CRISMA ICMS Architecture Document V2

Pascal Dihé, Martin Scholl, Sascha Schlobinski,
Thorsten Hell and Steven Frysinger, CIS
Peter Kutschera, Manuel Warum and Denis Havlik, AIT
Arnaud DeGroof and Yannick Vandeloise, SPB
Oren Deri, NICE
Kalev Rannat, TTU
Jussi Yliaho, VTT
Ari Kosonen, INS
Martin Sommer, EADS
Wolf Engelbach, FRA

28.2.2014

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 284552.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 284552 "CRISMA“

Deliverable No.		D32.2	
Subproject No.	3	Subproject Title	Integrated Crisis Modelling System
Work package No.	32	Work package Title	Crisis Modelling System Architecture
Authors		Pascal Dihé (CIS), Martin Scholl (CIS), Sascha Schlobinski (CIS), Thorsten Hell (CIS), Steven Frysinger (CIS), Peter Kutschera (AIT), Manuel Warum (AIT), Denis Havlik (AIT), Arnaud DeGroof (SPB), Yannick Vandeloise (SPB), Oren Deri (NICE), Kalev Rannat (TTU), Jussi Yliaho (VTT), Ari Kosonen (INS), Martin Sommer (CASS), Wolf Engelbach (FRA)	
Status (F = Final; D = Draft)		F	
File Name		CRISMA_D322_final.pdf	
Dissemination level (PU = Public; RE = Restricted; CO = Confidential)		PU	

Contact	Anna-Mari.Heikkila@vtt.fi Crisma.Coordinator@vtt.fi
Project	www.crismaproject.eu

Keywords	CRISMA, Architecture, Framework, Concepts, Building Blocks, Implementation
Deliverable leader	Name: Pascal Dihé Partner: CIS Contact: pascal.dihe@cismet.de
Contractual Delivery date to the EC	28.2.2014
Actual Delivery date to the EC	28.2.2014

Document Revision History

Version	Date	Who	Comments
0.1	11.11.2013	Pascal Dihé	Initial draft structure and description of designated content
0.2	02.12.2013	Pascal Dihé	Initial CIS Contributions
0.3	17.12.2013	Pascal Dihé	Updated Partner Contributions
0.4	08.01.2014	Pascal Dihé	Updated Mission chapter completed
0.4.1	13.01.2014	Sascha Schlobinski	Conceptual Business Logic chapter completed
0.5	14.01.2014	Pascal Dihé	Concepts chapter completed
0.6	22.02.2014	Pascal Dihé	Realisation and Implementation chapters completed
0.7	24.01.2014	Pascal Dihé	Annex A & B completed
0.8	07.02.2014	Pascal Dihé	Final draft for pre-review
0.9	14.02.2014	Reiner Güttler	Pre review
0.9.1	20.02.2014	Sascha Schlobinski	SP-Leader Pre-Review
1.0	27.02.2014	Pascal Dihé	Final Version and Updates after Review
1.0.1	27.02.2014	Sascha Schlobinski	SP-Leader Review & approval
Final	28.2.2014	Coordinator	SSA approved and styles corrected by the Coordinator

Table of Contents

Table of Appendices

List of Figures

List of Tables

Glossary of terms

Term	Definition
Access control	The ability to enforce a policy that identifies permissible actions on a particular resource by a particular subject. (SANY-SA, 2009)
Agent	A kind of "autonomous" and “active” OOI that does have control over its behaviour, can act in the environment, perceive, and reason.
API	An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other. See http://en.wikipedia.org/wiki/API
Application	See CRISMA application
Architecture	In computer science and engineering, computer architecture is the art that specifies the relations and parts of a computer system. See https://en.wikipedia.org/wiki/Computer_architecture
Attribute	A characteristic of an entity (e.g. of an object in a model or the model itself).
Authentication	Concerns the identity of the participants in an exchange. Authentication refers to the means by which one participant can be assured of the identity of other participants. (SOA-RM, 2006)
Authorisation	Concerns the legitimacy of the interaction. Authorisation refers to the means by which an owner of a resource may be assured that the information and actions that are exchanged are either explicitly or implicitly approved. (SOA-RM, 2006)
Building block	A Building Block is a CRISMA Federate that is provided as part of the CRISMA Framework to build CRISMA Applications. Building Blocks are generic, composable, adaptable as well as domain- and location-independent and thus transferable to different crisis management domains. There are three different types of Building Blocks: Infrastructure, Integration and User Interaction Building Blocks. Not all Building Blocks are mandatory in each CRISMA Application.
Crisis scenario	A scenario describing a crisis as it has happened or could happen in real life.
CRISMA application	A CRISMA Application is an integrated crisis management simulation system that is build according to the concepts of the CRISMA Framework Architecture. It is composed of (customised) Building Blocks of the CRISMA Framework and integrated or federated legacy components (simulation models, applications, systems,).
CRISMA federate	Any component that connects to the Middleware Infrastructure of the CRISMA Framework and is able to exchange Control and Communication Information with the Middleware Infrastructure. More specifically, a CRISMA Federate has to be aware of the API of the ICMM.
CRISMA federation	A number of CRISMA Federates that act together as a unit. A CRISMA Federation is a subset of a CRISMA Application.
CRISMA framework	A framework composed of ready-to-use Building Block and supporting tools that can be connected together to form a CRISMA Application.
Criteria	Criteria relate indicators to a qualitative assessment of the respective crisis situation. Indicators and corresponding criteria are basis for CRISMA decision support concept
Decision making	Decision making can be regarded as the cognitive process resulting in the selection of a course of action among several alternative scenarios. Every decision making process produces a final choice. The output can be an action or an opinion of choice. (Wikipedia, 2013)

Term	Definition
Decision point	A point in time where the user wants to introduce a decision to compare what will be the outcome of different options. In CRISMA, visualisation is done on a time line to show where a decision has been met.
Decision support	Decision Support Systems (DSS) make up a specific class of computerized information systems that support business and organizational decision-making activities. A properly designed DSS is an interactive software-based system intended to help decision makers to compile useful information from raw data, documents, personal knowledge, and/or business models to identify and solve problems and make decisions.
Decision tree	A time line with branches showing options in each decision point. Not only useful as visualisation of decisions but also for navigation to compare different states at the same or different time stamps.
Evaluation	Evaluation is “systematic investigation of the worth or merit of an object.” (Frechtling 2011). Evaluation is a valuable source of information on how a project is being implemented, specifically, what works and what should be modified.
Evolution scenario	A combination of several impact scenarios resulting from multiple hazards/incidents.
Feature	A "Feature" (ISO 19101, OGC 08-126) is "an abstraction of a real world phenomenon". A feature is considered "geographic feature" if it is associated with a location. According to OGC 08-126, the geographic features are "fundamental unit of geospatial information".
Federate	see CRISMA federate
Framework	An information architecture that comprises, in terms of software design, a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with application code. See http://www.opengeospatial.org/resources/?page=glossary More specific see CRISMA framework
Functional requirement	Defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. See https://en.wikipedia.org/wiki/Functional_requirements
Functional specification	A functional specification is an implementation independent description of a software components behaviour which means that operations are specified on an abstract level not defining specific data types or schemas. It is the basis for a Formal Specification.
Incident	An occurrence, natural or human-caused, that requires a response to protect life or property. Incidents can, for example, include major disasters, emergencies, terrorist attacks, terrorist threats, civil unrest, wild land and urban fires, floods, hazardous materials spills, nuclear accidents, aircraft accidents, earthquakes, hurricanes, tornadoes, tropical storms, tsunamis, war-related disasters, public health and medical emergencies, and other occurrences requiring an emergency response (FEMA Glossary, 2013).
Indicator	A thing that indicates the state or level of something (Oxford dictionaries). In CRISMA, indicators illustrate World State aspects that are representative for scenarios. The indicators can either be represented by a single existing value in the CRISMA World State or are aggregated by a function of several values from the World State (e.g. average, minimum or maximum values in a given region for a given time).

Term	Definition
Information model	An information model is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. The advantage of using an information model is that it can provide sharable, stable, and organized structure of information requirements for the domain context. (Lee, 1999)
Infrastructure	Infrastructure in a crisis management context covers for instance dikes, hospitals, rescue bases and critical infrastructure (as water, power, telecommunication networks) etc.
Integrated Crisis Management Middleware (ICMM)	The ICMM is a central Building Block in every CRISMA Application. It connects Crisis Management Simulations with the Analysis and Decision Support functionality of CRISMA by providing a central repository for harmonized world state and indicator information. The ICMM is fed by simulations providing the basic information to be used for world state analysis and decision support Building Blocks.
Integrated component	A component that takes part in an interaction of CRISMA Federates but is not itself a CRISMA Federate. It is not CRISMA-aware and thus does not interact with the CRISMA Middleware Infrastructure. An Integrated Component may be a member of a CRISMA Application but not of a CRISMA Federation.
Interface	In the context of IT, a named set of operations that characterise the behaviour of an entity. The aggregation of operations in an interface, and the definition of the interface, shall be for the purpose of software re-usability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations. (ISO_19119, 2003)
Interoperability	Capability to communicate, to execute programs, or to transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units. (ISO_19119, 2003)
Key performance indicator	Key performance indicator (KPI) help to characterise and compare alternative scenarios with respect to the crisis management measures that have been done.
Mashup platform	A component that allows to combine smaller components (widgets) providing specific functionality into a complete graphical user interface (GUI) to provide all the functionality needed
Meta-information	Descriptive information about resources in the universe of discourse. Its structure is given by a Meta-Information Model depending on a particular purpose. Note: A resource by itself does not necessarily need Meta-Information. The need for Meta-Information arises from additional tasks or a particular purpose (like catalogue organisation), where many different resources (services and data objects) must be handled by common methods and therefore have to have/get common attributes and descriptions (like a location or the classification of a book in a library). (RM-OA, 2007)
Meta-information model	Information-model for Meta-Information. See Meta-model and Meta-information.
Meta-model	A meta-model typically defines the language and processes from which to form a (description) model. Meta-modelling is the modelling methodology used in software engineering. See https://en.wikipedia.org/wiki/Meta_model. In this IT context "model" usually means "data model" or "information model" – a description of data structures.

Term	Definition
Meta-modelling	Meta-modelling is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for the modelling in a predefined class of problems. See https://en.wikipedia.org/wiki/Metadata_modeling
Model	A model is a hypothetical simplified description of a complex entity or process (Sterling & Taveter, 2009). A model can be considered as “an abstract representation of a system or process” (Carson, 2005). A model is a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process that has been designed for a specific purpose (NATO, 2010). Stachowiak (1973) describes a model using three features: the mapping feature (reproduction of the original), the reduction feature (abstraction of the original) and the pragmatic feature (addressing a purpose for its user).
Multi-hazard assessment	To determine the probability of occurrence of different hazards either occurring at the same time or shortly following each other, because they are dependent from one another or because they are caused by the same triggering event or hazard, or merely threatening the same elements at risk without chronological coincidence. (Commission staff working paper: “Risk assessment and mapping guidelines for disaster management”, European Commission, Brussels, December 2010)
Multi-risk assessment	To determine the whole risk from several hazards, taking into account possible hazards and vulnerability interactions (a multi-risk approach entails a multi-hazard and multi-vulnerability perspective). This would include the events: 1) occurring at the same time or shortly following each other, because they are dependent on one another or because they are caused by the same triggering event or hazard; this is mainly the case of cascading events; 2) or threatening the same elements at risk (vulnerable/exposed elements) without chronological coincidence (EU Matrix)
Non-functional requirement	A requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. See https://en.wikipedia.org/wiki/Non-functional_requirement
Object of interest	Object of Interest (OOI) is used in CRISMA to designate objects that are of interest to crisis management practitioners and therefore need to be represented and handled by a CRISMA Application. More precisely, the term is used for IT-representation of such objects within CRISMA. The term was initially introduced as disambiguation of the word "resources". However, the OOI can also represent objects which aren't considered resources by crisis managers, such as hospitals, dams or residential buildings. Since OOI instances always exist in a spatial and temporal context, OOI can be considered a specialization of the "Feature" as defined by ISO 19101 and OGC 08-126.
OOI	see Object of Interest
Pilot	A pilot within CRISMA manages the definition and implementation of a CRISMA application and provides the data for a specific crisis scenario and for a specific use case. This results in a demonstrator that is used during a demonstration and for experimentation purposes in order to validate the CRISMA software. CRISMA pilots (pilot sites) provide generic experimentation frame and validations for testing, validate and promote CRISMA system and provide necessary return of experience in order to validate CRISMA in relevant wide range of crisis management situations including multi-risk and domino effects.

Term	Definition
Point of interest	A specific point location that someone may find useful or interesting (see http://en.wikipedia.org/wiki/Point_of_interest). In CRISMA context, the Point of Interest (POI) is a map object, usually visualized with a marker/icon on the map. POI is an object with properties, like geographic location, and/or some properties characterizing this object. The POI can represent both passive (like buildings, bridges, etc.) and active objects (e.g. agents as representatives of some physical or virtual entities) on the map.
Preparedness	The knowledge and capacities developed by governments, professional response and recovery organizations, communities and individuals to effectively anticipate, respond to, and recover from, the impacts of likely, imminent or current hazard events or conditions.
Recovery	The restoration, and improvement where appropriate, of facilities, livelihoods and living conditions of disaster-affected communities, including efforts to reduce disaster risk factors.
Reference architecture	A reference architecture "is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted." (SOA-RM, 2006)
Reference scenario	Reference scenarios are abstract crisis and response scenarios that help to understand the relevant aspects of a typical scenario.
Requirement	A singular documented physical and functional need that a CRISMA Application should perform. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system for it to have value and utility to a user. (Source: http://en.wikipedia.org/wiki/Requirement)
Resource	Crisis management context: Resources are deployed in the scope of crisis management activities. Resources may include material (e.g. sandbags, medical products, and oxygen tank), personnel (e.g. medical officer, ambulance driver, and crisis manager), vehicles (e.g. fire trucks), protection infrastructure and facilities (e.g. hospital, shelters), installations (e.g. weirs). IT-context: Resource is every possible data object as part of the common CRISMA meta information model.
Resource management	Crisis management context: Management and optimal deployment of crisis management resources such as fire trucks, ambulances, personnel, equipment etc. for crisis management. IT-context: Management in terms of storage, creation, update and delete of data objects in the context of IT implementation.
Response	Immediate actions to save and sustain lives, protect property and the environment, and meet basic human needs. Response also includes the execution of plans and actions to support short-term recovery. (FEMA, 2010)
Risk analysis	A systematic use of available information to determine how often specified events may occur and the magnitude of their likely consequences.
Sample scenario	Crisis and response scenario representing either real past events or imaginary events specified in detail.
Scenario	A description of a projected situation and the causes leading to it. Scenarios are used to evaluate the outcomes of different courses of actions in changing conditions.
Scenario description	In the context of CRISMA it includes text, timeline and tables that describe a crisis management scenario.

Term	Definition
Scenario evolution	Sequential development of a scenario describing events as they could evolve based on alternative assumptions.
Service	In the context of IT, a service is a distinct part of the functionality that is provided by an entity through interfaces. (ISO_19119, 2003)
Simulation	A simulation is the manipulation of a model in such a way that it represents the expected behaviour of an individual actor or an entire system over time (NATO, 2010).
Simulation case	A simulation case describes the user context, the objects of interests, the simulation models and the user interactions that are relevant for the modelling and simulation of a crisis management scenario with a CRISMA application.
Simulation models loose integration	Describes a low level of integration between two or more simulation models, when a running simulation model is not aware of other running simulation models. The output of one simulation model can be used as the input for the next simulation model. This level of integration is required when two simulation models are running in-parallel with no need to share information during the run.
Simulation models tight integration	Describes a high level of integration between two or more simulation models at runtime. This level of integration is required when two simulation models are running in-parallel and need to interact with each other by sharing their intermediate output information during the run.
Simulation results	Simulation results are the output values of a specific simulation run. They provide the content of a specific scenario (series of World States).
Simulation run	A simulation run is the parameterisation and initiation of a simulation case that provides specific simulation results. A simulation run can happen with or without user interaction.
Simulation tool	A software that implements a model and allows simulation, no matter if it provides an own user interface or operates integrated in a larger software.
Simulation view	A CRISMA user interface that allows a CRISMA user to interact with a simulation case in terms of changing simulation parameters, starting, stopping and saving simulation runs.
Standard	Denotes relevant technical standards on interoperability and federated simulation.
System	System is a set of entities connected together to make a complex whole or perform a complex function (Sterling & Taveter, 2009). System can also be defined as a complex of interacting components and relationships among them that permit the identification of a boundary-maintaining entity or process (Laszlo & Krippner, 1998).
Technical requirements	Technical requirements define the functions or constraints needed by the system to achieve a goal. They specify how a goal should be accomplished by the system. Technical requirements can be separated according to their characteristics into functional and non-functional technical requirements. In CRISMA, technical requirements are created based on user stories.
Tight simulation integration	An approach towards coupling of several simulation models at a domain-specific, conceptual, and semantic level. The standards DIS and HLA are designed for such integrated simulations. An example would be a system that integrates several vulnerability models of different infrastructures and takes into account interrelationships between them at any time of a simulation run.
Transition	A transition is a relationship between two CRISMA World States. It can originate either from a CRISMA simulation model run or from a manual change by a CRISMA user.

Term	Definition
Use case	A list of steps, typically defining interactions between a user and a system, to achieve a certain goal (Cockburn, 1999). A use case is described as "a generalized description of a set of interactions between the system and one or more actors, where an actor is either a user or another system". (Cohn, 2004).
User requirements	User requirements represent what goals in crisis management need to be achieved by means of the system and what roles of stakeholders are involved in that. In CRISMA user requirements are represented by crisis-specific goal models and behavioural scenarios and the consolidated crisis management goal model for the ICMS.
Widget	A component providing a specialized user interface functionality. A complete user interface consists of a number of widgets.
World State	A CRISMA World State is a set of attributes and their values describing the state of the world in a crisis management scenario.
Cascade events	Cascade events describe a sequence of adverse events generated by a single or different sources. For example an earthquake that causes ground motion that triggers a landslide.
Cascade effects	Cascade effects describe the effects (consequences) of cascade events and thus allow evaluating indirect effects caused by the originating incident.
CRISMA system	In the perception of the architecture, the CRISMA System is the overall project results consisting of all CRISMA Pilot Applications.
Application architecture	An Application Architecture provides a specification of application-specific Simulation Cases in accordance to the Integrated System Viewpoint of the Conceptual Business Logic of the CRISMA Framework Architecture.
Transition Point	A Transition Point represents the type of the Transition that can be performed by the user at a certain defined point in a Simulation Case.
Transition matrix	A matrix-like representation of the conditional probabilities P(IM2|IM1), indicating the probability that a triggered event with intensity IM2 occurs given the occurrence of a triggering event with intensity IM1.
Decision node	Element of a decision tree which represents a decision (e.g., to assess different possible mitigation actions) to be taken in a particular segment of a decision tree.
Database of scenarios	A collection of plausible scenarios of cascading effects.
Time dependent vulnerability	Referring to physical vulnerability, time-dependent vulnerability is defined as the vulnerability affected by deterioration of elements characteristics due to ageing and/or damage. In a broader sense, time dependent vulnerability generally indicates the variation of vulnerability characteristics over time (in the understanding of CRISMA, this e.g. also includes spatio-temporal patterns of exposure or varying situation patterns during the process of evacuation).
ICMM	Integrated Crisis Management Middleware BB (ICMM) is a generic distributed resource-oriented Control and Communication Information Management System. Thereby it is important to note, that the term ‘resource-oriented’ in the ICMM refers to the concept of generic resources as used in the context of the Resource Oriented Architecture (ROA).
Path of analysis	A Path of Analysis is a consecutive sequence of Transitions through a Simulation Case Graph. A Path of Analysis that starts at the root of a Simulation Case and ends at a leaf can be called a Simulated Crisis Management Scenario.
Simulation Objective	A Simulation Objective is a well-defined purpose of a Crisis Management Simulation supporting specific planning, training or decision support activities. The physical representation or the means to reach a simulation objective is the Simulation Case.

Acronyms and technical terms

The table below lists technical terms and acronyms that are not part of the CRISMA glossary but that are nevertheless needed to understand the content of this deliverable.

Term	Definition
Application Programming Interface (API)	An application programming interface (API) specifies how some software components should interact with each other. See https://en.wikipedia.org/wiki/Application_programming_interface
Cascading Style Sheets, Level 3 (CSS3)	Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. See https://en.wikipedia.org/wiki/Cascading_Style_Sheets
Core Control and Communication Information Model (CCIM)	The core meta-information model of a CRISMA Federation that provides a formal specification of the Conceptual Business Logic of the CRISMA Framework.
DIESIS	The EU FP7 Project “Design of an Interoperable European Federated Simulation Network for Critical Infrastructures”. See http://www.diesis-project.eu
Distributed Interactive Simulation (DIS)	An IEEE standard for conducting real-time platform-level war gaming across multiple host computers and is used worldwide, especially by military organizations but also by other agencies. See https://en.wikipedia.org/wiki/Distributed_Interactive_Simulation
Data Distribution Services (DDS)	The Data Distribution Service for Real-Time Systems (DDS) is an Object Management Group (OMG) Publish/Subscribe (P/S) standard. See http://omg.org/
Decision Support System (DSS)	A software system supporting a decision maker by providing information relevant for his decisions.
Enterprise Service Bus (ESB)	A software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in service-oriented architecture (SOA). See https://en.wikipedia.org/wiki/Enterprise_service_bus
Event-driven architecture (EDA)	A software architecture pattern promoting the production, detection, consumption of, and reaction to events. See https://en.wikipedia.org/wiki/Event-driven_architecture
Extensible Markup Language (XML)	A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. See (XML, 2010)
FI-WARE	Future internet core platform. Delivers a novel service infrastructure, building upon elements (called Generic Enablers) which offer reusable and commonly shared functions making it easier to develop Future Internet Applications in multiple sectors. (FI-WARE, 2011)
File Transfer Protocol (FTP)	File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet. See https://en.wikipedia.org/wiki/FTP
GeoServer	GeoServer is an open source software server written in Java that allows users to share and edit geospatial data. See http://geoserver.org
Geography Markup Language (GML)	The Geography Markup Language (GML) is the XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features. See https://en.wikipedia.org/wiki/Geography_Markup_Language

Term	Definition
High-level architecture (HLA)	A general purpose architecture for distributed computer simulation systems. HLA is an open international standard developed by the Simulation Interoperability Standards Organization (SISO) and published by IEEE. See https://en.wikipedia.org/wiki/High-level_architecture_%28simulation%29
Hypertext Transfer Protocol (HTTP)	A standard protocol for data transport in the world wide web (WWW). See (RFC_1945, 1996) and (RFC_2616, 1999)
Integrated Modelling, Mapping, and Simulation (IMMS)	A framework for planning exercises. See SUMMIT.
JavaScript (JS)	JavaScript is an interpreted computer programming language. As part of web browsers, implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. See https://en.wikipedia.org/wiki/JavaScript
Local Area Network (LAN)	A computer network within one organisation unit, usually within one company or building.
Mission, Concepts, Realisation and Implementation (MCRI)	The Mission, Concepts, Realisation and Implementation schema used to structure the CRISMA Framework Architecture into four architectural layers that build on each other.
Model as a Service (MaaS)	An external provider offers IT infrastructure providing a model ready to use.
Network Common Data Format (NetCDF)	NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. See http://www.unidata.ucar.edu/software/netcdf/
OAuth	An open standard for authorization. OAuth provides a method for clients to access server resources on behalf of an end-user (OAuth, 2012)
Object Management Group (OMG)	An international, open membership, not-for-profit computer industry standards consortium. See http://www.omg.org/
Observations and Measurements (O&M)	One of the OGC standards. Specifies an XML implementation for the OGC and ISO Observations and Measurements conceptual model. See (O&M, 2011).
Open Geospatial Consortium (OGC)	The Open Geospatial Consortium (OGC) is an international industry consortium of 480 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards. OGC® Standards support interoperable solutions that "geo-enable" the Web, wireless and location-based services and mainstream IT. The standards empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications. (OGC, 2013)
OpenId	A standard for user identification. See chapter 4.4 Interoperability standards
Open Modelling Interface (Open-MI)	A standard defines a set of interfaces that allowing time-dependent models to exchange data at run-time. See https://en.wikipedia.org/wiki/OpenMI_Standard
Remote procedure call (RPC)	A concept describing the access to computer program procedures located not in the own program but anywhere in a computer network, completely hiding complexity .There are many different realisations of this concept.. See (RPC, 2013)
Representational State Transfer (REST)	A style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant web service design model. See (REST, 2013)

Term	Definition
RiskScape	A framework for multi-risk modelling. Website at www.riscscape.org.nz not available (2013-03-06)
Run-Time Infrastructure (RTI)	A service Bus providing HLA services. See High-level Architecture (HLA) and Enterprise Service Bus (ESB).
Sensor Model Language (SensorML)	One of the OGC standards. Specifies models and XML encoding of sensor descriptions. See (SensorML, 2007)
Sensor Observation Service (SOS)	One of the OGC standards. Specifies a service interface and data encoding methods for sensor data access. See (SOS, 2012
Sensor Planning Service (SPS)	One of the OGC standards. Specifies service interface and data encodings to control sensors, where sensor is a very wide concept ranging from simple things like a thermometer to earth observation satellites and also, including simulation models. See (SPS, 2011).
Sensor Web Enablement (SWE)	A suite of standards developed by Open Geospatial Consortium (OGC) describing basic data and service models used in other OGC standards. See (SWEdata, 2011) and (SWEservice, 2011)
Service Oriented Architecture (SOA)	A style of software architecture for designing and developing software in the form of interoperable services. See https://en.wikipedia.org/wiki/Service-oriented_architecture
SOAP (Simple Object Access Protocol)	A protocol specification for exchanging structured information in the implementation of Web Services in computer networks. See https://en.wikipedia.org/wiki/SOAP
Software as a Service (Saas)	An external provider offers IT infrastructure and software ready to use.
Standard Unified Modelling, Mapping & Integration Toolkit (SUMMIT)	A modelling & simulation software environment that enables to seamlessly access integrated suites of modelling tools & data sources. See https://dhs-summit.us
Test and Training Enabling Architecture (TENA)	Architecture designed to promote integrated testing and simulation-based acquisition. See https://www.tena-sda.org
The Open Group Architecture Framework (TOGAF®)	A framework for enterprise architectures which provides a comprehensive approach for designing, planning, implementing, and governing an enterprise information architecture. See https://en.wikipedia.org/wiki/TOGAF
Unified Modelling Language (UML)	A standardized general-purpose modelling language. Defines different diagram types and a data exchange format. Many tools are available; some even allow generation of e.g. program code from the diagrams. See (UML, 2013)
Web Coverage Service (WCS)	One of the OGC standards. Can be compared to WMS, but delivers actual data instead of pictures of the data. So in contrast to WMS the data can be used for further processing. See (WCS, 2010).
Web Distributed Authoring and Versioning (WebDAV)	Web Distributed Authoring and Versioning (WebDAV) is an extension of the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers. See https://en.wikipedia.org/wiki/WebDAV
WebKit	WebKit is an open source web browser engine. See http://www.webkit.org/
Web Map Service (WMS)	One of the OGC standards. Defines request parameters and output formats for the delivery of Maps as pictures. See (WMS, 2006).
Web Processing Service (WPS)	One of the OGC standards. Used to control processes, e.g. simulation model runs. Defines rules for parameters, input and output data requests. See (WPS, 2007)

Term	Definition
Wirecloud	Wirecloud is a web mashup platform to integrate heterogeneous data, application logic, and UI components (widgets/gadgets) sourced from the Web. See http://conwet.fi.upm.es/wirecloud/

Executive Summary

The document at hand is a revision of the initial version of the architecture specification of the CRISMA Framework (D32.1, 2013) and the addresses updates and aspects not considered in V1. This document is a deliverable of Sub-Project 3 (Integrated Crisis Modelling System) corresponding to WP32 (Crisis Modelling System Architecture), reporting on the results of the work performed in the context of the tasks Task 3.2.2 "ICMS Functional Architecture Update" (m20–m23) and Task 3.2.4 "ICMS Implementation Architecture Update" (m21–m24).

The main objective of this document is to provide an exhaustive specification of the architecture of the CRISMA Framework. Thereby it outlines the implementation independent Functional Architecture, describing the general aspects of the CRISMA Framework as well as the Implementation Architecture, specifying a software framework which constitutes a concrete realisation of the Functional Architecture.

In particular, the architecture specification includes the definition of the architectural concepts in relation to project objectives and user requirements as well as the specification of Functional Building Blocks of the CRISMA Framework. In addition, the document discusses the relevant standards used for implementation of the framework, introduces the concepts of the Implementation Architecture and presents a template for defining the pilot-specific Application Architectures as well as a specification of an information model supporting interoperability and integration aspects and a set of rules and guidelines for the implementation of Building Blocks.

The first architecture specification (D32.1, 2013) took into account the explicit user requirements expressed in D23.1 (2013), research issues and technical requirements identified in D31.1 (2012) as well as the requirements inherent to CRISMA goals, and expressed in the DoW (2013) and D11.1 (2012). The resulting Functional Architecture has set the stage for the first round of technical developments at the level of SP3 (D34.1, D35.1), as well as at the level of SP4 (primarily the D42.1, D43.1 and D44.1) and SP5 (D51.x). Prioritization, elaboration of the Building Block specifications and implementation thereof continued in the scope of D34.1. Soundness of the proposed architectural concepts has been re-evaluated in the scope of D31.2, taking into account the feedback from external experts expressed in D23.2, as well as the findings of D41.2 on interface specification, and the pilot specifications.

The second version of the architecture specification has been largely influenced by the work performed in WP34 "Implementation" and the early validation of the framework performed in SP5.The focus of the second of version of CRISMA Framework Architecture has been laid on the specification of the Implementation Architecture which has been carried out jointly in T3.2.3 and T3.4.1.

The main achievements of this document are: (1) the clarification of the main architectural concepts answering the various CRISMA requirements (chapter 3); (2) the definition of the Functional Architecture and Building Blocks realising these concepts (chapter 4); and (3) the specification of the Implementation Architecture as basis for the implementation of the CRISMA Framework and the development of individual CRISMA Applications of the pilots (chapter 5) in cooperation with teams working on SP3, SP4 and SP5.

1. Introduction

1.1. Purpose of this document

The goal of this document is to specify the CRISMA Framework Architecture both on functional and implementation level. In particular, this includes the definition of the architectural concepts, Functional Building Blocks and their interactions. In addition, the document also discusses the relevant standards and introduces the concepts of the Implementation Architecture which deals with the implementation of Building Blocks as Software Components and the development of individual CRISMA Applications with help of the CRISMA Framework.

CRISMA developments are positioned in a conflicting field linking the expectations and state of the art software systems used by crisis management practitioners and regional planners on the one side, technological requirements and inherent limitations of the Simulation Models on the other, and environmental and geospatial ICT dictating the standardized access services and information models for large portion of the data used in crisis management. In this setup, building an architecture from scratch, without taking into account existing infrastructure, legacy systems, services, data and standards is not feasible.

Four major challenges in developing the CRISMA Functional Architecture are thus posed by: (1) the need to leverage the legacy and often conflicting ICT systems outside of CRISMA control; (2) to define a coherent set of functions answering implicit and explicit requirements of our (potential) users and existing infrastructure; (3) to maximize the re-use of legacy software within CRISMA while identifying the gaps; and (4) to maximize the usability of the CRISMA result in stand-alone setups as well as in setups where CRISMA components are used as add-ons to existing software.

1.2. Intended audience

The target readers of this document are the CRISMA developers in SP3 and SP4, the technical partners in charge of the software integration in WP5.x (Pilots) as well as individuals interested in the CRISMA project.

1.3. Structure of the document

The structure of the document and the relationships between the different chapters and annexes is as follows:

Chapter 1 (this chapter) introduces the document.

Chapter 2 defines the “mission” of the CRISMA Framework Architecture including its purpose and scope in terms of architectural goals, properties and constraints which have been derived from the project objectives and pilot expectations documented in (DoW, 2013) and (D11.1, 2012), the research issues and technical requirements identified in (D31.1, 2012) and (D31.2, 2013), and the detailed user requirements expressed in (D23.1, 2013).

Chapter 3 elaborates the architectural concepts describing various aspects of the CRISMA architecture resulting from the mission defined in chapter 2.

Chapter 4 introduces the functional Building Blocks and gives an overview about standards relevant for CRISMA interoperability.

Chapter 5 provides a mapping of the functional Building Blocks defined in chapter 4 to Software Components, specifies a unified information model for CRISMA, introduces a template for the specification of Application Architectures and defines rules and guidelines for the development of Building Blocks and CRISMA Applications.

Chapter 6 provides the conclusions and outlook on follow-up activities.

Chapter 7 lists the references to literature used in preparation of this deliverable.

Annex documents provide a formal specification of the core information model of the CRISMA Framework and the complete Application Architecture Template.

1.4. Changes to this document

The major achievements and results of the second version of the CRISMA Framework Architecture in relation to the chapters and sections of this document that have been updated and introduced are listed in the following table:

Table 1: Improvements to the CRISMA Framework Architecture V2.

Updates and improvements	**Sections affected**
Description of relation to other deliverables and SP2, SP4 and SP4 updated.	1.5, 1.6
Mission of the CRISMA Framework Architecture updated with the latest findings of the 2nd technical requirements specification document (D32.2, 2013), especially regarding non-functional requirements.	2, 2.2
General small updates and improvements to all sections of this chapter. Sections with major updates are listed in the following.	3
Extension of the Conceptual Business Logic and the associated terminology, refinement of concepts of Crisis Management Simulation Scenario Analysis, consideration of recent SP2 and SP4 work related to Indicators, Criteria, Cascade Events and Time-Dependent Vulnerability, introduction of the concept of Transition Points and numerous other improvements.	3.1
Update according to the latest improvements to the Integrated Crisis Management Middleware in WP34.	3.5
Update according to the latest improvements to the Core Control and Communication Information Model in WP34 and WP32.	3.6
Terminology updated according to the extensions introduced in D34.1 (D34.1, 2013).	3.7
Complete refinement of the GUI Integration Approach under special consideration of aspects regarding integration and compatibility with legacy systems.	3.12
Improvement of the Data Integration Approach regarding the integration of legacy data sources which are not compatible to (OGC) standards.	3.13
Improvement and extension of the Simulation Model Integration Approach and description of the support for Cascade Events.	3.14

Updates and improvements	Sections affected
New section to highlight the aspects of compatibility and integration with legacy systems including a summary of the integration concepts pursued by the architecture.	3.15
New section to highlight the aspects of transferability of the CRISMA Framework and its Building Bocks as envisioned by the architectural concepts described in this document.	3.16
Alignment of the Training Concepts to the updated Conceptual Business Logic.	3.17
Refinement and update of the Resource Management Concepts to reflect the latest improvements to the Resource Management related Building Blocks and the integration effort of WP35.	3.18
Alignment to improved SP4 concepts of Cascade Events and Time-Dependent Vulnerability (Transition Matrix).	3.19
Description of the Access Control Concept and its relation to Building Blocks updated.	3.20
Introduced administration as a general system concept.	3.21
Update of all Building Block descriptions according to the latest changes and improvements made in WP32, WP34 and W35, respectively, and provided references to the CRISMA Catalogue.	4.1, 4.1.1, 4.1.2 and 4.1.3
List of interoperability standards and their intended usage within the CRISMA Framework updated.	4.4
Specification of the Implementation Architecture of the CRISMA Framework completed.	5
Description of all Software Components that have been selected, adapted or developed in WP34 for the realisation of CRISMA Framework Building Blocks.	5.1
New specification of the Core Control Information Model of the CRISMA Framework Architecture.	5.2
New specification of the Application Architecture Template as basis for the development of individual CRISMA Applications.	5.3
Introduction of Rules and Guidelines for the development of Building Blocks and CRISMA Applications.	5.4
New appendices for the complete technical specification of the Application Architecture Template and the Core CCIM.	APPENDIX (A) and APPENDIX (B)

1.5. Relation to other deliverables

This section explains the relationship of the CRISMA Framework Architecture represented by the Architecture specification document at hand and the deliverable D32.2, respectively, to other deliverables of the CRISMA Project. Since this document is largely based on the first version of the Architecture specification (D32.1, 2012), the relations of the former D32.1 deliverable are included in the next section.

1.5.1. Relations of the architecture specification V1

As shown in the Figure 1, the first Architecture specification (D32.1, 2013) took into account the explicit user requirements expressed in D23.1 “(User) Requirements and Use

Cases” (D23.1, 2013), research issues and technical requirements identified in D31.1 “Technology, Concepts & Technical Requirements Report V1“(D31.1, 2012) as well as the requirements inherent to CRISMA goals, and expressed in the DoW (DoW, 2013) and D11.1 “Consolidation Report” (D11.1, 2012).

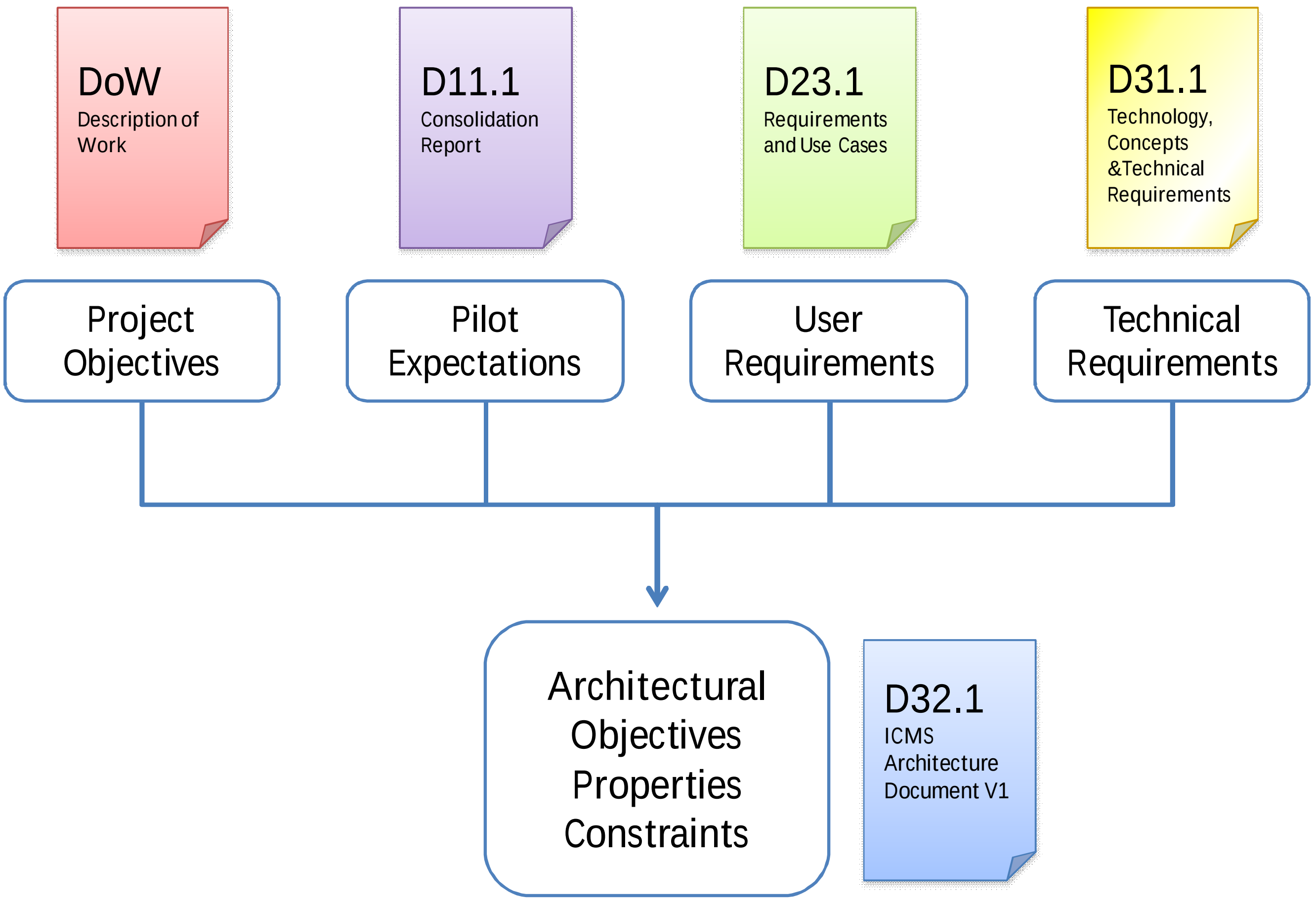

DoW
Description of Work
D11.1
Consolidation Report
D23.1
Requirements and Use Cases
D31.1
Technology, Concepts &Technical Requirements
Project Objectives
Pilot Expectations
User Requirements
Technical Requirements
Architectural Objectives Properties Constraints
D32.1
ICMS Architecture Document V1

Figure 1: Sources of architectural objectives, properties and constraints.

The examination of aforementioned results of various CRISMA work packages resulted in a clear and precise definition of the mission (chapter 2) of the CRISMA Framework Architecture. The definition of the mission enabled the architecture team of WP32 to develop and formulate the core concepts of the CRISMA Framework Architecture (chapter 3) and thus to establish a sound basis for their realisation (chapter 4). Mission, Concepts and Realisation together form the Functional Architecture (Figure 1), the major outcome of the CRISMA deliverable D32.1 “ICMS Functional Architecture Document V1”.

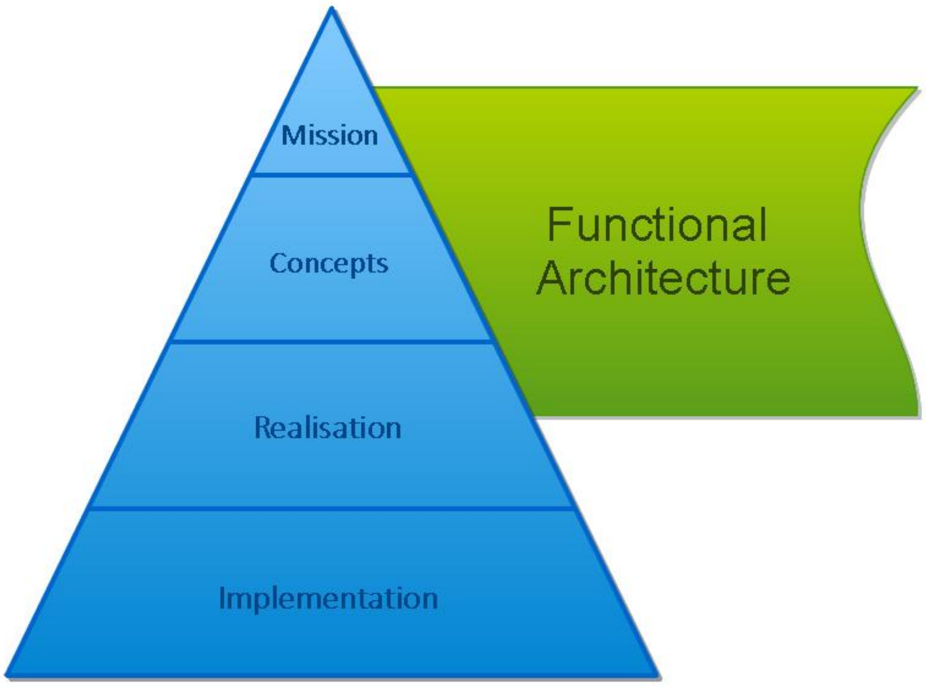


Figure 2: Functional Architecture of the CRISMA Framework.

As illustrated in Figure 3, the Functional Architecture set the stage for the first round of technical developments at the level of SP3 (D34.1 “ICMS Building Blocks V1”, D35.1 “ICMS Framework V1”), as well as at the level of SP4 (mainly D41.2, D42.1, D43.1 and D44.1).

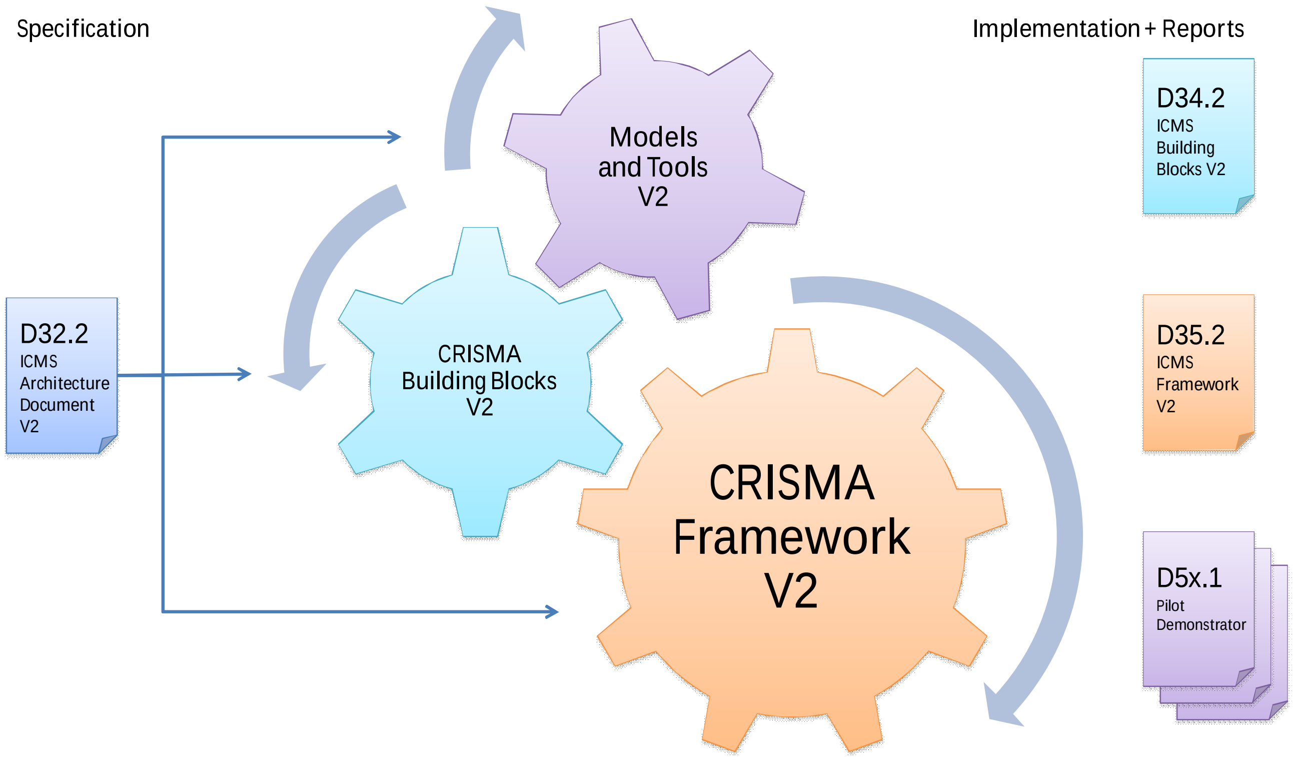


Figure 3: Deliverables and work packages dependent on the outcome of D32.1.

The prioritization and elaboration of the Building Block specifications and their implementation continued in the scope of D34.1 “ICMS Building Blocks V1”.

The soundness of the proposed architectural concepts established has be re-evaluated in the scope of D31.2 “Technology, Concepts &Technical Requirements Report V2”, taking into account the feedback from external experts expressed in D23.2 “External recommendations to the CRISMA Use Cases”, as well as the findings of D41.2 "(Model) Interface specification" and the pilot specifications (D51.2 “Pilot User Guide” and D51.1 "Site Status"). The workflow evaluating the first iteration of the CRISMA Framework Architecture is illustrated in Figure 4 below.

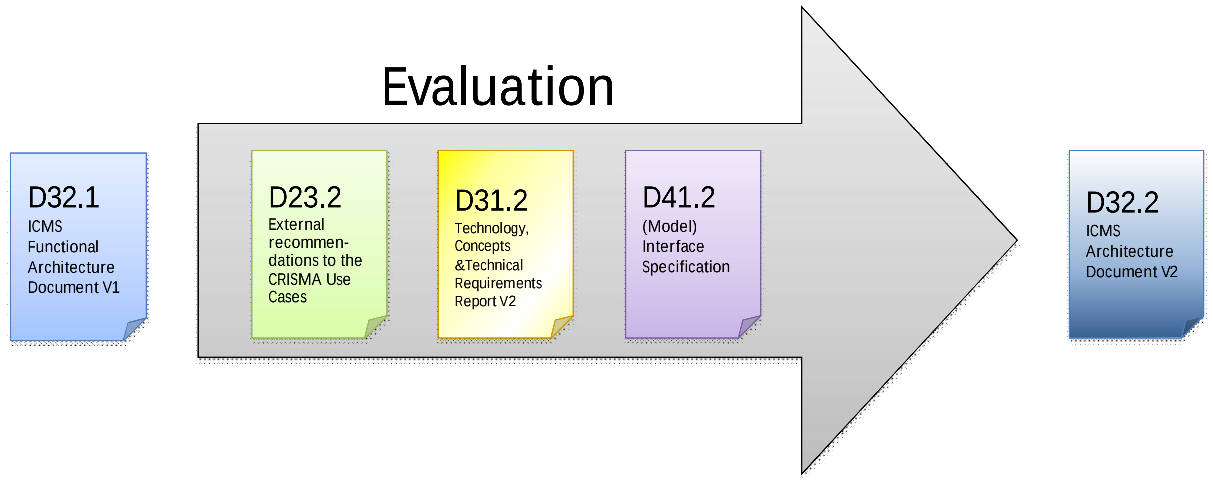

Evaluation
D32.1
ICMS Functional Architecture Document V1
D23.2
External recommen-dations to the CRISMA Use Cases
D31.2
Technology, Concepts &Technical Requirements Report V2
D41.2
(Model) Interface Specification
D32.2
ICMS Architecture Document V2

Figure 4: Evaluation of the CRISMA Framework Architecture.

As shown in Figure 5, D51.2 referenced concepts established during the preparation of D32.1. In particular, the concept of an Application Architecture was considered in this first deliverable of SP5.

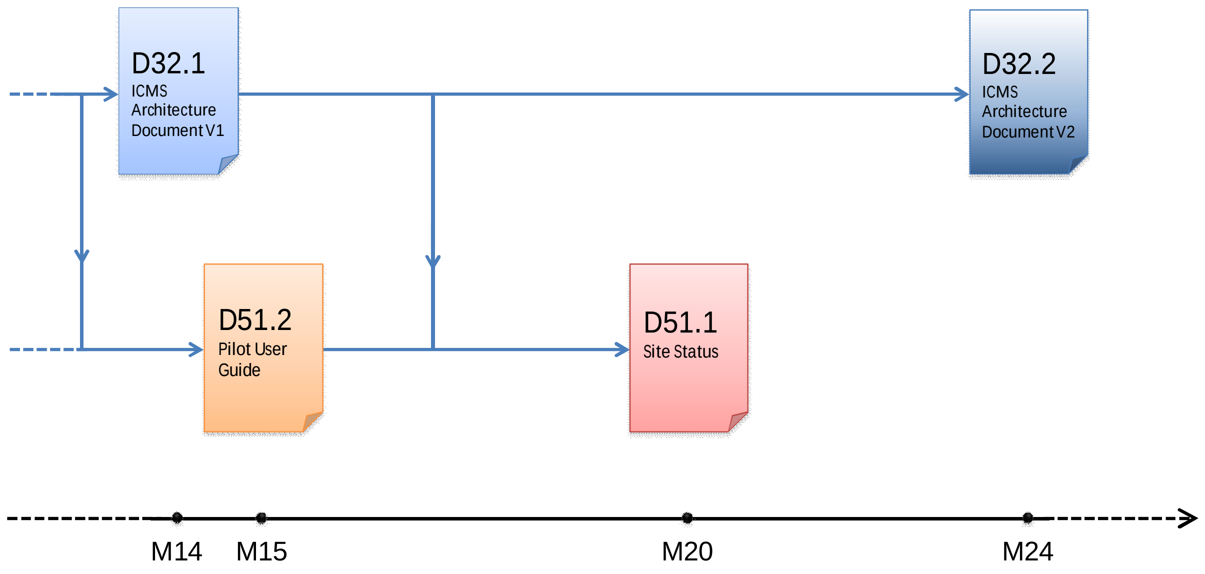


Figure 5: Relation of SP3 and SP5 deliverables.

D51.2 has given a user-oriented overview of major functionalities and datasets which are used in the five pilots. These pilot descriptions together with the template for the specification of Application Architectures (section 5.3) serve as basis for the development of pilot-specific CRISMA Applications with help of the CRISMA Framework provided by SP3 and SP4.

1.5.2. Relations of the architecture specification V2

The second version of the Architecture specification has been largely influenced by the work performed in implementation workpackes WP34 “Implementation” and the early validation of the framework performed in SP5 “CRISMA Pilots”. Figure 6 gives an overview on CRISMA deliverables that provided major input to D32.2 “ICMS Architecture Document V2”.

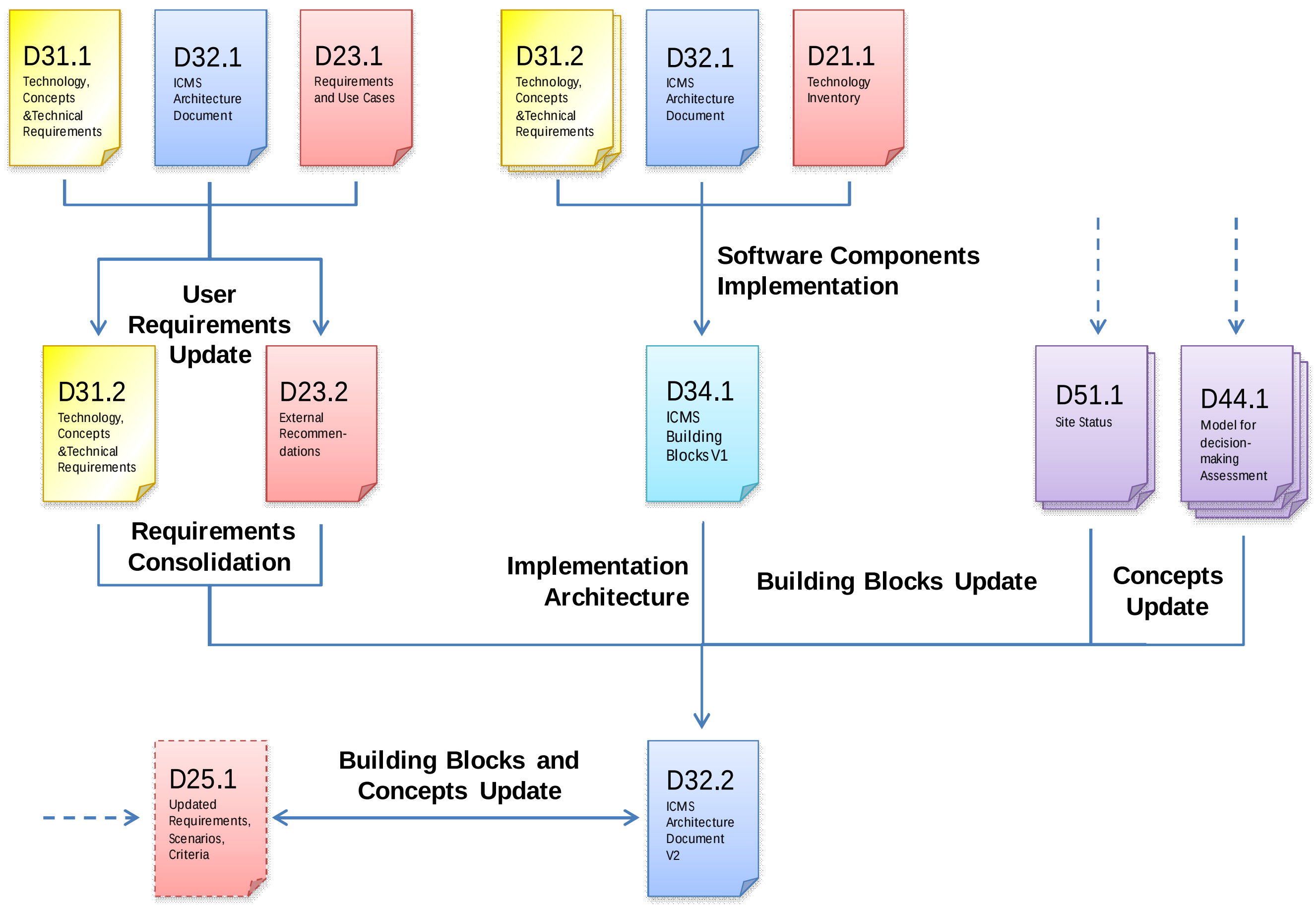

D31.1 Technology, Concepts &Technical Requirements
D32.1 ICMS Architecture Document
D23.1 Requirements and Use Cases
D31.2 Technology, Concepts &Technical Requirements
D32.1 ICMS Architecture Document
D21.1 Technology Inventory
User Requirements Update
Software Components Implementation
D31.2 Technology, Concepts &Technical Requirements
D23.2 External Recommen-dations
D34.1 ICMS Building Blocks V1
D51.1 Site Status
D44.1 Model for decision-making Assessment
Requirements Consolidation
Implementation Architecture
Building Blocks Update
Concepts Update
D25.1 Updated Requirements, Scenarios, Criteria
Building Blocks and Concepts Update
D32.2 ICMS Architecture Document V2

Figure 6: Linking the first and the second SP3 development cycle.

The second technical requirements specification (D31.2, 2013) and valuable recommendations by external experts (D23.2, 2013) provided consolidated functional, non-functional and technological requirements that led, among others, to a major improvement of Functional Building Blocks (chapter 4), an exhaustive definition of technical rules and guidelines (section 5.4) as well as a refinement and more detailed definition of the overall mission of the CRISMA Framework (chapter 2).

As already mentioned, the focus of the second architecture specification has been laid on the Implementation Architecture (Figure 7). The design of the Implementation Architecture was carried out jointly in T3.2.3 “ICMS Implementation Architecture V2” and T3.4.1 “ICMS Building Block Implementation V1. Thus, the Software Components (section 5.1) reported in this deliverable are based on the results of D34.1 “ICMS Building Blocks V1” (D34.1, 2013).

Further input for updating Building Blocks and Software Components was contributed by the report on pilot site status (D51.1, 2013) which provided an exhaustive description of the usage of Building Block by CRISMA Applications that are being developed for the different Pilots.

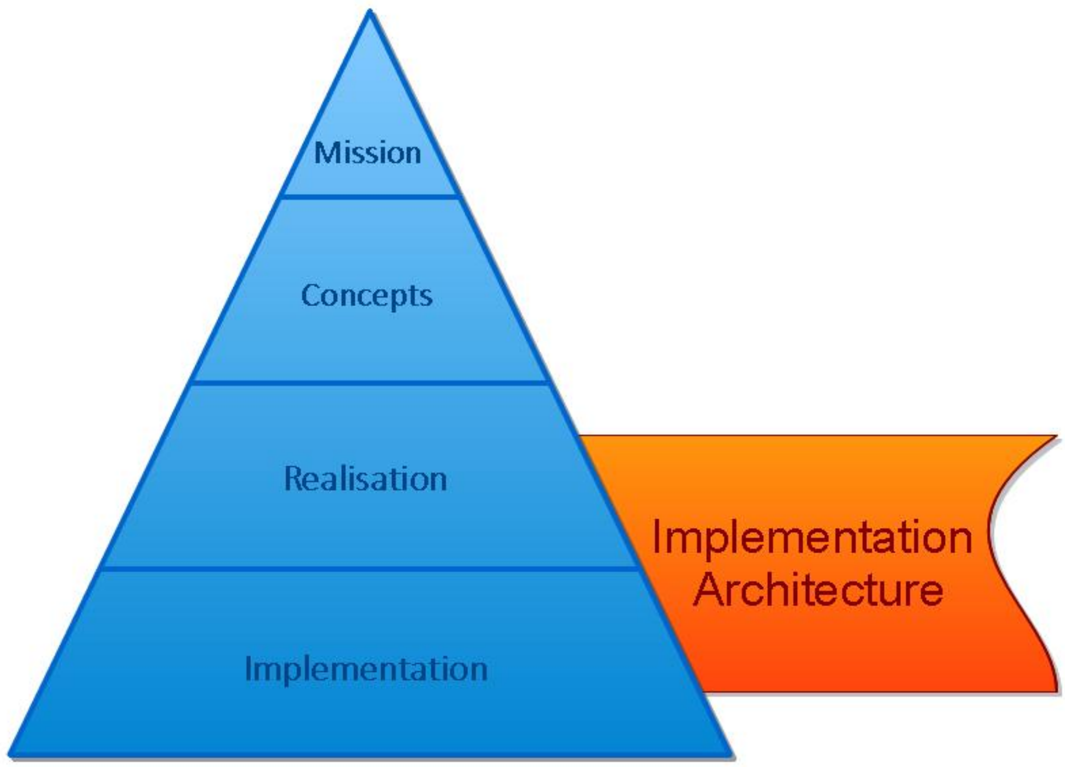


Figure 7: Implementation Architecture of the CRISMA Framework.

Update and refinement of architectural concepts (chapter 3) took also results from SP2 and SP4 into account. Particularly mentionable is D44.1 “Model for decision-making assessment, Economic Impacts and consequences, and simulation” (D44.1, 2013) and the related deliverables addressing Cascade Events D42.1 “Cascade Effects on Crisis-Dependent Space-Time Scales“ (D42.1, 2013) and Time-Dependent Vulnerability D43.1 “Time-Dependent Vulnerability for Systems at Risk” (D43.1, 2013).

D25.1 “Updated technology inventory, requirements, scenarios, criteria and key performance indicators” (D25.1, 2014) which was produced in parallel to D32.2 benefitted from the newly specified Implementation Architecture and the respective Pilot Application Architectures produced in SP5, thereby performing a considerable update of reference scenarios which are of major importance for integration of the CRISMA Framework in D35.1 “ICMS Framework V1” (D35.1, 2014).

Consequently, D32.2 established the link between SP2 “Scenarios, Requirements and Criteria for Crisis Management Modelling”, SP3 “Integrated Crisis Modelling System” and SP4 “Models for Multi-Sectoral Consequences” and thus laid the foundations for the development of a software framework to build custom Crisis Management Simulation Applications (CRISMA Applications) in SP5 “Experimentation and Testing”.

A major step towards the CRISMA Software Framework has been taken by Task 3.5.1 “Framework & Model Integration V1. As shown in Figure 10, the implementation of the CRISMA Framework will be continued in Task 3.4.2 “ICMS Building Block Implementation V2” and Task 3.5.2 “Framework & Model Integration V2” which rely on the results of this deliverable.

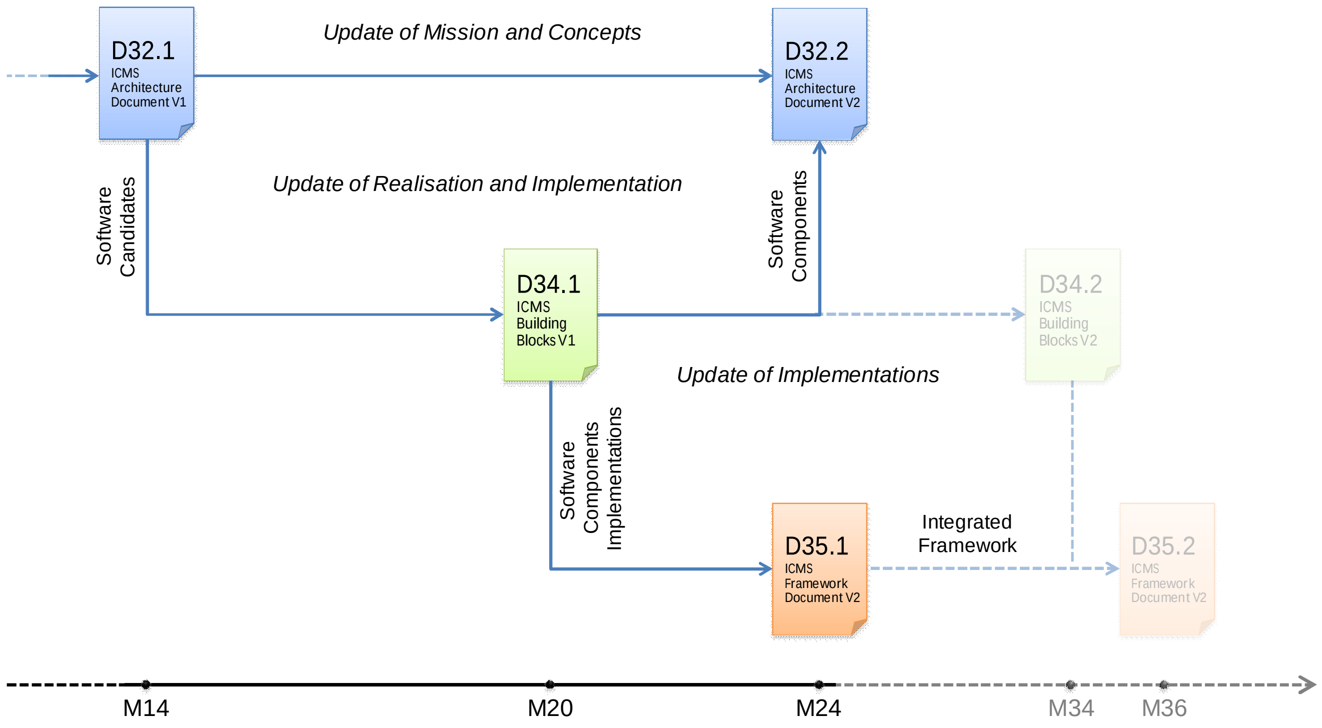

D32.1
ICMS Architecture Document V1
Update of Mission and Concepts
D32.2
ICMS Architecture Document V2
Software Candidates
Update of Realisation and Implementation
Software Components
D34.1
ICMS Building Blocks V1
D34.2
ICMS Building Blocks V2
Update of Implementations
Software Components Implementations
Integrated Framework
D35.1
ICMS Framework Document V2
D35.2
ICMS Framework Document V2
M14
M20
M24
M34
M36

Figure 8: From Architecture to Software.

1.6. *Relation to SP4 and SP5*

While the previous section provided a description of relationships on the level of distinct deliverables and tasks, the goal of this section is to give a more general picture on the relation of SP3 to SP4 and SP5 under consideration of central concepts of the of CRISMA Framework Architecture. This general relationship of the work performed in SP3 to the results and objectives of SP4 "Models for Multi-Sectoral Consequences" and SP5 "Experimentation and Testing" is depicted in Figure 5 and Figure 9 below.

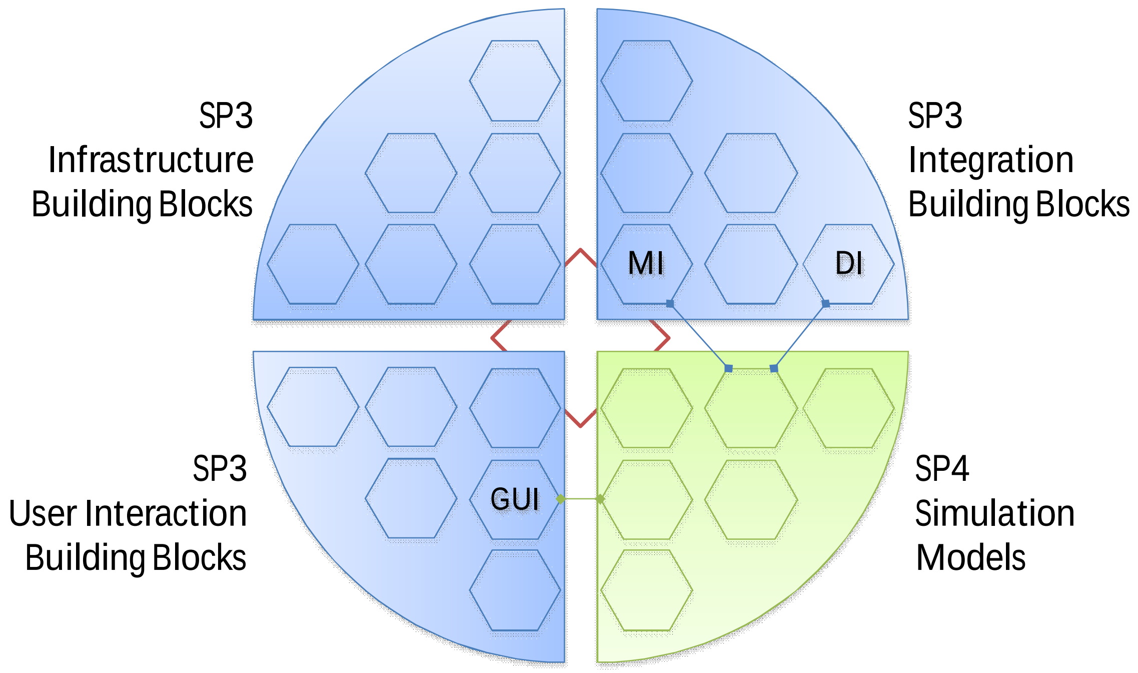


Figure 9: SP3 and SP4 Contributions to the CRISMA Framework.

As shown in Figure 5 SP3 provides three types of Building Blocks that cover the infrastructure, integration and user interaction aspects of the CRISMA Framework. Crisis management specific aspects are contributed by SP4 in form of Simulation Models (Time-Dependent Vulnerability, Cascade Effects, Economic Impact and Decision Support). While SP4 delivers Simulation Models as services in accordance to the Simulation Model Integration (MI) and Data Integration (DI) concepts and Building Blocks of the CRISMA Framework Architecture it does not provide new graphical user interfaces (GUI) for model configuration or model results visualization. Providing user interfaces for the Simulation Models of SP4 which can be seamlessly integrated into a CRISMA Application is therefore the job of SP3 which develops the respective User Interaction Building Blocks.

In SP5, Building Blocks of SP3 and Simulation Models of SP4 are assembled together with pilot specific Simulation Models and Software Components to build distinct CRISMA Applications (Figure 9) – one for each pilot.

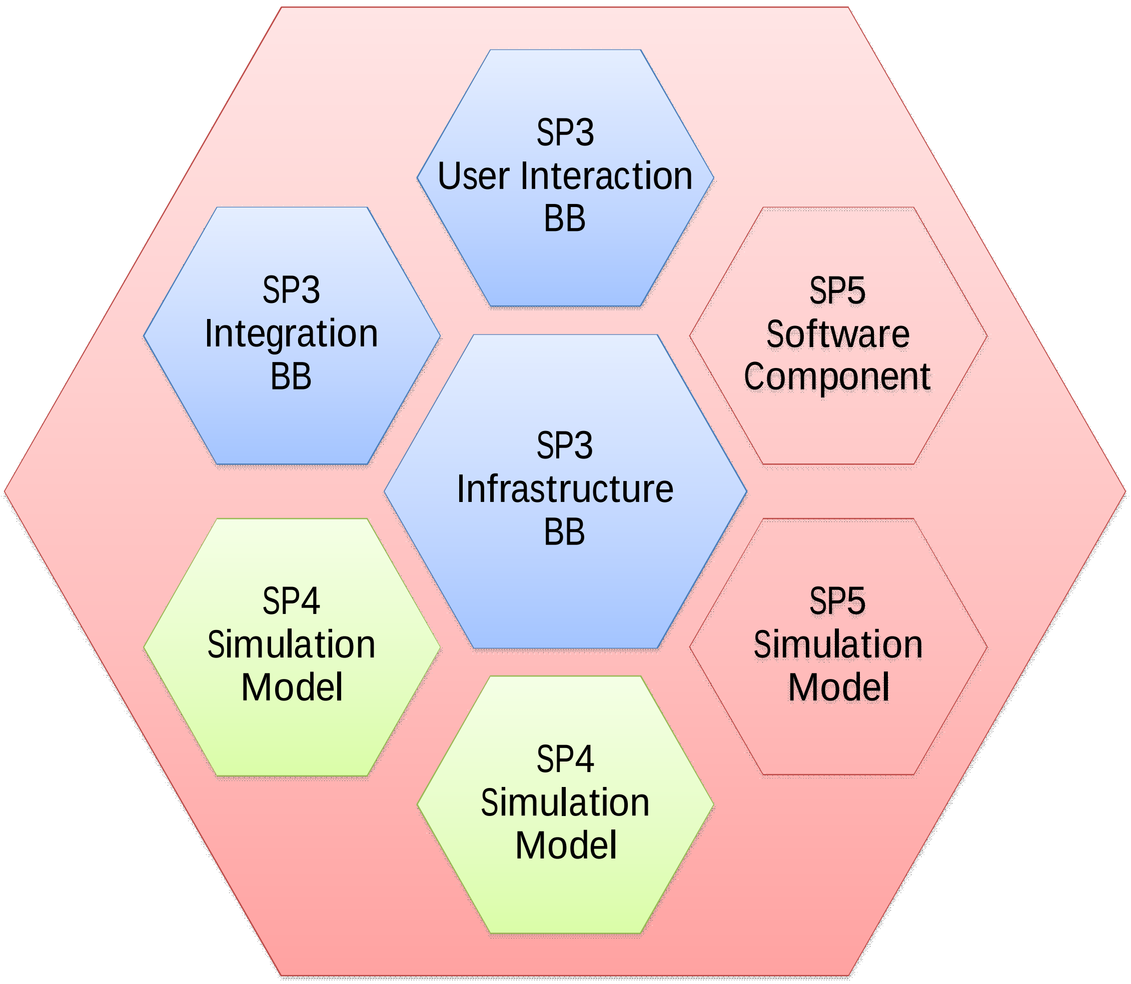


Figure 10: Assembling Software Components of SP3, SP4 and SP5.

1.7. Architectural approach

In the context of the CRISMA project a sound architectural approach for the design and implementation of the CRISMA Framework has been developed. An architectural approach defines a methodology and process of *how* an architecture is designed. Rather than designing and specifying the architecture itself it describes the architectural objectives, the applied concepts, their realisation and also to a certain extent their implementation. The approach presented in this document is based on long-term experience in architecture design, procedures and concepts established in architectures of comparable systems. Thereby, several important aspects have been considered and a thorough assessment of the available options and their implications has been performed.

1.7.1. Preliminary considerations

In detail, the following initial steps towards the definition of a suitable architectural approach for the CRISMA Framework Architecture have been undertaken:

1. Recapitulation of different architectural reference models and frameworks, standards, architectures and projects which seem relevant for the CRISMA Framework Architecture.
2. Considerations of crucial issues that have a severe impact on the architectural process and the resulting CRISMA Framework.
3. Discussion and decision on fundamental design decisions on basis of the previously discussed considerations and the definition and selection of an overall architectural approach.
4. Selection and/or definition of formal procedures, rules and guidelines for the specification and description of components, information models.

At first, a considerable number of projects, architectures and simulation-specific standards that provide interesting architectural concepts which have strong relevance for the CRISMA Framework Architecture and the CRISMA project in general have been assessed. Among the most relevant projects, architectures and standards partially recommended in D21.1 “Technology Inventory for Crisis Management Tools & Models“ (D21.1, 2012) and D31.1 “Technology, Concepts &Technical Requirements Report V1“ (D31.1, 2012) were:

- **IEEE HLA** - High Level Architecture (IEEE, 2010) and HLA Evolved (IEEE, 2011)
- **Open-MI** - Open Modelling Interface Standard (Moore, 2010)
- **TENA** - Test and Training Enabling Architecture (Powell, 2012)
- **DIESIS** - Design of an Interoperable European Federated Simulation network for critical InfraStructures (Beyer, 2010)
- **IMMS/SUMMIT** - Integrated Modelling, Mapping, and Simulation / Standard Unified Modelling and Mapping Integration Toolkit (Plantenga, 2010)
- **RiskScape** - Framework for multi-risk modelling (Schmidt, 2011)
- **OGC SWE** – Open Geospatial Consortium Sensor Web Enablement
- **OMS** - Object Modelling System (David, 2010)
- **RM-ODP** - Reference Model of Open Distributed Processing (ISO/IEC (1998))
- **RM-OA** - Reference Model for the ORCHESTRA Architecture (RM-OA, 2007)
- **SANY-SA** - Specification of the Sensor Service Architecture V3 (SANY-SA, 2009)
- **TOGAF** - The Open Group Architecture Framework (TOGAF, 2011)
- **OASIS SOA-RM** – Service Oriented Architecture Reference Model (SOA-RM, 2006)

Second, based on the assessment of the aforementioned projects, the following important topics and questions to be answered before designing the architecture have been identified:

1. **Architectural Goals**
   What are the principal goals and objectives of the CRISMA Framework Architecture and the CRISMA Framework? What are the most important ones?
2. **Architectural Properties**
   What are the Architectural (non-functional) Requirements of the CRISMA Framework Functional Architecture? How, at which layer or Building Block are they considered and realised?
3. **Architectural Reference Models and Frameworks**
   Do we have to follow an Architectural Reference Model or Framework when specifying CRISMA Framework Architecture? To which degree and level of detail shall the CRISMA Framework Architecture follow the selected Reference Model(s) and Frameworks(s)?
4. **Architectural Style**
   What is the general architectural style of the CRISMA Framework Architecture?
5. **Overall System Design**
   What is the overall System Design? Will there be *one* central Simulation / Management / Training / etc. GUI that covers all aspects of CRISMA?
6. **Supported Crisis Management Phases**
   What are the Crisis Management Phases directly supported by the CRISMA Framework Architecture (by the overall architectural design and/or supporting services and tools)?
7. **Distribution Scope and Communication Network**
   What is the scope of distribution of the CRISMA Framework Architecture and what are the communication networks to be supported? Is there a difference at different layers (tiers) of the architecture?
8. **Communication Models**
   What communication model shall be supported by the CRISMA Framework Architecture at which level?
9. **Communication Infrastructure**
   What is the general communication infrastructure of the CRISMA Framework Architecture, are there different infrastructures at different layers of the architecture?
10. **(Model) Runtime Infrastructure**
    What type of Runtime Infrastructure (RTI) shall be provided by the CRISMA Framework Architecture? Do we support the dynamic composition of workflows?
11. **Levels of Interoperability**
    What levels of interoperability should be supported by the CRISMA Framework Architecture? How and at which layer or Building Blocks is a certain level supported? What is the level of interoperability between different CRISMA Federations (Applications)?
12. **Information Model Approach**
    What is the information model approach to be pursued in the CRISMA Framework Architecture? Will there be a global information model and/or a meta-model that defines modelling rules and commonalities?
13. **Overall GUI Approach**
    What is the overall GUI development approach of the CRISMA Framework Architecture?
14. **Overall Integration Approach**
    What is the primary overall integration approach (Data Integration, Application Integration and/or GUI Integration) of the CRISMA Framework Architecture?

15. **Simulation Model Integration Approach**
    What is the Simulation Model Integration approach that should be followed by the CRISMA Framework Architecture? How and at what layer or Building Blocks is a specific level supported?
16. **Data Integration Approach**
    Does the CRISMA Framework Architecture support the integration of heterogeneous data sources and if so how? What are the Building Blocks and Tools that support the integration of data sources?
17. **Application Integration Approach**
    What is the CRISMA Framework Architecture's approach towards the integration of legacy applications?
18. **GUI Integration Approach**
    What is the general strategy for GUI integration of the CRISMA Framework Architecture? Does the CRISMA Framework Architecture also support the integration of autonomous legacy GUIs and if so how? What are the Building Blocks and Tools that support the integration of GUIs?

The topics and the different approaches concepts and options to choose from have been thoroughly elaborated, analysed and compared. Finally, architectural design decisions have been taken collaboratively. The complete set of design decision along with a rationale referring to project objectives is given in Appendix A of the initial Architecture specification (D32.1, 2013) and the document on the updated technology inventory, requirements, scenarios, criteria and key performance indicators (D25.1, 2014), respectively.

The larger part of the above topics is further considered in conceptual description of the CRISMA Framework Architecture (chapter 3). The first four topics Architectural Goals, Architectural Properties (non-functional requirements), Architectural Reference Models and Frameworks and Architectural Style mainly define the architectural design approach as such and the overall mission of the CRISMA Framework Architecture (Chapter 2).

1.7.2. Approach and methodology

The architectural design approach and the chosen methodology take several concepts from the previously assessed projects, architectures and simulation-specific standards into account while they do not directly adopt a certain architecture or reference model. Although the CRISMA Framework Architecture can be considered a reference architecture as explained in more detail in section 3.2 it does not strictly follow a specific architectural reference model like RM-ODP, RM-OA, SOA-RM or TOGAF. Specifying architectures in such a formal and strict manner may lead to a clear and structured design. However, the considerable effort needed to produce formal specifications and follow extensive meta-models and rules does not per see guarantee interoperability between Integrated Systems. Nevertheless, we adopt certain common concepts that are helpful for a structured architectural design process. Those concepts are:

- We apply the four-layered MCRI scheme introduced in the DIESES project (Rome, 2009) to structure the specification of the CRISMA Framework Architecture.
- We identify and describe the overall goals of the architecture and its main properties and constraints in chapter 2.
- We develop and explain the key concepts of the architecture in chapter 3 in relation to its goals.

- We separate the architecture into an implementation dependent (Implementation Architecture and Application Architectures) and an implementation independent (Functional Architecture) part as in chapters 4 and 5 in relation to concepts introduced in the Reference Model for the ORCHESTRA Architecture (RM-OA, 2007).
- We specify the architecture as a framework that provides the tools, Building Blocks and guidelines to create a concrete Application Architecture.
- We provide templates for
  - the abstract description of functional Building Blocks,
  - the selection of software candidates for the realisation of functional Building Blocks, and
  - the (draft) specification of Application Architectures
- We adopt the concept of Service Platform from SOA-RM (SOA-RM, 2006) that specifies basic properties of a SOA like message format, schema language, etc. and apply it to the specification of the Implementation Architecture.
- We identify Functional Building Blocks and apply a stepwise hierarchical and functional decomposition into a set of components (e.g. tools, services) with their interfaces and interaction patterns.
- We identify software candidates for the implementation of the Functional Building Blocks and provide a mapping to the Functional Building Blocks.
- We select and define methodologies and rules for data, model and application integration and select, define or specify the respective tools and standards.

The Architecture specification is structured along four main viewpoints according to the MCRI pyramid (see Figure 11) which stands for mission, concepts, realisation and implementation. The mission viewpoint (chapter 2) defines the primary goals of the CRISMA Framework and documents how the architectural goals, properties and constraints have been derived from project objectives and user requirements. The concepts viewpoint (chapter 3) defines the key concepts that are applied to realise the goals. They encompass general architectural concepts like reference architecture, Building Blocks, Federations, several integration concepts and user oriented concepts for resource management, training and so on. The realisation viewpoint (chapter 4) addresses the realisation of the concepts by a set of functional Building Blocks and their interaction patterns. Functional Building Blocks provide an abstract description of the general functionalities to be provided by the CRISMA Framework. A further decomposition of functional Building Blocks into concrete Software Components is performed in the implementation part (chapter 5) of the architecture.

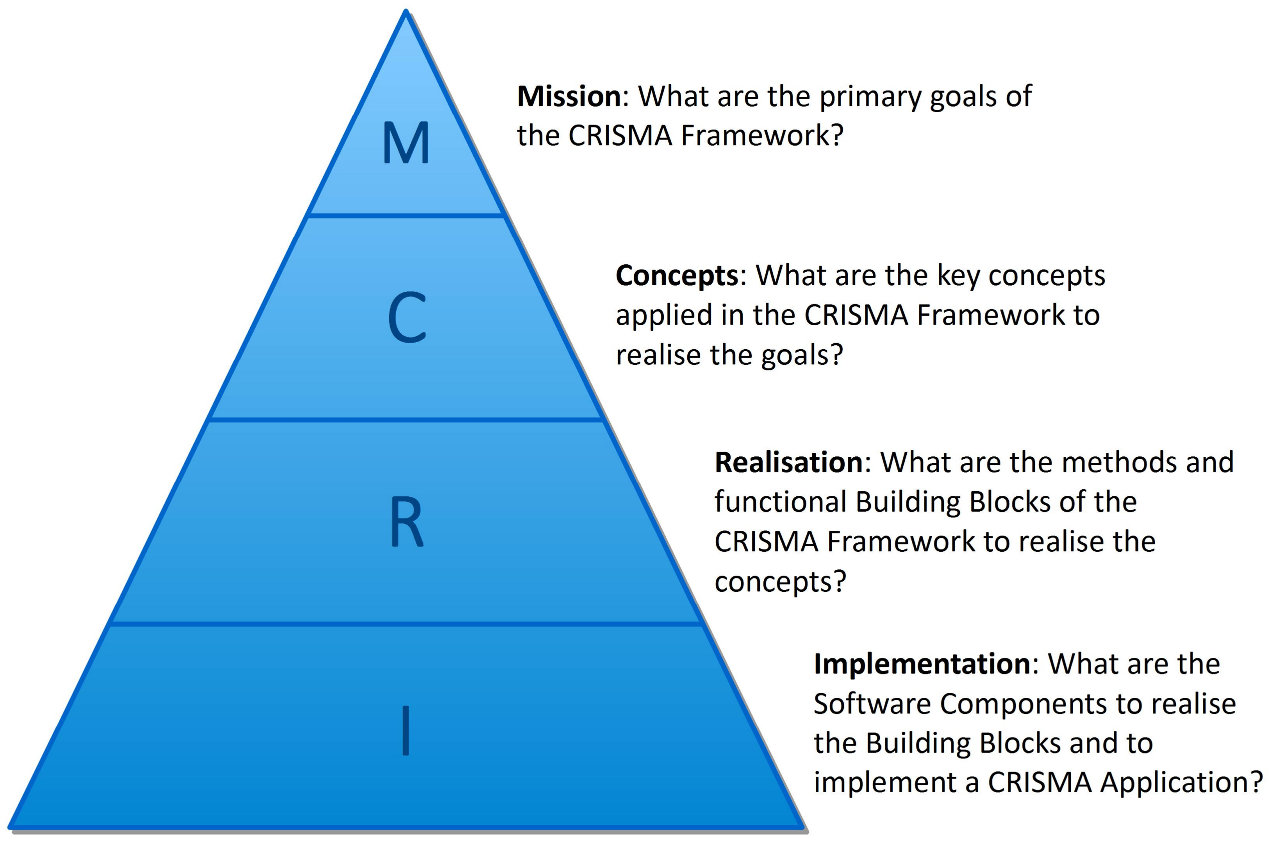

M
C
R
I
Mission: What are the primary goals of the CRISMA Framework?
Concepts: What are the key concepts applied in the CRISMA Framework to realise the goals?
Realisation: What are the methods and functional Building Blocks of the CRISMA Framework to realise the concepts?
Implementation: What are the Software Components to realise the Building Blocks and to implement a CRISMA Application?

Figure 11: MCRI pyramid.

1.7.3. Functional and implementation architecture

As explained in more detail in sections 2.2 and 3.2 the overall architecture of the CRISMA Framework consists of an implementation independent part which we call Functional Architecture complemented by an Implementation Architecture. Using the words of the MRCI pyramid (Figure 11) the Functional Architecture covers the mission, concepts and realisation viewpoints while the Implementation Architecture addresses the implementation viewpoint. Implementation independence in the context of the Functional Architecture does not mean that the Architecture specification will be independent of technologies and standards. In contrary, the selection of appropriate technologies and standards is integral part of the concepts and realisation viewpoints. Moreover, the realisation viewpoint describes on abstract level how and by what Functional Building Block the expected functionality of the overall CRISMA System is realised rather than how and by which specific Software Component as this is the purpose of the Implementation Architecture which can be considered a realisation of the Functional Architecture.

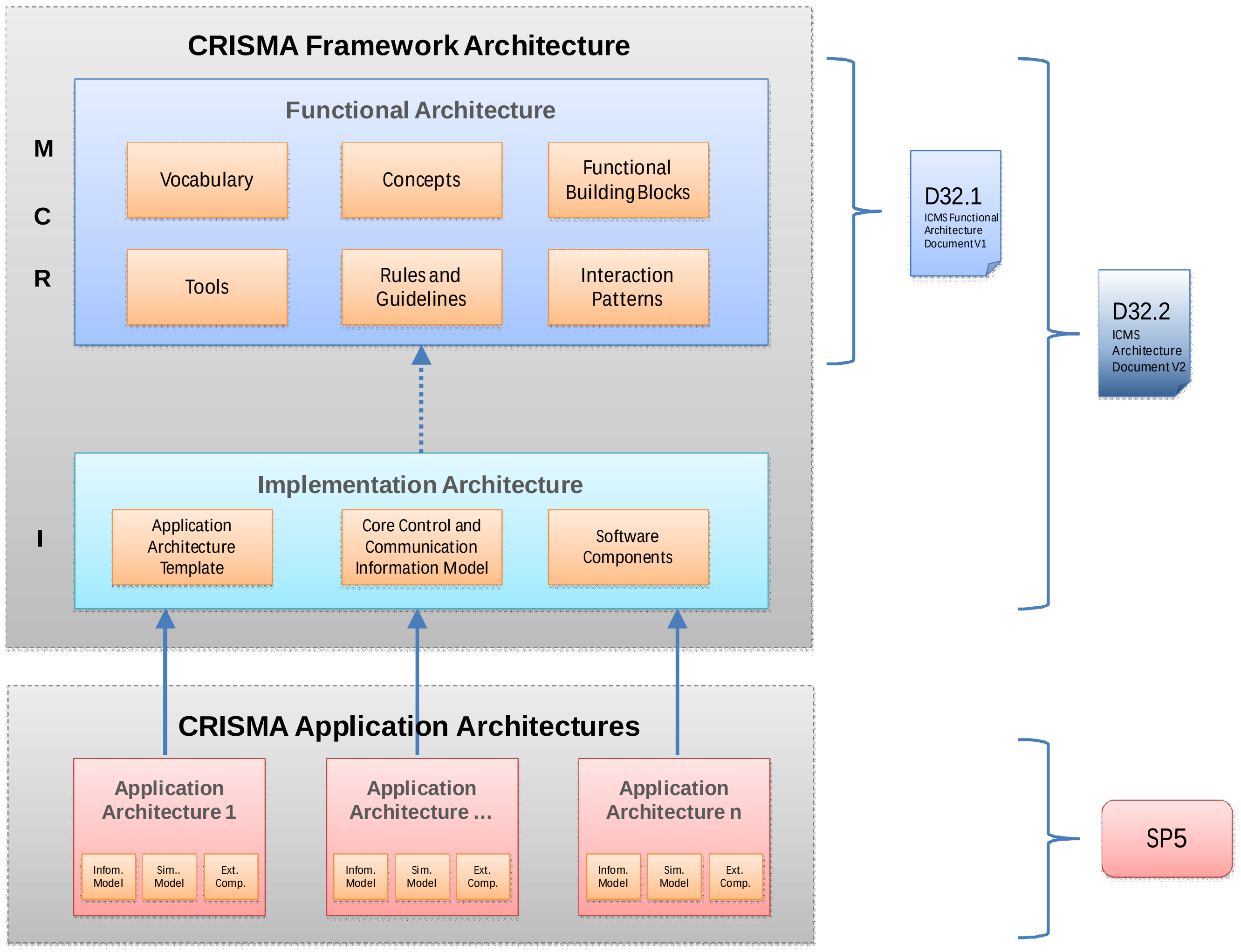

CRISMA Framework Architecture
M
C
R
Functional Architecture
Vocabulary
Concepts
Functional Building Blocks
Tools
Rules and Guidelines
Interaction Patterns
I
Implementation Architecture
Application Architecture Template
Core Control and Communication Information Model
Software Components
CRISMA Application Architectures
Application Architecture 1
Application Architecture ...
Application Architecture n
Infom. Model
Sim.. Model
Ext. Comp.
D32.1 ICMS Functional Architecture Document V1
D32.2 ICMS Architecture Document V2
SP5

Figure 12: CRISMA Framework Architecture.

As shown in Figure 12 the first version of the Architecture specification (D32.1, 2013) focused on the Functional Architecture, whereby some general aspects of the Implementation Architecture were already addressed. The complete picture is given in the current version of the Architecture specification by augmenting it with the complete Implementation Architecture (chapter 5). Thereby, it is important to highlight that the Implementation Architecture is a domain independent transferable specification which is the basis for the implementation of the CRISMA Framework (WP34 “ICMS Framework Building Block Implementation” and WP35 “ICMS Framework Integration”)

An additional architectural concept that is shown in Figure 12 is the concept of the Application Architecture. An Application Architecture is a specialisation of the common Implementation Architecture and describes the architecture of a concrete CRISMA Application (e.g. a pilot application), thus adding domain and location context. Among others, it specifies the extensions of the core information models, what Software Components are used in the context of the application, what models, data and systems need to be integrated, which types of World State Transitions are performed by the application and so on. Anticipating the concept of Federations introduced in section 3.3 we can already state that an Application Architecture is the architecture of a CRISMA Federation.

2. Mission

This chapter defines the general mission of the CRISMA Framework Architecture including its purpose and scope. As shown in Figure 1, this mission is expressed in terms of architectural goals, properties and constraints which have been derived from the project objectives and pilot expectations, research issues and technical- and user requirements. Thereby, this chapter addresses primarily common architectural concerns rather than concrete system functionalities. Detailed functionalities of the CRISMA Framework are described by means of Functional Building Blocks that are specified in chapter 4 – Realisation.

2.1. Preliminary considerations

At first, we give a brief overview on numerous objectives and requirements collected from different sources before we assess the consequences for the architectural design process described in section 1.7 and provide an exhaustive description of the overall mission of the architecture.

According to the DoW (DoW, 2013) and D63.1 "Report on market analysis" (D63.1, 2013) the primary objectives of the Integrated Crisis Management System (ICMS) to be developed by the CRISMA project are:

- To model possible **multi-sectoral crisis scenarios** and assess the consequences of an incident,
- to **simulate possible impacts** resulting from **alternative actions**,
- to support **strategic decisions** on capabilities, related investments, reserves and inventories,
- to **optimize the deployment of resources** dedicated to crisis response in-line with the evolvement of a crisis, and
- to **improve action plans** for the preparedness and response phases of the crisis management

Although these objectives already provide a comprehensive overview on the mission of the ICMS and the CRISMA project a as whole, they don’t define the general architectural design of the perceived system. Nevertheless, they serve as a good starting point towards the identification of more detailed system functionalities which was performed during the three workshops in the first four months of the CRISMA project. The results of those workshops are documented in D11.1 “Consolidation Report” (D11.1, 2012). The most valuable outcome of this report regarding the architectural design process are the pilots’ general targets and expected outputs related to CRISMA objectives as well as several critical challenges that can be considered an initial set of non-functional requirements for the architecture.

While the majority of those pilot expectations refer to system functionalities and thus are considered user requirements which have been further developed as goal models, behavioural scenarios and use cases in D23.1 “(User) Requirements and Use Cases“ (D23.1, 2013) the expectations relevant for the architecture in general are:

- The CRISMA System should also support field training, with real deployment of people and resources and thus provide ”mixed reality” **applications allowing ”real” and ”virtual” field workers to seamlessly interact** (**life, virtual and constructive**).

- The CRISMA System should be **adaptable to the requirements of different categories of end users**, with different operating modules related to Crisis Management activities (e.g. planning, training, capacity and resource allocation, real time operational management, cost benefit analyses, etc.).
- The CRISMA System needs both an unsecured feature that is **accessible through the web**, and a **secured component**.
- The CRISMA System needs to be able to **combine data generated by different risk assessment models**, such as an earthquake causing a chemical leak with toxic plume dispersion and fire.
- The CRISMA System should function as a **standalone as well as distributed system**.
- The CRISMA System should support the **transfer of scenarios, assumptions and experiences from one region to anothe**r using an abstraction level named Reference Scenarios.
- The CRISMA System might also support **real time communication** with field workers (emergency capacities).
- The CRISMA System should support the **co-existence of generic models** with high uncertainties and models which are fine-tuned to a specific spatiotemporal area.

From those expectations we can already derive some architectural goals like the objective to provide a middleware for real-time information exchange and key architectural properties like extensibility, flexibility and interoperability.

Even though not yet focused on the provision of a Framework for building Crisis Management Simulation systems, D11.1 “Consolidation Report” (D11.1, 2012) gives a brief outlook on the architecture and its primary objectives. Together with the general project objectives of the DoW (DoW, 2013) and the objectives of several other crisis management projects and systems assessed in D21.1 “Technology Inventory for Crisis Management Tools & Models“ (D21.1, 2012) those initial architectural objectives were further detailed in D31.1 “Technology, Concepts & Technical Requirements Report V1“ (D31.1, 2012) and can be summarised as follows:

Technically, the architecture should be flexible enough to:

- Support **interaction with existing** decision support and crisis management **systems,**
- allow for **de-centralised development** and management, and **easy integration** of new features,
- assure **timely information flow** to decision makers and relief forces,
- allow for **dynamic exchange of services** providing the **real time data** (e.g. from sensors used in real crisis situations) with services providing faked/modelled data (used in planning and for the training purpose),
- provide data models and services required for crisis management applications that are capable of **integrating the existing legacy systems** (data, models) used in crisis management today,
- be **compatible with on-going developments** in the field of environmental monitoring and environmental information management (e.g. **SISE, INSPIRE, GMES**),
- assure **controlled and secure access** to all data and services (authentication, authorisation, data consistency),

- provide a **service infrastructure** as a basis for the **harmonized**, **standardized** and **user friendly integration** of **new and existing Simulation Models,**
- support the integration of **sensors** as well **as data sources of various types,**
- take into account **electricity and communication infrastructure failures** at the level of the architecture (in case someone wants to adapt the CRISMA Framework for crisis management and mitigation). However, we can safely assume that these issues do not occur in the planning and training phases (**reliability** requirement),
- support **views depending on the organization and the user's operative role** (user management requirement),
- support the **communication with existing IT tools** for crisis management to address decision making alternatives, perform what-if scenarios**, update system information with real time data**, derive quantitative and qualitative data from existing models and **introduce new features to existing simulation tools** if required,
- consider mechanisms for near-real-time exchange of the information about field workers and other features of interest in the real world, as well as those in the virtual world (mixing life, virtual and constructive elements),
- exhibit the **right level of genericity** to be able to support the very different pilot applications as well as applications beyond in the crisis management context.

In addition to those objectives which help us to formulate the primary architectural goals we consider also some of the research issues and common problems defining and implementing simulation systems identified in D31.1 “Technology, Concepts &Technical Requirements Report V1“ (D31.1, 2012):

- **Composability and interoperability** of crisis management models and simulations coupled for some particular use
- **Reliability** of communication infrastructure
- **Adaptability** and scalability of environmental models
- Granularity of models
- **Scalability** of the ICMS system
- Problems with hybrid models
- Location intelligence of tools and models
- **Uncertainty** issues
- Concerns about **integrating GIS**
- **Evolution of crisis scenarios**
- Conceptual modelling and crisis management technologies
- **High-level interoperability**
- Appropriate level of interoperability
- **Reuse of existing models and simulations**
- **Data issues**
- **Adjustability** of environmental models

Most of those issues relate mainly to Simulation Model integration and Simulation Model interoperability; a major topic of the CRISMA project which has to be addressed by the CRISMA Framework Architecture in by a dedicated subproject: SP4 “Models for Multi-Sectoral Consequences”.

Furthermore, D21.1 “Technology Inventory for Crisis Management Tools & Models“ (D21.1, 2012) poses a valuable source of user requirements which serve mainly as source of functional requirements for the Functional Building Blocks (section 4).Nevertheless, we can extract also some requirements which are highly relevant for the architecture design itself:

- The system should **use common European standards** (e.g. **OGC** compliant) so that impact/damage models running in remote servers can exchange data with CRISMA.
- The system should be able to develop impact scenarios from **external tools** (like hazard models – e.g. ALOHA – Areal Locations of Hazardous Atmospheres for plume dispersal).
- The system should be able to combine data generated by different risk assessment models (e.g. an earthquake cause a chemical leak with toxic plume dispersal and fire).
- The system should provide a "none secured" feature that is accessible through the web, and a **secured component** accessible using login and password parameters (or more advanced security systems).
- The system should allow to **integrate local maps** and realistic connection times on the transport network (e.g. between hospitals and places of accidents).
- The system should **constantly keep track** of the search and rescue operations.

Another valuable source of non-functional requirements and architectural properties, respectively, which are relevant for the CRISMA Framework Architecture, is the SP2 deliverable D23.1 “Requirements and Use Cases” (D23.1, 2013). The consolidated crisis management goal models therein define besides the behavioural (functional) aspects of the overall CRISMA System (the entirety of all individual crisis management applications provided by the CRISMA project) also so called quality goals (shown in Figure 13). The quality goals shown in Figure 13 have been translated into architectural properties (section 2.3). Further architectural properties could be derived from the user requirements specification tables (APPENDIX C of D23.1).

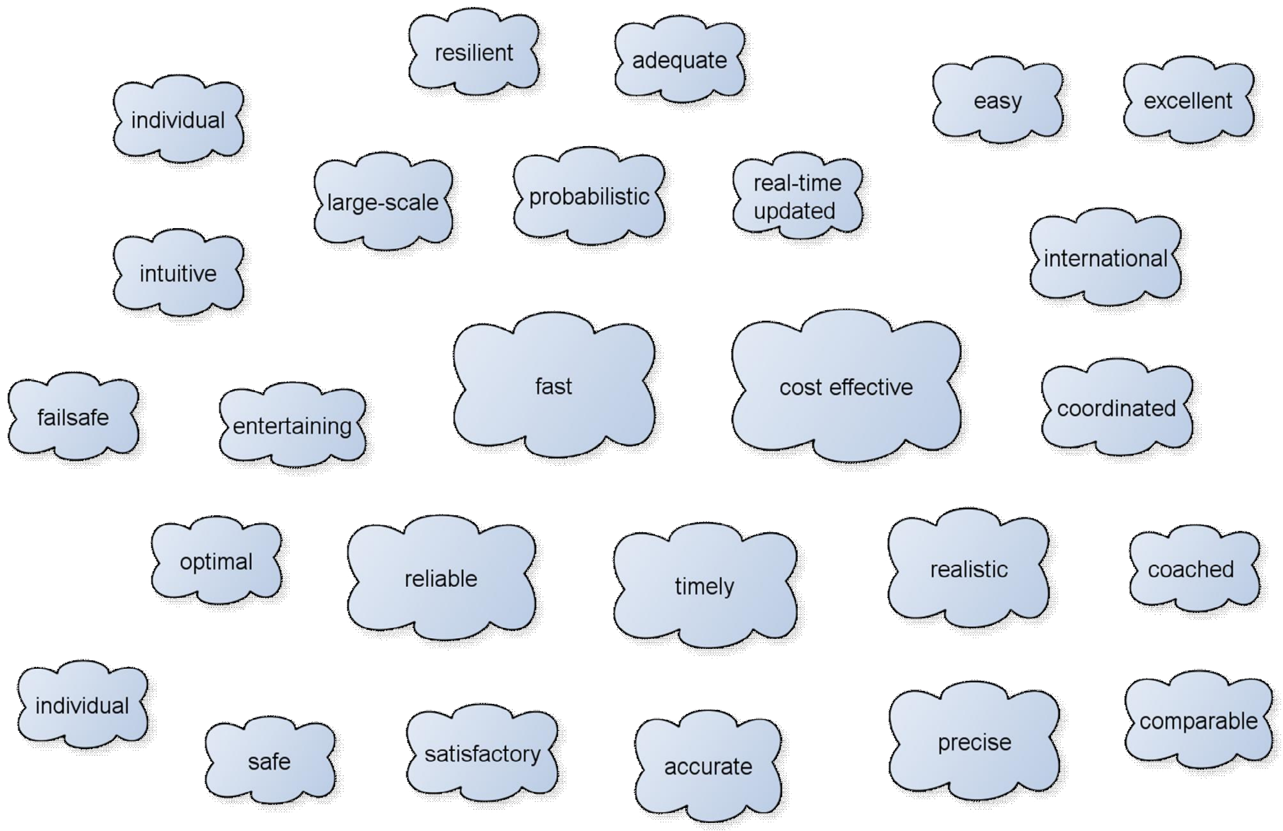


Figure 13: Quality Goals of the consolidated Crisis Management Goal Model.

Last but not least, a thorough assessment of related research projects and comparable simulation management systems like SUMMIT (Plantenga, 2010), DIESIS (Beyer, 2010), TENA (Powell, 2012), HLA (IEEE, 2010) and others severed as input for the formulation of the architectural goals and the identification of important architectural properties.

Figure 14 gives an overview on the main sources of architectural goals and properties. Since the extraction and processing of those non-functional requirements of the CRISMA Framework Architecture is mainly the result of “Task 3.1.2: ICMS Technical Requirements” they are also documented in condensed form in D31.2 “Technology, Concepts &Technical Requirements Report V2” (D31.2, 2013).

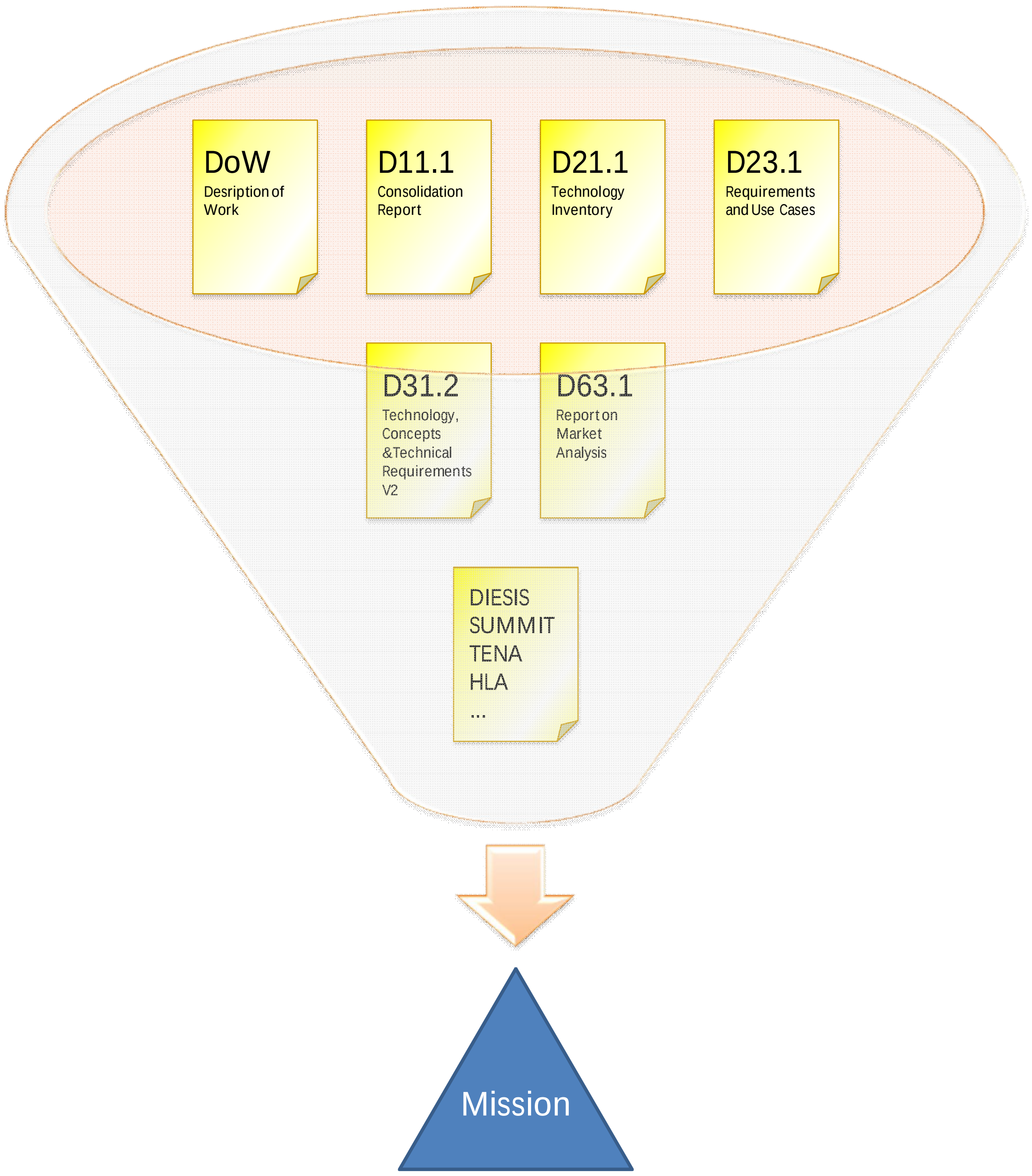


Figure 14: Sources of Architectural Goals and Properties.

2.2. Architectural goals

From the previous discussions we can conclude that the main challenge of the CRISMA Framework Architecture lies within the broad field of system integration. This encompasses model integration, data integration, application integration as well as GUI integration. Thus we can formulate one key objective of the CRISMA Framework Architecture as

> **The CRISMA Framework Architecture has to support integration with existing systems, existing tools, existing models and existing data sources at multiple levels of interoperability.**

Unlike in other projects we have assessed the systems, models, tools and data sources to be integrated are not yet known. This means, that the architecture must be flexible enough to support integration at many levels. Therefore, also the usual integration approach to agree on *one* common (global) data model and *one* set of common interfaces is not feasible for the CRISMA Framework Architecture. Furthermore, the five different pilots will each rely partially on different existing models, tools, etc. Thus, another major conclusion is that there won't be *one* CRISMA System as initially perceived by the idea of a single ICMS. Instead, a framework has to be provided which allows building a CRISIS Management Application around and/or on top of legacy systems. Conceptually, this architectural objective can be formulated as

> **The CRISMA Framework has the characteristics of a reference architecture framework, that is, a template and framework to build concrete Application Architectures of the crisis management domain.**

As a consequence and as already anticipated by the work plan (DoW, 2013) the CRISMA Framework Architecture is split into an implementation independent and an implementation dependent part. While the Functional Architecture is concerned with the general concepts, relationships and Functional Building Blocks within the crisis management domain covered by the CRISMA project the Implementation Architecture provides a concrete selection of Software Components and information models. From this point of view, we can formulate the following goals for the Functional and the Implementation architecture:

> **The Functional Architecture has to establish a common vocabulary which describes the core concepts and the relationships between the elements of the architecture.**

Such a vocabulary defines scope and context of the different architectural elements and thus enables architects and developers to gain a common and harmonised view on the architecture. In the Architecture specification at hand the vocabulary is represented by the glossary as well as the by the concepts described in chapter 3.

> **The Functional Architecture has to define a methodology for describing a CRISMA Federation in terms of a set of interacting Building Blocks. It has to provide the respective tools as well as the functional specification of the Building Blocks.**

The Building Blocks are the core elements of the architecture and encapsulate the functionality that is provided by the CRISMA Framework. Besides the Functional Building Blocks the architecture has also to provide templates for their description as well as an explanation of their interactions and their relation to other architectural elements like Federates (see section 3.8) and Integrated Components and systems (see section 3.10.3).

> **The Functional Architecture has to ensure composability and reusability of Building Blocks especially in the context of integrating them with legacy systems.**

An important aspect of the architecture is that not only existing Software Components or complete legacy systems may be wrapped into CRISMA Building Blocks or CRISMA-aware components (see section 3.7) but also the Building Blocks provided by the CRISMA Framework should provide the possibility to be integrated into legacy systems. Thus, composability and reusability are key requirements to be fulfilled by Building Blocks.

Furthermore, the architecture has to employ suitable methods to ensure that those requirements can be met by Software Components implementing the Functional Building Blocks.

The Functional Architecture has to select consistent standards and specifications for the exchange of data and Control and Communication Information between Federates and Integrated Components.

Formal standards, data formats and agreements are one of the main concepts to enforce interoperability between different systems and components. The architecture has to pre-select and define such standards.

The Functional Architecture has to define a common communication protocol as well as a common language used for communication between all Federates.

Although the CRISMA Framework has to support many different components and systems of which each may use its own standards and formats and communication protocol, one common denominator has to be found which leverages a unified integration.

The Functional Architecture has to define a common infrastructure and a middleware for (meta-) information exchange and integration of heterogeneous data, models and systems.

On basis of the common communication protocol and language a Middleware Infrastructure has to be specified which connects all internal (E.g. Building Blocks) and external (e.g. Integrated Systems) components of a CRISMA Federation.

The Functional Architecture has to define an open API for the Middleware Infrastructure.

Well defined service interfaces toward the Middleware Infrastructure realised as open and easy to use APIs have to be provided by the architecture in order to leverage the development and integration of Software Components.

The Functional Architecture has to define the concepts for the specification of modular information models for Control and Communication Information exchanged with the Middleware Infrastructure.

While different CRISMA Applications may rely on different (Control and Communication) information models, there is the need for a minimal common information model that specifies the shared concepts on the level of the CRISMA Framework. The architecture has to establish concepts that allow the definition of modular and application specific extensions of the global information model.

A global data model approach when applied to the whole set of CRISMA Applications is unfeasible since existing data has to be transformed to comply with the global model. Instead a minimum global CCIM has to be designed by the Implementation Architecture which can be extended by application-specific CCIMs.

The Implementation Architecture has to define common, minimal, generic and modular Control and Communication Information Models (Core CCIM) that can be extended by each individual Application Architecture.

One of the main goals of the Implementation Architecture is to identify and assess Software Components that are suitable for the implementation of the Functional Building Blocks. Thereby, a strong focus lies on extensibility and reusability of those Software Components to be adapted to CRISMA needs.

> **The Implementation Architecture has to select and/or specify a set of composable and reusable Software Components that can be utilized to implement the Functional Building Blocks.**

Besides those architectural goals listed above there are also other more specific architectural requirements which we consider rather high level functional requirements than general architectural goals. Among those high level functional requirements are e.g. requirements to provide an event data management system, to provide repositories for storing different types of information entities, to provide management tools for administration and integration, and many more. Those high level functional requirements are further decomposed in the specification of the respective Building Blocks (section 4).

2.3. Architectural properties

High-Level architectural requirements sometimes also referred to as architectural properties or architectural principles (RM-OA, 2007), (SANY-SA, 2009) are non-functional requirements with high relevance for the architectural design process and the resulting architecture. Some of those requirements are specific to CRISMA and the Crisis Management Domain while others are common non-functional requirements of distributed integration architectures. The non-functional requirements influence the overall architectural design while the functional requirements drive the specification and development of Building Blocks.

The architectural properties of the CRISMA Framework Architecture and their relevance and applicability are explained in the following:

Clean and structured system design
The CRISMA Framework Architecture shall be based upon well adopted and commonly used architectural design principles. The architecture has to provide tools and templates that support the structured specification and documentation of individual Application Architectures.

Use of concepts and standards
The architecture shall make use of proven concepts and standards in order to decrease dependency on vendor-specific solutions and help ensure the openness of the CRISMA Framework and support the evolutionary development process of the architecture.

Interoperability
The architecture shall employ mechanisms and guidelines to enforce interoperability between independently developed internal (CRISMA Federates) and external (Integrated Components) of a CRISMA Application.

Loosely coupled components
The Building Blocks involved in the CRISMA Framework shall be loosely coupled, where loose coupling implies the use of mediation to permit existing components to be interconnected without changes.

Extensibility and flexibility
The CRISMA Framework Architecture shall not be a “closed” system with a fixed set of functionalities. It must be possible to easily integrate new Simulation Models, Building Blocks or other types of components with additional functionality into the CRISMA Framework or an Application.

Security and confidentiality
The CRISMA Framework Architecture shall be designed to allow state of the art security mechanisms to be incorporated. These mechanisms shall include user management (authentication, authorisation), as well as control of access to data, services and tools.

The CRISMA Framework Architecture shall also consider mechanisms to restrict sharing of confidential information across organisational boundaries, e.g. to support CRISMA Applications that cannot rely on the transmission of sensitive or confidential information over unsecured public networks like the internet.

Performance and scalability
The CRISMA Framework Architecture shall be able to take scalability and availability issues (throughput, response time) into account, e.g. to handle various levels of operational load and external conditions.

Reliability, dependability, robustness and safety
The CRISMA Framework Architecture shall support the development of CRISMA Applications that are resilient to security threats or breakdowns.

Ubiquity of access
The CRISMA Framework Architecture shall support mechanisms to realise ubiquitous access to its functionalities, which is access regardless of location or device. Functionality might be limited because of technical reasons (bandwidth, screen size) or security considerations. While the architecture shall enable remote access it has to be considered what functionality is actually needed and if a specialized user interface is required (e.g. mobile phones used for training).

Supportability, adaptability and maintainability
Since every pilot and later every user will have special needs there won’t be one CRISMA System just to install but each CRISMA Application will need adaption to user’s needs. So the CRISMA architecture shall enable easy adaption and maintaining to enable support of many different Applications with reasonable efforts.

User-friendliness and usability
The architecture shall provide mechanisms and guidelines that leverage the development of user-friendly graphical user interfaces that are adaptable to the workflows and needs of different types of users.

2.4. Architectural constraints

As already observed in D31.1 “Technology, Concepts &Technical Requirements Report V1“ (D31.1, 2012), user requirements for developing the CRISMA Framework cannot be unconditionally met because of availability and suitability of appropriate modelling and simulation tools and technologies. This applies also to the CRISMA Framework Architecture in general. Therefore, a good compromise between the architectural goals and properties and the feasibility and applicability of different architectural approaches and their influence on the implementation possibilities had to be reached. This is reflected in the descriptions of the major concepts (chapter 3) as well as in the description of the functional Building Blocks (chapter 4) of the architecture. Furthermore, it is documented in a set of architectural design decisions (D32.1, 2013) which provide also a rationale regarding architectural goals and constraints.

Focus on preparation phase

The CRISMA Framework is based upon a robust and generic architecture that supports the development of Crisis Management Applications capable of modelling and simulating all phases of a real crisis. The CRISMA Framework will therefore provide Building Blocks to support the simulation of actions related to all crisis management phases. Although the architecture has been designed having properties like reliability, fault tolerance and dependability in mind, the consideration of crisis-induced issues like power outages and breakdown of communication infrastructure are beyond the scope of the project.

Restricted automatic GUI creation

While most of the GUIs exposed by the Building Blocks can either be automatically generated and adapted to the use cases and information models of an individual CRISMA Application, the CRISMA Framework does not provide generic GUIs for the configuration of arbitrary complex models nor is it feasible to develop specific GUIs for the configuration of models having very complex input files. This issue is further discussed in section 3.12.

Implementation effort for Simulation Model integration

The CRISMA Framework Architecture has been designed to make integration of disparate Simulation Models services and data as easy as possible. However, this high level of adaptability and genericity of the CRISMA Framework implies some additional effort when building an individual CRISMA Application. Integration of Simulation Models and transformation of I/O data e.g. has to happen at the level of model wrappers or data adapters which have to be developed as part of the implementation of a CRISMA Application.

No formal meta model

Although the CRISMA Framework Architecture is considered a reference architecture, it does neither define nor follow a specific meta model like RM-OA (RM-OA, 2007) or TENA (Powell, 2012).

The practical applicability of a formal meta model approach, especially when model driven automatic code generation is involved highly depends on a strict adherence to a formal methodology. This may not only conflict with the employment of rapid prototyping methods applied in CRISMA project but also pose substantial specification overhead.

No global information model

When integrating existing components and data (thus existing information models) it is problematic to develop one global information model that encompasses all pre-existing information models. While it might be possible to define a global information model for a single CRISMA Application Architecture, developing a global information model for a family of Federations (e.g. one global information model for all possible CRISMA Application Architectures) is not feasible.

No adaptation to legacy systems

When it comes to the integration of CRISMA functionality into legacy systems and Software Components it depends on those systems and components whether they can incorporate services and tools based on open standards provided by the CRISMA Framework. That means an external system that wants to interact with CRISMA Building Blocks has to adhere to the interfaces and integration concepts supported by the CRISMA Application and the CRISMA Framework. Since CRISMA Building Blocks have to be transferable and domain independent they are not indented to be adapted to legacy systems.

3. Concepts

This chapter describes the general concepts of the CRISMA Framework Architecture. Those concepts are realised by or applied to one or more Building Blocks (chapter 4).

3.1. Conceptual business logic

In CRISMA the role of the Conceptual Business Logic is to describe the key concepts of Crisis Management Simulation, the analysis of simulation results and decision support. The logic represents an analysis viewpoint meaning: rather than looking at the “real” evolvement of a crisis the logic focuses on the **Path of Analysis** of a Crisis Management Simulation that offers well defined possibilities to interfere and modify a simulation (**Transition Points**, see section 3.1.4).

3.1.1. Analysis viewpoint

In CRISMA we focus on Crisis Management Simulations that support specific planning, training or decision support objectives (**Simulation Objectives**). For this we need to consider (only) the aspects of the crisis relevant for the respective analysis represented through the concepts of the business logic. As a consequence the analysis viewpoint defines the granularity (e.g. temporal, spatial) of the view on results of a simulation. E.g. if we want to train decision makers regarding specific evacuation strategies, the specific phases from the viewpoint of the analysis of a simulation of a possible evacuation set the “pulse” (i.e. the number of World States) of how we “look” on simulation results. In general the number of World States that are interesting from an analysis point of view is much lower than the number of World States that are part of an actual simulation (e.g. states of objects moving on a map).

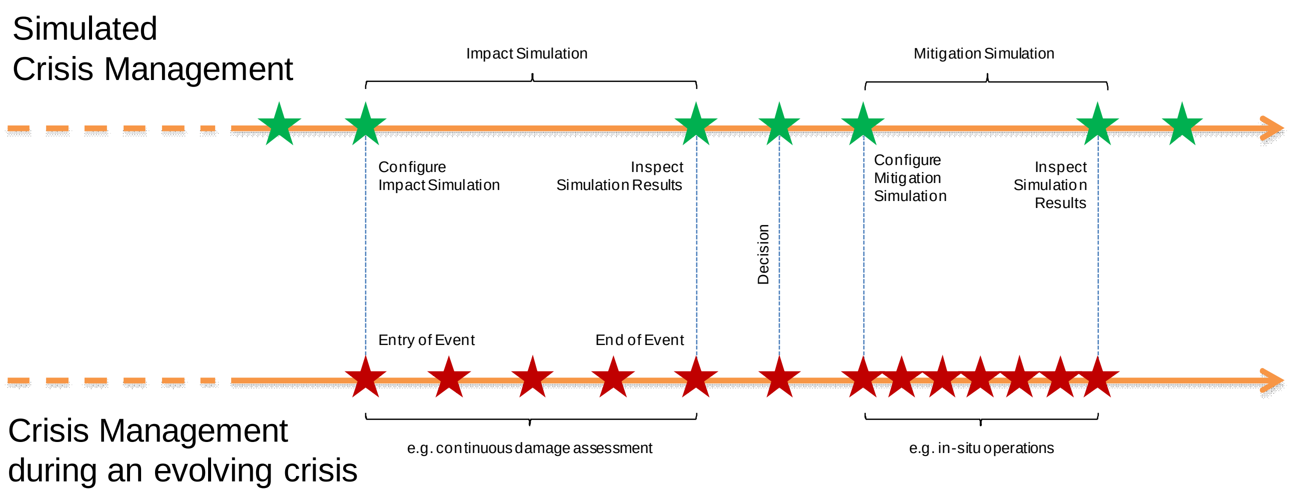

Simulated Crisis Management
Impact Simulation
Mitigation Simulation
Configure Impact Simulation
Inspect Simulation Results
Decision
Configure Mitigation Simulation
Inspect Simulation Results
Entry of Event
End of Event
Crisis Management during an evolving crisis
e.g. continuous damage assessment
e.g. in-situ operations

Figure 15: Simulation and Analysis Viewpoint.

Figure 15 visualizes the difference between the Simulation Viewpoint (Crisis Management during an evolving crisis) and the Analysis Viewpoint (Simulated Crisis Management).

While the Simulation Viewpoint has a higher number of World States, the Analysis Viewpoint focuses only on those World States relevant in the context of the respective Simulation Objective.

3.1.2. Central concepts for crisis management simulation

Figure 16 gives an overview on the central concepts for Crisis Management Simulation explained in the following subsections.

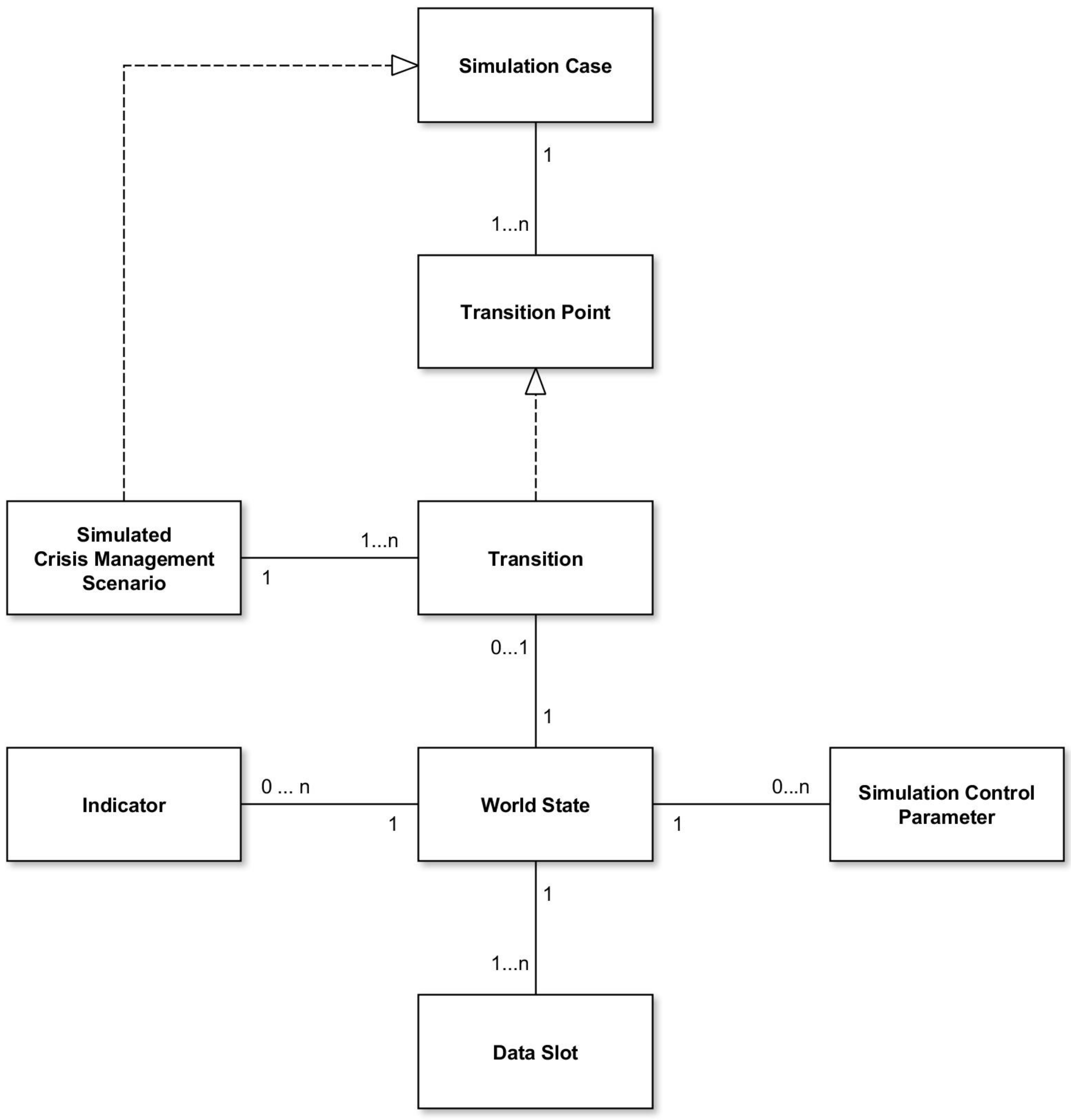


Figure 16: Central Concepts for Crisis Management Simulation.

3.1.2.1. World state space and world state elements

The basis for our considerations is the space of parameters describing a Simulated World (**World State Space**) in a crisis management context. A snapshot of the Simulated World is called **World State** and consists of all data related to a specific Crisis Management Simulation. In other words World States represent a situation or snapshot in the context of a simulated crisis management activity. World States can be either changed directly by the user or indirectly by the use of Simulation Models driven by a change in **Simulation Control Parameter**s. Conceptually, any change of a World State produces another World State.

World States consist of three groups of elements:

- Data,
- Simulation Control Parameters,
- Indicators.

3.1.2.2. World state transition and world state completeness

A change from one World State to another is called **World State Transition**. The World State Transition maintains the structure of the World State and does neither change the dimension of the Simulated World nor the type of elements that constitute the World State (Homomorphism).

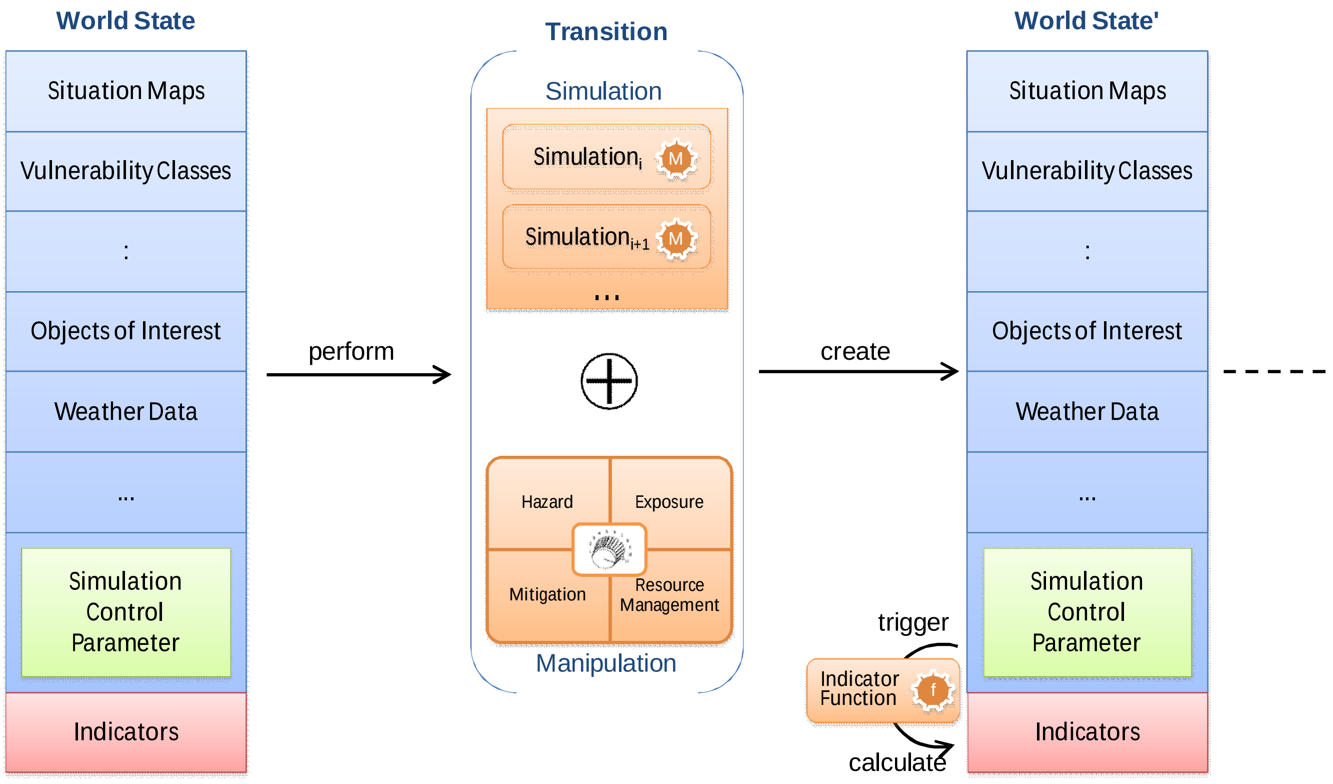


Figure 17: World State Transition, Basic Elements.

As shown in Figure 17 and Figure 18, a Transition is either induced by a Manipulation of existing World State Data or by the execution of a Simulation having the respective World State as input. The Simulation is parameterized through a set of Simulation Control Parameters that are also considered part of the World State as they might contain information required to evaluate a World State. The actual input data for each Simulation consist of a subset of the elements of the World State. In contrast, each Simulation produces a full World State to assure comparability of World States. To be able to produce a complete World State as output of a Transition, World State Data not affected by the simulation needs to be added or referenced.

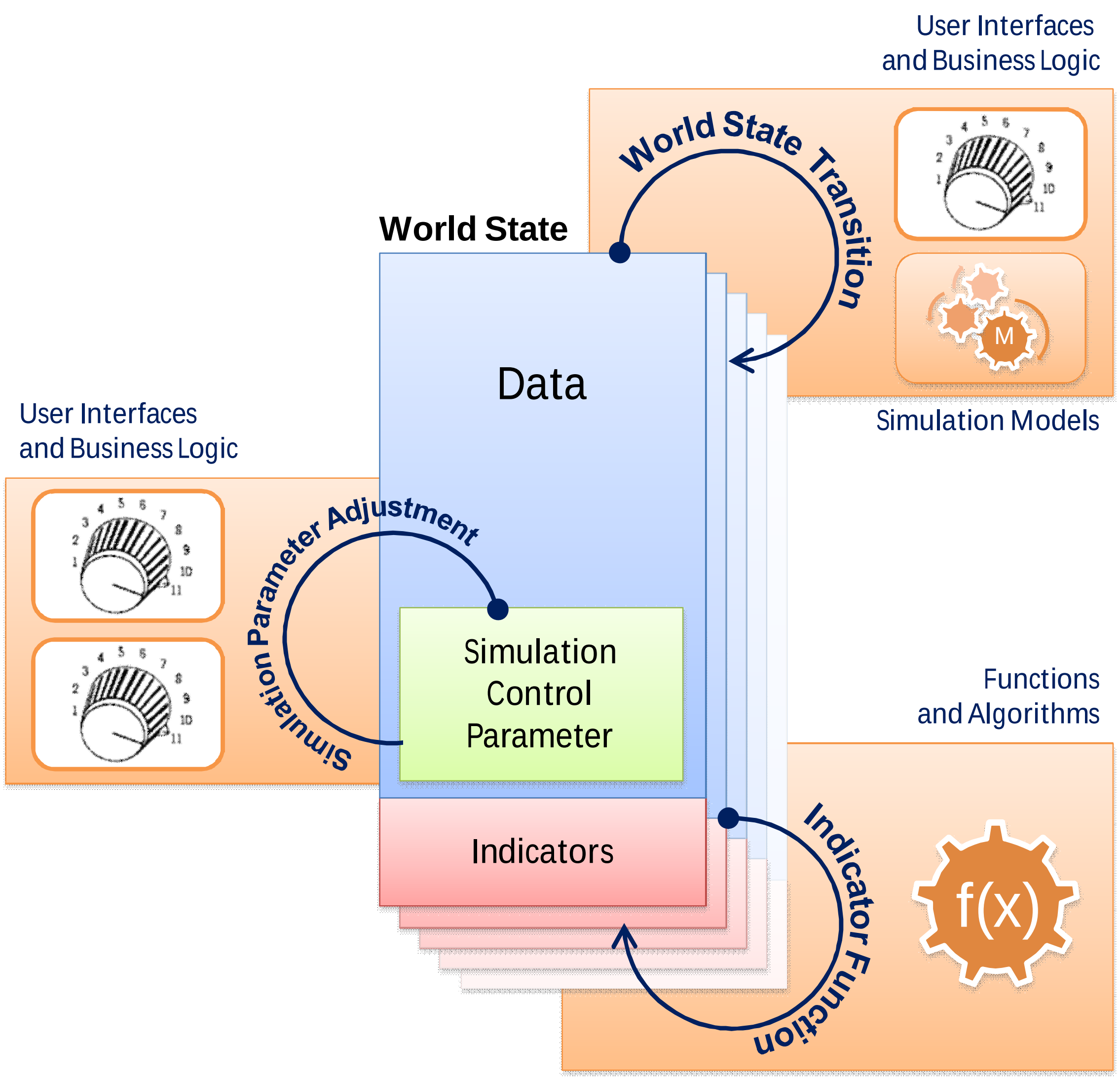

User Interfaces and Business Logic
World State Transition
World State
Data
Simulation Models
User Interfaces and Business Logic
Simulation Parameter Adjustment
Simulation Control Parameter
Functions and Algorithms
Indicators
Indicator Function
M
f(x)

Figure 18: World State Transition, Details.

3.1.2.3. Indicators and indicator functions

The creation of a new World State triggers the production of corresponding **Indicators** serving as common denominator between World States and a **Crisis Management Simulation Analysis** (e.g. Economic Impact Analysis).

From the point-of-view of the analysis of often complex simulation results, Indicators are essential elements of a World State. However, on concept level they are not produced by a Simulation but rather through the use of a specific **Indicator Function** that is the same for all Transitions in a specific **Simulation Case**. The execution of an Indicator Function is triggered immediately after the Transition has produced the result World State Data. Only after Indicators have been added to World State Data and Simulation Control Parameters a new and complete World State is obtained.

3.1.2.4. Simulation objective, simulation case, transition point, alternatives and path of analysis

A **Simulation Objective** is a well-defined purpose of a Crisis Management Simulation supporting specific planning, training or decision support activities. The physical representation or the means to reach a simulation objective is the **Simulation Case**. A

Simulation Case defines a graph of possible Transitions on a World State Space for a specific Simulation Objective.

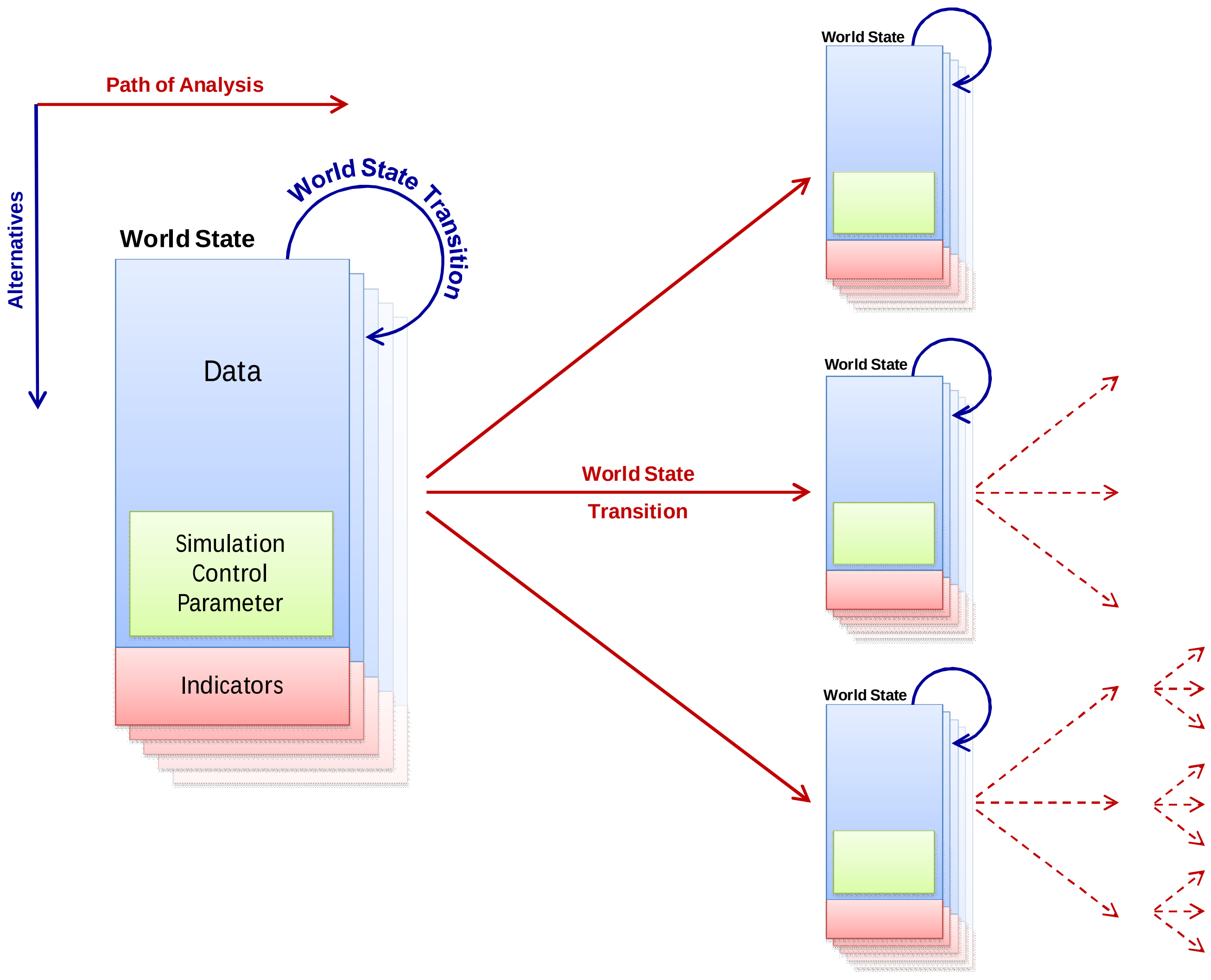


Figure 19: World State Transition, Alternatives and Path of analysis at a Transition Point.

A node in this graph pinpointing the possibility to interfere in the simulation is called a Transition Point. Alternatives are represented by the triples (World State, Transition, {World State', World State'',…, World State^{n}}) at a specific Transition Point (Figure 19).

A consecutive sequence of Transitions through the Simulation Case Graph can be called a Path of Analysis. Finally, a Path of Analysis (Figure 20) that starts at the root of a Simulation Case and ends at a leaf can be called a **Simulated Crisis Management Scenario** (for more details see section 3.1.4).

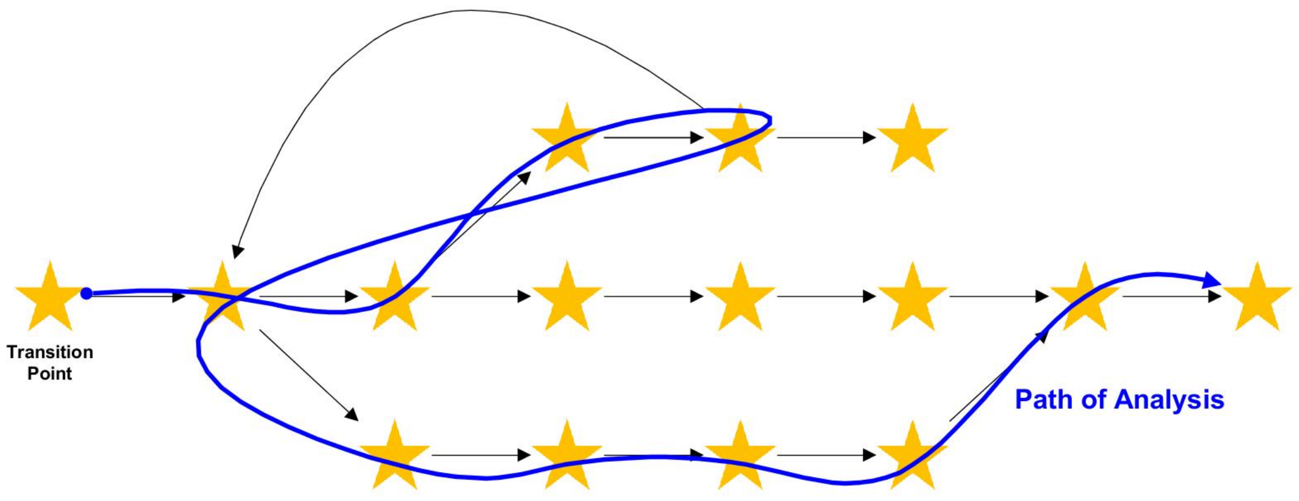


Figure 20: Simulation Case, Transition Point and Path of Analysis.

This understanding of a Scenario as consisting of a sequence of Transitions changing World States is essential to the understanding of the overall concept. Moreover, it is important to note that the concept is independent of crisis management phases as well as the elapsed time of a crisis. This is because all aspects of time are considered as elements of a World State.

3.1.3. Central concepts for crisis management simulation analysis

Comparison and analysis of different World States, Alternatives and related Transitions enables crisis managers to explore the effects of changes in mitigation strategies affecting vulnerabilities based on related World State Data.

3.1.3.1. Crisis management simulation analysis

In CRISMA the concept of Crisis Management Simulation Analysis has three variants:

- The assessment of planning options
- Training i.e. the education of stakeholders
- Decision support for crisis management activities

For each of the three variants CRISMA supports Crisis Management Simulation Analysis on three levels:

- Exploration of World State Data
- Comparison and assessment of aggregated data
- Algebraic Comparison that enables e.g. ranking

The core concepts for the upper two levels of this approach are Indicators and Criteria being the basis for a possible Multi-Criteria Analysis.

3.1.3.2. Indicator, indicator vector, indicator space and indicator function

From the viewpoint of Crisis Management Simulation Analysis we can extend the concept of Indicators and related terms as shown in Figure 21.

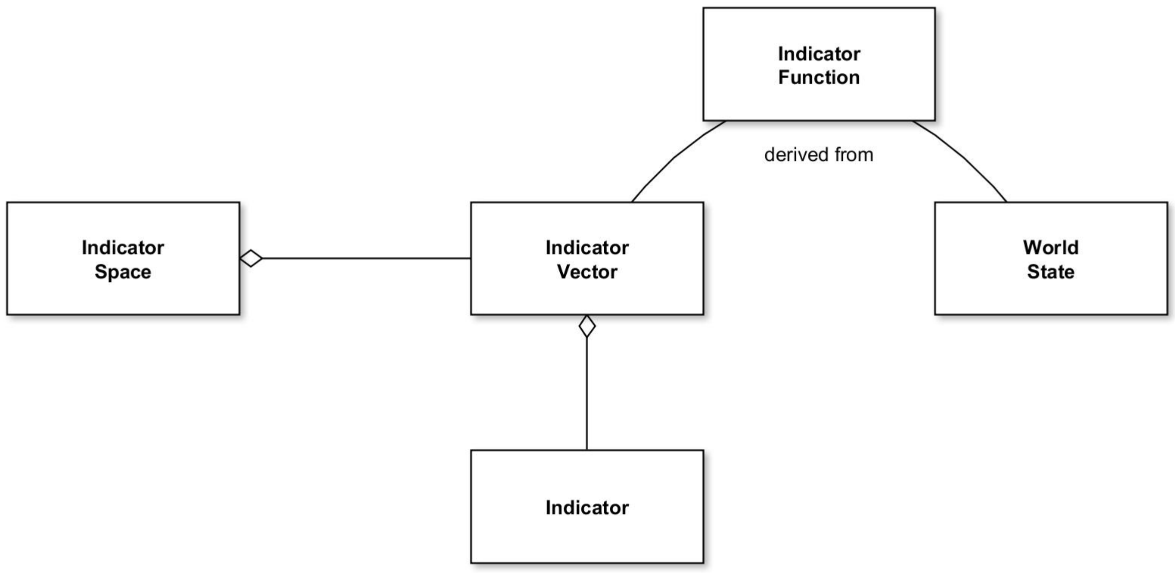


Figure 21: Indicator, Indicator Vector, Indicator Space and Indicator Function.

An Indicator is an aggregation of elements of a World State. The **Indicator Space** is a mathematical vector space so we can scale and add its elements. Each vector in this space represents an aggregated image of a World State. By this an **Indicator Vector** is an instance in an Indicator Space and contains a number of Indicators as elements. An Indicator Function produces an Indicator Vector where the dimension and complexity of the World State Space is drastically reduced.

In other words, Indicators are a concept that helps us to bring more structure in the Simulated World aggregating thus losing information so we can do e.g. algebra on this representation of World States for the benefit of being able to better evaluate complex World State Data.

3.1.3.3. Criteria, criteria vector and criteria function

A **Criterion** sets an Indicator in relation to a Simulation Objective. More precisely, Criteria measure the level of satisfaction of an Indicator in the context of a specific Simulation Objective. A Criteria Function maps an Indicator Vector to a corresponding Criteria Vector. The dimension of an Indicator Vector and a Criteria Vector is usually the same.

The use of the Criteria concept is the basis for **Algebraic Comparison** of World State based Crisis Management Scenarios that enables us to introduce e.g. ranking methods.

3.1.3.4. Algebraic comparison, multi-criteria analysis and ranking

To be able to give a decision maker hints regarding which Crisis Management Scenario performs best according to specific Criteria we have to further aggregate information (Figure 22) and introduce methods to set the scenarios into an order relation. On the basis of this order relation we can algebraically compare scenarios and produce e.g. a ranking of scenarios with respect to specific criteria.

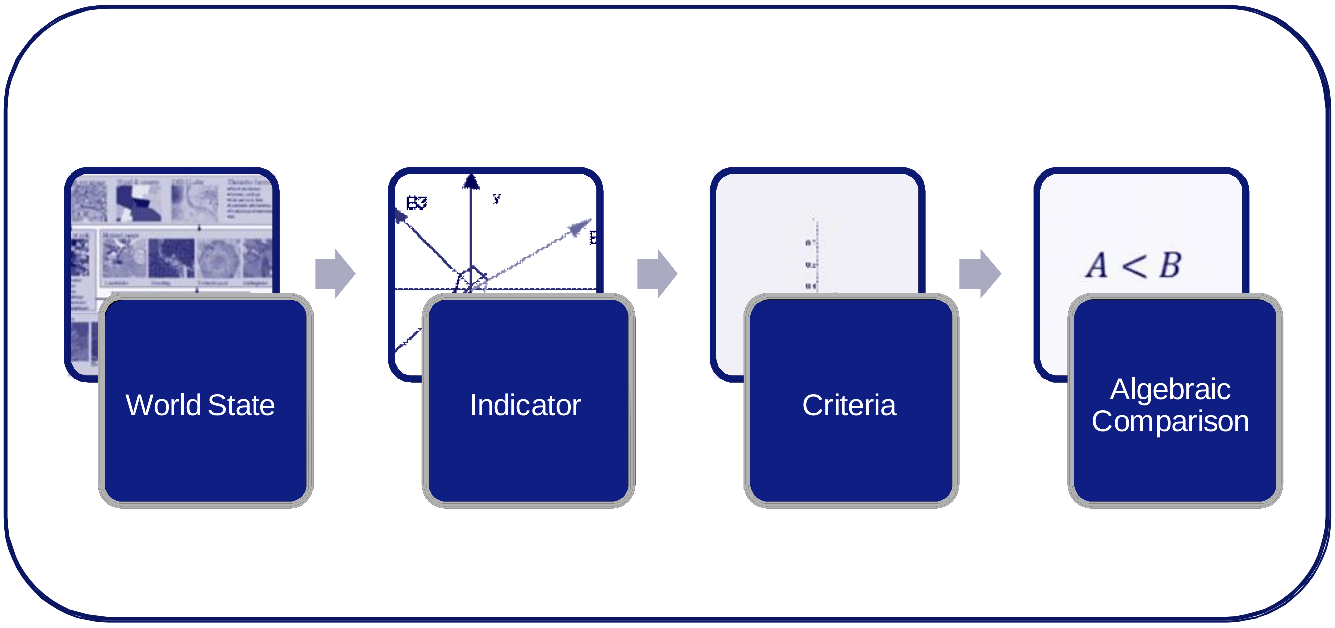


Figure 22: Aggregation Steps for Crisis Management Simulation Analysis.

There are a number of prominent methods in the field of Multi-Criteria-Analysis to essentially map a Criteria Vector to a representative single scalar value. CRISMA has selected the OWA (Ordered Weighted Averages) method that allows the definition of a specific decision strategy by combining the means to emphasise specific Criteria and their achieved level of satisfaction. For more details refer to the specification of the Model for decision-making assessment, economic impacts and consequences and Simulation (D44.1, 2013) and section 3.19 of this document.

3.1.4. Simulation case graph

As already described in section 3.1.2.4, each Simulation Case is initially specified as a set of Transition Points that are the nodes of a Simulation Cases Graph (Figure 23). This graph defines a Path of Analysis that can be followed in a CRISMA Application.

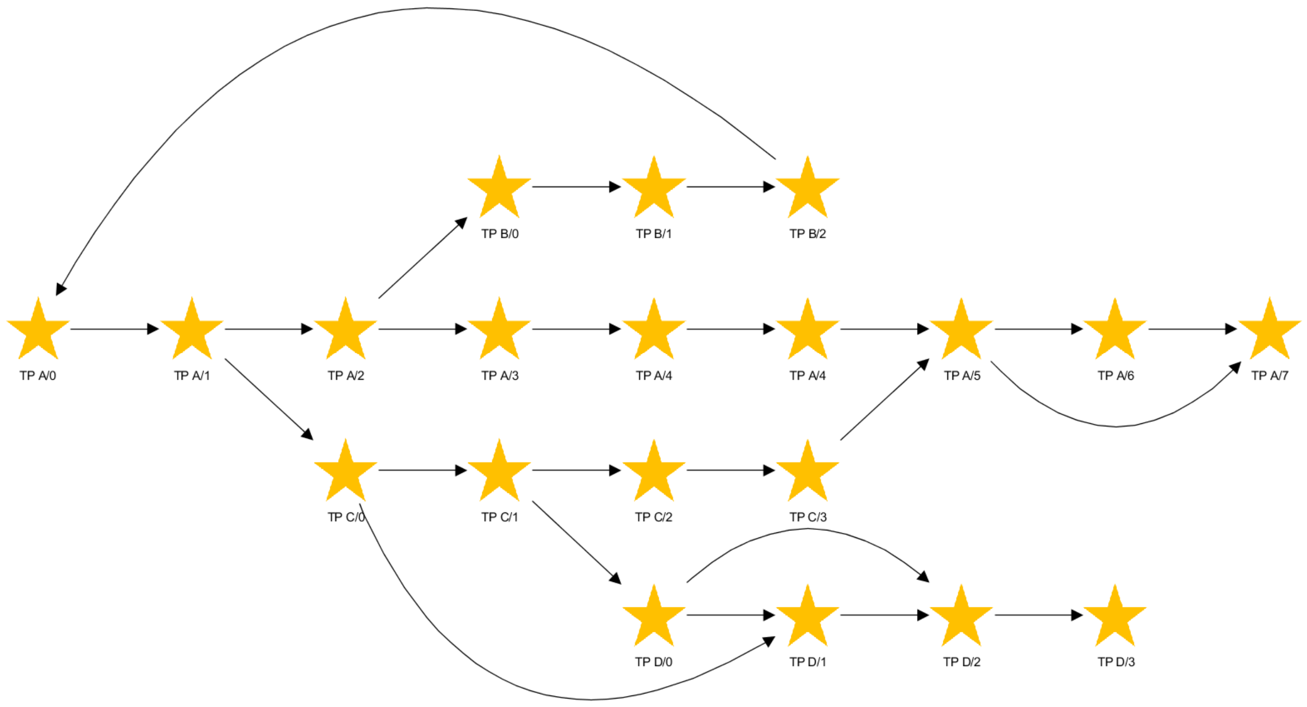


Figure 23: Simulation Case Graph.

A Transition Point represents the type of the Transition that can be performed by the user at a defined point in a Simulation Case. Thus, the temporal or logical evolvement of a Simulation Case and its concrete instances, the Simulated Crisis Management Scenarios, respectively, can be expressed by a sequence of Transition Points (Simulation Case Graph). Transition Points can be freely defined by the designer of the Simulation Cases and the CRISMA Application, respectively.

While in a certain Simulation Case an arbitrary number of Transition Points could be defined, it is advisable to limit the number of Transition Points to the ones required to support the decision objective. For this purpose, a Simulation Case should be divided into discrete steps whereby at each step the CRISMA Application should allow the user to perform e.g. important or alternate decisions, an analysis of the current World State, a comparison of Indicators, etc which are then related to Transition Points. Thus, the Simulation Case Graph actually defines the Analysis Viewpoint (section 3.1.3) of a CRISMA Application.

Both concepts Transition Points and World State Definitions (specifications of the World State Data Slots) act as input for individual Crisis Management Scenarios which consist of a sequence of concrete Transitions and World States (Experiments). Figure 24 shows an example of a Crisis Management Scenario which follows a specific path through the Simulation Case Graph.

This path is chosen by the user during the execution of the scenario. The Simulation Case Graph can therefore be considered as a set of suggested Paths of Analysis, i.e. sequences of Transitions that are required to meet particular Simulation Objectives (see section 3.1.2.4) of a Simulated Crisis Management Scenario.

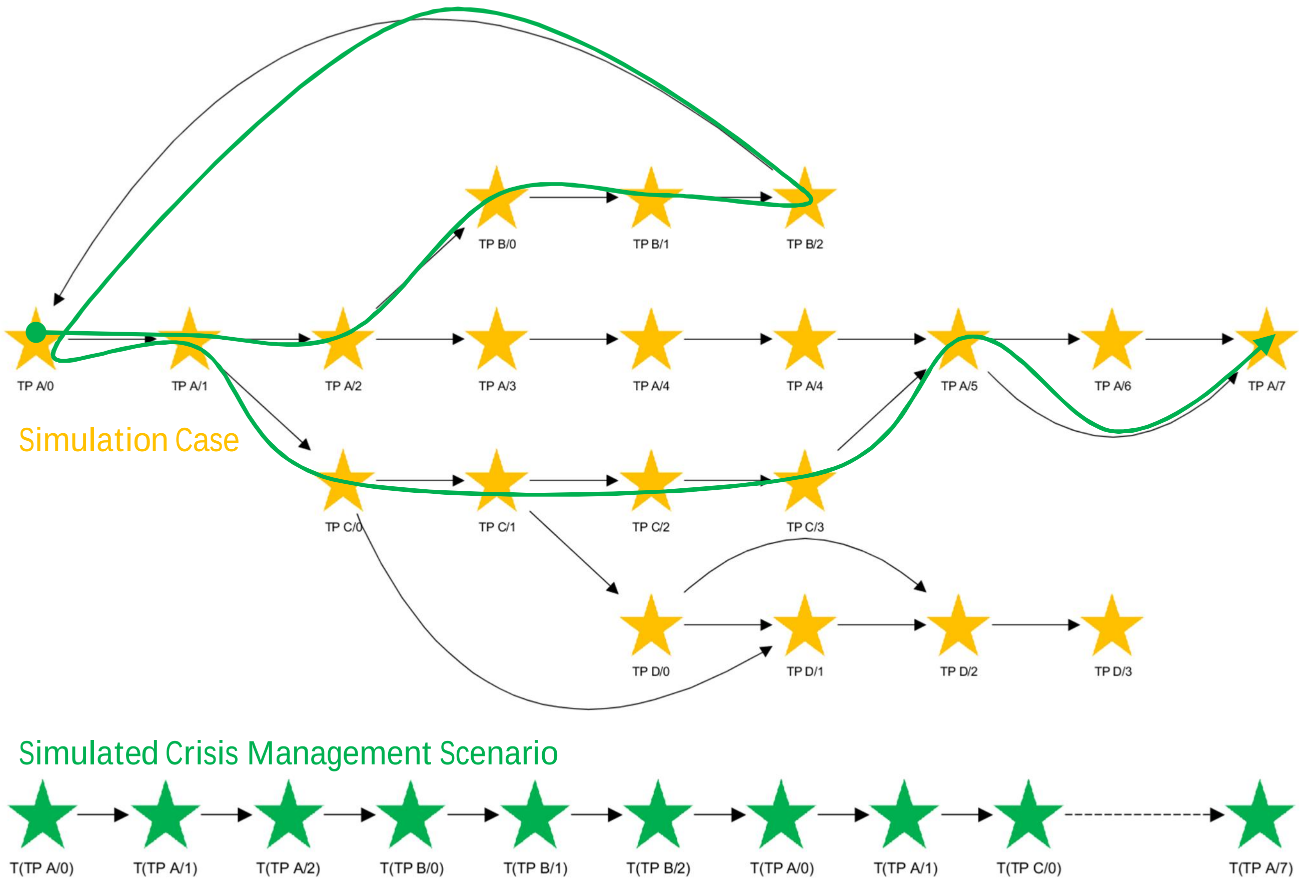


Figure 24: Simulated Crisis Management Scenario as Instance of Simulation Case.

As shown Figure 24, the Scenario Graph permits different alternative paths. Except for root (TP A/0 in the example above) and leaf nodes (TP A/7 and TP D/3 in the example above), a Transition Point has one or more predecessors and successors. In general, it is up to the developer of the CRISMA Application to enforce the predefined order of Transition Points. Therefore a user of the Application may or may not be allowed to perform Transitions which deviate from the predefined paths of the Simulation Case Graph.

The self reference of the Transition Point emphasises the possibility to create a set of alternative Transitions which can then be compared during a Crisis Management Simulation Analysis (e.g. comparing Indicators of different alternative decisions).

As already shown in Figure 23, the Simulation Case Graph itself permits alternate paths to be followed according to the temporal evolvement of a crisis or the logical evolvement of a specific use case. The graph defines mainly in which order which *types of Transitions* can be created in the Application.

Additionally, at each Transition Point the user has the possibility to create different alternate variations of a Transition, e.g. by choosing different Simulation Control Parameters (earthquake intensity, number of resources deployed, ...).

The difference between choosing an alternative path in the Simulation Case Graph and performing an alternate Transition in a Simulated Crisis Management Scenario might not be easily understandable in the first place. Therefore, one should keep in mind that a

Transition Point represents a type (what *can* be done) and the Transition a concrete instance (what *is* done).

An example of six different Crisis Management Scenarios (CMS1 + CMS6) which all follow the same path of the Simulation Case Graph of Figure 24 is shown in Figure 25 below.

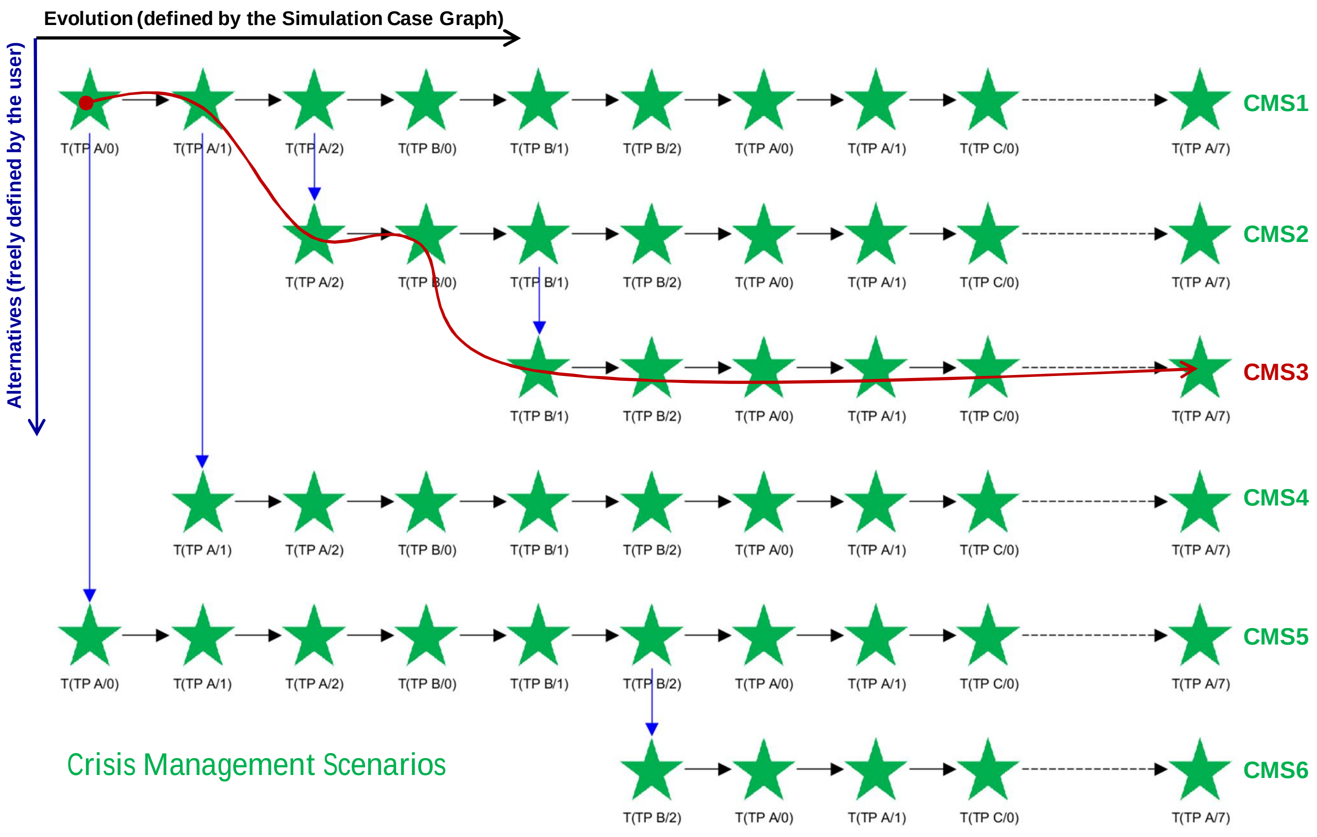

Evolution (defined by the Simulation Case Graph)
Alternatives (freely defined by the user)
T(TP A/0)
T(TP A/1)
T(TP A/2)
T(TP B/0)
T(TP B/1)
T(TP B/2)
T(TP C/0)
T(TP A/7)
CMS1
CMS2
CMS3
CMS4
CMS5
CMS6
Crisis Management Scenarios

Figure 25: Crisis Management Scenarios.

Like a Transition Point, a Transition as shown in Figure 25 above has also one predecessor and possibly multiple successor Transitions, except for the root Transitions and Leaf Transitions. While a Simulation Case can be modelled as *graph* of Transition Points, a concrete Crisis Management Scenario is always represented as a path in a *tree* of Transitions.

3.1.5. Relevance of the conceptual business logic

In relation to the Conceptual Business Logic the objective of SP3 is to provide software elements (Building Blocks and respective Software Components) that implement the business logic in the form of a software framework. As a consequence the framework will be most useful for applications that follow/adopt the Conceptual Business Logic.

After all, the Conceptual Business Logic is not only the conceptual basis of the CRISMA Framework but equally important as the foundation of more advanced Crisis Management Simulation Concepts that are in the focus of CRISMA i.e. Cascade Events and effects (D42.1, 2013) and Time-Dependent Vulnerability (D43.1, 2013).

3.2. Reference architecture

Essentially, the largest part of the CRISMA Framework Architecture is the Functional Architecture e.g. specifying the functional Building Blocks of the Architecture on an abstract level rather than describing or selecting existing Software Components. Obviously, Functional (abstract) Building Blocks will in general not be implemented from scratch and a decision for a specific Software Component has to be made at a certain point in time. However, due to the diversity of the envisioned CRISMA Applications in terms of requirements, objectives and availability of models and tools to be reused the CRISMA Framework Architecture cannot always anticipate the decision how and by which Software Component a certain Building Block has to be realised (or implemented). Therefore, one goal of this architecture is to define how existing applications and systems can be complemented by CRISMA functionality.

Thus, a proper balance between the requirement to be able to incorporate legacy applications or legacy components available at the pilot sites on the one hand and the ability to build a custom CRISMA Application on basis of ready-to use Building Blocks contributed by the CRISMA Framework on the other hand has to be found.

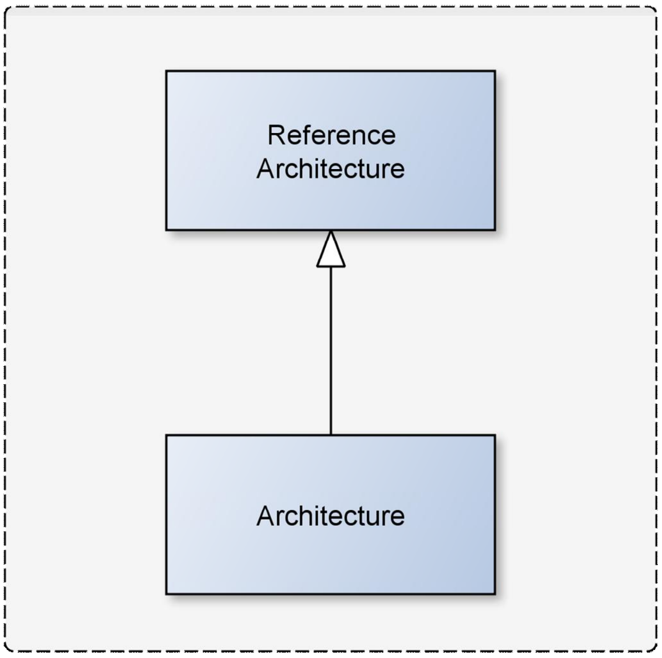


Figure 26: Reference architecture.

As a consequence, the CRISMA Functional Architecture is designed as a reference architecture that follows a lightweight and pragmatic reference model approach. A reference architecture is an architecture to build concrete architectures (Figure 26).

Therefore, a realisation in the context of a reference architecture is a concrete architecture. As a reference architecture the Functional Architecture provides the tools, Building Blocks and guidelines to create a concrete architecture of a CRISMA Application, a so called Application Architecture.

However, the CRISMA Framework Architecture does not stop at the abstract level; it provides also an Implementation Architecture (chapter 5). The Implementation Architecture selects and defines already all Software Components for the implementation of the Building Blocks of the Functional Architecture. Furthermore, it defines the common and generic Core Control and Communication Information Model which can be extended by the individual Application Architectures.

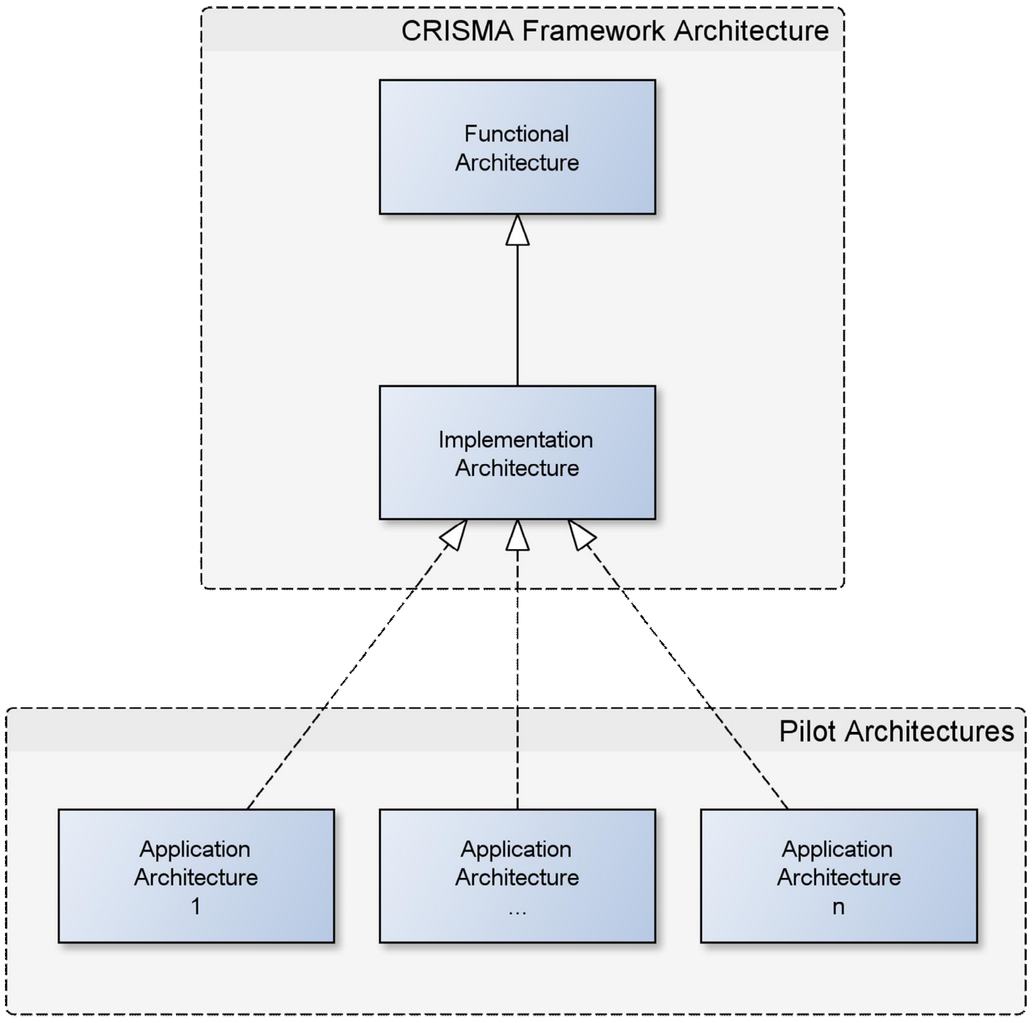

CRISMA Framework Architecture
Functional Architecture
Implementation Architecture
Pilot Architectures
Application Architecture 1
Application Architecture …
Application Architecture n

Figure 27: CRISMA Architecture Relationships.

The relationship between the different architecture types is visualised in Figure 27: The Implementation Architecture is a realisation of the Functional Architecture. There exists exactly one Functional Architecture and exactly one Implementation Architecture.

An Application Architecture is a specialisation of the general Implementation Architecture and thus an intermediate realisation of the Functional Architecture. The number of Application Architectures is not limited. Following this approach, the CRISMA project will deliver 5 different Application Architectures, one for each pilot. Since an Application Architecture inherits the concepts and generic interfaces of the Functional Architecture interoperability between different Application Architectures is fostered.

3.3. CRISMA federations, applications and scenario orientation

As can be seen in Figure 28, a CRISMA Application is an implementation of an Application Architecture. Moreover, it is realised with the help of Software Components that are provided by the CRISMA Framework. Like the CRISMA Framework Architecture which provides the means to create (specify) a specific Application Architecture, the CRISMA Framework is implemented as integration Toolkit which offers a set of defined functionalities represented by Software Components, Building Blocks exposing well defined APIs, and general guidelines, templates, Control and Communication Information Models and so on. The CRISMA Framework thereby implements, selects and adapts all Building Blocks that are the basis to build an entire CRISMA Application.

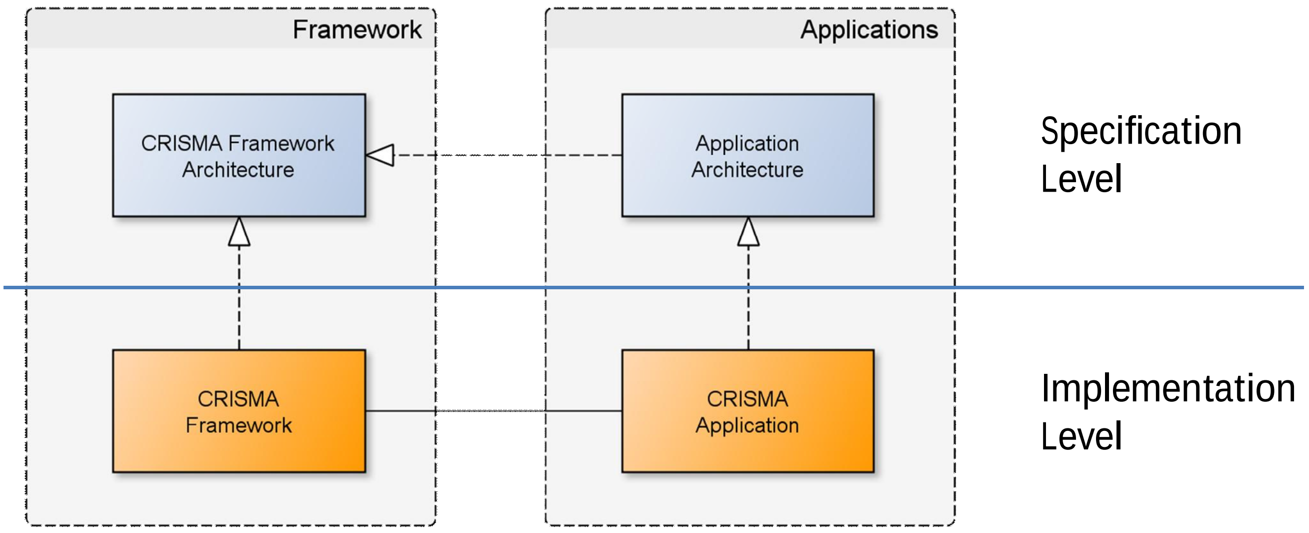


Figure 28: CRISMA Framework and applications.

The terminology of Functional Architecture, Implementation Architecture, Application, etc. is used to illustrate the relationships and dependencies between the different levels of specifications and the implementation as well as to establish a common conceptual basis for the design and implementation of CRISMA Applications on the basis of a common architecture and framework.

Furthermore, to illustrate important interoperability and integration concepts and to distinguish between CRISMA-aware and non-aware as well as external (integrated) and internal components we introduce the terms of Federates and Federation. Please be aware, that the terms CRISMA Application and CRISMA Federation are used synonymously. The terms Federate and Federation have been adopted from HLA, High Level Architecture (IEEE, 2010).

Using this terminology and as illustrated in Figure 29, a CRISMA System which comprises the overall outcome of the CRISMA project is a Federation of Federations. Although interoperability between different Federations is supported by architectural concepts this is out of scope of the project even though inter-Federation interactions open the possibility for various cross-pilot activities, e.g. propagation of cascading effects from one pilot application to another.

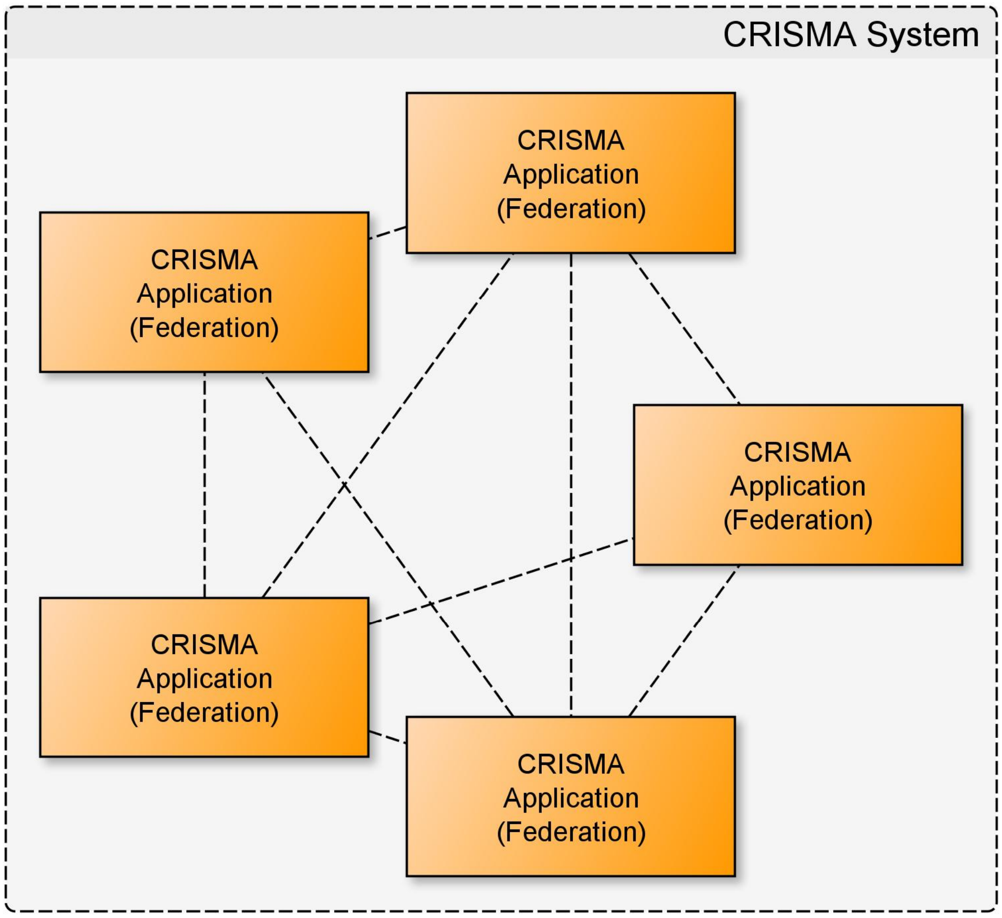

CRISMA System
CRISMA Application (Federation)
CRISMA Application (Federation)
CRISMA Application (Federation)
CRISMA Application (Federation)
CRISMA Application (Federation)

Figure 29: CRISMA System as Federation of Federations.

As can be seen in Figure 30, a CRISMA System is not a single CRISIS Management System but a system of different CRISIS Management Applications which are built using a common framework.

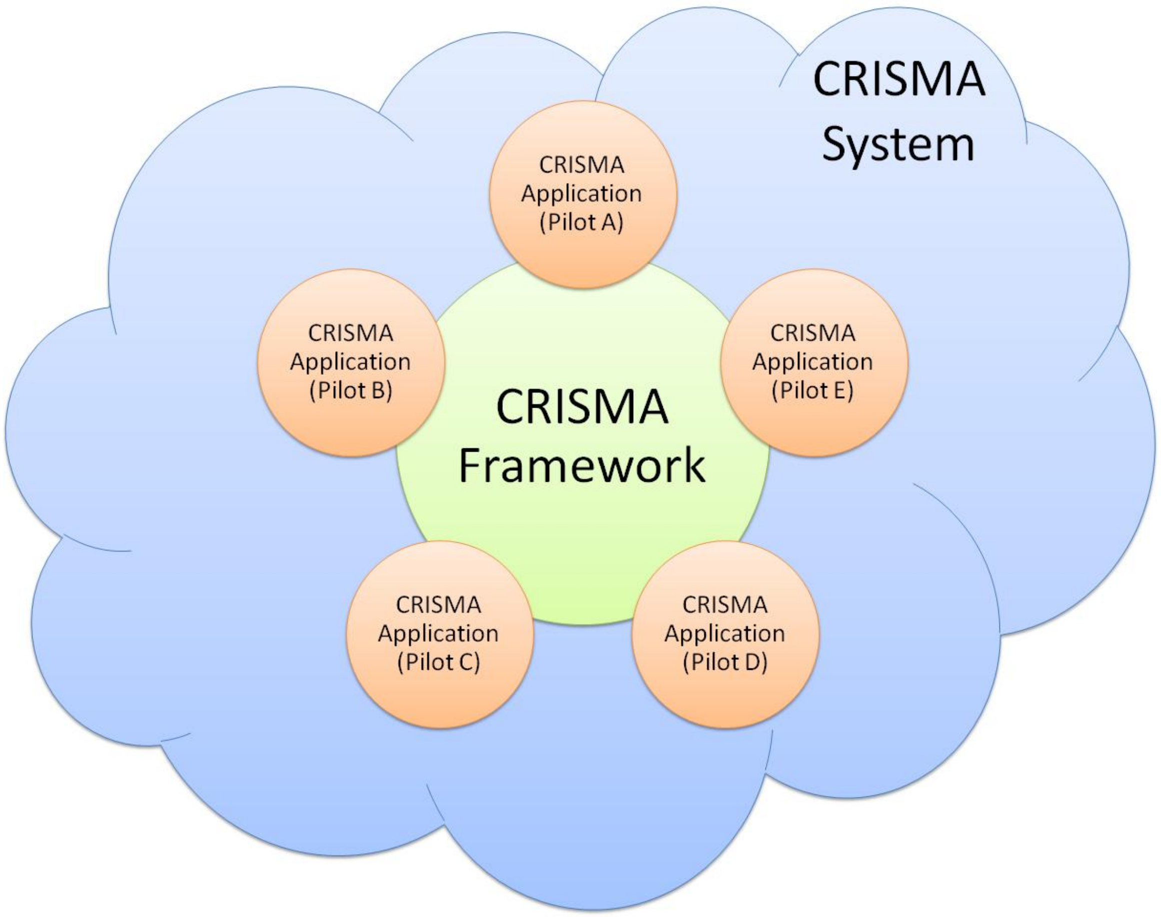

CRISMA System
CRISMA Framework
CRISMA Application (Pilot A)
CRISMA Application (Pilot B)
CRISMA Application (Pilot E)
CRISMA Application (Pilot C)
CRISMA Application (Pilot D)

Figure 30: CRISMA Applications.

A CRISMA federation is a scenario oriented Crisis Management Application composed of CRISMA Federates and built with the help of Building Blocks (services, tools, models, etc.) and guidelines provided by the CRISMA Framework. Scenario orientation (Rome, 2009) in this context means, that the CRISMA Framework cannot anticipate the management of all possible types of crisis scenarios in detail but rather provides the means to define and use simulated crisis scenarios for training, planning and decision making. As already observed in the architectural constrains (section 2.4), the CRISMA Framework is a generic integration and development framework but not a generic Crisis Management System. To leverage the development of individual CRISIS Management Application, the CRISMA Framework provides several pre-integrated Reference Applications. The design and implementation of Reference Applications is focused on the support of a set of Reference Scenarios as described in section 3.15.

CRISMA Framework Building Blocks must be composable and configurable, and thus adaptable to the needs and requirements of the CRISMA Application to be built. Of course, CRISMA Pilot Applications don't have to be realised exclusively with help of the CRISMA Framework Building Blocks. Application specific legacy applications may either be wrapped into the CRISMA Federates, e.g. by implementing a specific interface, or just be integrated on a lower e.g. data level.

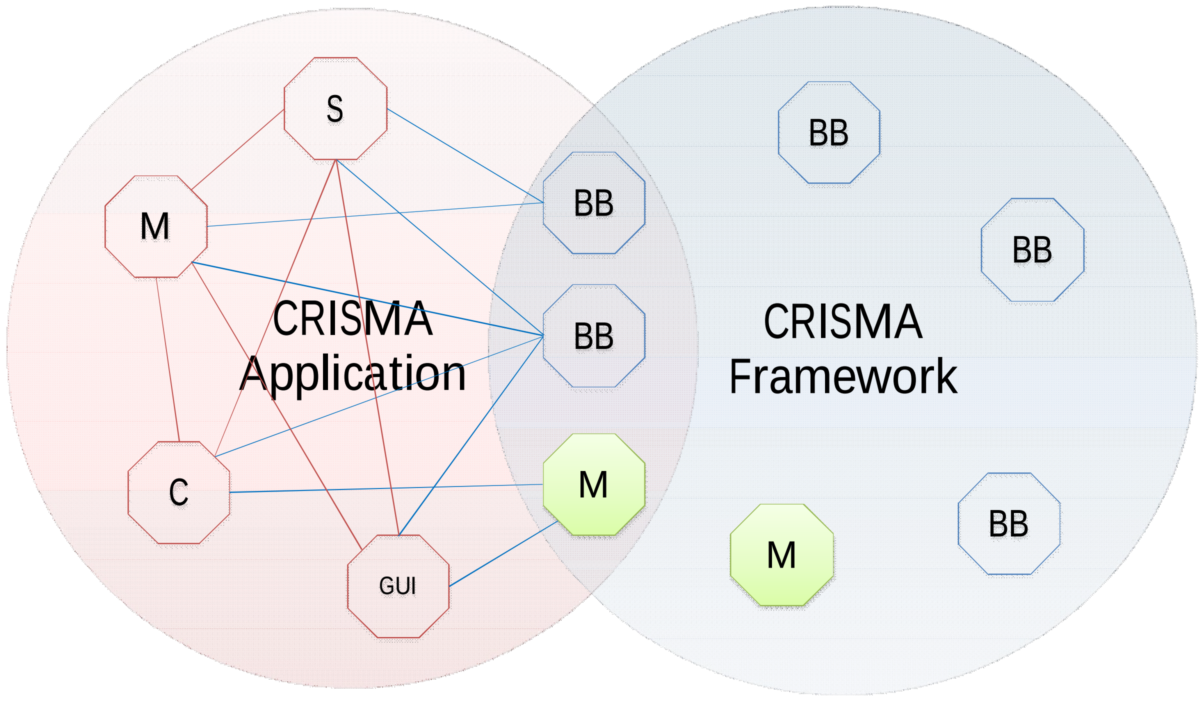


Figure 31: Relation of a CRISMA Application to the CRISMA Framework.

As shown in Figure 31, a CRISMA Application may be built around an existing system as long as the core concepts of the CRISMA Framework Architecture are respected and the essential Infrastructure Building Blocks are used. Thus, a CRISMA Application may consist of application specific models, components, GUIs or even complete legacy systems which are not part of the CRISMA Framework.

The relation of the different types of Building Blocks, Federations and Integrated Systems and components as well as a precise definition is given in section 3.7.

3.4. Communication paradigms and spatial distribution

Since a CRISMA Application should function standalone as well as distributed system (see section 2.1 for more details), the Functional Architecture is based on paradigms for distributed systems. Due to this, the CRISMA Framework Architecture is designed as multi-tiered architecture that supports the creation of potentially spatially distributed CRISMA Applications (Federations) which may interact and exchange data over public networks (e.g. the internet) as well as systems deployed on a single and isolated machine.

Considering this in combination with strong interoperability requirements it is obvious that the very flexible distributed system design approach has to be applied to the CRISMA Framework Architecture. Additionally, it has to be clarified how the communication and information exchange between disparate actors can be harmonised.

A possible solution to this problem based on global information models and one common communication infrastructure is provided by High Level Architecture (HLA). However, as already observed in section 2.4 applying the HLA approach to a CRISMA Federation might impose too many restrictions on the participating Federates and Integrated Systems. Not only models but any other component of the Federation would have to adhere to HLA interfaces and principles and thus route all data through a HLA compliant Runtime Infrastructure (RTI). Furthermore, all participants of the Federation would have to agree upon one Common Information Model, the Federation Object Model (FOM).

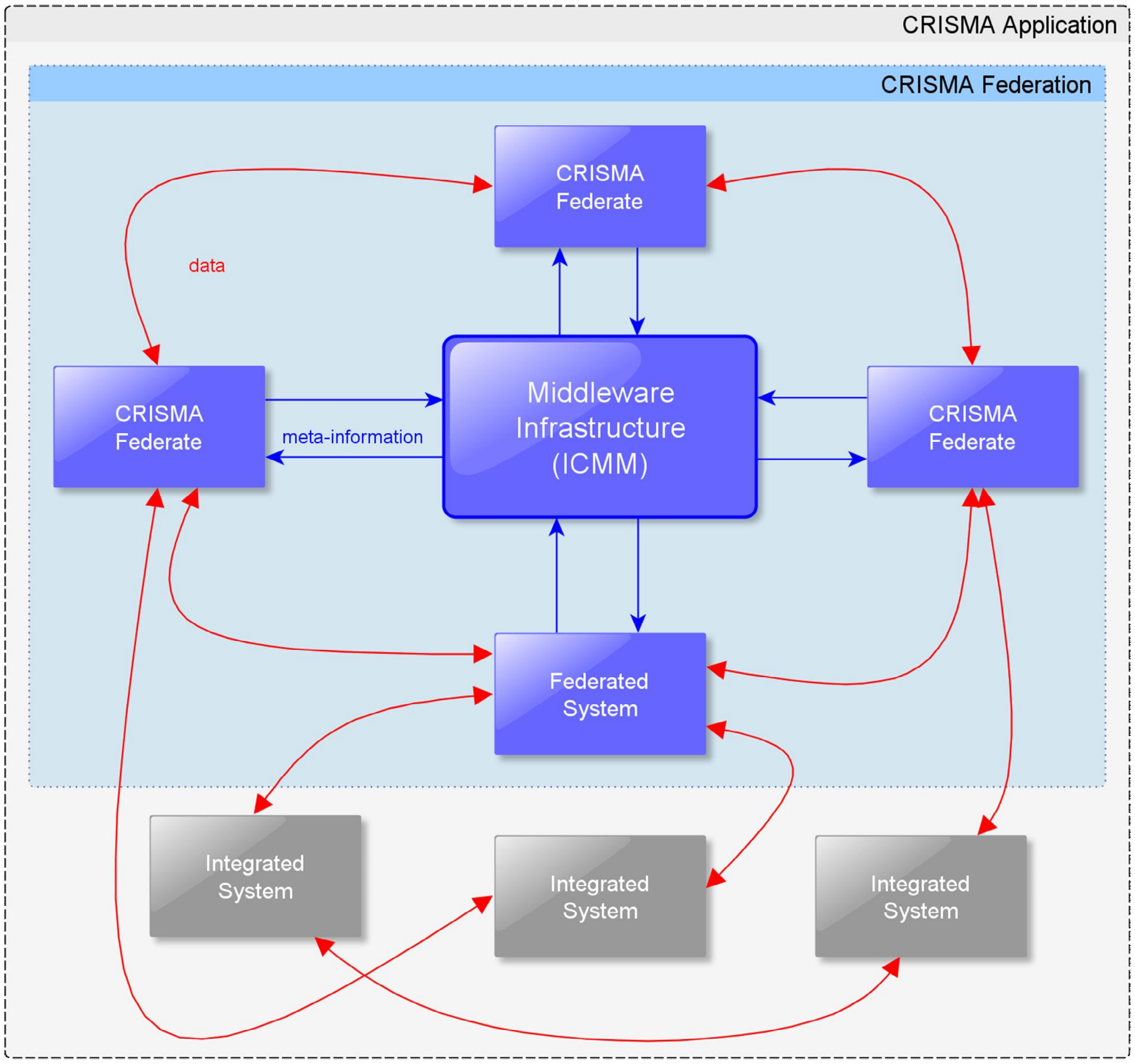


Figure 32: Information Flows.

Therefore, in the CRISMA Framework Architecture a different approach is followed which applies an event based Control and Communication Information management paradigm on top of existing client/server infrastructures. One of the fundamental concepts is the separation of Control and Communication Information and data flows (section 3.11). Thereby strict rules regarding communication paradigms, encodings, protocols, etc. for the Control and Communication Information flows have to be enforced by the architecture. On the other hand, the rules for actual data flows are recommendations and subject to interpretation during the implementation of an individual CRISMA Application. Although we claim that the Functional Architecture is implementation and technology independent, these rules have also to include some technological aspects with potential impact on the architecture in general.

As shown in Figure 32, Control and Communication flows (blue) are handled by a Middleware Infrastructure which connects Federates in a uniform, simple and transparent way (section 3.5). Data flows (red) may be established between any Federates and Integrated Systems, components, or models directly. For an explanation of the difference between Federated and Integrated Systems or components please refer to section 3.7. The communication model applied in the interaction between Federates and the Middleware Infrastructure can best be described as resource-oriented and event based.

Thereby, communication styles like publish-subscribe and the request-response are supported in the communication with the Middleware Infrastructure. The publish-subscribe pattern is mainly used for the propagation of events while the request-response pattern is used for the general access to Control and Communication Information. It is important to note, that Control and Communication Information shall not be exchanged between Federates directly and thus must be transferred solely using the APIs provided by the Middleware Infrastructure (section 3.5). In accordance to common REST principles the mandatory communication protocol used in this type of communication is the hypertext transfer protocol (HTTP).

The exchange of data (maps, time series, …) occurs in general in a peer to peer fashion where the actors in the data exchange are directly connected. Although HTTP is the obvious choice, any other protocol may be permitted as long as it is required to communicate with an Integrated System or component. To foster interoperability on data level the architecture proposes several standardised protocols and message formats for direct data exchange between Federates.

3.5. Middleware infrastructure

CRISMA Federates are interconnected through a central Middleware Infrastructure which acts as interoperability layer between disparate systems and components. This Middleware Infrastructure is driven by Control and Communication Information and provides common functionalities and a model for the interaction of heterogeneous components. It can be seen as the core of the CRISMA Framework and thus of each CRISMA Application. The Middleware Infrastructure is realised as a generic distributed resource-oriented Integrated Crisis Management Middleware (ICMM) which in turn is one of the most important Infrastructure Building Blocks of the CRISMA Framework Architecture. An overview of the ICMM Building Block can be found in section 4.1.

To be able to integrate legacy applications, models and data sources this Integrated Crisis Management Middleware has to support arbitrary Control and Communication Information Entities, including Simulation Models, Simulation Model adapters and Simulation Cases. Moreover, it needs to allow the management of Control and Communication Information. Thus it provides basic create, update and delete (CRUD) operations for all types of Control and Communication Information Entities. The functionality of the ICMM is exposed through easy to use and easy to understand web-based RESTful APIs (some examples shown in Figure 33) that are usable by a wide variety of different client applications like capacity resource & assets management systems, model wrappers, mobile apps or web browsers.

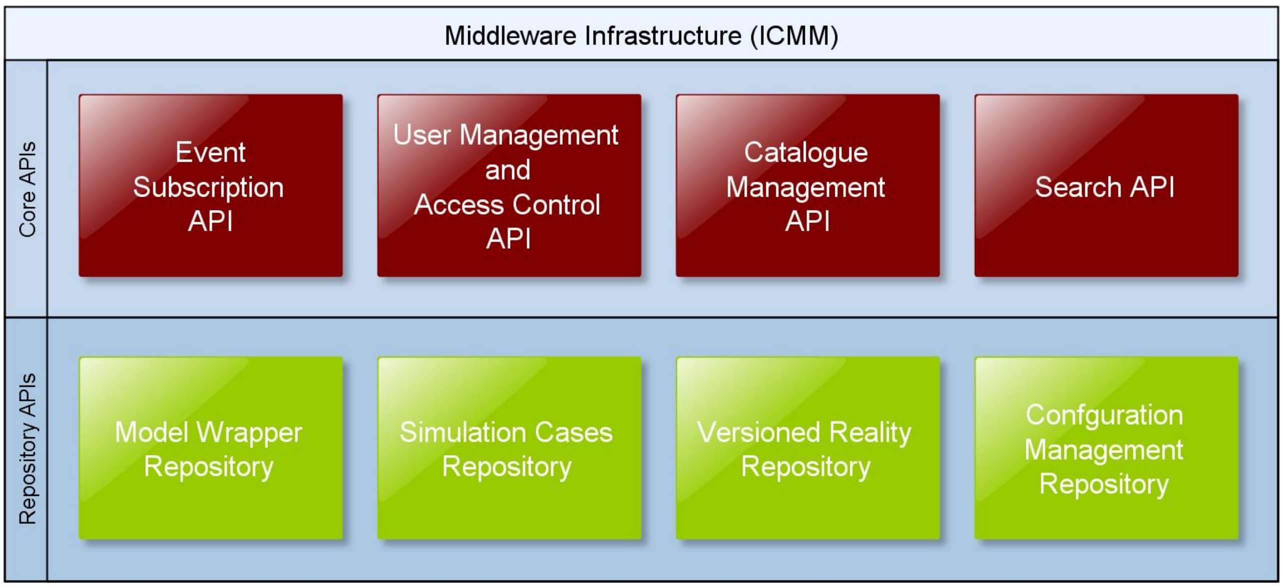

Middleware Infrastructure (ICMM)
Core APIs
Event Subscription API
User Management and Access Control API
Catalogue Management API
Search API
Repository APIs
Model Wrapper Repository
Simulation Cases Repository
Versioned Reality Repository
Confguration Management Repository

Figure 33: Examples of Middleware Infrastructure APIs.

As shown in Figure 33, the ICMM provides also repositories and APIs that enable secured and controlled access to Control and Communication Information Entities, including support for authentication and authorisation.

Another important ability of the ICMM is to build modularised Control and Communication Information Models (CCIMs) and to allow the introduction of new entity types without causing changes to the core API and clients of the ICMM or the infrastructure. The development of CCIMs goes hand in hand with the provision of Control and Communication Information Repositories and related repository APIs. Figure 33 also shows some example repository APIs to be provided by the ICMM as well as a number of core APIs. The actual standard repository APIs of the ICMM are defined by the Core CCIM which is described in chapter 5.

Figure 34 shows the relation between CCIMs, Repositories and APIs. The CCIM represents both the information schema (e.g. a database schema) that defines the structure of the repository (e.g. a data base) as well as the formal specification of the repository's API. Due to the RESTful approach followed by the CRISMA Middleware Infrastructure, Control and Communication Information Entities defined by the CCIM and stored in the Control and Communication Information Repository can be mapped to URLs exposed by a REST API. Ideally, the definition of a new repository is just a matter of providing a new CCIM if supported by the Software Component that implements the ICMM API.

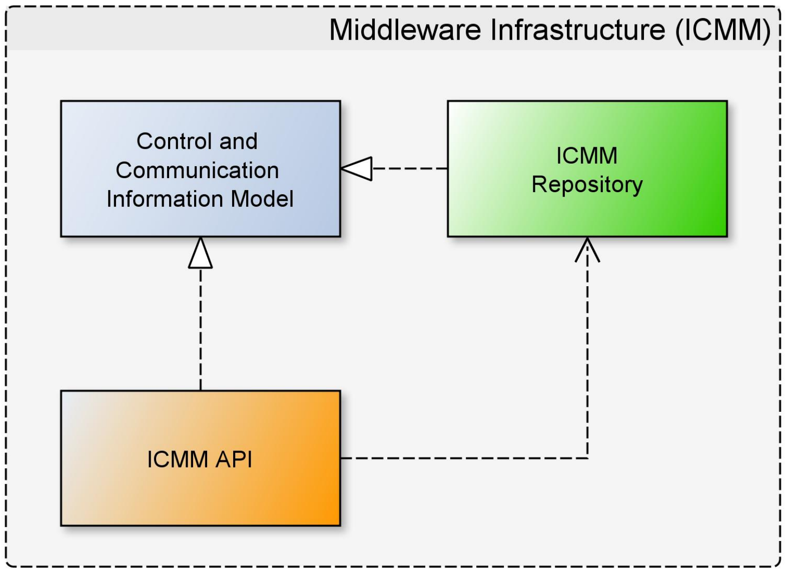

Middleware Infrastructure (ICMM)
Control and Communication Information Model
ICMM Repository
ICMM API

Figure 34: Control and Communication Information Repositories.

It is important to note, that the term ”resource oriented” in the context of the ICMM refers to the concept of generic resources as used in the specification of the Resource Oriented Architecture. An ICMM “resource” may represent Control and Communication Information on Simulation Models, Simulation Model wrappers or Simulation Cases. Therefore a clear distinction has to be made between the management of so called *Objects of Interest* (OOI) which represent abstractions of real world phenomena (e.g. capacity resources) and of “virtual” Control and Communication Information Entities. Of course the ICMM may also provide and manage Control and Communication Information describing CRISMA relevant aspects of OOIs. To avoid ambiguities with the term “resource” we call “resources” managed by the ICMM Control and Communication Information Entities or short entities.

In accordance with the publish-subscribe pattern the ICMM supports mechanisms to subscribe to events related to Control and Communication Information Entities, e.g. the state of a Simulation Model run (not started yet, running, finished, cancelled, ...). Other functionalities of the ICMM include the ability to save and restore the actual state of a Control and Communication Information Entity, to support the creation of catalogues and search functions.

3.6. Modular control and communication information model

As already mentioned in section 3.5, Control and Communication Information managed by the ICMM plays an important role for discovery, access, integration and interpretation of data and models. With the help of the ICMM and related Control and Communication Information Repositories, it will be possible to describe all aspects of a CRISMA Federation and to manage the Control and Communication Information and the actual data independently.

The ICMM manages Simulation Case information like basic configuration parameters, location and access to input data while the actual input data and results are provided by different components (DDS, WMS, SOS, RDMBS …). A dedicated repository which is part of the ICMM will store Control and Communication Information about data transformation adapters and Simulation Model wrappers or may even provide machine-processable transformation rules e.g. encoded in Extensible Style Sheet Language Transformations (XSLT) or even a scripting or programming language.

The actual content of those repositories is defined by modular Control and Communication Information Models. Modular in this context means that a generic Core CCIM exists which can be extended by user defined modules. The core model is defined by the Implementation Architecture and the modules by the different Application Architectures.

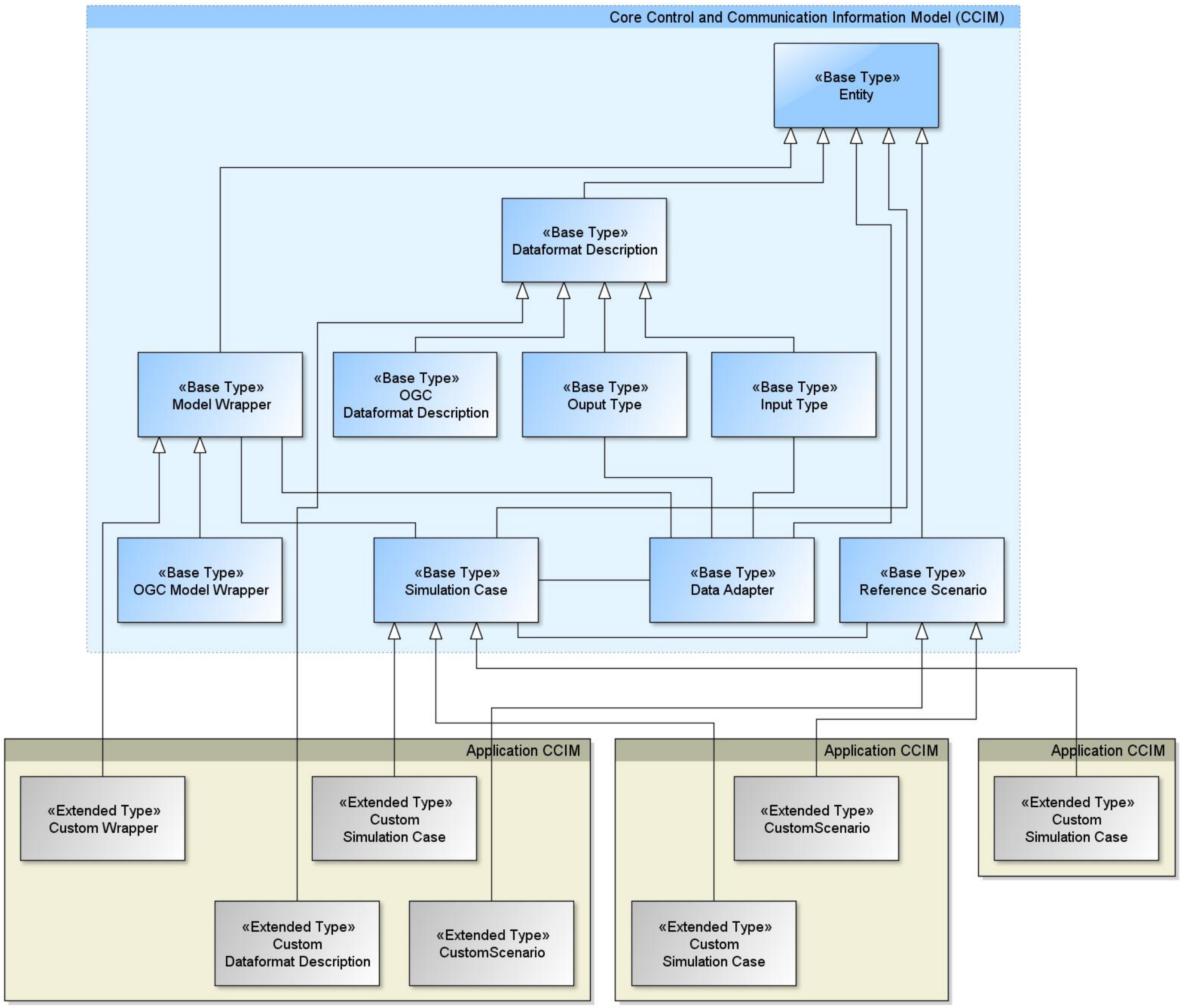

Core Control and Communication Information Model (CCIM)
«Base Type» Entity
«Base Type» Dataformat Description
«Base Type» Model Wrapper
«Base Type» OGC Dataformat Description
«Base Type» Ouput Type
«Base Type» Input Type
«Base Type» OGC Model Wrapper
«Base Type» Simulation Case
«Base Type» Data Adapter
«Base Type» Reference Scenario
Application CCIM
«Extended Type» Custom Wrapper
«Extended Type» Custom Simulation Case
«Extended Type» Custom Dataformat Description
«Extended Type» CustomScenario
Application CCIM
«Extended Type» CustomScenario
«Extended Type» Custom Simulation Case
Application CCIM
«Extended Type» Custom Simulation Case

Figure 35: Example of Modular CCIMs.

It is worth noting, that the Core CCIM does not only enable interoperability among different CRISMA Federations, it is also the basis for the development of generic and configurable Building Blocks (see section 3.7). This implies that generic Building Blocks that are designed to operate on basis of the core information model shall be adaptable to user defined modules e.g. by offering plug-in or extension mechanisms.

Figure 35 shows an example of a modular CCIM. It describes some core Control and Communication Information Entities which are then extended by the application specific models. The model shown in Figure 35 is a simplified example to illustrate the general concepts. The actual Core CCIM is described in the Implementation Architecture (chapter 5).

We foresee also that the ICMM reserves the possibility to import Control and Communication Information that originally resides in another system. Either an initial import or a continuous integration with direct access to the other systems data stores is supported. This kind of ”live” Control and Communication Information integration is especially required when a CRISMA Application and an Integrated System shall coexist and Control and Communication Information about objects of interest for the purpose of integration and interoperability has to be provided.

Among the main properties of the modular CCIM are

- The ability to support relationships between Control and Communication Information Entities like generalisation / specialisation, links, aggregations, and others;
- the ability to provide a (lifetime) unique reference for identifying (URN) and accessing (URL) a Control and Communication Information Entity; and
- the ability to support complex (hierarchical) Control and Communication Information Entities that are made of references to other entities.

In accordance to the RESTful principles supported by the ICMM REST API (section 3.5) references can be expressed as URIs and thus be resolved on demand.

3.7. Federates and integrated components

One fundamental concept of architectural design is to establish a consistent terminology which unambiguously defines the various elements of the architecture and explains their relationships. While most of the key terminology of the CRISMA Framework Architecture has already been used in the previous section and the glossary this section intends to give a formal definition of terms required to describe the concepts of a CRISMA Federation and to establish a link to the previously introduced architectural concepts. Figure 36 visualises the relationship between the different Federation related terms established by the CRISMA Framework Architecture (Federates, Building Blocks, Integrated Components, etc.) Please note that the dark blue boxes represent arbitrary examples of instances of the respective Federates. The Federation concept terminology is further extended by the CRISMA GUI Integration Approach which introduces additional types of Federated .and Integrated GUI Components. Those GUI related terms are defined in section 3.12.

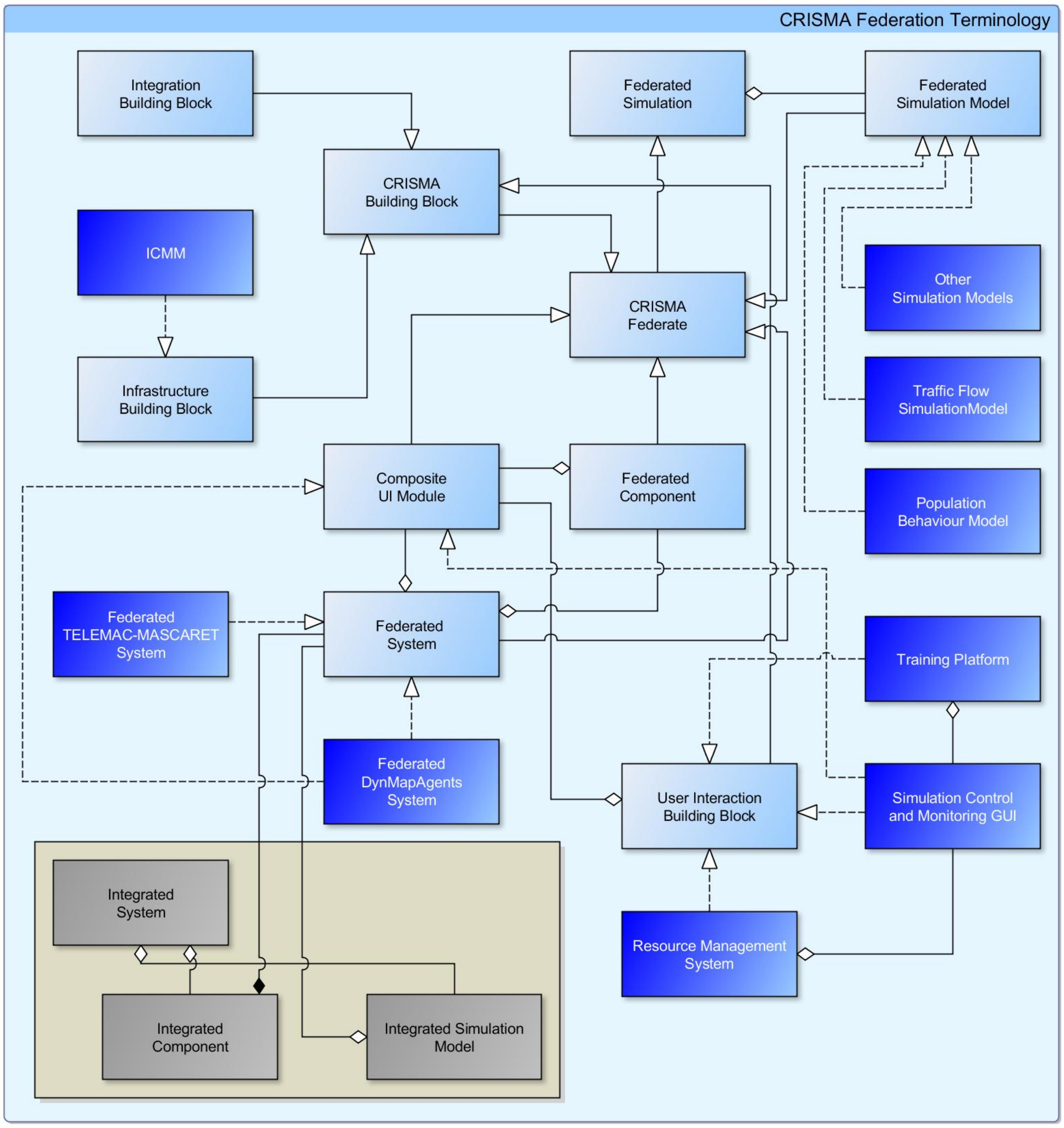

CRISMA Federation Terminology
Integration Building Block
Federated Simulation
Federated Simulation Model
CRISMA Building Block
ICMM
CRISMA Federate
Other Simulation Models
Infrastructure Building Block
Traffic Flow SimulationModel
Composite UI Module
Federated Component
Population Behaviour Model
Federated TELEMAC-MASCARET System
Federated System
Training Platform
Federated DynMapAgents System
User Interaction Building Block
Simulation Control and Monitoring GUI
Integrated System
Resource Management System
Integrated Component
Integrated Simulation Model

Figure 36: Terminology of the CRISMA Framework Architecture.

3.8. CRISMA federates

A CRISMA Federate is a Software Component (including CRISMA Building Blocks) that is able to interact with the Middleware Infrastructure and thus exchange Control and Communication Information with the ICMM. A CRISMA Federate may but does not have to expose a dedicated RESTful CRISMA API that is aligned to the APIs of the ICMM.

Accordingly, an existing Software Component that wants to join a CRISMA Federation and thus become a CRISMA Federate has to be aware of the ICMM API but does not necessarily have to expose such an API. This keeps the effort for integrating existing Software Components at a minimum (especially components with GUI). Other Federates, especially CRISMA Building Blocks need to be integrated more tightly. As a general rule they offer their programmatically accessible functionality through well-defined RESTful APIs. CRISMA Federates are explicitly allowed to communicate with non-CRISMA components (Integrated Components) e.g. a CRISMA-aware GIS Viewer (CRISMA Federate) is allowed to communicate with a non-CRISMA-aware Web Map Service (WMS) while the ICMM is not involved in this communication at all (see section 3.11). In summary, we can state that a component that is aware of the ICMM API and the Core CCIM and thus can interact with the Middleware Infrastructure to access Control and Communication Information Entities is called CRISMA Federate. In this spirit, we can also state that such a component is CRISMA-aware. Thus, the process of modifying or wrapping an existing system, a Simulation Model or a component to become a CRISMA Federate is called "making it CRISMA-aware".

3.9. CRISMA building blocks

We can distinguish between Federates, that are an integral part of the CRISMA Framework and other Federates that are only relevant in the context of a specific CRISMA Application. Federates that are provided by the CRISMA Framework are called **CRISMA Building Blocks**. CRISMA Building Blocks shall be generic, composable and configurable and should expose their GUI as CRISMA web based Composite UI Modules (see section 3.12). The generic and configurable aspects of a Building Block refer to its ability to operate both on the basis of the Core CCIM and application specific extensions (modules). To give an example, a generic Building Block for the configuration of Simulation Models may be designed according to the Simulation Model Control and Communication Information definition specified by the Core CCIM of the CRISMA Framework (section 3.6). Since a CRISMA Application may extend this model and provide an application-specific Simulation Model definition as part of its Application Architecture the implementation of the aforementioned Building Block shall be easily adaptable to a new model.

CRISMA Building Blocks can be categorized as **Infrastructure Building Blocks** (section 4.1), **User Interaction Building Blocks** (section 4.3) and **Integration Building Blocks** (section 4.2). End users of a CRISMA Application will directly interact with the User Interaction Building Blocks only. An example for an Infrastructure Building Block is the ICMM as the central Infrastructure Building Block and an example for an Integration Building Block is a Simulation Model Integration (Simulation Model Wrapper) Building Block.

3.10. The federation concept

The CRISMA Federation Concept explains the relationships between the different components that may become part of a CRISMA Application.

3.10.1. Federated components and federated systems

Application-specific Federates are also called **Federated Components**. In general, those Federated Components are existing Software Components which originally haven't been designed to be part of a CRISMA Application but have been modified, wrapped or extended to become CRISMA-aware. This term can also be extended to **Federated System** when several Federated Components act together. Hence, the difference between a Federated Component and a Building Block is that the Federated Component doesn't need to be generic and configurable and therefore it is not part of the CRISMA Framework but of a CRISMA Application only. Thus, in contrast to a Building Block, a

Federated Component may but does not have to be implemented as a transferable Software Component that can be reused in the context of a different location and domain.

3.10.2. Federated simulation models and federated simulations

Another type of Federate is the Simulation Model. Simulation Models can be wrapped into CRISMA-aware Federates and thus become **Federated Simulation Models**. Since Simulation Models may be specialized for a certain CRISMA Application or domain they are not considered Building Blocks. The concept of model integration in CRISMA is described in section 3.14. The term federated simulation is used to highlight that a federated simulation is composed of several distinct Simulation Models.

3.10.3. Integrated components, simulation models and systems

Any other component, Simulation Model or system that is not aware of the ICMM API and the Core CCIM but that takes part in an interaction of CRISMA Federates is called **Integrated Component**. The difference between a CRISMA Federate and an Integrated Component is that the Integrated Component is not CRISMA-aware. It has been integrated on a lower level e.g. by offering standardised services that are accessible by CRISMA Federates or allowing direct access to its internal data stores or public service interfaces (APIs). The same applies to integrated Simulation Models which are not integrated according to the Model Integration Approach presented in section 3.14. Their results however are made available to the system somehow. With help of the ICMM and a CCIM, a CRISMA Application may be able to establish a “CRISMA-aware view” on the data of an Integrated System without the need to interfere with the Integrated Systems itself (see section 3.6). In general, integrated components, Simulation Models and system don’t have to be modified or extended to become part of a CRISMA Application.

3.10.4. CRISMA federations and applications

As already explained in section 3.3, while a **CRISMA Federation** is composed of Federates only a **CRISMA Application** may additionally consist of Integrated Components, Simulation Models and Systems (Figure 37). This distinction is necessary to highlight the fact that it must be possible to build CRISMA Applications that are composed of CRISMA Building Blocks and legacy Software Components without the need to modify the legacy components. This is an important prerequisite to foster the integration of legacy systems (section 3.15). Thereby the loosely integrated Software Components are left entirely unchanged and are not made CRISMA-aware. In the case that a tighter integration is necessary Software Components may be changed or wrapped to become Federated Components (see section 3.10).

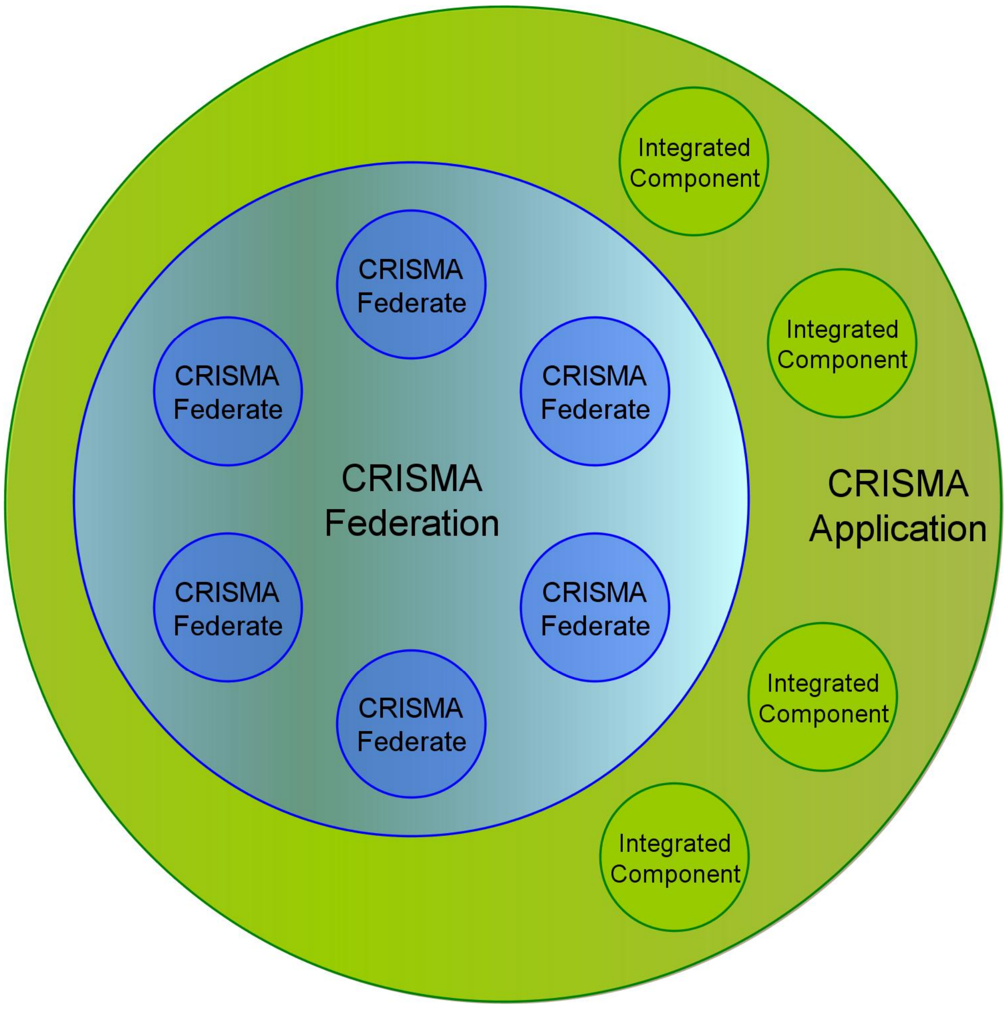


Figure 37: Relation between CRISMA Federation and CRISMA Application.

3.11. Separation of data and control and communication information flows

An important concept of the ICMM is the separation of data and Control and Communication Information flows. While all Control and Communication Information is routed through the ICMM, data can be directly exchanged between Federates and Integrated Components. For CRISMA Federates there are two binding rules regarding the flow of Control and Communication Information:

- Control and Communication Information is exchanged exclusively via the Middleware Infrastructure through the ICMM API.
- Control and Communication Information exchanged must be conformant to the core information model of the CRISMA Framework (Core CCIM).

The first rule guarantees that all Control and Communication Information is routed through the Middleware Infrastructure and thus auxiliary functions like event propagation, versioning, logging, access control, cataloguing, indexing, etc. are applied by the ICMM and related Infrastructure Building Blocks.

The second rule enforces interoperability between Federates on the Control and Communication Information level. All Federates can unconditionally rely on the Core CCIM and thus interoperate.

For data flows there are no strict rules but recommendations focusing on the data exchange services and standards. A list of recommended standards, their usage and applicability in the CRISMA Framework Architecture is provided in section 4.4.

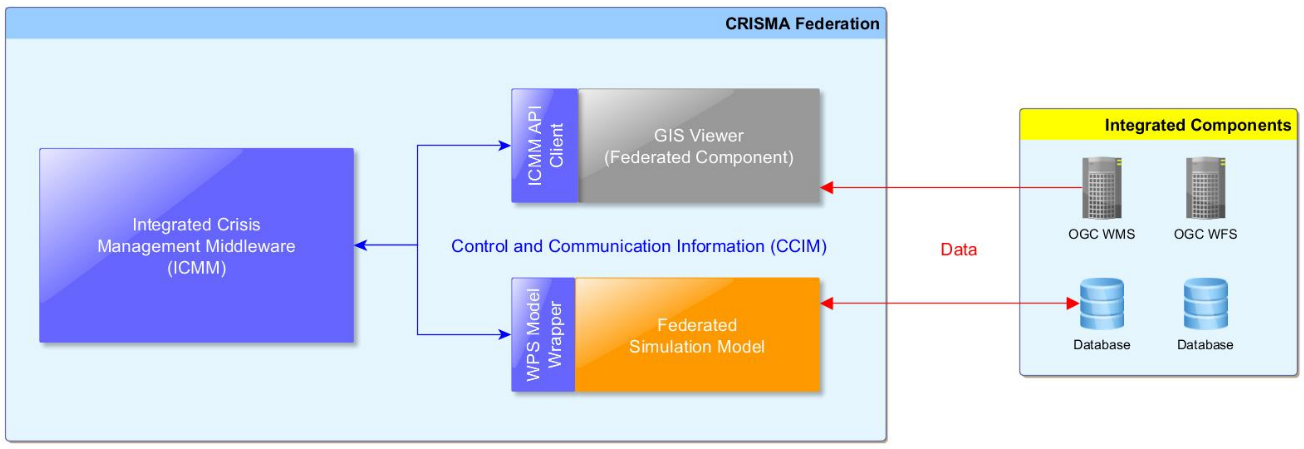

CRISMA Federation
Integrated Crisis Management Middleware (ICMM)
ICMM API Client
GIS Viewer (Federated Component)
Control and Communication Information (CCIM)
WPS Model Wrapper
Federated Simulation Model
Data
Integrated Components
OGC WMS
OGC WFS
Database
Database

Figure 38: Separation of Data Flows (Example).

Whether an information set is considered as data or Control and Communication Information depends on its purpose and usage context, therefore it is not always easy to distinguish the two. However, we can consider any information produced by Integrated Components (non-CRISMA-Federates) as data. A Web Map Service e.g. that is not related to the ICMM provides data (maps, features). A CRISMA-aware Simulation Model and GIS viewer e.g. may produce or consume both Control and Communication Information and data. In the example shown in Figure 38, the CRISMA-aware Federated Simulation Model stores its output (data) in a WMS-accessible database (peer to peer) and notifies clients via the ICMM API (publish/subscribe) that a new Simulation Model result is available.

3.12. GUI integration approach

The CRISMA Framework must provide a set of default composable GUIs which can be interactively combined to support specific workflows of CRISMA applications. However, potential CRISMA users already have powerful legacy applications potentially making use of CRISMA functionality as add-on. This is one reason why the CRISMA Framework does not aim to provide one overall and coherent GUI for all possible CRISMA Applications (CRISMA Federations). Instead, some CRISMA Applications will be realised as a conglomeration of disparate GUIs varying from separate and isolated desktop applications to HTML mashups. Those CRISMA GUIs need to consider different integration approaches. Existing applications or Building Blocks realised with help of existing Software Components already use their legacy user interfaces and client-side applications. Other potential clients would prefer to integrate CRISMA GUI components into their user interface and sometimes vendors of Simulation Models that come without any GUI might want to benefit from the CRISMA model integration GUI components.

For that reason CRISMA will introduce central CRISMA mashup user interfaces that are capable to work as a standalone user interface but that can also be consumed by external CRISMA-aware systems. The basic concept is to create web applications that expose CRISMA web based Composite UI Modules for arbitrary browser environments. Such web based Composite UI Modules can be embedded into any application GUI as explained in the next paragraph.

When considering embedding the GUI of an application in the GUI of another application there are several challenges such as technological barriers, interoperability and usability. CRISMA is targeting a markup based application (e.g. HTML5 and JavaScript) that can work standalone or embedded in a desktop application by using an appropriate web engine[1]. Such a pure web application GUI can be consumed by a desktop application or another markup based application. It can be embedded in a markup based HTML application with help of the concepts of HTML frames and browser plug-ins[2].

Exchange of data and events between two disparate GUI applications (tight integration) can become a challenging task as will be explained later. The Composite UI Modules will enable bidirectional communication (e.g. publish-subscribe) between the legacy application and other CRISMA components via those APIs and the CRISMA Middleware Infrastructure.

The main features of the composite CRISMA UI modules are:

- UI modules are standalone and depend on the CRISMA web application server side only.
- UI modules can share a common CRISMA state on the server side (with help of the ICMM).
- UI modules are integrated into the CRISMA security model and may require authentication and authorization in order to be initialized.
- UI modules are developed using pure web technologies, such as HTML, JavaScript and CSS in order to have a platform-independent support.
- UI modules are operating in the context of a specific user.

Example for potential Composite UI Modules:

- Run simulation Composite UI Module: Select the simulation and the relevant parameters (such as simulation type, initial state, simulations input and output parameters) and run the simulation (immediately, schedule based).
- Monitoring simulations lifecycle composite: enable to track all active simulations and understand a current state in the simulation lifecycle (such as pending, running, completed, failed, and aborted).
- Search simulation results Composite UI Module: Enable search in the simulation results catalogue based on keywords, parameters and hierarchical browsing.
- Simulation visualization composite UI Modules: Enable a view of simulation results in several visual representations, such as native result (for debugging and demonstrations), Tabular visualization, Geo map (per vendor ESRI, Google Maps …).
- Capacity resources management composite UI Modules: Enable the life-cycle management of OOI and other resources that are manipulated by the different models. Support operations such as Add\Modify new resources, Import resources from file, Search resources, View resources types and current state.

Since the GUI Integration Approach is a fundamental concept for the non-intrusive integration of the CRISMA results into legacy system and vice versa (see section 3.15), a

[1] Examples for web engines are MSHTML, Mozilla Gecko, WebKit

[2] Examples are ActiveX and Java applets

comprehensive set of rules for the design and development of Composite UI Modules has been defined. The fist and most important rule in this regard is, that

> **Each Composite UI Module shall be realised as distinct website which consists of a static part (HTML/CSS + other static resources) defining layout and structure of the user interface and a dynamic part (JavaScript) providing dynamic content and interaction functionalities.**

Additional rules are needed to ensure technical interoperability and composability, especially regarding the UI Integration Platform and the UI Mashup Platform. The complete set of rules for composite UI Modules is presented in the following.

There are few technical requirements that set the boundaries for the development of Composite UI Modules and the Integration Platform Building Block. Of course, since composite UI Modules and the UI Integration Platform are by definition CRISMA Federates, all technical and interoperability requirements on CRISMA Federates apply also to Composite UI Modules and the UI Integration and Mashup Platforms. Technically, and as shown in Figure 39,

> **Composite UI Modules must be implemented in HTML5, CSS3 and JavaScript.**

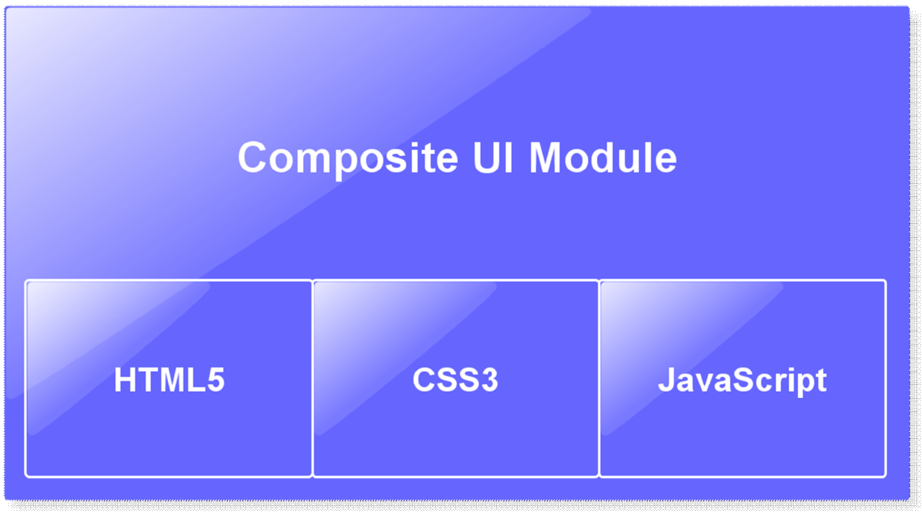


Figure 39: Composite UI Module Overview.

Accordingly, we can also define what Composite UI Modules are not made of:

> **Composite UI Modules must not rely on any type of external plug-ins like Oracle Java, Adobe Flash, Microsoft Silverlight, and others.**

Additionally, Composite UI Modules must be standalone and self-contained, which means they must not rely on a certain server infrastructure in the backend to function properly.

> **Composite UI Modules shall be realised as purely client-side and server-agnostic solutions that do not require any specific backend server-side infrastructure (e.g. enterprise application server, servlet container, portal server, ...), services or components apart from the Infrastructure services Building Blocks provided by the CRISMA Framework (e.g. ICMM, UI Mashup Platform for Mashable Composite UI Modules, etc.).**

More precisely, a Composite UI Module is a static HTML page and its dynamic behaviour is realised entirely by HTML5 and AJAX techniques in conjunction with CRISMA APIs (e.g. the ICMM APIs). Techniques that rely on server-generated dynamic content like Portlets, Servlets, PHP Scripts, etc. are not suitable for the implementation of Composite UI Modules. Moreover, a Composite UI Module should also operate properly when loaded from the local file system involving no HTTP web server whatsoever.

Furthermore, and especially in regard to security sensitive information exchanged within a CRISMA Application, the usage of proprietary and vendor-specific APIs (e.g. Google APIs, Yahoo APIs, ...) by a Composite UI Module is strongly discouraged. Although not a rule, the following recommendation is given:

> **Composite UI Modules should not rely on proprietary APIs or services that pose the risk of exposing security and privacy sensitive information to third parties.**

Whether such a service or API is save to be used has to be decided on a case by case basis by the architects and developers of the respective CRISMA Application. While it might be o.k. to use the Google Maps API for displaying background maps, storing sensitive information in cloud services like Dropbox[3] or Google Drive[4] might be in violation to data protection rules or non-disclosure agreements.

Since Composite UI Modules may also provide functionalities that can be called by other Composite UI Modules or even other types of components, there is a need for a well defined API exposed by each Composite UI Module. However, in contrary to the initial GUI integration concept presented in the first version of this Architecture specification (D32.1, 2013) this API should not exclusively be realised as RESTful API. Since pure HTML components are not able to offer such a RESTful API it would have to be provided by a separate server side component. The initial Composite UI Modules interaction concept shown in Figure 40 has therefore been extended by a more flexible approach which is based on JavaScript and ICMM APIs.

According to Figure 40, Composite UI Modules should be split into a client-side and a server side part. The server-side part would expose a REST API that accepts incoming calls from other components which want to call functions of the Composite UI Module.

Thus, besides the creation of HTML, CSS and JavaScript snippets the development of a Composite UI Module would also include the implementation of a RESTful service that can be deployed on a web server. Since the implementation of such an additional server-side component would place an unnecessary burden on developers, integrators and maintainers of a CRISMA Application, the second iteration of the CRISMA Framework Architecture introduces an improved interaction concept for Composite UI Modules.

However, one must realise that in some cases a server-side component for encapsulating the business logic of a Composite UI Module might be required. For those cases, either an ICMM Actions API or an Extension API can be used to outsource time and resource demanding tasks (see Figure 48). In consequence, we can state that

[3] https://www.dropbox.com

[4] https://drive.google.com/

- it is o.k. to split a Composite UI Module into a server-side and a client-side part when the role of the server-side part is to encapsulate business logic of the Composite UI Module; and
- it is not o.k. to split a Composite UI Module into a server-side and a client-side part when the role of the server-side part is to provide an endpoint for other components to communicate and interact with the Composite UI Module.

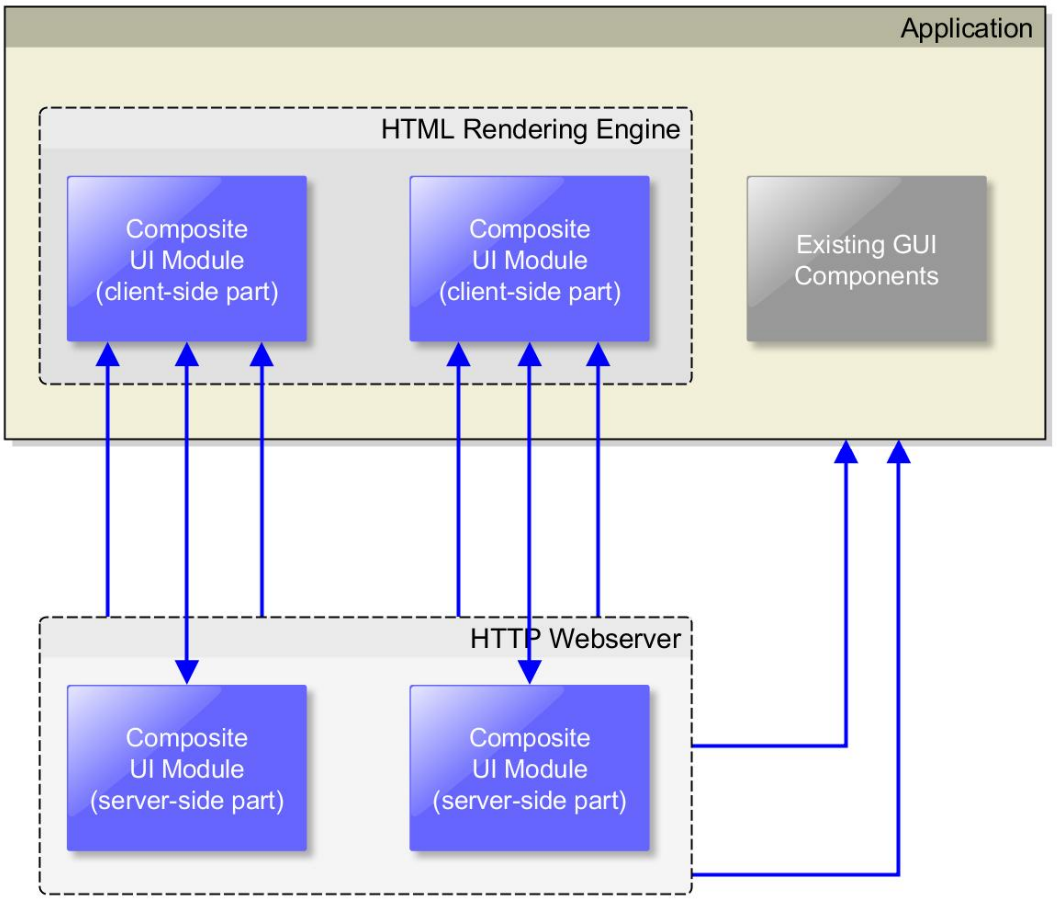

Application
HTML Rendering Engine
Composite UI Module (client-side part)
Composite UI Module (client-side part)
Existing GUI Components
HTTP Webserver
Composite UI Module (server-side part)
Composite UI Module (server-side part)

Figure 40: Initial Composite UI Modules Interaction Concept.

For the direct communication and interaction with Composite UI Modules a new concept has been introduced. This concept is based on the fact that web pages are able to call JavaScript functions of other web pages acting in the same scope (e.g. in the same web browser window). Furthermore, a HTML/JavaScript engine (e.g. a web browser) is also able to directly call JavaScript functions of the web pages it is rendering. Therefore, a further rule for Composite UI Modules can be defined:

> **Composite UI Modules shall expose their programmatically accessible functionality via a JavaScript API.**

However, in some cases a component that wants to interact with a Composite UI Module may not have the possibility or capability to directly invoke JavaScript functions. In this case, an alternative possibility is provided by the ICMM. The ICMM Action and Event Delegation APIs can be used to forward arbitrary action requests to a Composite UI Module that is registered with the ICMM. Figure 41 shows how different types of components can interact with the Composite UI Modules' JavaScript API.

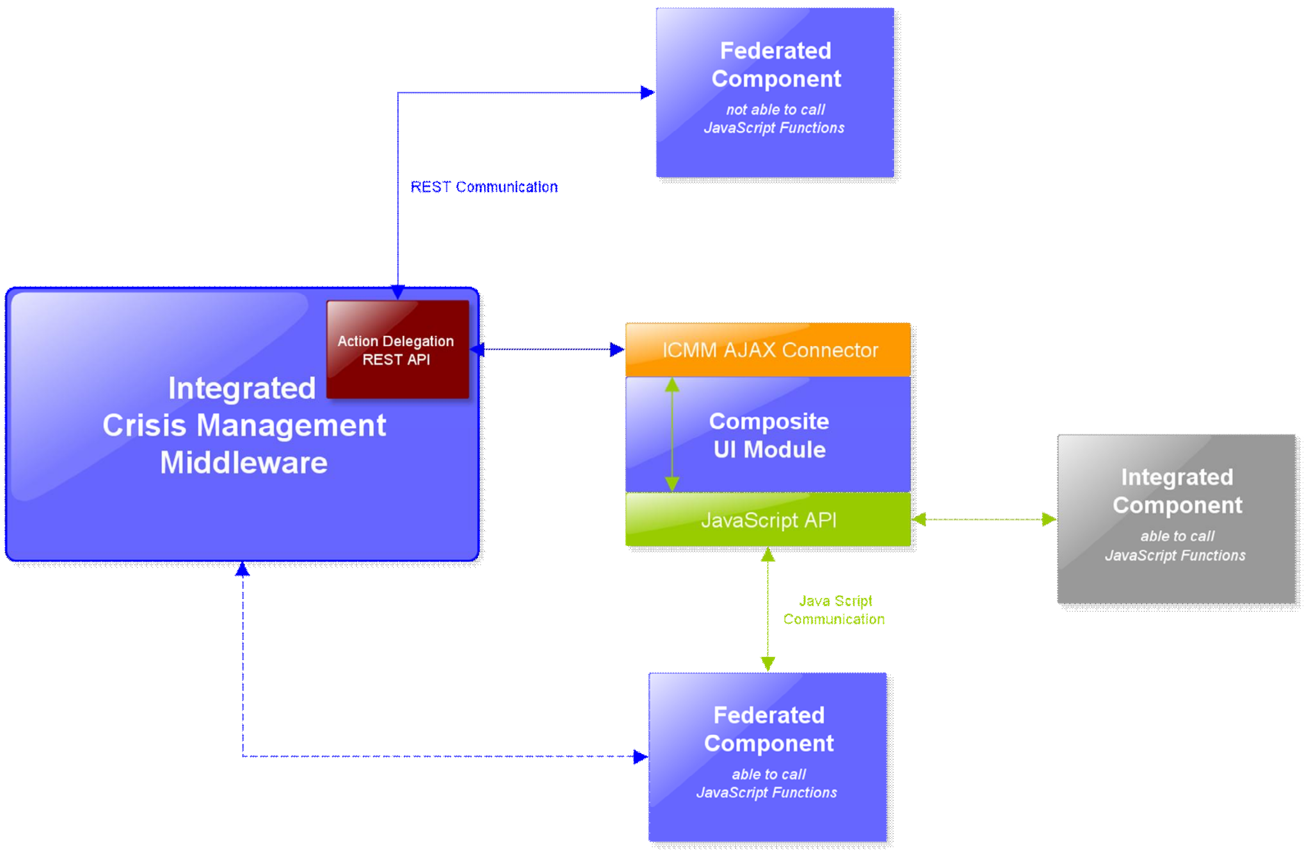

Federated Component
not able to call JavaScript Functions
REST Communication
Action Delegation REST API
Integrated Crisis Management Middleware
ICMM AJAX Connector
Composite UI Module
JavaScript API
Integrated Component
able to call JavaScript Functions
Java Script Communication
Federated Component
able to call JavaScript Functions

Figure 41: Improved Composite UI Modules Interaction Concept.

Depending on its capabilities a Federated Component may either use the mandatory Composite UI Modules' JavaScript API or the ICMM Action REST API to interact with the Composite UI Module. An Integrated Component which is not aware of any ICMM API may only interact with the Composite UI Modules through the JavaScript API.

Please note that in this case Composite UI Module and Integrated Component must reside in the same scope, e.g. the same web page or the same web browser window (or frame).

An important prerequisite for communicating via the optional Action API is the ability of the Composite UI Module to accept and answer incoming requests sent by a web server. In principle, there exist two techniques for realising this kind of server induced communication with HTML/JavaScript based web clients: WebSockets and Server-Sent Events (SSE). While WebSockets natively support bidirectional communication through a two-way communication channel SSE connections can only push data from the web server to the client. However, there exist client- and server-side frameworks like the Atmosphere Framework[5] which provide transparent bidirectional client-server communication regardless of the underlying transport technique. Even fallback transport like long-polling can be enabled if the client cannot accept incoming requests at all. In any case, bidirectional communication between the ICMM and CRISMA Federates is a key requirement for event notifications (Event API) and action requests (Action API). Therefore also this requirement applies to Composite UI Modules:

> **As any other Federate, Composite UI Modules shall support <u>bidirectional communication</u> with the ICMM. Specifically, Composite UI Modules must be**

[5] https://github.com/Atmosphere/atmosphere

able to receive action requests or event notifications dispatched by the ICMM. Thereby, appropriate transport techniques like WebSockets, Server-sent Events or long-polling shall be supported by a Composite UI Module.

It is important to note that the ICMM Action API is a generic API for executing arbitrary actions in the context of the ICMM and each action has to be implemented for each Composite UI Module communication. Moreover, the approach of action delegation though the ICMM is considered as workaround when no JavaScript Communication with Composite UI Modules is possible.

The ICMM Action API can also be used in conjunction with the Event Delegation API to forward action requests to Composite UI Modules. The Event Delegation API is realised by the Publish/Subscribe Context Broker Infrastructure Building Block. The Publish/Subscribe Context Broker is used by the ICMM to delegate events to CRISMA Federates. Thereby, a CRISMA Federate can be notified once a new CCIM entity of a specific type is created, deleted or updated in the respective ICMM repository. However, the Publish/Subscribe Context Broker can also be triggered by the Action API to delegate action requests to respective Federates.

Another general rule for Federates that has to be slightly extended when applied to Composite UI Modules is the rule for self-describing components:

A Composite UI Module has to provide a self-description that informs about its capabilities. Specifically, this self-description has to be provided as JSON-encoded CCIM and has to contain general descriptive and technical meta-information as well as a description of the Composite UI Modules' JavaScript API.

Having formulated those basic rules for the realization and behaviour of Composite UI Modules we can now discuss how Composite UI Modules can be embedded or composed into whole CRISMA Applications. Ideally, the GUI of a CRISMA Application is composed entirely of Composite UI Modules. However, the GUI integration approach of the CRISMA Framework Architecture takes also into account that a CRISMA Application may be based upon existing non-CRISMA-aware GUIs and possibly also a mixture of Composite UI Modules and existing GUIs. Therefore the concepts and Building Blocks, respectively, of the UI Integration and UI Mashup Platform were introduced.

The UI Integration Platform is a client-side component for displaying Composite UI Modules. A technical requirement and basic rule for this UI Integration Platform is:

The UI Integration Platform must be able to render and execute Composite UI Modules. Thus, it must support HTML5, CSS3 and JavaScript.

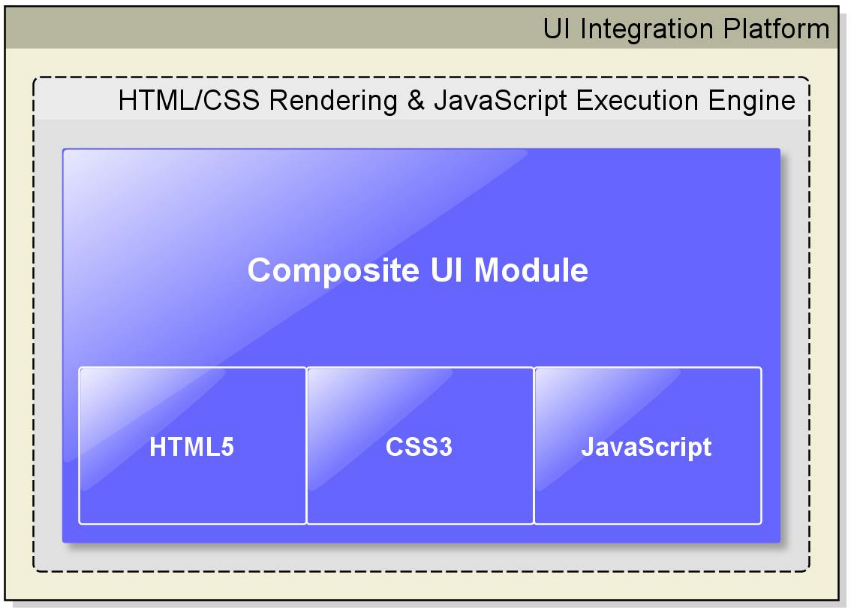


Figure 42: UI Integration Platform Basic Overview.

Technically, this basic requirement is met by most modern web browsers as they implement the respective rendering and execution engines (Figure 42).

In consequence, the GUI of a CRISMA Application may be realized as browser-based rich internet application (RIA). Nevertheless, additional requirements on the UI Integration Platform arise when Composite UI Modules shall be able to interact with other types of existing GUIs or even be embedded into existing GUIs. Before going further into detail, additional aspects when building RIAs have to be considered first.

Those are the aspects of modularity and composability. Composite UI Modules are self-contained and modular by itself. As mentioned before different Composite UI Modules must possibly be composed to form one whole application.

Although there are several techniques and frameworks for building modularized RIAs, an overall concept and a separate Building Block that seamlessly fits into the CRISMA Framework Architecture had to be defined – the UI Mashup Platform. Basically, the UI Mashup Platform is a server-side layout container and execution environment for interactively composing (mashing up) HTML widgets (Figure 43).

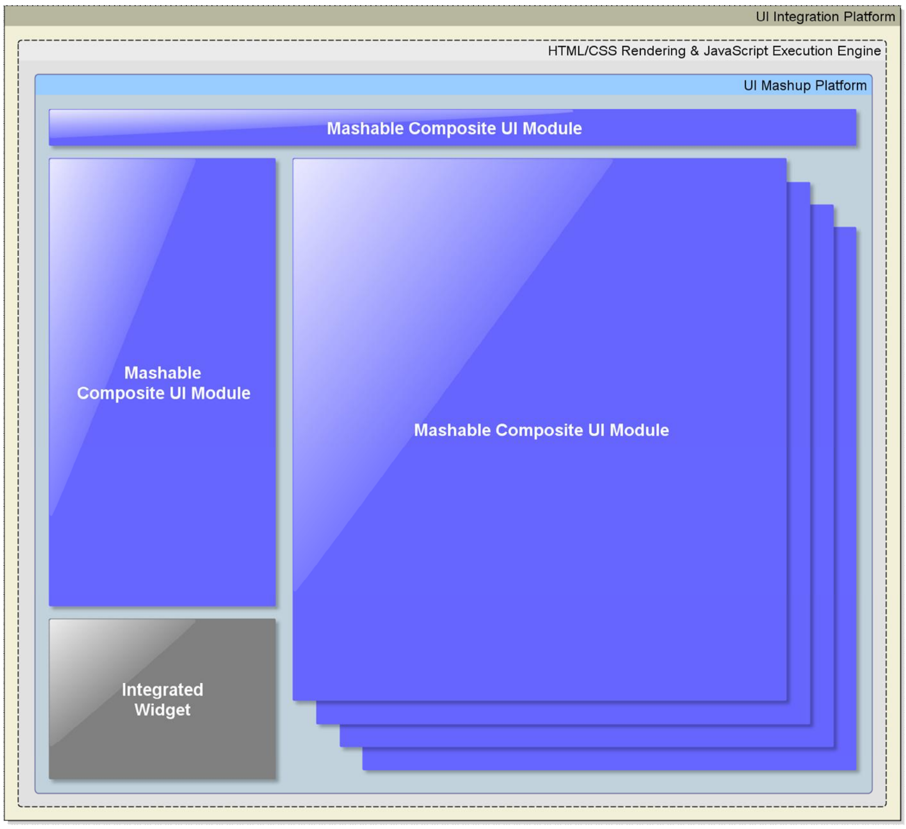

UI Integration Platform
HTML/CSS Rendering & JavaScript Execution Engine
UI Mashup Platform
Mashable Composite UI Module
Mashable Composite UI Module
Mashable Composite UI Module
Integrated Widget

Figure 43: UI Mashup Platform Basic Overview.

The UI Mashup Platform provides auxiliary functions exposed as JavaScript APIs, further one referred to as Mashup APIs. Among others, this Mashup APIs can be consumed by widgets to leverage communication with widgets among each other or with the platform itself. Following the concept of CRISMA-aware (federated) Components that are able to interact with the CRISMA Middleware Infrastructure (ICMM) through its RESTful API we introduce the concept of "Mashable" Composite UI Modules that are aware of the Mashup JavaScript APIs.

> **A Mashable Composite UI Modules must be able to interact with the UI Mashup Platform or other Mashable Composite UI Modules through the Mashup JavaScript APIs which are provided by the UI Mashup Platform.**

Likewise, the general rule for the UI Mashup Platform is:

> **The UI Mashup Platform shall support the interactive composition of Mashable Composite UI Modules and Integrated Widgets to rich internet applications.**

Interactively means, that a graphical designer tool for creating the overall application layout and for connecting ("wiring") widgets should be provided by the UI Mashup Platform. Otherwise there would be no real benefit compared to an ordinary RIA framework.

Mashable Composite UI Modules shall also inform the UI Mashup Platform about their composability with other widgets. This rule is already covered by the general rule of self-description. However, the format of the self-description (capabilities) expected by a particular UI Mashup Platform (e.g. XML Widget Description Language[6], portlet.xml, etc.) may be different from the capabilities JSON document provided by an ordinary Composite UI Module. Also the interaction between widgets may have to be realized with help of UI Mashup Platform specific methods instead of the JavaScript API and the ICMM Action API. These are potential conflicts; therefore the distinction between Composite UI Modules and Mashable Composite UI Modules and their relationships to the UI Integration and the UI Mashup Platform respectively is an important aspect to be considered.

While a Composite UI Module may exist independently of a particular UI Integration Platform (e.g. not bound to a specific web browser or web server), a Mashable Composite UI Module is always depended on a certain UI Mashup Platform (e.g. a portal server like Liferay or an actual widget mashup platform like Wirecloud).

The CRISMA Framework intends to deliver generic and composable Building Blocks. However, it is not guaranteed that every CRISMA Application will use the UI Mashup Platform Building Block or a specific implementation of this Building Block. Therefore, the following rule is introduced:

> **Mashable Composite UI Module should be based upon Composite UI Modules which can be used independently of a particular UI Mashup Platform in a CRISMA Application.**

In consequence, a Mashable Composite UI Module should be implemented first as (non-mashable) Composite UI Module and then wrapped into a Mashable Composite UI Module (Figure 44). How this wrapping is performed depends on the particular UI Mashup Platform Implementation. The UI Mashup Platform Implementation of the Implementation Architecture of the CRISMA Framework is the Wirecloud Platform. In principle, functionalities required by the enclosing UI Mashup Platform for Mashable Composite UI Modules, e.g. for being able to connect (wiring) widgets, should be mapped to the JavaScript API of the Composite UI Module.

> **Communication and Interaction between Mashable Composite UI Modules among each other and between the UI Mashup Platform should be based upon functions exposed by the JavaScript API of wrapped Composite UI Modules.**

[6] http://conwet.fi.upm.es/docs/display/wirecloud/XML+Widget+Description+Language

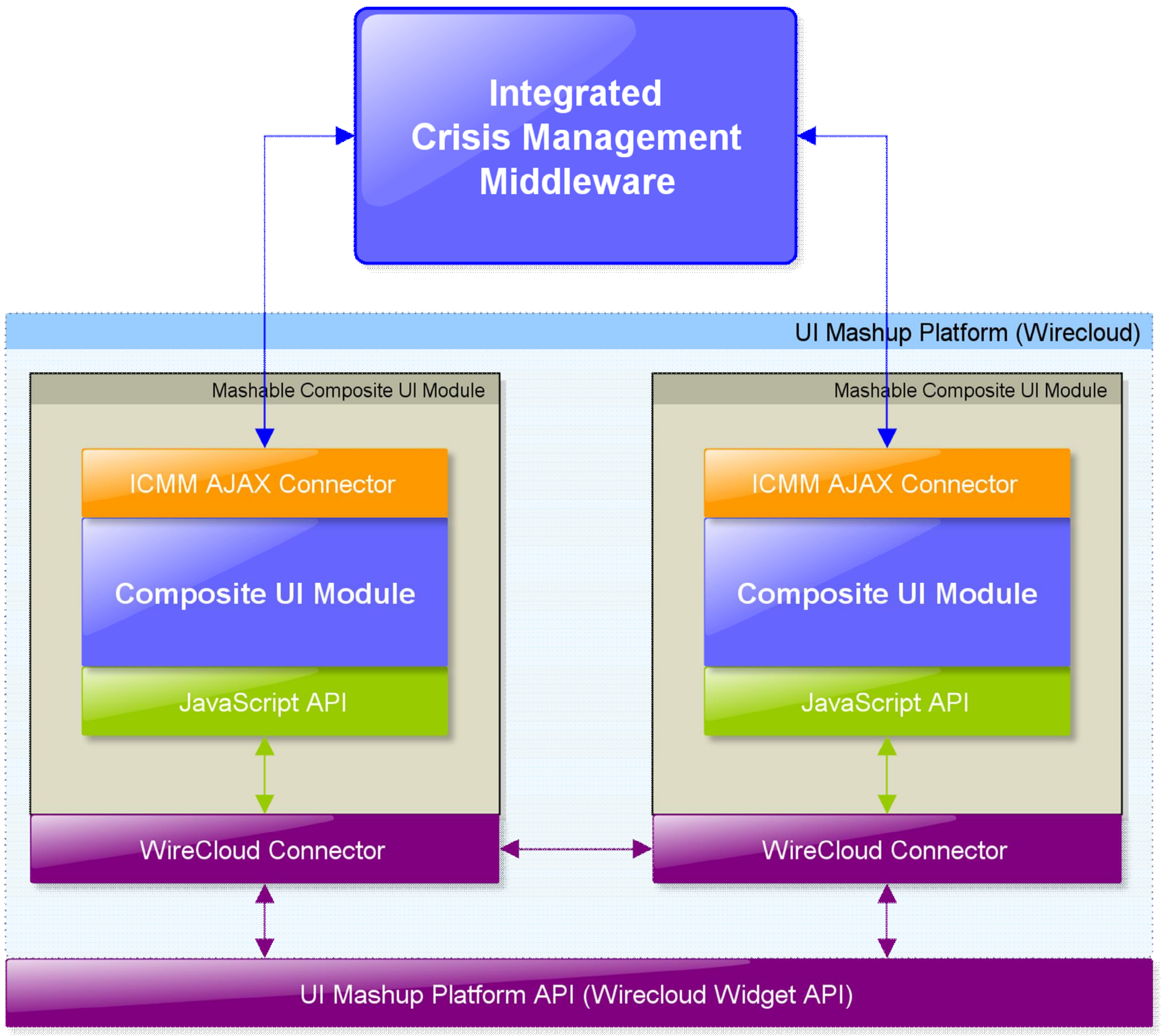

Integrated Crisis Management Middleware
UI Mashup Platform (Wirecloud)
Mashable Composite UI Module
ICMM AJAX Connector
Composite UI Module
JavaScript API
WireCloud Connector
Mashable Composite UI Module
ICMM AJAX Connector
Composite UI Module
JavaScript API
WireCloud Connector
UI Mashup Platform API (Wirecloud Widget API)

Figure 44: Wrapping of Composite UI Modules into Mashable Composite UI Modules.

Figure 45 brings the concepts of the UI Integration and the UI Mashup Platform into relation to the general terminology of the CRISMA Framework Architecture (section 3.7).

Interestingly, the UI Mashup Platform may also be used to mash-up Composite UI Modules with other types of widgets. Following the definition of Integrated Components which are not CRISMA-aware but can be integrated into a CRISMA Application, we call such widgets Integrated Widgets.

It is important to note, that the Mashup JavaScript API is by no means a replacement of the RESTful ICMM API but an optional extension that leverages interactive composability (shown also in Figure 44). Mashable Composite UI Modules are by definition CRISMA Federates and as a matter of course all rules and restrictions (see chapter 5.4.2) especially regarding the exchange of CCIM with the ICMM, apply also to Composite UI Modules.

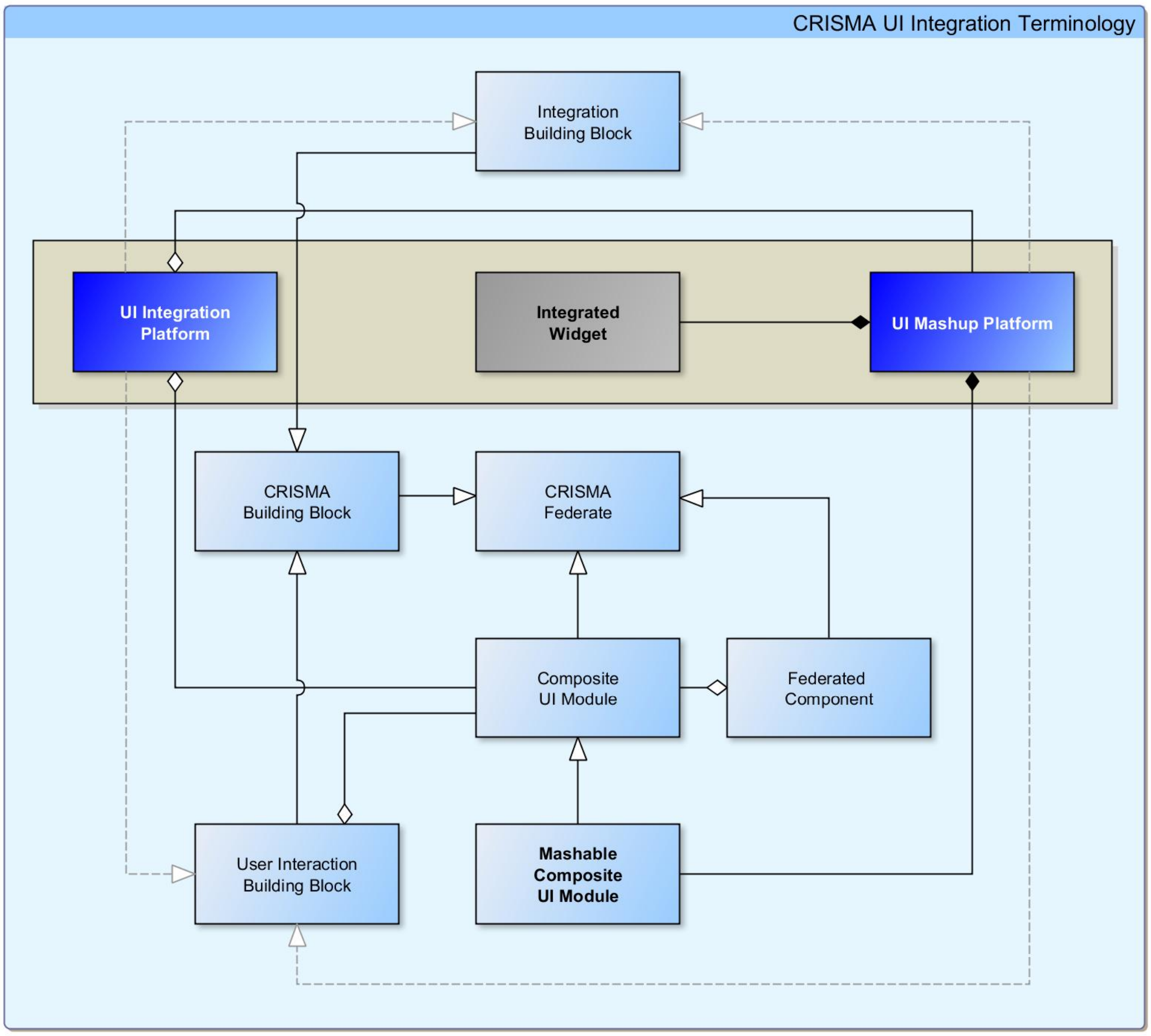


Figure 45: UI Integration Terminology.

Also notably in Figure 45 is, that the realization relationships between UI Integration Platform and UI Mashup Platform, respectively, to the Integration and User Interaction Building Blocks are shown greyed out to highlight a weak relationship. Whether the UI Integration Platform or the UI Mashup Platform have to be realized as Building Block and thus as CRISMA Federates depends on the design of a concrete Application Architecture.

An Application Architecture that defines one central GUI which is based on the UI Mashup Platform and the widgets contained therein does not necessarily need a sophisticated federated UI Integration Platform. Since the UI application logic is already covered by the mashed up RIA, a simple web browser will be able to play the role of the UI Integration Platform.

In the opposite case, when a Composite UI Module shall be integrated into an existing application, the UI Integration Platform should be able to support interaction and communication between the existing application and the Composite UI Module. Preferably via the Composite UI Module's JavaScript API as defined previously (Figure 46).

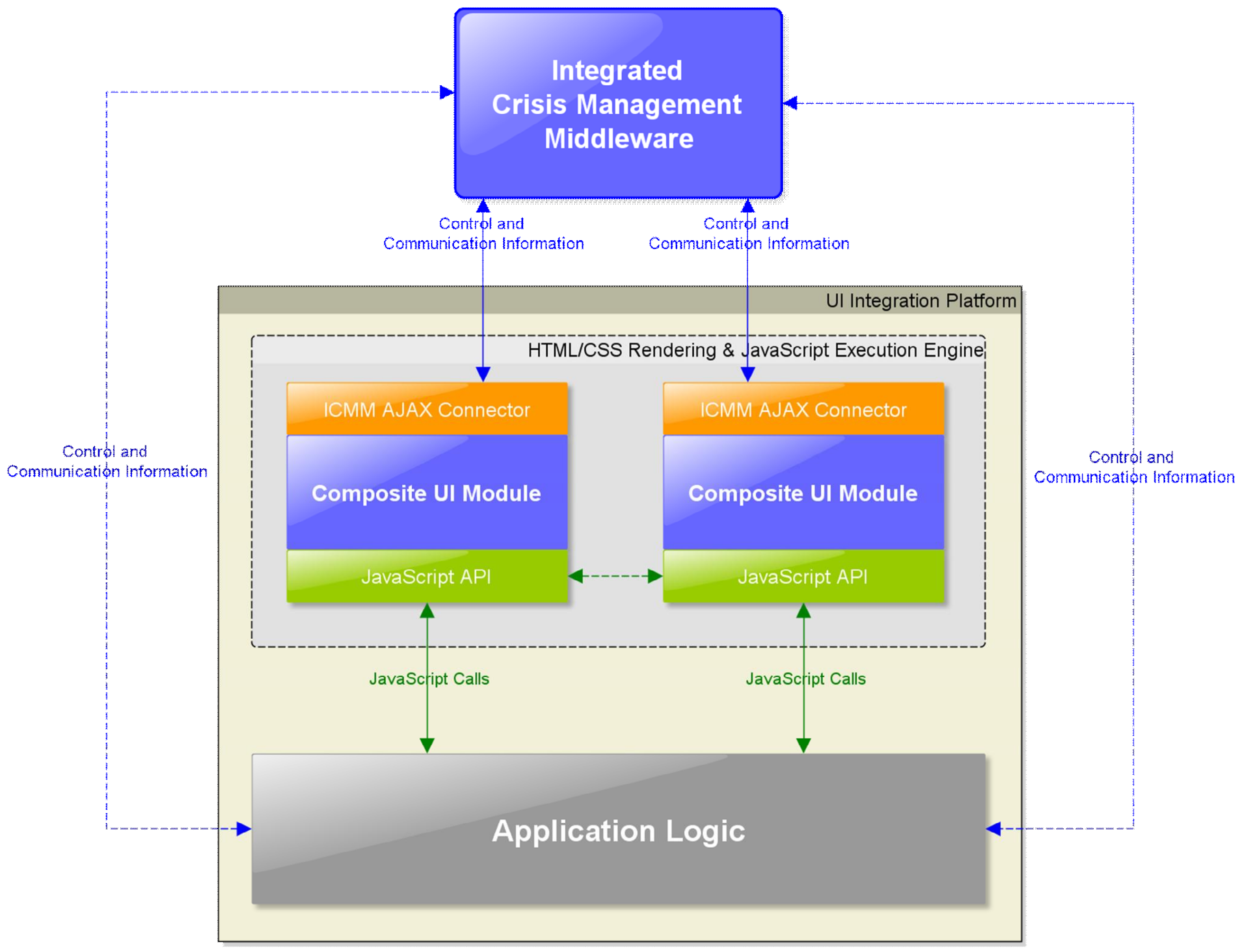

Integrated Crisis Management Middleware
Control and Communication Information
Control and Communication Information
Control and Communication Information
Control and Communication Information
UI Integration Platform
HTML/CSS Rendering & JavaScript Execution Engine
ICMM AJAX Connector
Composite UI Module
JavaScript API
ICMM AJAX Connector
Composite UI Module
JavaScript API
JavaScript Calls
JavaScript Calls
Application Logic

Figure 46: UI Integration Platform interacting with Composite UI Modules.

Furthermore, when several Composite UI Modules shall be integrated into an existing application the UI Mashup Platform should be used to assemble them into one RIA. Consequently there are several different possibilities for UI Integration and Interaction (Figure 47).

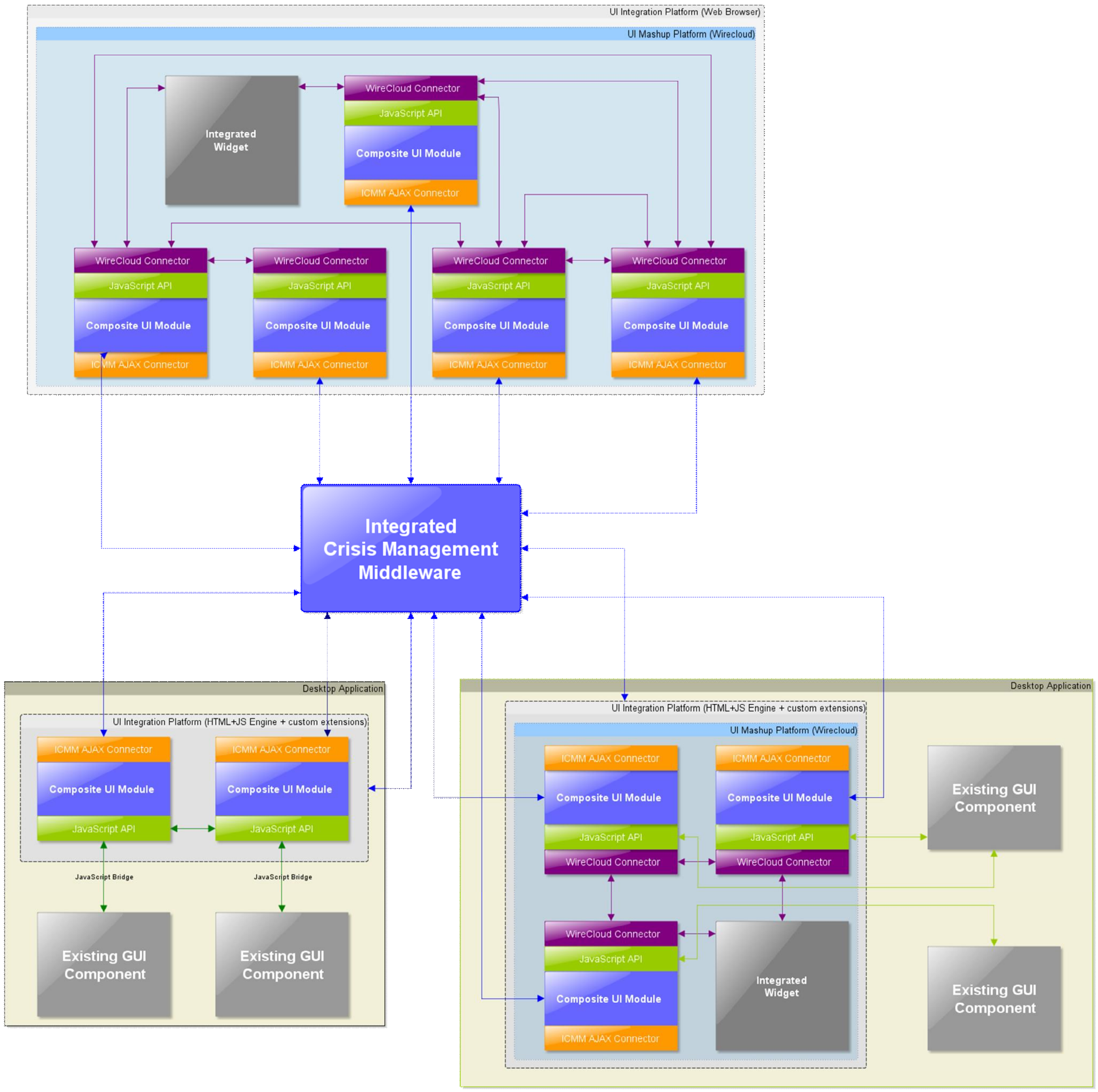

UI Integration Platform (Web Browser)
UI Mashup Platform (Wirecloud)
Integrated Widget
WireCloud Connector
JavaScript API
Composite UI Module
ICMM AJAX Connector
Integrated Crisis Management Middleware
Desktop Application
UI Integration Platform (HTML+JS Engine + custom extensions)
JavaScript Bridge
Existing GUI Component

Figure 47: UI Integration and Interaction Possibilities.

Another important topic that has already been mentioned but not yet been considered in detail is the fact that web applications should be optimised for responsiveness. Thus, load times, server communication, complex processing of data, etc. should be kept at a minimum. Some CRISMA Building Blocks realised as Composite UI Modules may however require complex business logic. Especially Building Blocks dealing with World State Transition and Simulation Model execution possibly need to transfer and process large amounts of data and support fault tolerant and recoverable transactions.

Thus, pure web applications are not suited for demanding and long-running processes like managing Transitions and starting and monitoring model executions. The business logic for such processes should therefore be executed by a server-side component and exposed via dedicated APIs.

The business logic of Composite UI Modules that need to perform demanding and long-running processes should be encapsulated in a dedicated server-side component.

There are two possibilities for realising such a server-side component. The first possibility has already been presented in the context of UI Module communication through action delegation (Figure 41). It is based on the ICMM Action API. The Action API is a generic extension point of the ICMM to execute arbitrary actions in the context of the ICMM. Being a core API of the ICMM the Action API is rather generic and basically designed to support arbitrary asynchronous tasks.

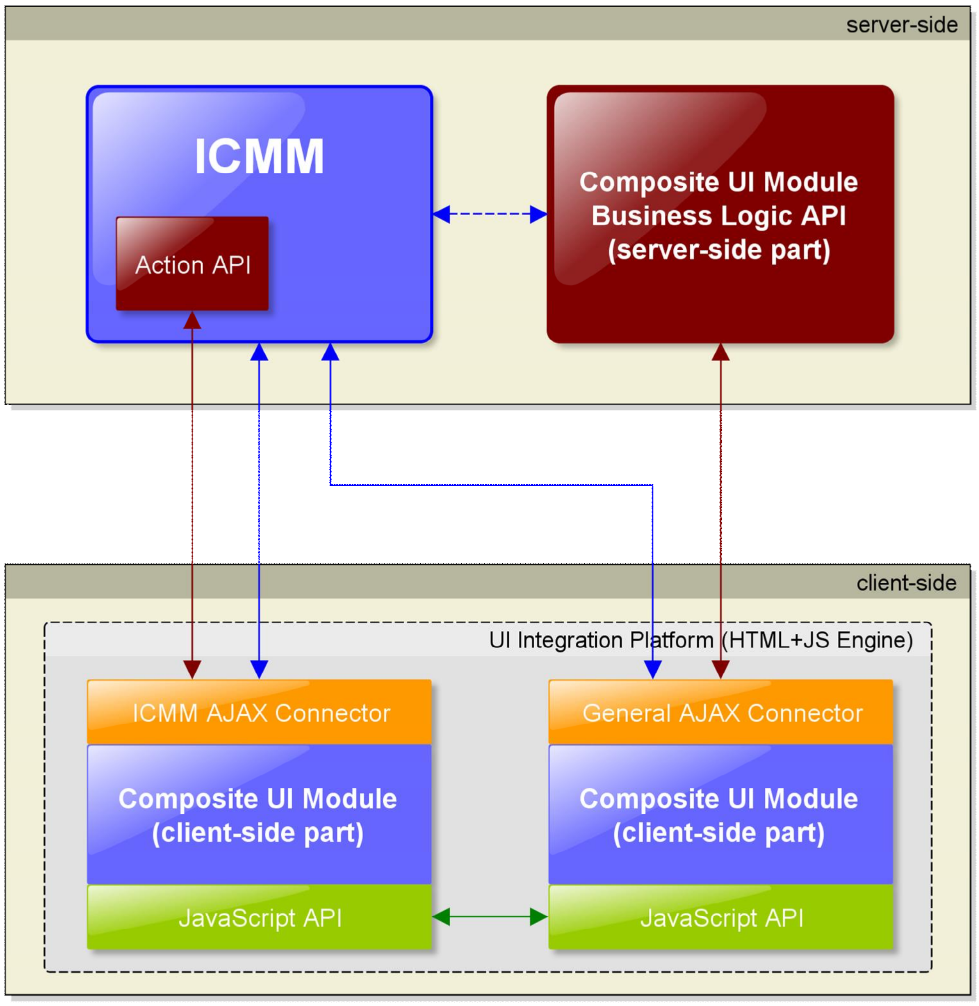

server-side
ICMM
Action API
Composite UI Module Business Logic API (server-side part)
client-side
UI Integration Platform (HTML+JS Engine)
ICMM AJAX Connector
Composite UI Module (client-side part)
JavaScript API
General AJAX Connector
Composite UI Module (client-side part)
JavaScript API

Figure 48: Composite UI Modules Server-Side Business Logic.

If a more fine grained service interface is needed, the server-side component should be realised by a dedicated RESTful Web Service. Of course, this server-side component should also follow the rules for Building Block Implementations and RESTful APIs. Apart from that, no further rules or guideless are given since the realisation of this component depends on the business logic of the respective Building Block.

In conclusion, plug-in interfaces and open standards based (web) service interfaces are the most suitable means to support integration of CRISMA functionality in legacy GUI applications across hardware and software-boundaries. For newly developed user interfaces, Mashup Interfaces are adequate, as they allow to reuse widgets and to combine individual user interfaces in order to support varying application contexts in the CRISMA pilot sites.

Furthermore, the following recommendations regarding user interfaces apply for the implementation of functional Building Blocks and the integration of legacy applications with GUIs: Although the GUI of a CRISMA Building Block should be realised by Composite UI Modules, this might not always be possible depending on the Software Component used to implement the functional Building Block. However, an existing Software Component should be wrapped into a Composite UI Module if technologically feasible. CRISMA Building Blocks that cannot be wrapped into a Composite UI Module due to technological barriers as they are based on existing software should provide a plug-in-interface or any other means to integrate generic Composite UI Modules provided by other Building Blocks.

3.13. Data integration approach

The preferred method for direct data transfer between Federates (see section 3.11) is based upon open standards and services such as those of OGC (see section 4.4). Therefore, the CRISMA Framework must provide respective Integration Building Blocks that leverage standards based access to a wide variety of different data sources.

Furthermore, besides the mere access to data the CRISMA Framework has also to support mechanisms that leverage a harmonisation of different data models. However, as observed in section 3.6, the transformation of disparate data models into one federation-wide data model is unrealistic due to the diversity of formats and information schemas used in legacy systems. Instead, the data integration approach followed in CRISMA is based on open standards, lateral coupling as introduced by the DIESIS project (Rome, 2009) and Meta-Information as described by (Denzer, 2003). The lateral coupling approach relies on re-usable coupling solutions (data adapters) that ensure syntactic and semantic interoperability of the data exchanged between Federates. As the name implies data providers and data consumers are coupled pair wise by data adapters. Data consumer and producers may be represented by CRISMA Federates or arbitrary data sources like services, files, data bases. Meta-information which is represented by Control and Communication Information in CRISMA context comes into play when the input as well as the output formats of producers and consumers is syntactically and semantically described. The same applies to data adapters that mediate between different formats.

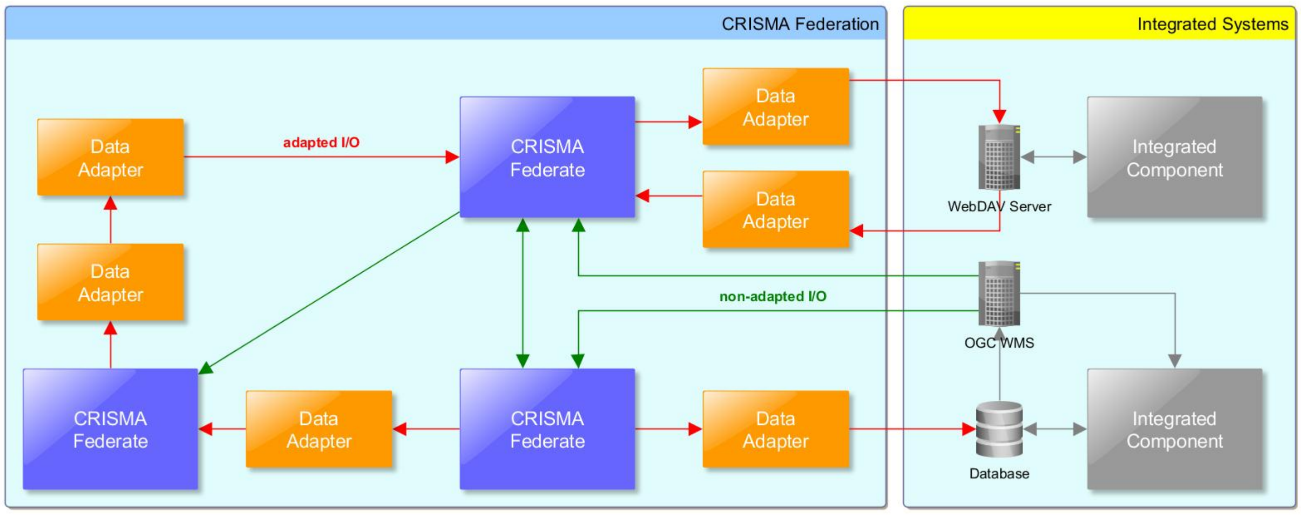


Figure 49: Data Integration Approach.

According to Figure 49 we can also distinguish between adapted and non-adapted I/O. Many Federates will be able to interpret data provided in several standardised formats with the consequence that no adapter is needed when the data of the producer (e.g. an OGC Web Map Service) is natively understood by the consumer (e.g. an OGC Web Map Client). To foster interoperability between Federates the CRISMA Framework features a common Control and Communication Information Repository of data formats and adapters shared among all CRISMA Federations. This repository can store shared data type definitions and information about the adapters. Each CRISMA Federation contributes to the shared repository. Data Formats (structure and syntax) can be described according to a widely adopted standard e.g. XML-Schema, IDL and JSON. Furthermore, a set of default data types for Objects of Interest (OOIs, see section 3.18) like date, geographic location etc. as well as generic types representing concepts relevant for crisis management is defined whereby each Federation may add specific types as extension of default types.

As said before, the most relevant standards to foster interoperability within a CRISMA Federation are those of the OGC, namely Web Feature Service (WFS and WFS-T primary for vector data), Web Map Service (WMS, for map images), Web Coverage Service (WCS, for raster data) and Sensor Observation Service (SOS and SOS-T, for time series of data). The usage of those OGC services offers not only a standard data encoding and transport mechanism but even more important a standardized method of data description by providing meta-information.

If data cannot be encoded matching the OGC rules the usage of the WebDAV (Web-based Distributed Authoring and Versioning) standard is foreseen. WebDAV is preferred over the File Transfer Protocol (FTP) because of security considerations (encrypted data transfer can be enabled by default) and for practical reasons because of easier access over firewalls. Another point is that WebDAV can offer data versioning, which is useful to reproduce results later on. Unlike OGC standards WebDAV allows any file format to be used. If possible file formats including meta-information should be used, e.g. the Network Common Data Format (NetCDF).

3.14. Simulation model integration approach

One major aspect of CRISMA is the usage of Simulation Models in order to enable users to assess the evolution of events in case of a crisis and thus to support decision making to mitigate direct and indirect consequences. Hence the CRISMA Framework allows the integration of newly developed Simulation Models as well as already existing ones. However, especially due to the heterogeneity of existing Simulation Models CRISMA has to ensure that they can be integrated in a standardized way and ultimately become a CRISMA Federate (see section 3.6.4). External accessibility is a requirement for components participating in a CRISMA Federation as CRISMA aims at offering software the Software as a Service (SaaS) way, more specifically the Model as a Service (MaaS) way for Simulation Model integration. Moreover CRISMA recommends the usage of well-known and widely adapted standards to handle data flow between Federates (see section 3.7). Ultimately due to the variety of different Simulation Models, their various runtime environments and other boundary conditions a standard to be used within CRISMA has to be versatile and also needs to fulfil a certain set of requirements common to Simulation Models:

- Any Simulation Model shall be able to provide a human readable description of its purpose.
- Any Simulation Model shall be able to provide information about the supported, required and optional parameters that have to be available in order to perform a new model run. This includes information how this data can be provided, e.g. if it can be transmitted prior to the actual execution request and where it shall reside, if it shall be part of the actual execution request, if the actual execution request may contain links to the actual data. Additionally it must state what formats and encodings are supported for each parameter as well as being able to provide validity ranges if applicable.
- Any model shall be able to provide information about its result data. This includes information on supported data formats and means for retrieving the data (Example: the result of a run is map raster data and can be fetched at a specific WMS using a certain URL).
- Any model shall be able to provide information about the status of a specific execution. Requesters may be informed at any time whether an execution is
  - Running
  - Failed
  - Finished
- Any model shall be able to either indicate the expected execution time for a certain run and the remaining execution time or at least provide an indication of the actual status. This requirement is directly dependent on the capacity of the model to provide this kind of information.
- Any model shall be able to provide detailed error messages e.g. in case of a failed execution

The **OGC Web Processing Service (WPS)** standard is perfectly suited for these needs. It enables the web-based execution of processes that can be any algorithm, calculation or model operating.

The WPS provides rules for standardizing the way how the processes and their inputs and outputs are described, how a client can request the execution of the processes and how the outputs of the process are delivered. The WPS defines a standardized interface to configure, initiate, manage and access the results of services. It facilities the publishing, the discovery of and binding to these processes by clients. The WPS supports computationally expensive models with the asynchronous execution mode. The processing time of the models is an important point of the models integration approach. In CRISMA, several models require considerable computation time. For example, time processing of the coastal submersion model (TELAMAC-2D) exceeds 5 hours.

In addition, the WPS provides several functions:

- encode requests for process execution
- encode responses from process execution
- embed data and metadata in process execution inputs/outputs
- reference web-accessible data inputs/outputs
- return process status information
- return processing errors
- request storage of process outputs
- assure the interoperability of the process

The WPS also supports Simulation Model chaining which is needed for the calculation of cascading effects. More information on how the conceptual model for considering cascading effects is realised by the Simulation Model Integration Approach is provided in section 3.19.

Although a WPS process is normally an atomic function, it can be integrated into a chain of different WPS processes. Therefore, a Simulation Model chain can be easily wrapped into a WPS and the outputs of one Simulation Model can be used as input of another Simulation Model as shown in Figure 50.

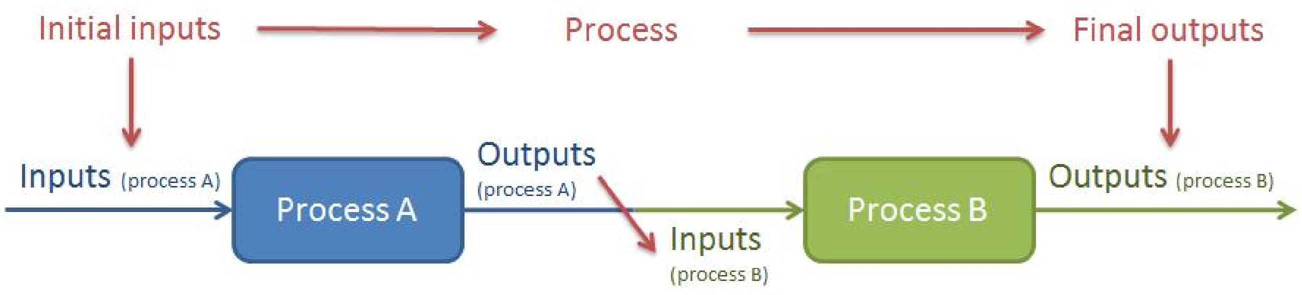


Figure 50: WPS Process Chaining.

In CRISMA, Simulation Model chains will be controlled by a specific WPS that takes care of the correct orchestration of several single WPS-wrapped Simulation Models. The chaining of Simulation Models is implemented using workflow management techniques. Workflow management allows combining simple or basic Simulation Models into a new complex Simulation Model providing added value results to the users. This new complex Simulation Model is also available as a WPS. Of course this kind of chaining does not prevent the other simple or basic Simulation Models to be accessible as WPS.

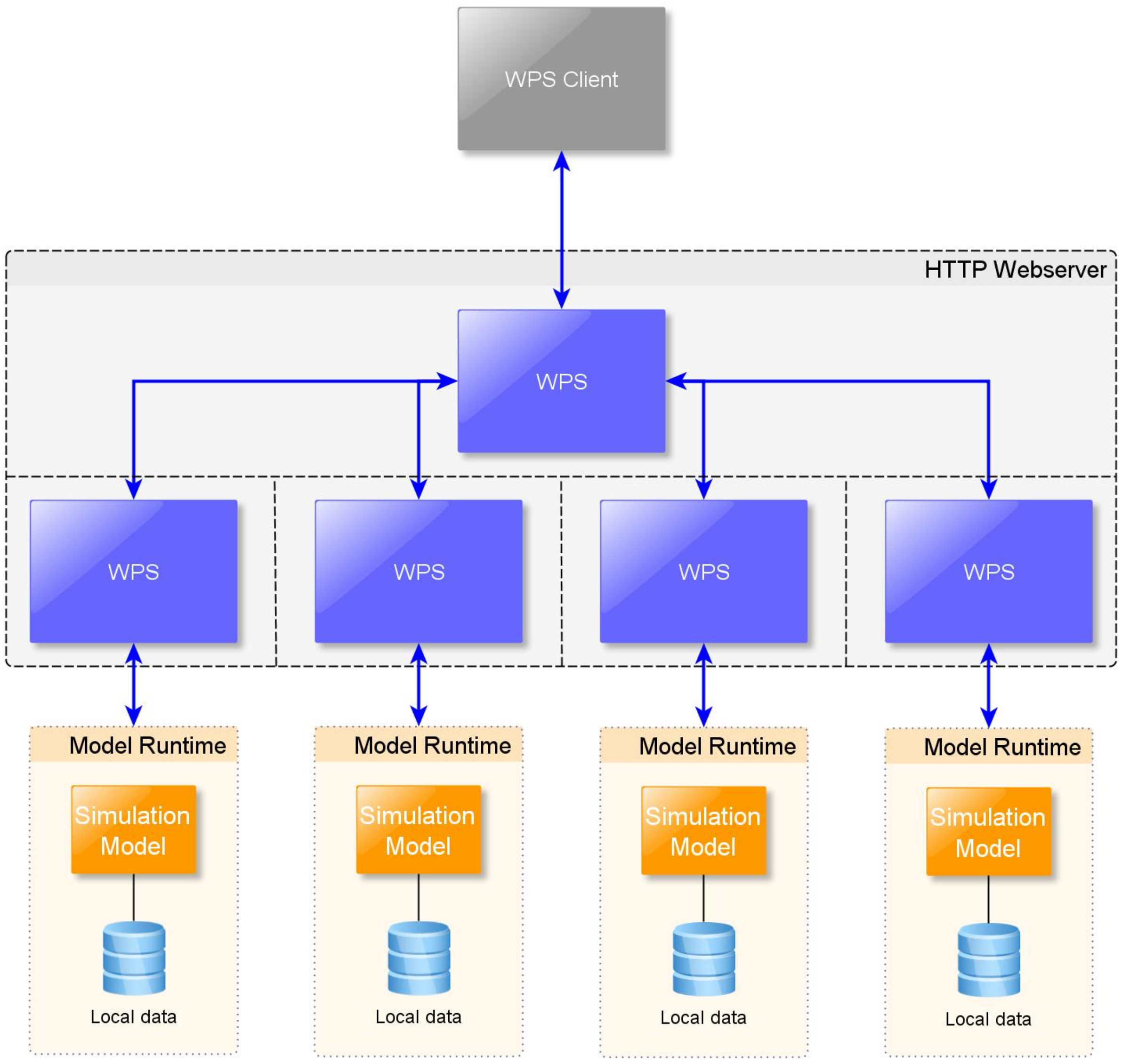


Figure 51: Model Chaining in CRISMA.

The approach shown in Figure 24 enables users of CRISMA to access Simulation Model chains in a uniform way, i.e. they can use model chains like single Simulation Models using the OGC WPS standard. This way, even external non-CRISMA users may benefit from a CRISMA outcome.

When Simulation Model Integration is done using the OGC WPS standard the level of integration is that of Integrated Simulation Models according to section 3.6.5. In order to fully participate in a CRISMA Federation it is required that any model has to be able to connect to the CRISMA Middleware Infrastructure (see section 3.4). Hence any Simulation Model shall on the one hand become an Integrated Simulation Model through adopting the OGC WPS standard and on the other hand be aware of the ICMM API as well as provide a standardised RESTful interface that complies with the requirements of tight integration (see section 3.6.1).

The latter will be achieved by a CRISMA-aware WPS wrapper that translates between CRISMA Federates and the Integrated Simulation Model. With this approach Simulation Models integrated in CRISMA will ultimately offer maximal interoperability as they will be

Federated Simulation Models as well as Integrated Simulation Models at the same time. This way CRISMA models can:

- participate in a CRISMA Federation / Application
- be accessed by ordinary non-CRISMA WPS clients
- be accessed by proprietary clients (e.g. if the model originates from already existing software)

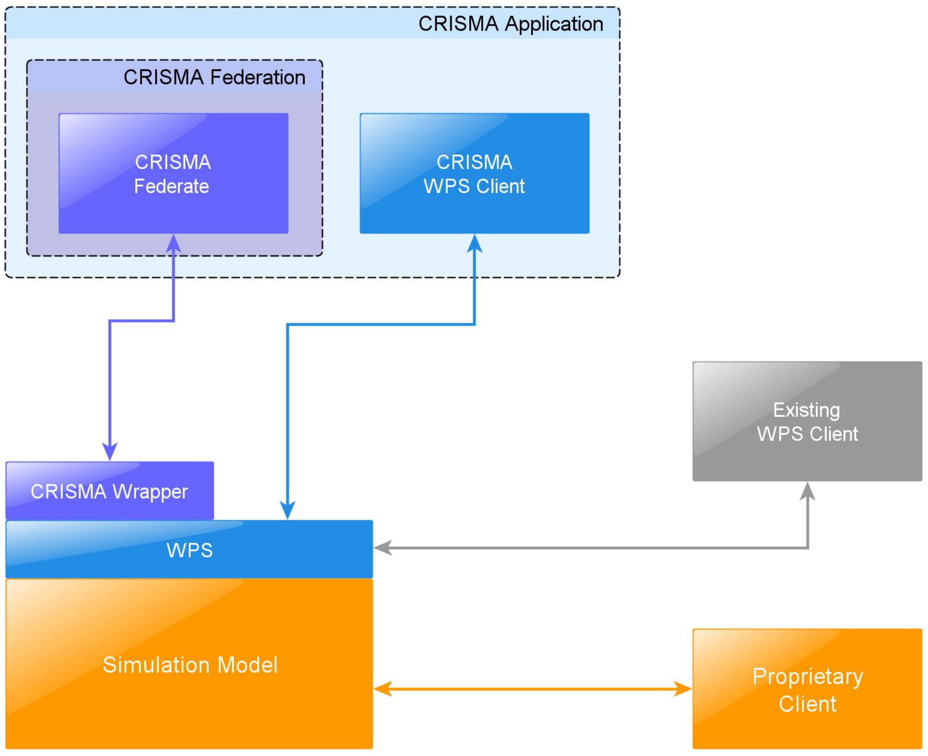


Figure 52: Federated Model Access Possibilities.

The CRISMA Framework will provide a Model Interaction UI (Simulation Interaction View Building Block, see section 4.3.12) as a CRISMA Building Block that basically consists of an embeddable UI according to the GUI integration approach (see section 3.8) that helps users manage their actual model execution which is part of a simulation run. This UI will allow users to:

- List the actually available Simulation Models and their properties
- Provide required and optional model input data
- Create single or batch executions
- Get information about their current executions
- Fetch model output data

3.15. Compatibility with legacy systems

The ability to integrate and interoperate with legacy systems and related data, services, models and user interfaces is one of the primary objectives of the project and thus also of the CRISMA Framework. These issues are addressed by the CRISMA Framework Architecture in depth and various approaches for integration on different levels of interoperability including the integration of CRISMA results in legacy systems and vice versa have been defined.

Conceptually, the architecture supports two dimensions of system integration: The first dimension is that of a Federated System (see section (3.10) whereby the legacy system to be integrated is aware of the CRISMA Framework and in particular the CRISMA Middleware Infrastructure (ICMM Building Block). Thus it is able to interact directly with CRISMA Building Blocks and information models (e.g. Core CCIM and its extensions). The legacy system needs to be adapted to take advantage of the extended crisis management support functionalities offered by the CRISMA Framework. This tight integration approach is commonly followed when a CRISMA Application is built around a legacy crisis management system or the legacy system is enhanced by CRISMA functionality. The second dimension is that of an Integrated System (see section 3.10.3). This loose and non-intrusive integration approach leaves the legacy system completely unchanged. This approach is commonly followed when only parts of the legacy system (data, models, user interfaces) are used in the overall CRISMA Application.

Apart from the integration dimensions, three general system integration approaches have been taken into account thoroughly during the specification and design of the CRISMA Framework Architecture. Those are: (1) integration on data level, (2) integration on business logic level and (3) integration on user interface level. (Probst, 2010) writes about those integration approaches:

"In the past decades, software engineering has been changing from building one-piece monolithic programs to ***assembling systems from modular and distributed components****. Furthermore, as the number of pre-existing Software Components is continuously increasing, integration of those components has become an important part of software engineering. Today, developers and integrators should be enabled to* ***connect software modules without having to know about other components' internal functionality*** *except for* ***well-defined interfaces****. Reality, however, most often shows a different picture: integrating software applications requires large coordination efforts, and component developers and integrators need a lot of knowledge about how other components actually work internally. Integrating applications requires a common understanding of those applications as well as the information they process. Software systems most often consist of three layers: the* ***data source layer****, the* ***business logic layer****, and the* ***user interface layer****. Therefore, the integration of software systems can be performed on three levels:*

- ***Integration on the data source level*** *provides a unified view on heterogeneous data sources.*
- ***Integration on the business logic level*** *unifies different implementations of business logic, each using its own data sources, under a common user interface.*
- ***Integration on the user interface level*** *unifies different user interfaces in one common system, e.g. a portal or a plugin-based user interface."*

While in fact those different approaches are not mutually exclusive the CRISMA Framework Architecture pursues each of the three integration approaches to certain extent supported by different concepts and Building Blocks. Figure 53 gives an overview on the levels of system integration supported by the architecture and the related Building Blocks. The architecture defines furthermore respective concepts for Data Integration (section 3.13), Simulation Model Integration (section 3.14) and GUI Integration (section 3.12) which rely on well defined standards and APIs to reach compatibility with legacy systems. Thereby it must be noted that the Simulation Model Integration Approach as well as the corresponding Building Blocks are suitable for the integration of arbitrary processes, not only Simulation Models.

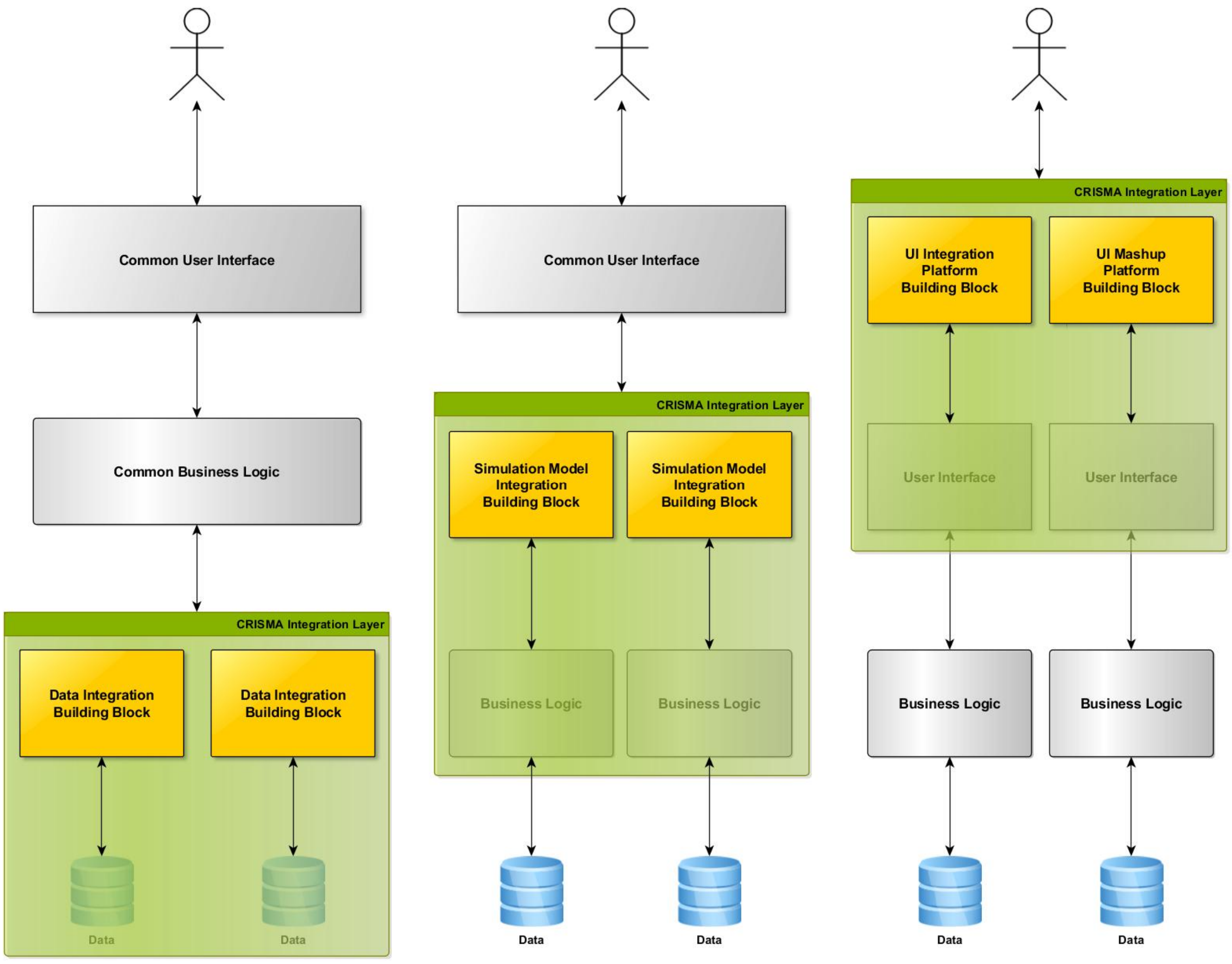


Figure 53: Levels of System Integration.

CRISMA Framework Building Blocks are implemented as Software Components (services and GUI modules) that take advantage of state of the art web technologies and thus can be integrated seamlessly in existing infrastructures. Figure 54 provides an example on how Mashable Composite UI Modules can be used to enhance the user interface of a legacy application. This integration of CRISMA User Interaction Building Blocks into autonomous graphical user interfaces of legacy systems is a major topic of the GUI Integration Concept and is explained thoroughly in section 3.12.

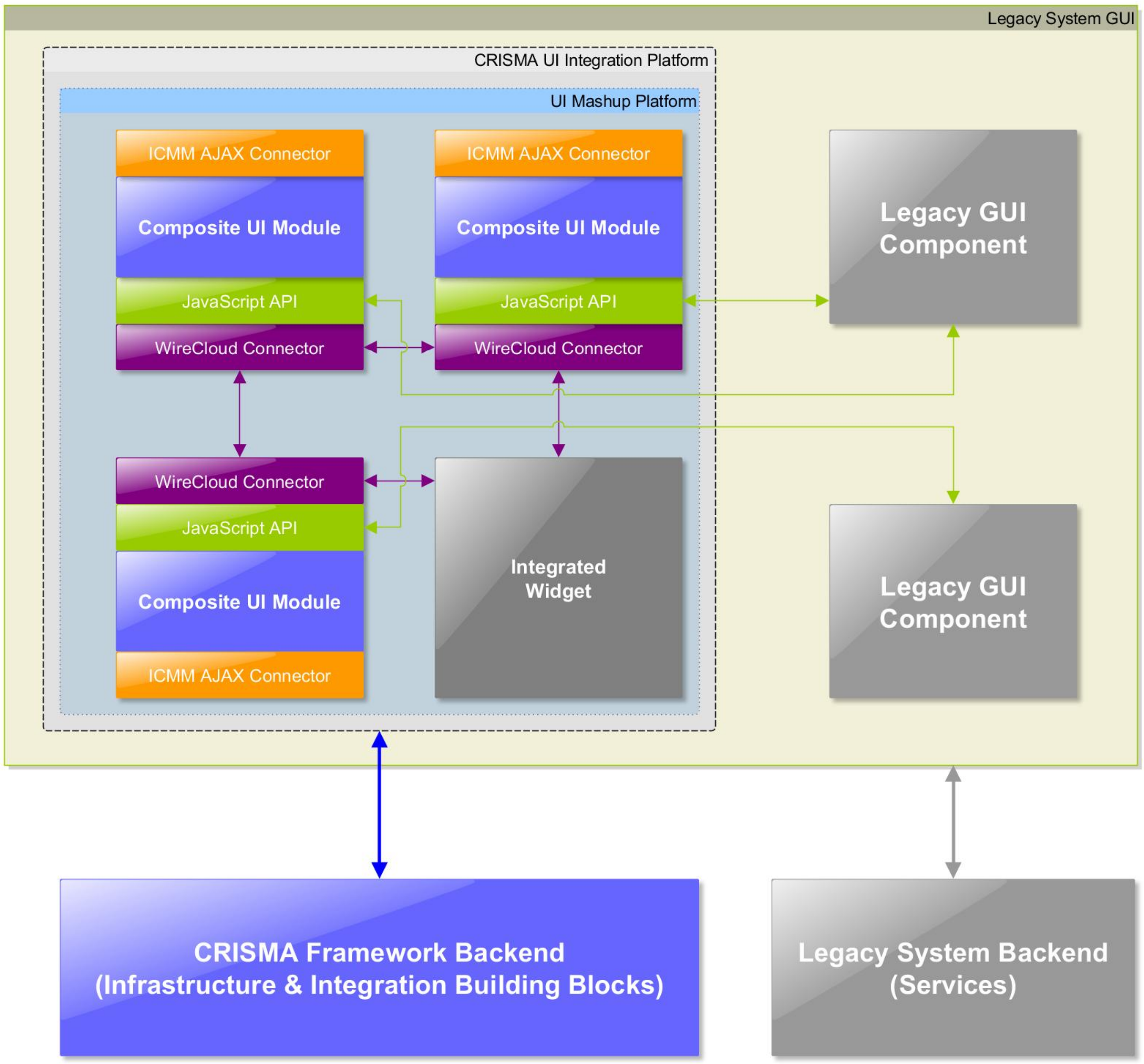


Figure 54: Compatibility with Legacy Systems (GUI Integration).

In conclusion, the CRISMA Framework Architecture establishes concepts and approaches that allow interaction and integration with legacy decision support and crisis management systems used in crisis management today.

3.16. Transferability

The transferability of the CRISMA Framework and thus the applicability of CRISMA results in other crisis domains and locations other than the ones addressed in the Pilot Applications has been a key requirement of the architectural work from the beginning. The approach to guarantee transferability is already laid down in the project structure as SP3 work is crisis domain and location independent and the actual domain and location context is added by SP5 by building Pilot Applications.

Several architectural concepts have been established to create a Software Framework that consists of generic, reusable and extensible Building Blocks which are essentially independent from domain or location specific aspects. A major concept which must be mentioned in this context is the Core Control and Communication Model (CCIM) of the CRISMA Framework (see section 3.6 for more details). Although completely generic and domain independent the Core CCIM explicitly defines extension points for plugging-in domain specific information. In each CRISMA Application, those extensions are used to define concrete Data Slots, Simulation Models, etc. that are only relevant in the scope of this particular Application.

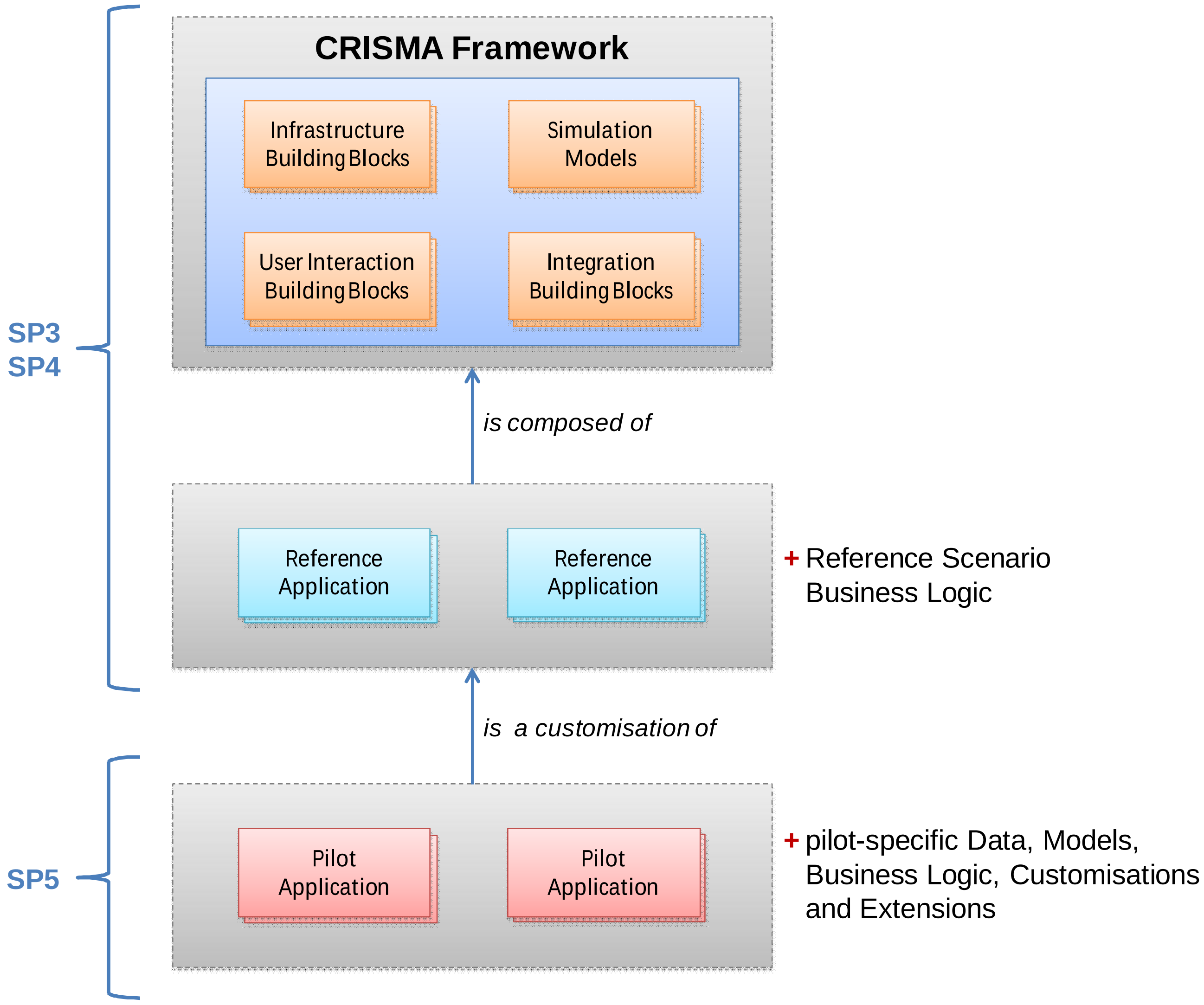

CRISMA Framework
Infrastructure Building Blocks
Simulation Models
User Interaction Building Blocks
Integration Building Blocks
SP3
SP4
is composed of
Reference Application
Reference Application
+ Reference Scenario Business Logic
is a customisation of
SP5
Pilot Application
Pilot Application
+ pilot-specific Data, Models, Business Logic, Customisations and Extensions

Figure 55: Reference Applications and Pilot Applications.

Furthermore, the implementation of Software Components (realisations of Functional Building Blocks) of the CRISMA Framework is driven by consolidated and general functional requirements that are derived from more concrete and domain specific user requirements. Thus, software provided by SP3 covers most functionality requested by pilot users without being bound to a particular domain or Pilot Application.

In fact SP3s main activities WP32 Architecture, WP34 (Building Blocks) and WP35 (Framework integration) are focusing on the creation of a general software framework for Crisis Management Simulation that is also usable by other scenarios and organisations than those foreseen in the five pilot cases of the CRISMA project.

Besides the mere provision of individual and independent Software Components for building Crisis Management Simulation Applications, the CRISMA Framework offers also several pre-integrated Reference Applications. A Reference Application represents the transferable composition of CRISMA Framework Building Blocks and Simulation Models (Software Components, respectively) without any pilot-specific logic. As shown in Figure 55 Reference Applications serve as basis for the development of the Pilot Applications.

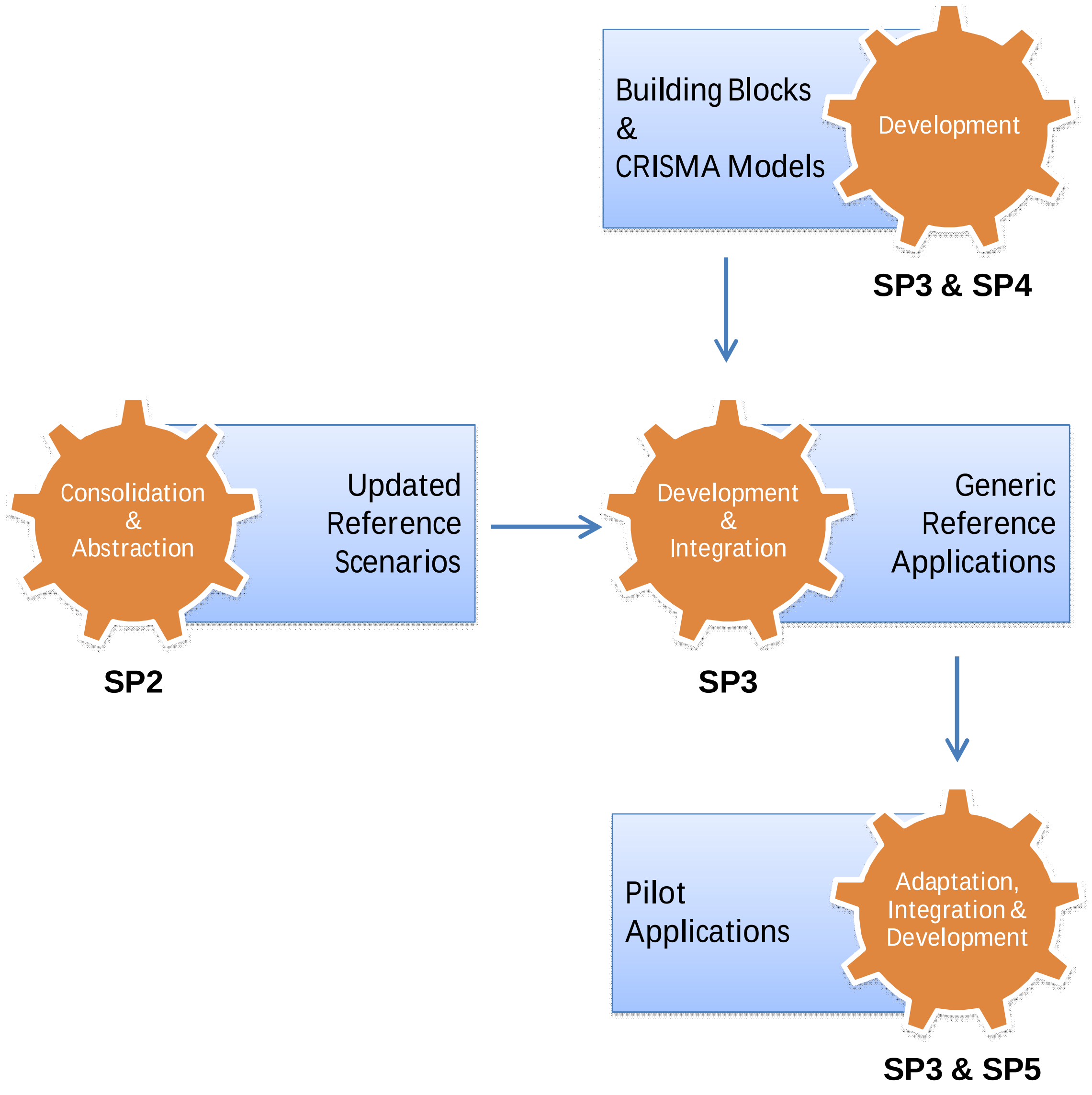

Building Blocks
&
CRISMA Models
Development
SP3 & SP4
Consolidation & Abstraction
Updated Reference Scenarios
SP2
Development & Integration
Generic Reference Applications
SP3
Pilot Applications
Adaptation, Integration & Development
SP3 & SP5

Figure 56: Development of Reference Applications.

Thereby, the main objective of a Reference Application is to provide a customizable, extensible and transferable implementation of a Reference Scenario (D25.1, 2014).

As described previously, the Reference Application can then be customised, extended and adapted to create a more specific CRISMA Application by adding specific data models, location context, etc. Figure 56 shows the general development process of Reference Applications and the involved CRISMA SPs.

3.17. Training

The support of Crisis Management Training is one of the main CRISMA targets. Preparedness action is carried out within the context of disaster risk management and aims to build the capacities needed to efficiently manage all types of emergencies and achieve orderly transitions from response to recovery. The Crisis Operator is normally the first person to become aware of a developing emergency situation. The actions of the operator can often make the difference between the inconvenience of managing a minor incident or managing a full scale emergency.

Operator competency is based on knowledge, skills and experience. It is crucial that Crisis operators have the proper training to do their jobs effectively. Operator mistakes can be both costly and potentially life threatening. Therefore, training and field exercises are a major part of Preparedness activities. As described in section 3.1.1, CRISMA focuses on Crisis Management Simulations that support specific decision support objectives. Thereby, Crisis Management Simulation Analysis concepts have been established (see section 3.1.3) that allow decision makers to judge which Simulated Crisis Management Scenario performs best according to specific Criteria.

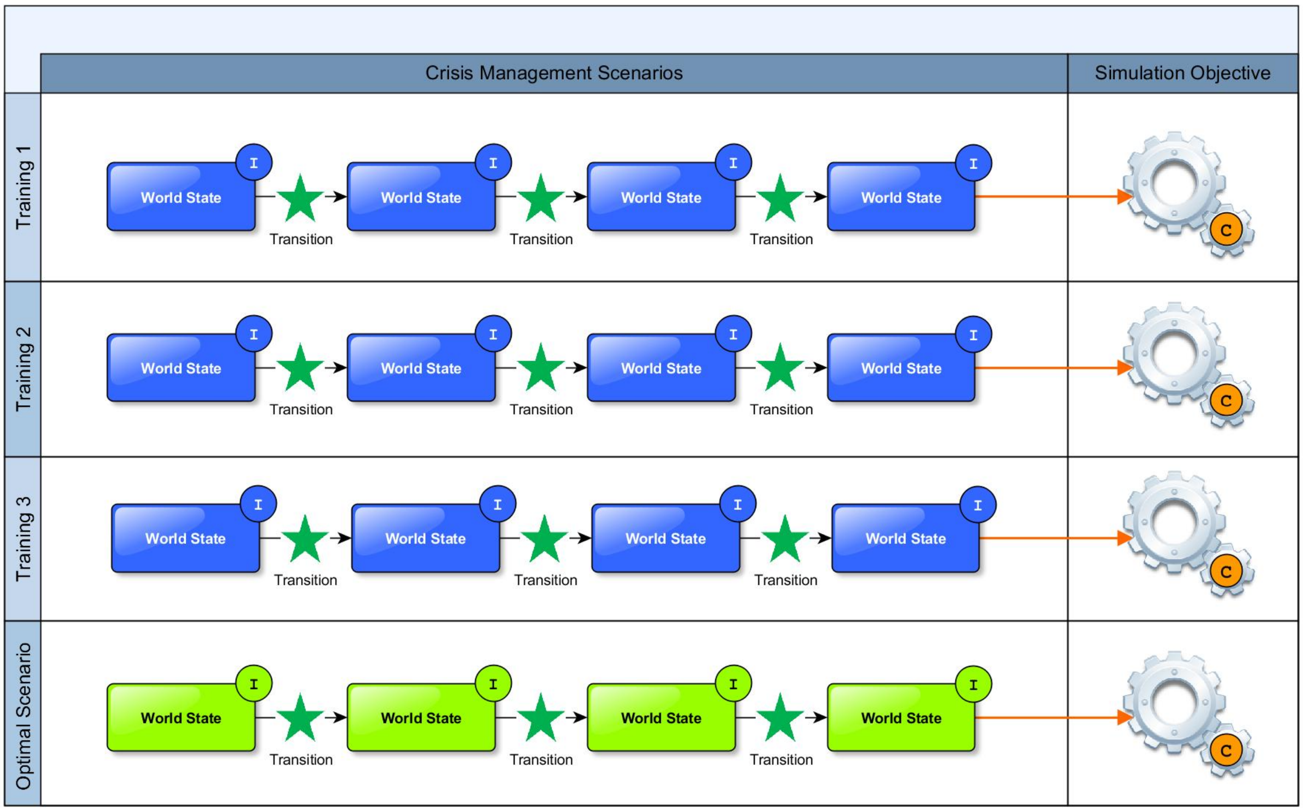


Figure 57: Mapping Training to the World State Transition Concept.

In case of a training-related Simulation Case this judgment has already been by taken by a Training Operator for one or more concrete Crisis Management Simulations. Thereby, a Simulation Case defines the outline of the overall training exercise, that is, the actions (Transition Points) that can be performed by a trainee. A concrete training session then corresponds to an instance of the Simulation Case, thus a Crisis Management Simulation.

A training operator may define an Optimal Crisis Management Scenario leading to an optimal level of satisfaction for a given Simulation Objective (shown in green in Figure 57). The training sessions of a trainee can be recorded as World States and Transitions (shown in blue in Figure 57), thus leading to alternate Crisis Management Scenarios. Indicator based comparison as well as Multi Criteria Analysis can then be used to (vertically) compare the scenarios of the trainee with the optimal scenario defined by the training operator.

In fact, judging the performance in a training exercise does not differ from regular Crisis Management Simulation Analysis except for the fact that an “optimal” solution is known beforehand and can be used as a reference. Besides the relation to the Conceptual Business Logic, the support of Crisis Management Training in CRISMA encompasses several other aspects which are briefly explained in the following.

A system supporting Crisis Management Training can be perceived as a CRISMA Federation (as described in section 3.2 and 3.6.1) which is fully aware of the CRISMA services. It internally hosts CRISMA Composite UI Modules. A Training Federation enables a trainee to learn emergency management in a virtual environment. The trainee-user is a control room operator or an on-scene commander. The (optional) coach-user is a trainer that is able to add additional incidents or change incident parameters during a training session.

The main features of a Training Federation are:

- Plan custom training scenario: using an easy-to-use planning tool and a BPM workflow/rule correlation engine, trainers plan incident scenarios. The trainer can define the following:
  - The roles involved (stakeholders)
  - The procedures by building a hierarchical task-based scenario
  - The resources involved in the training scenario
- Execute training scenarios manually on demand or based on a predefined schedule.
- The trainee is able to evaluate and monitor the situation using a Geographical Information System (GIS).
- The trainee can interact with field trainees via two-way communication methods such as radio and telephony.
- Use the CRISMA Framework in order to incorporate simulation in training scenarios.
- Debrief of a training session with ability to reconstruct multimedia resources, user actions and simulation usage, generate reports and provide training scores.

Integration of the CRISMA Infrastructure Building Blocks and the Training Federation is performed in the following way:

- Integrate predefined simulations results with real-time incident management by using the Simulation Result Catalogue module.
- As part of the incident management process use the CRISMA Run simulation UI to collect user input, fetch the relevant simulations results and visualize them on the GIS map that supports the CRISMA simulation result standards such as OGC.
- Common operating picture: User sees a hybrid view of the current state integrated with simulation results. E.g. the expected fire direction along with the amount of fire brigade in the area.
- User takes decisions based on the presented information. E.g. direct more fire brigade forces to the relevant high risk areas.
- User verifies the situation in real time using map overlays of real time sensors information and live/playback video.

3.18. Resource management

The term resource in this context is used for the information representing an entity that can be used to potentially affect the operations and resolving of serious hazard incidents. Crisis managers use resources in the context of response actions including aspects of logistics, deployment, evacuation and monitoring. As shown in Figure 58, in a crisis simulation, crisis managers deal with the information representing incident response resources (such as ambulances and paramedics), incident affected entities (such as patients), hazard related properties (such as effected radius) and environmental properties (such as weather conditions, roads and traffic). As part of CRISMA terminology these resources are called Object of Interest and in short OOIs.

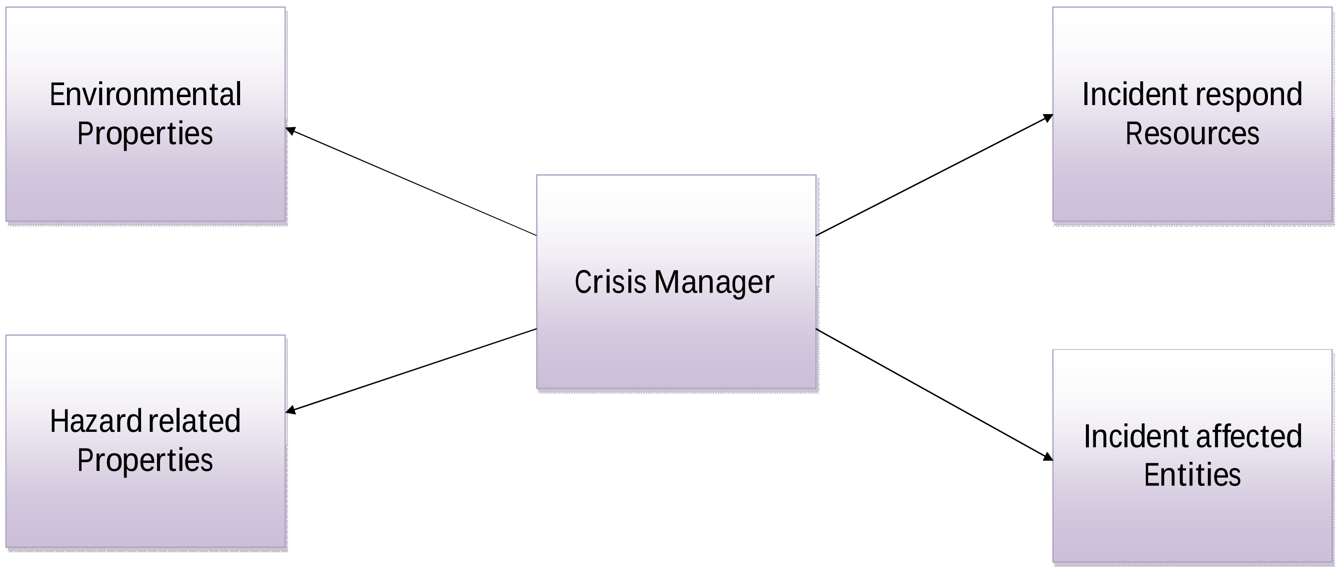

Environmental Properties
Incident respond Resources
Crisis Manager
Hazard related Properties
Incident affected Entities

Figure 58: Relevant OOIs in Crisis Management.

Managing this kind of information encompasses:

- Identify the relevant OOI types.
- Initialize instances of these types.
- Associate instances to a specific simulation exercise or training session.
- Execute commands (such as dispatch resources).
- Let these instances develop their state over time (according to the resource management models).
- Capture snapshots of the resources in a World State format.
- Visualize the status of these instances (over time, final, on a map, summary, …)
- Enable analysis of simulation sessions by comparing World States and visualize Indicators and measurements.

The OOI Resource Management Domain Model is used to define the entities (OOI Instances) which are manipulated by the user or automatically by some resource management mathematical model. It contains a flat list of properties, each constructed from one of the following types: Number, Text, Date and Time and Geometry (Geo data). The relation between the different entities is shown in Figure 59.

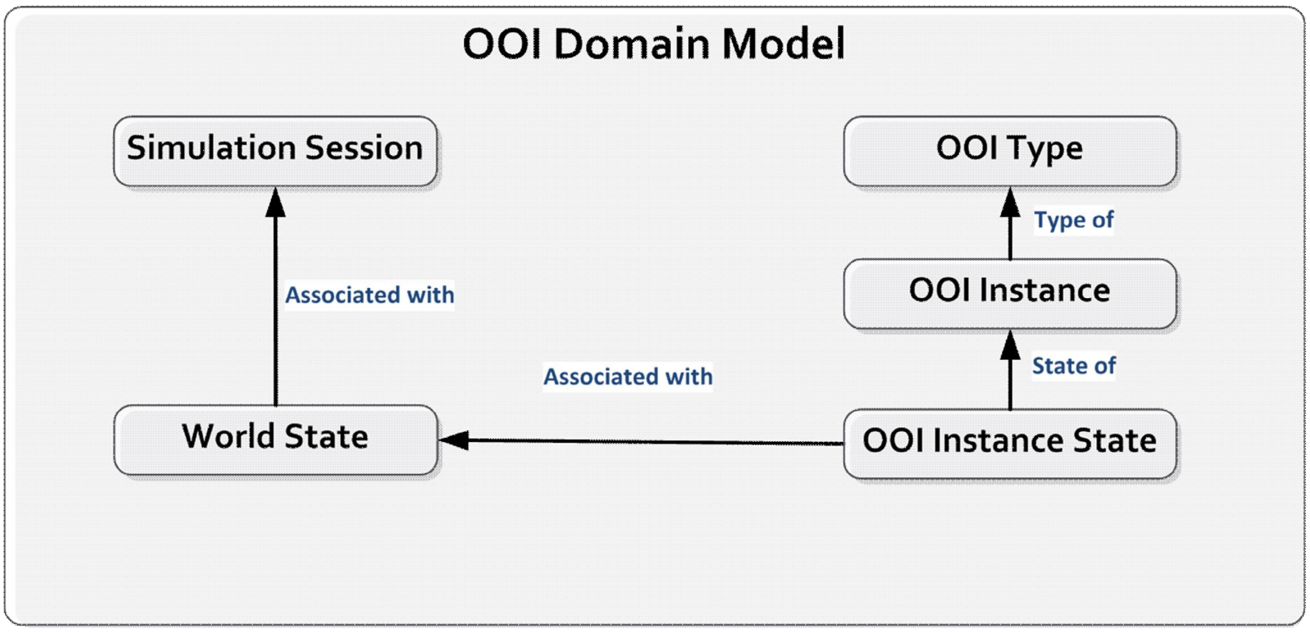


Figure 59: OOI Types and Instances Relations.

By defining an **OOI Type** an initial type including the available properties of a resource is set. According to the definition of an OOI (feature representation), a distinction between two generic property types can be made (shared properties and unique properties). Figure 60 provides an overview on OOI Types Class Hierarchy.

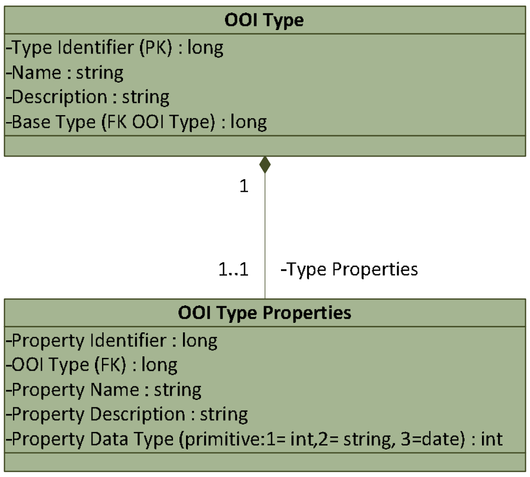

OOI Type
-Type Identifier (PK) : long
-Name : string
-Description : string
-Base Type (FK OOI Type) : long
1
1..1
-Type Properties
OOI Type Properties
-Property Identifier : long
-OOI Type (FK) : long
-Property Name : string
-Property Description : string
-Property Data Type (primitive:1= int,2= string, 3=date) : int

Figure 60: Overview of generic OOI Types Class Hierarchy.

All OOI Types share the following properties:

- Type Identifier – Object kind definition. Identifies this object type so it is unique in a Federation of Federates.
- Name – Object short description text.
- Description - Object long description text.
- Base type – The base object type. Indicate inheritance of all base type properties.
- Geometry – Used to express the spatial component of the OOI.
- Symbol – Resource identifier for symbol display. Usually raster or vector image file.
- Properties – Collection of properties that characterise this OOI Type.

All OOI Type Properties share the following properties:

- Type Identifier – Object kind definition.
- Name – Property short description text.
- Description - Property long description text.
- Associated OOI Type – The OOI Type that contains this type as part of its Properties collections.

By **defining an OOI Instance** an instance of a type including the available properties of a resource are set. Figure 61 provides an overview on the OOI Instances Class Hierarchy.

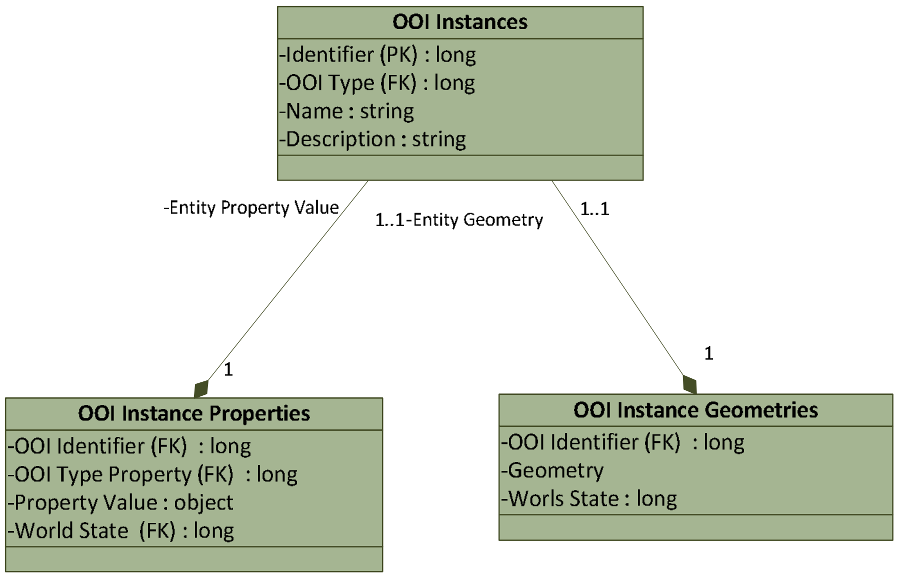


Figure 61: Overview of OOI Instances Class Hierarchy.

All OOI Instances share the following properties:

- Identifier – Object instance identifier. Identifies this object so it is unique in a Federation of Federates.
- OOI Type – Object kind definition.
- Name – Object short description text.
- Description - Object long description text.

All OOI Instance Properties share the following properties:

- OOI Identifier – The OOI instance that this property is associated with.
- OOI Type Property – The property type that this property represents.
- Property Value - a primitive that represents the actual value of this property.
- World State – The point in time, snapshot, that this property is related to. A specific OOI Instance and property combination can be related to a single World State at a specific point in time. World States represent the evolvement of OOI properties data across time.

All OOI Geometry Properties shares the following properties:

- Identifier – Object instance identifier.
- World State – The point in time, snapshot, that this property is related to.

- Geometry – Represents a geographical representation of the OOI Instance. The geometry data type enables the representation of geometric shapes. The supported OOI Geometry Types are shown in Figure 62 below:

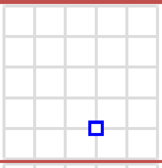
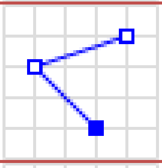
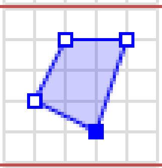
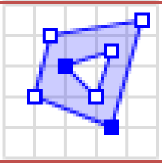
Type		Examples
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Figure 62: Supported OOI Geometry Types.

The main aim of the Resource Management Concept is to provide a unified OOI World State storage and management system that enables user interface components (Mashable Composite UI Modules) to create the initial World State and to record and encode resource management related user commands. Then a Resource Management Model can decode this user input and the initial World State and execute the commands while producing new time stamped World States.

Consumers such as UI views visualize the World State to the user. Other analysis components can inspect the World State evolvements and provide additional insights.

Figure 63 shows a simplified example of an architecture of a resource management related CRISMA Application that uses CRISMA Infrastructure Building Blocks, such as the Pub/Sub Context Broker for publishing events between components and the ICMM in order to register the simulation World State and enable external tools to access and investigate the resource management simulation results. Please note, that the example architecture is further elaborated and detailed in the context of the specification and development of the respective CRISMA Application.

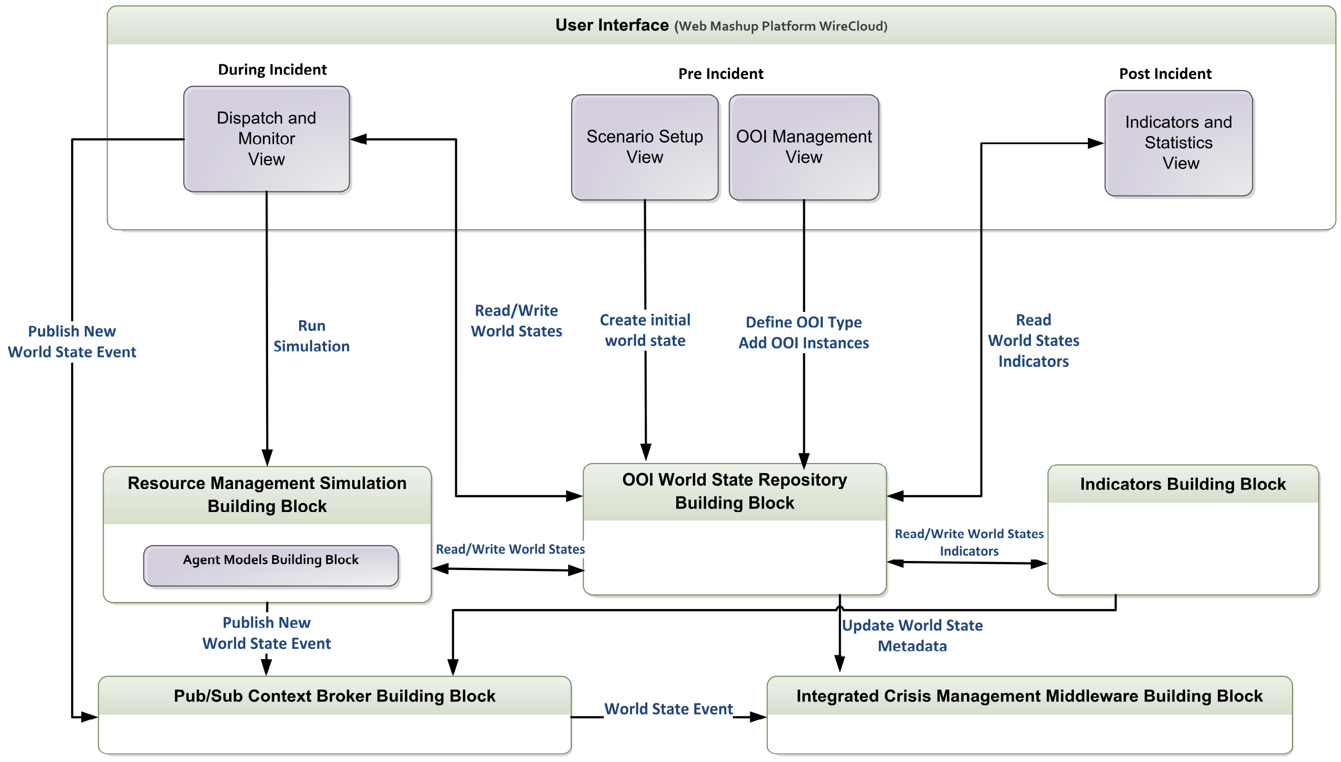

User Interface (Web Mashup Platform WireCloud)
During Incident
Dispatch and Monitor View
Pre Incident
Scenario Setup View
OOI Management View
Post Incident
Indicators and Statistics View
Publish New World State Event
Run Simulation
Read/Write World States
Create initial world state
Define OOI Type Add OOI Instances
Read World States Indicators
Resource Management Simulation Building Block
Agent Models Building Block
OOI World State Repository Building Block
Indicators Building Block
Read/Write World States
Read/Write World States Indicators
Publish New World State Event
Update World State Metadata
Pub/Sub Context Broker Building Block
World State Event
Integrated Crisis Management Middleware Building Block

Figure 63: Architecture of a Resource Management Application (simplified).

3.19. Relation to models for multi-sectoral consequences

The CRISMA Framework Architecture has to support the development of Models for Multi-Sectoral Consequences (SP4 Models) taking place in SP4 as well as their integration into the CRISMA Framework. To achieve this, the architectural concepts and Building Blocks of SP3 have to be put in relation to the concepts and models developed in SP4. While the general relationship between SP3 and SP4 has already been described in the introduction to this document (section 1.6), this section intends to give further details and point to relevant documents that explain the SP4 point of view.

From the (Simulation) Model Integration perspective (section 3.14) of the CRISMA Framework Architecture Software Components (“models” and “tools”) developed in SP4 are generally considered as Federated Simulation Models and thus have to be implemented and integrated according to a WPS-based approach. In this regard, both conceptual and technical compliance of SP4 Software Components with the SP3 part of the CRISMA Framework is ensured. The aforementioned integration approach considers also methods for chaining of different models thus it perfectly fits to the realisation of the Cascade Events and Time-Dependent Vulnerability concepts of SP4.

When talking about models, it is important to point out the difference between Simulation Models with the meaning of hazard models, impact models, etc. that are specific to a particular CRISMA Application (see section 3.3) and Models for Multi-Sectoral Consequences that are an integral part of the CRISMA Framework.

Although a distinction between those two types of models seems only relevant on conceptual level (technically, both are exposed by WPS), it is important to highlight the fact that dedicated User Interaction Building Blocks for Models for Multi-Sectoral Consequences are provided by SP3.

Theoretically, the Simulation Interaction View (see section 4.3.12) which provides, amongst others, functionalities to configure and monitor a simulation run could also be used as client for models provided by SP4. Due to its genericity and adaptability the Simulation Interaction View perfectly fits as client for (domain and application specific) Simulation Models. Genericity is imposed by the requirement for transferability of CRISMA Framework (section 3.16). Therefore a certain amount of configuration and implementation effort is required to provide an application-specific customisation of this Building Block that is tailored to the needs of a particular Simulation Model and CRISMA Application, respectively.

In contrast, Models for Multi-Sectoral Consequences are by definition also transferable results that shall support Crisis Management Simulations as envisaged by the CRISMA Framework. Thus, SP3 provides dedicated but still domain and application independent User Interaction Building Blocks for those models. Therefore, the relation of Models for Multi-Sectoral Consequences to the concepts of the CRISMA Framework Architecture as well as to relevant Building Blocks of SP3 is explained in subsequent sections.

3.19.1. Crisis management strategies and planned action

Task 44.3 “Simulation model and virtual application for crisis management strategies and planned action” concentrates on the development of a conceptual model for Resource Management. As has been explained in CRISMA D41.1 (D41.1, 2013), resources are various assets that can be actively used by a crisis manager. The crisis manager can take advantage of resources for the purpose of danger defence, crisis alleviation, civil protection, etc., as long as adequate information about these resources is available. Resources include materials (e.g. sandbags, medical products), personnel (e.g. fire brigades), protection infrastructure and facilities (e.g. hospital, shelters), and installations (e.g. weirs).

The conceptual model for Resource Management specified in (D44.1, 2013) was designed using agent-based modelling, which includes active agents and passive objects and enables the comparison of alternative Crisis Management Scenarios, which can be usefully applied for decision support in resource management. According to (D41.1, 2013) the conceptual architecture of a decision-making agent can be described as having five major components:

1. The first is a set of facts representing knowledge about the world that the agent currently holds. These facts get updated as the agent experiences the world.
2. The second component of the agent is a specification of goals that the agent is trying to achieve.
3. The third component represents the set of actual goals the agent is currently trying to achieve - its intentions.
4. The fourth component is a library of plans. Plans are sequences of concrete actions for achieving the agent goals.
5. The fifth component is a reasoner or interpreter. The role of the reasoner is to orchestrate the overall behaviour of the agent.

The agent-based Resource Management Simulation Models used in CRISMA Applications are implemented by SP4 on the basis of an Agent-Oriented Simulation Models Building Block (section 4.2.1). Additionally, SP3 provides various User Interaction Building Blocks that realise general conceptual models for Resource Management and thus act as clients for the Resource Management Simulation Models. Those Building Blocks are:

- Resource Management Tactical Training (section 4.3.8)
- Resource Management Training Dispatch and Monitor View (section 4.3.9)
- Resource Management Training Indicators and Statistics View (section 4.3.10)
- Resource Management Training Simulation Scenario Setup View (section 4.3.11)

In section 3.18, the CRISMA Framework Architecture defines furthermore a concrete Resource Management Architecture for Objects of Interest (OOI).

3.19.2. Economic impacts and consequences

The conceptual model Economic Impacts and Consequences defined in (D44.1, 2013) presents the economic impacts assessment approach used as well as the position of the evaluation application in the overall CRISMA concept, for example the connection to the CRISMA World States.

The economic evaluation in CRISMA is a multidimensional problem and various aspects need to be taken into account in the evaluation. Two different types of economic evaluation models can be distinguished: (1) the economic evaluation at the general level and (2) the pilot-specific economic evaluation. In WP44, the main focus is on developing the general evaluation framework to be applied for the economic evaluation of impacts of preparedness and the consequences of a crisis. It needs to cover the evaluation of economic damages and losses and the evaluation of mitigation measure options.

The Economic Impacts and Consequences Model of SP4 follows the logic of the Conceptual Business Logic of the CRISMA Framework Architecture. Thereby, it takes a World State snapshot, created by Simulation Models, as a static input (called as a scenario description for the economic evaluation), and calculates the cost of a crisis and/or the economic impact of different possible mitigation actions. The simulation of economic parameters is done by the economic evaluation model but all other simulation runs are to be done by other CRISMA models.

The concept is implemented by two Building Blocks: The Economic Impact Calculation View (4.3.1), a User Interaction Building Block provided by SP3 and the Economic Impact Calculation Service, the actual implementation of the model provided by SP4.

3.19.3. Multi-criteria decision analyses

Multi-Criteria Decision Analyses (MCDA) as introduced in (D44.1, 2013) makes it possible to consider a large number of data (i.e. inputs from several criteria), relations and goals which are generally present in the decision making process, so that the specific problem can be studied from multiple angles. Also, decision makers are supported by Crisis Management Simulation Analyses that suitably combines ICC (Indicators, Criteria and Cost) according to standard rules.

ICC are automatically determined and appended to the World State, using dedicated Indicator functions or models, and depend on the GIS information and databases describing the world as well as on Simulation Control Parameters (SCP).

ICC parameters are essentially used in WSA in order to summarize the World States by providing some relevant measures that are useful to take decisions. ICC parameters, after having been selected, generally need a further elaboration with the aim to obtain a synthetic judgment. This can be made by a Multi-Criteria Analysis, a powerful method that is able to take into account a large number of information, relations and goals and mostly the possibility to include intangible effects (e.g. value of people).

Indicators, Criteria and Cost are a central component of the Conceptual Business Logic of the CRISMA Framework as described in section 3.1. Thus, the MCDA concepts specified by SP4 perfectly fits into the SP3 architecture. A method proposed in (D44.1) for the realisation of Multi-Criteria Decision Analyses is based on Ordered Weighted Average (OWA). It will be used together with Crisis Management Simulation Analyses, which, in fact, as result of the workflow of a simulation includes cascading effects of WP42 and time dependent vulnerability of WP43.

Besides its inherent support for MCDA on conceptual level (see section 3.1.3), SP3 defines the Multi Criteria Analysis and Decision Support View (section 4.3.5) and Scenario Analysis and Comparison View (section 4.3.12) User Interaction Building Blocks for the definition and execution of alternate decision strategies based on the OWA method and for the Indicator- and Criteria based visualisation and comparison have to be provided.

3.19.4. Cascade events and time-dependent vulnerability

The term Cascade Effect in CRISMA describes a chain of events affecting each other. These events can be series or parallel sequences of adverse events generated by one or more sources, e.g. an earthquake that causes ground motion triggering a landslide in a given area. CRISMA will enable the end user to determine the related events and possible cascading effects for each hazard (directly or indirectly) and also to evaluate indirect effects caused by the incident.

In (D42.1, 2013), a methodology to define the possible scenarios of cascading effects is introduced while in (D42.2, 2013) a data base and an actual model for cascading effects is defined. The Conceptual Business Logic of the CRISMA Framework relies on the concept of World State and Transitions between World States that are driven either by events, by model outputs and/or by user manipulation. Within this concept, cascading effects are integrated as sequences of triggered events following an initial ”triggering event” occurring at a given time and affecting the World State (WS) at that given time.

There are three different concepts that are introduced for the integration of the cascading effects into the CRISMA Framework:

- **Cascading Effects Database**
  The scope of the cascading effects database is to communicate to the system the possible cascading scenarios that can be developed after the occurrence of a given event at a given time.
- **Transition Matrix**
  The transition matrix is a representation of the conditional probabilities of the form P(T1|E1), which means the probability of having the triggered event T1 given the occurrence of a triggering event E1.
- **Time-Dependent Vulnerability Model**
  A Time-Dependent Vulnerability Model calculates the probable damage distribution after each event, meaning: given (as input) the hazard intensity and the inventory of elements at risk, the model calculates as an output, for each class of the elements at risk, the number of elements that are likely to sustain the different damage levels considered in the damage scale.

Supporting Cascading Effects implies major technological challenges, since it requires that two or more Simulation Models will share data and exchange data at run-time, which includes also updating inventories of elements at risk as envisaged by the Time-Dependent Vulnerability Model.

According to the Simulation Model Integration Approach defined in this document (section 3.14), the CRISMA Framework supports cascading effects by defining common interfaces (OGC) and standards that will be supported by all Simulation Models that require participating in a cascading effect scenario. A Cascade Effects Scenario is realized by a dedicated WPS which allows executing the distinct (WPS-wrapped) Simulation Models and mediating between different producers and consumers of model input and output. Such a WPS model chain is defined by a Path of Analysis that is chosen by the user from a Transition Matrix. Support for Cascade Effects on the GUI Level is provided by the Cascade Events Configuration and Interaction View Building Block (4.3.1).

3.20. Access control

The topic of Information System security covers the protection of information from unprivileged use. Most aspects of information system security like the encryption of messages sent through communication channels is usually handled on lower layers of the protocol stack. Due to this, most aspects of information system security are transparent for the architectural concepts discussed in this section. One exception is the topic of Access Control mechanisms where usually some type of credentials need to be managed and used for authentication and authorisation purposes and thus need to be considered on an architectural level even though established methods, standards and solutions can be used.

In CRISMA we presume that Access Control to resources controlled by specific Federates and Integrated Components needs to be managed by the very same Federates and Integrated Components. In that sense the owner of a Federate or Integrated Component is free to choose any Access Control mechanism that suits the individual security requirements. From the CRISMA point of view required credentials are only part of the information needed to gain access to a particular resource (service or data). For this reason credential information are handled the same way as any other information enabling the access to the resources of a Federate and thus need to be made available to the ICMM or provided e.g. by user interaction at the time of the access.

Even though there are no restrictions on conceptual level the CRISMA Framework Architecture recommends the use of state-of-the-art authentication and authorization solutions for this purpose[7] because it is the general project policy to foster relevant open standards like: OpenId and OAuth and solutions using these standards are recommended to be used in CRISMA Applications. See section 4.4.3 for more information.

3.21. Administration

As already described in D23.1 “Scenarios, Requirements and Criteria for Crisis Management Modelling” (D23.1, 2013), administration and configuration in the scope of the CRISMA project can be perceived from different perspectives. This section intends to provide a brief recapitulation of the aforementioned considerations and to discuss their

[7] Quite often the users will have authentication and authorization systems capable of Federation and providing the single sign on in place. If and when this is not the case, the FI-Ware Identity Management generic enabler may provide a good solution to a problem at hand (see https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.IdentityManagement)

relevance and applicability context of the CRISMA Framework. While in the user requirements rather specific administration tasks were described from the perspective of the individual pilots the CRISMA Framework has to concentrate on common administration tasks and describe them in a more general and crisis domain independent fashion.

As CRISMA doesn’t intend to deliver one monolithic system but a toolbox or framework for building custom applications, there cannot be only one common approach for the administration of a CRISMA Application. Instead common administration tasks such as the administration of deployed service instances, user and policy management, administration of the information models, etc. can only be performed in relation to a specific building Block. Those tasks are then usually carried out with the help of administration APIs or GUIs provided by the Software Components implementing the respective Building Block. Furthermore, due to the open nature of the CRISMA Framework, where Building Blocks can be implemented using different approaches, programming languages, libraries and legacy tools, etc., each featuring possible configuration options and methods, the CRISMA Framework cannot enforce one harmonised administration interface for all kinds of tasks and Software Components.

In conclusion, administration is a topic that has to be considered individually on the level of Building Blocks, or more precisely on the level of the implementation (Software Component) of a Building Block. Therefore, when applicable, the descriptions of Software Components (D34.1, 2013) report on the supported administration mechanisms and tools.

4. Realisation

This chapter explains how a number of the concepts defined in chapter 3 are applied to realise the mission of the CRISMA Framework Architecture defined in chapter 2. Thereby, it introduces Functional Building Blocks and gives an overview about various standards that are relevant for their implementation.

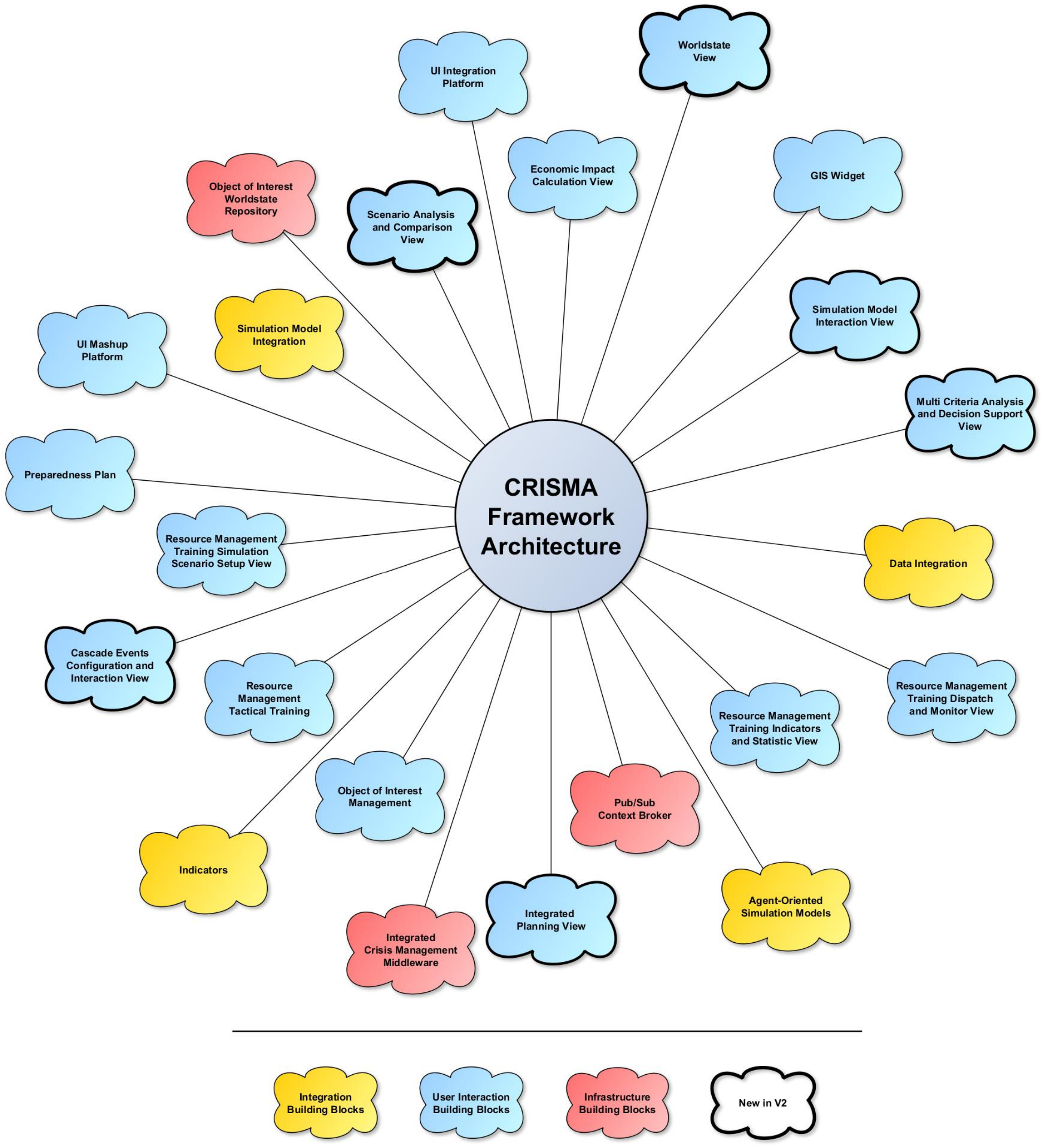

Worldstate View
UI Integration Platform
Economic Impact Calculation View
GIS Widget
Object of Interest Worldstate Repository
Scenario Analysis and Comparison View
Simulation Model Interaction View
Simulation Model Integration
UI Mashup Platform
Multi Criteria Analysis and Decision Support View
Preparedness Plan
CRISMA Framework Architecture
Resource Management Training Simulation Scenario Setup View
Data Integration
Cascade Events Configuration and Interaction View
Resource Management Tactical Training
Resource Management Training Dispatch and Monitor View
Resource Management Training Indicators and Statistic View
Object of Interest Management
Pub/Sub Context Broker
Indicators
Agent-Oriented Simulation Models
Integrated Planning View
Integrated Crisis Management Middleware
Integration Building Blocks
User Interaction Building Blocks
Infrastructure Building Blocks
New in V2

Figure 64: Functional Building Blocks Overview.

Figure 64 shows all Functional Building Blocks defined by the CRISMA Framework Architecture. The following colours are used to distinguish between Building Block

Categories: red stands for Infrastructure, orange for Integration and blue for User Interaction. New Building Blocks that have been introduced in the second version of the architecture specification are indicated by a thick outline.

The first Architecture specification (D32.1, 2013) provided comprehensive summaries and detailed functional specifications of CRISMA Framework Building Blocks as integral part of the deliverable document. The report on Building Block Implementation (D34.1, 2013) then introduced a central entry point for up to date description of CRISMA software results (Software Components): The CRISMA Catalogue.

http://catalogue.crismaproject.eu/

The major advantage of an online catalogue compared to a static document is that the catalogue always contains the most current information. The CRISMA Catalogue is continuously updated and therefore keeps up with the progress made in the CRISMA project.

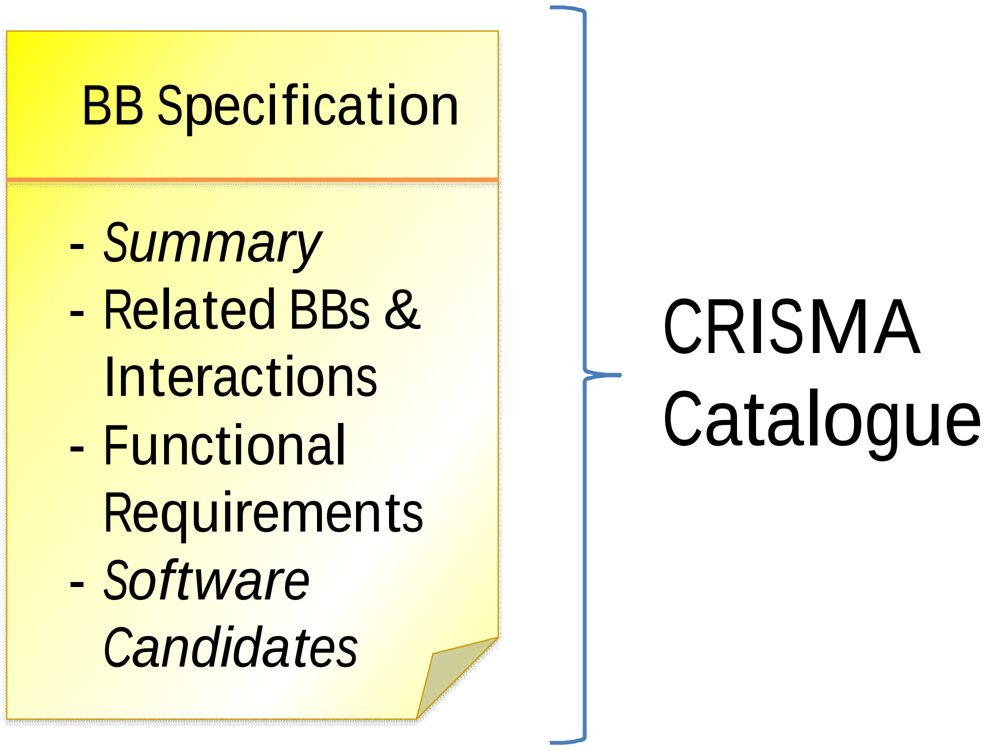


Figure 65: Building Block Specifications in CRISMA Catalogue.

Therefore, this document, the second and last specification of the CRISMA Framework Architecture, follows the same approach. Nevertheless, to make this deliverable a self contained document, each Building Block is briefly described in the following sections. Those condensed Building Block descriptions are sufficient for the understanding of the role and scope of the Building Block, its main functionalities and its position in the CRISMA Framework.

Developers of the Building Block Software Components or architects of individual CRISMA Applications will find the latest detailed functional specification in the CRISMA Catalogue.

https://crisma-cat.ait.ac.at/bb

As shown in Figure 65, this document provides a brief summary with general information on the Building Blocks (chapter 4) as well as on their implementation (chapter 5) while the catalogue additionally contains functional requirements (D31.2, 2013) and information on relations and interaction (diagrams, etc.).

4.1. Infrastructure building blocks

Infrastructure Building Blocks are working in the background building the core of the CRISMA Framework. Figure 66 shows all Infrastructure Building Blocks defined in the final Architecture specification of the CRISMA Framework.

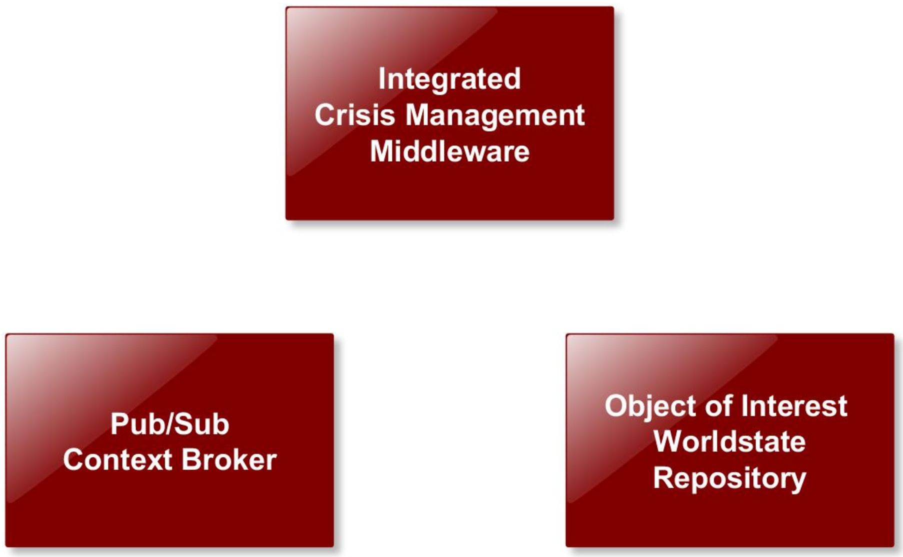

Integrated Crisis Management Middleware
Pub/Sub Context Broker
Object of Interest Worldstate Repository

Figure 66: Infrastructure Building Blocks of the CRISMA Framework.

4.1.1. Integrated crisis management middleware

The Integrated Crisis Management Middleware (ICMM) is the Building Block that represents the Middleware Infrastructure of the CRISMA Framework Architecture (see section 3.5). It is a generic, distributed and Integrated Crisis Management Middleware. It relies on concepts of the resource oriented architecture (ROA) and RESTful web services for the management of arbitrary Control and Communication Information Entities. It exposes several RPC based HTTP APIs and provides also an event based publish-subscribe mechanism that allows clients to subscribe to certain properties of a Control and Communication Information Entity. Thereby, clients may use the publish-subscribe mechanism e.g. to monitor a simulation execution or to trigger events that influence a running simulation. In principle, any CRISMA Federate is a client of the ICMM since a fundamental requirement of CRISMA Federates is to be able to interact with the ICMM.

The ICMM has also to support the common, minimal and generic control and Communication Information Models (CCIMs) of the Implementation Architecture (chapter 5) and the individual extension of these models provided by the different CRISMA Application Architectures. Additional functionalities of the ICMM include the ability to define user dependent-views and catalogues of Control and Communication Information, search functions, access control and others.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Integrated-Crisis-Management-Middleware-BB

4.1.2. Object of interest world state repository

The Object of Interest World State Repository (OOI-WSR) is an Infrastructure Building Block and enables archiving, querying and manipulation of OOI World State data. This module serves as a Repository service for OOI data that can be consumed or manipulated by other interaction- or functional building blocks and resource management models.

The OOI-WSR supports the basic functionality to store, query and manipulate OOI data requested by Resource Management related Building Blocks and Federations. User Interaction Building Blocks require a quick and simple way to access World State information in order to visualize it via plain text, GIS display, tabular display and graphs. Functional BBs that analyse simulation results require direct data access in order to generate Key Performance Indicators (KPI) and identify trends. CRISMA Models require access to the initial OOI World State and update the new World State based on simulation results.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/OOI-World-State-Repository-BB

4.1.3. Publish/subscribe context broker

The Publish/Subscribe (Pub/Sub) Context Broker is an Infrastructure Building Block and a cross-over between an event broker which accepts events and dispatches them to subscribers and an access service providing the information on the current state of the "world".

It simplifies the design of CRISMA Applications by minimizing the need for direct interaction between CRISMA Federates (and Building Blocks, respectively) and for hard-coding such interactions. A Context Broker exposes the (standard) interfaces for retrieval of the context information, events and other data from the Context or Data/Event Producers by the Context or Data/Event Consumers. The consumer doesn't need to know where the data are located and what the native protocol for their retrieval is. It will just communicate to the Context Broker through a well-defined interface specifying the data it needs in a defined way: on request or on subscription basis.

Event handling is also a central concept of the Integrated Crisis Management Middleware (ICMM). The Publish/Subscribe Context Broker Building Block can be used to transparently extend the ICMM with event subscription and event delegation functionality.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Publish-Subscribe-Context-Broker-BB

4.2. Integration building blocks

Integration Building Blocks are used for the integration of data and Simulation Models into a CRISMA Application. Figure 66 shows all Infrastructure Building Blocks defined in the final Architecture specification of the CRISMA Framework.

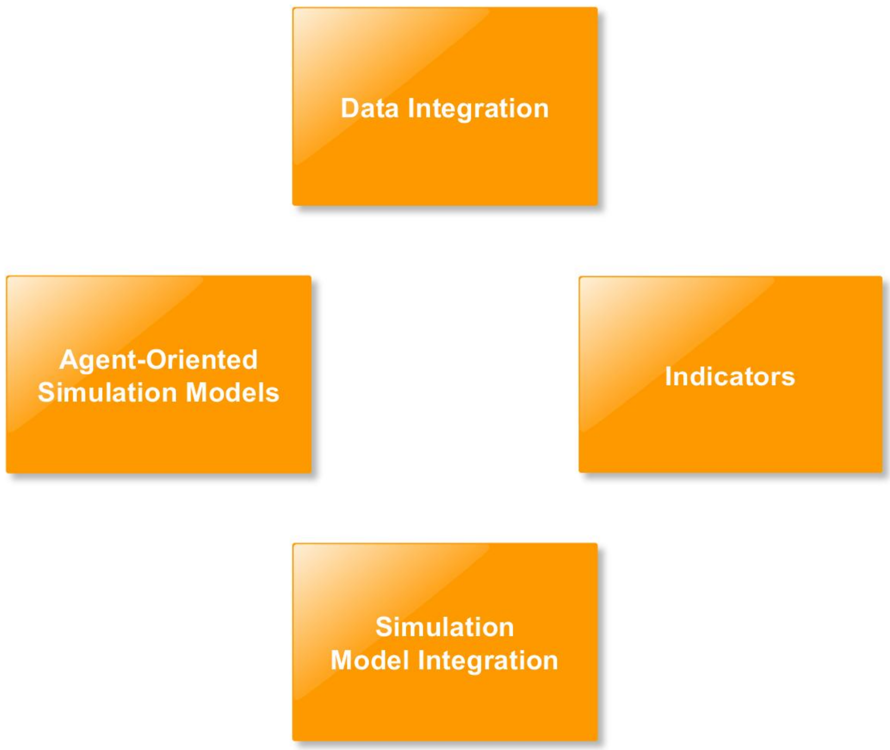

Data Integration
Agent-Oriented Simulation Models
Indicators
Simulation Model Integration

Figure 67: Integration Building Blocks of the CRISMA Framework.

Please note that from the architecture’s point of view and explained in section 3.19 in contrast to Models for Multi-Sectoral Consequences (SP4) application-specific Simulation Models are not Building Blocks of the CRISMA Framework. However, the components that are needed for the integration and/or implementation (like the Agent-Oriented Simulation Models Building Block) of the respective Simulation Models are Building Blocks.

4.2.1. Agent-oriented simulation models

The Agent-Oriented Simulation Models Building Block serves for the development of dynamic maps – specific (individual-based) Simulation Models composed of interacting software agents situated in some environment. This Building Block comprises a collection of generic agents and interaction templates for dynamic map construction, for describing, defining and specifying points, areas and layers of interest.

It provides furthermore a dynamic-map-based user interface for interaction and visualization. Thus, it can be considered both an Integration and User Interaction Building Block.

For decision support in crisis management it is essential to provide good situational awareness, including data fusion from and dissemination to different sources. Agent-oriented simulations, e.g. agent-based simulations of the crisis evolving in some environment, include Simulation Models for the environment and agents' behaviours and interaction patterns. Simulations serve for developing better understanding of situations and relationships between processes in the crisis area/aspect of interest.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Agent-Oriented-Simulation-Models

4.2.2. Data integration

The Data Integration Building Block is an Infrastructure Building Block that provides components that can be used to easily offer data in a CRISMA-compliant (OGC open standard compatible) way so that other Building Blocks may use them for further processing like viewing or editing. That way data can be made accessible for CRISMA components. The envisioned technique to make data available is to use standardized OGC WMS (WMS, 2006), WFS-T (WFS, 2010) and SOS-T (SOS, 2012) services. Future versions might include WCS (WCS, 2010) and WCPS (WCPS, 2009) also.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Agent-Oriented-Simulation-Models

4.2.3. Indicators building block

The Indicators Building Blocks supports the execution of Indicator Functions which is a central concept of the Conceptual Business Logic of CRISMA. According to this, the Indicators Building Block is used for calculation of World State indicators, key performance indicators. Indicators are calculated in order to provide feedback regarding the impact of a specific crisis management action and to allow comparison between different decisions. Impact in this sense refers to the question of how specific actions affect the current mission (i.e. material costs, lost productivity, number of death, time needed for normalization).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Indicator-Building-Block

4.2.4. Simulation model integration

The Simulation Model Integration Building Block is an Integration Building Block and provides components that can be used to integrate Simulation Models in a CRISMA Application. Being aware of the heterogeneity of existing Simulation Models CRISMA has to ensure that they can be integrated in a well-defined and standardized way and ultimately become a CRISMA Federate.

The approach to be used is the so called wrapping. The Simulation Model Integration Building Block provides a piece of software that can be used to make (existing) Simulation Models CRISMA-aware with as little effort as possible with respect to the specifics of the different Simulation Models.

The Simulation Model integration approach of the CRISMA Framework Architecture (section 3.14) is based on OGC Web Processing Service (WPS, 2007). Thereby, also several Simulation Models may be exposed as one WPS interface thus supporting model chaining, the simulation of cascading events, etc.). For developers, it is important to note that the Simulation Model Integration Building Block does not have to expose an API other than that supported by the OGC WPS standard.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Simulation-Model-Integration-BB

4.3. User interaction building blocks

This section describes the User Interaction Building Blocks that provide GUIs allowing direct interaction of the end user. They constitute the graphical user interface of a CRISMA Application. Figure 68 shows all User Interaction Building Blocks defined in the final Architecture specification of the CRISMA Framework.

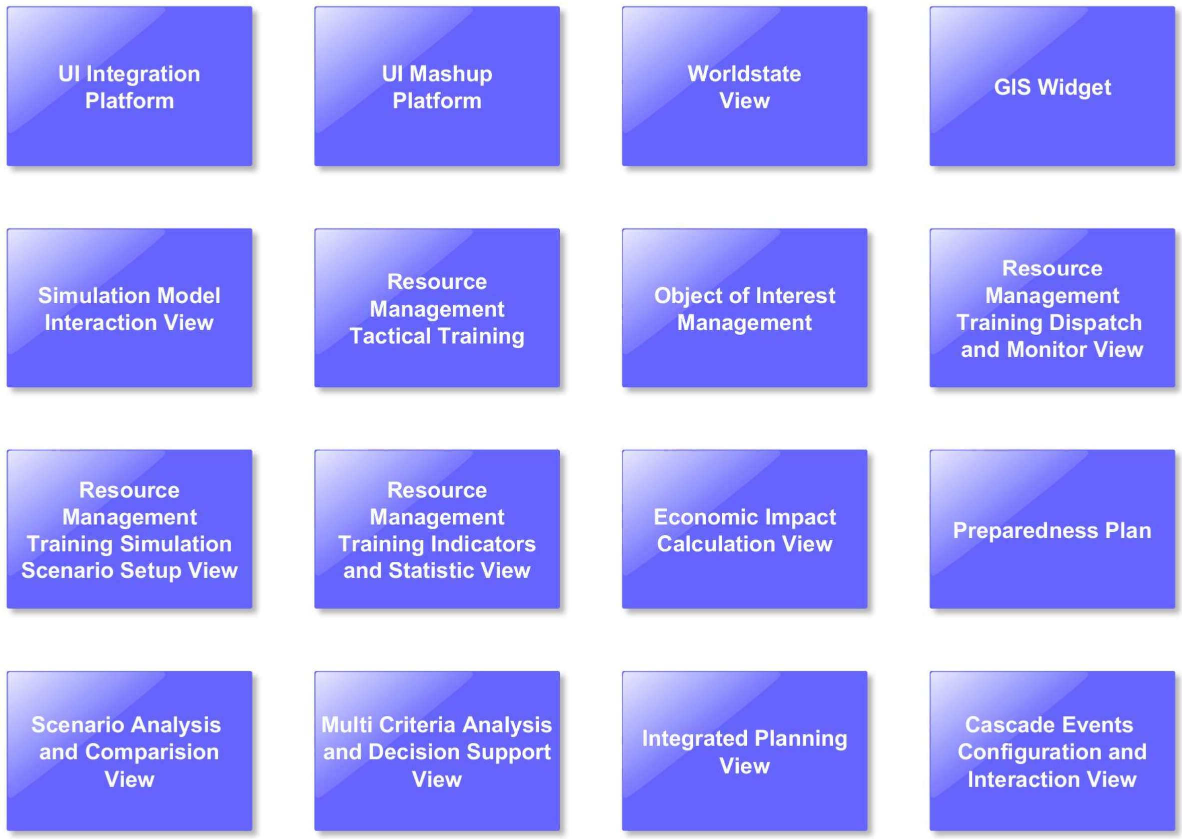

UI Integration Platform
UI Mashup Platform
Worldstate View
GIS Widget
Simulation Model Interaction View
Resource Management Tactical Training
Object of Interest Management
Resource Management Training Dispatch and Monitor View
Resource Management Training Simulation Scenario Setup View
Resource Management Training Indicators and Statistic View
Economic Impact Calculation View
Preparedness Plan
Scenario Analysis and Comparision View
Multi Criteria Analysis and Decision Support View
Integrated Planning View
Cascade Events Configuration and Interaction View

Figure 68: User Interaction Building Blocks of the CRISMA Framework.

4.3.1. Cascade events configuration and interaction view

The Cascade Events Configuration and Interaction View is a User interaction Building Block that allows a user to configure and run a Cascade Events and Effects Scenario. The user can select a triggering event (e.g. an earthquake). On the basis of the selection the View shows the possible cascade events that are available according to the respective Transition Matrix that contains the possible Paths of Analysis. In this way, the user may select a specific chain of events.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Cascade%20Events%20Configuration%20and%20Interaction%20View

4.3.2. Economic impact calculation view

This Building Block is an economic evaluation tool to support crisis management and to be used in the preparedness phase for planning and training purposes. The main objective of an economic evaluation in CRISMA is:

- to present the Economic Impacts arising from crises (ex post performance) and
- to assess different mitigation options and their costs/benefits (ex ante planning).

This Building Block‘s purpose is two-fold. Firstly, to collect cost related data (economic parameters) which are not readily available from the World State description and are needed in Economic Impact calculation from a user thus serving as Simulation Configuration Widget. Secondly, its purpose is to show results related to economic impact analysis.

From an architectural point of view this Building Block is a GUI client invoking the Economic Impact Calculation Service which is provided by SP4 - Models for Multi-Sectoral Consequences (Task 4.4.2 - Simulation of Economic Impacts and consequences).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Economic-Impact-Calculation-View

4.3.3. GIS widget

The GIS Widget is a User Interaction Building Block that enables the visualisation and manipulation of geo-spatial data. Geospatial data plays a predominant role in all crisis management related applications, because most if not all information playing a role in crisis management has a geospatial component.

The GIS Widget must be able to handle data delivered by the Data Integration Building Block (section 4.2.2) but also other well-known GIS formats. Furthermore, it must be able to visualise geo-spatial data from multiple sources simultaneously to allow a comparison of different scenarios (GIS layers). Moreover, the GIS Widget shall not only be able to show static information like background maps (e.g. road transport infrastructure, topographic maps) but also dynamic content which is updated according to a progressing Crisis Management Simulation.

For example, the current location of resources and actors, as well as the status of other objects of interest should be shown in their spatial context. Additional user interaction features of the GIS Widget include, for example, the ability to interactively manipulate objects on the map.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/GIS-Widget-BB

4.3.4. Integrated planning view

The Integrated Planning View is a generic integrated view for the configuration and inspection of arbitrary Crisis Management Scenarios in planning situations. In (D25.1, 2014) the Integrated Planning View is described as interaction concept that can be tailored to all use cases related to planning tasks. It is a suitable reuse of common components and ensures a harmonized usability and user experience in CRISMA applications that are built on the basis of the Integrated Planning View.

Thereby, the Integrated Planning View is realised by combining several other User Interaction Building Blocks and their Software Components (Widgets), respectively, to support application-specific planning use cases.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Integrated%20Planning%20View

4.3.5. Multi-criteria analysis and decision support view

The Multi Criteria Analysis and Decision Support View is a User Interaction Building Block that allows performing a ranking of different Crisis Management Scenarios with respect to specific Criteria. Together with the Scenario Analysis and Comparison View it realises the concepts for Crisis Management Simulation Analysis as defined by the CRISMA Framework Architecture (section 3.1.3) and the Model for Decision-Making Assessment (D44.1, 2013). While the Scenario Analysis and Comparison View only allows a comparison of Indicators and Criteria for different scenarios, the Multi Criteria Analysis and Decision Support View adds supplemental decision support functionalities.

The Multi Criteria Analysis and Decision Support View is composed of two different Widgets: The Decision Strategy Widget and the Decision Ranking Widget. The Decision Strategy Widget allows defining a weighting strategy for different Criteria to ultimately allow a ranking of Crisis Management Scenarios on basis of a Multi Criteria Analysis. Thereby, a static weighting factor can be assigned to each Indicator. This factor specifies the contribution of the particular Criteria to the overall level of satisfaction of a particular Crisis Management Scenario. Additionally, a dynamic weighting factor can be specified which emphasises specific Criteria in relation to the achieved level of satisfaction according to the OWA (Ordered Weighted Averages) method (refer to D44.1 for more details).

The Decision Ranking Widget allows selecting a previously defined Decision Strategy and to apply it to a set of Crisis Management Scenarios. Crisis Management Scenarios are represented by World States (leaf nodes in a World State Tree).

The Decision Support View can therefore be invoked from the Worldstate View (section 4.3.16) by selecting different World States for the comparison. After the Decision Strategy has been applied, the widget will present a list of World States ordered according to their level of satisfaction. Further visualisation and comparison of Indicators and Criteria can then be performed with help of the Scenario Analysis and Comparison View (section 4.3.12).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Multi%20Criteria%20Analysis%20and%20Decision%20Support%20View

4.3.6. Object of interest management

The Object of Interest (OOI) Management Building Block is a User Interaction Building Block that enables the user to view and edit the actual data available for a specific scenario or simulation.

It enables the system administrator to detect the OOI data type’s properties based on the OOI Information Models definitions. It also enables the user to monitor and edit a specific snapshot context (time, user, and workflow).

The OOI Management Building Block has a generic UI for creating and instantiating different data types. It enables creating a new instance from an existing data type or editing an existing object instance. Support for new data types can be done by creating a dedicated plug-in that contains a data adapter and an optional UI data visualization.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/OOI-Management-BB

4.3.7. Preparedness plan

The Preparedness Plan Building Block represents a decision support mechanism which helps the decision maker to allocate resources for response and recovery tasks in different scenarios according to pre-planned patterns and up-to-date resource and situation data.

Different phases of an emergency require different kinds of actions. For example, in the early warning phase the actions can be reservations of resources according to existing contracts. If the situation further evolves to an actual emergency, the proposed actions may be e.g. dispatching of the response units or taking actions needed for evacuation of the affected people. Proposed actions for a certain situation may be provided, for example, as check lists or communication plans.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Preparedness%20Plan%20BB

4.3.8. Resource management tactical training

The Resource Management Tactical Training Building Block (RMTT) enables a Trainee to learn emergency management in a virtual environment. The Trainee-User is a control room operator or an on-scene commander.

The Trainee-User needs to be trained in his natural environment, which is the command and control, information management or situational management application.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Resource-Management-Tactical-Training-BB

4.3.9. resource management training dispatch and monitor view

The Resource Management Training Dispatch and Monitor View is a User Interaction Building Block realised as Mashable Composite UI Module. It provides a high-level overview on the simulation's state. Its purpose is to display one s state at a time and allow the user to distribute resources (ambulances, etc.) among different areas of the crisis scenario.

Trainee operators are presented with the number of resources available at the current time in the resource management simulation, incidents and OOI states (depending on whether the administrator chose not to hide these details from trainees).

The map contains a predefined number of incidents depending. The operator can dispatch resources to scenes if they are available, or recall already dispatched resources. The operator is also always informed about the state of each resource at his disposal as well as the incident evolvement (aggregated patient states, severity of incidents, etc.).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Resource-Management-Training-Dispatch-and-Monitor-View

4.3.10. Resource management training indicators and statistics view

The Resource Management Training Indicators and Statistics View Building Block is a User Interaction Building Block realised as Mashable Composite UI Module. It focuses on the visual presentation of statistics and Indicators of a given World State in order to provide a quick overview on the situation and to allow the comparison of two given World States. The component performs no complex calculations by itself. It only takes data either directly from the World State, or from other Building Blocks providing such Indicators unless they are not already part of the World State (e.g. the Indicators Building Block).

The CRISMA Resource Management Concept (section 3.18) demands an easy-to-grasp representation of the state and the evolution of resources in a given simulation by visually highlighting the most important details. Resource Management tasks require benefit from the visualization of resource states at a given point in time as well as the distribution of available resources within a simulation run. There is no need for complex number aggregation or arithmetic operations, which allows these visualizations to run on the client and on demand.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Resource-Management-Training-Indicators-and-Statistics-View

4.3.11. Resource management training simulation scenario setup view

The Resource Management Training Simulation Scenario Setup View is a User Interaction Building Block realised as Mashable Composite UI Module.

It allows the creation and modification of resource management simulations. It allows the user to create incidents and scenes as well as to create and manage objects of interest (OOI) instances.

It consists of a map widget displaying the world map as well as layers for items that have been added through a toolbox widget (consisting of simply buttons) that contains controls to create OOI instances and other map elements as well as the ability to manage and edit properties of elements created this way and the scenario itself (name, description).

This is an important view focusing on the creation and management of simulations that are later used as configurations for initial World States. It allows the definition of simulation parameters, placement and configuration of object of interest instances, scenario evolution and other details that form a fully configured simulation for training purposes.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Resource-Management-Training-Simulation-Scenario-Setup-View

4.3.12. Scenario analysis and comparison view

The Scenario Analysis and Comparison View visualises Indicator and Criteria data in a way that users are able to analyse and compare different Simulated Crisis Management Scenarios and ultimately come to a decision which fits the simulation objective best for a specific Simulation Case.

As Indicators and Criteria data have the same format both of them use the same visualisation facilities which are provided by the Indicator Visualisation and Comparison and the Criteria Visualisation and Comparison Widgets. Those Widgets provide a plain table-like visualisation for all values of all select World States. Additionally, they provide a bar chart visualisation to give an impression on how much the single values differ. Moreover, the widgets are able to visualise the data as spider charts in order to give a fast impression of the overall performance of the different scenarios.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Scenario-Analysis-and-Comparison-View

4.3.13. Simulation model interaction view

The Simulation Model Interaction Widget Building Block is a Composite UI Module that lets end users interact with the various simulation models exposed by the Simulation Model Integration Building Block. It is responsible for collecting simulation model parameters, launching the respective simulation model, showing status information and retrieving simulation model results. It also constitutes the GUI to the Simulation Model Integration Building Block.

Since the current model integration approach described in section 3.14 of this document is based on OGC Web Processing Service (WPS), the Simulation Model Interaction Widget is essentially a CRISMA-aware WPS client.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/Simulation-Model-Interaction-View

4.3.14. UI mashup platform

The UI Mashup Platform is an Infrastructure, Integration and User Interaction Building Block that acts as a runtime environment for Mashable Composite UI Modules (widgets) and provides inter-widget communication, persistent per-widget configuration as well as proxy capabilities.

The platform allows the user (in this case the person configuring a CRISMA application) to create a HTML5-based front-end by selecting certain user components (widgets and operations) that can be combined into a mashup. The user components are provided by the catalogue, and can be installed and connected (“wired”) into a mashup using a graphical editor.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/UI-Mashup-Platform

4.3.15. UI integration platform

The UI Integration Platform is both a User Interaction and Integration Building Block that is able to host Composite UI Modules. Composite UI Modules are User Interaction Building Blocks that are realised as HTML5 and JavaScript widgets. The UI Integration Platform BB constitutes the Runtime Environment of the Composite UI Modules as they - by their nature - cannot be used as stand-alone applications.

In general, the UI Integration Platform will not be used to render a single Composite UI Module but a set of Mashable Composite UI Modules (e.g. Wirecloud Widgets) which have been combined into a Mashup Application with help of the UI Mashup Platform (e.g. Wirecloud).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/UI-Integration-Platform-BB

4.3.16. World state view

The World State View Building Block is a set of generic Widgets (Mashable Composite UI Modules) that allow visualising Control and Communication Information (CCIM) related to World States and World State Transitions. Thus, it operates mainly on common meta-information about the world rather than the real data of the world (World State Data Slots).

The World State View consists of the following four independent and generic Widgets:

- **Scenario Evolution Widget**
  The Scenario Evolution Widget visualises the evolution of a specific scenario from the beginning (the "initial" World State or Root World State) to the currently selected World State (leaf in the World States tree).
- **Scenario List Widget**
  The Scenario List Widget visualises the currently available Scenarios.
- **World State Tree Widget**
  The World State Tree Widget visualises the World State Trees of a CRISMA Application.
- **World State Widget**
  The World State Widget is capable of visualising and editing the details of a World State, in fact the actual World State data (with the help of World State specific extensions). As a World State may actually contain lots of information the World State Widget creates a logical grouping of several aspects and visualises a single group with full detail whereas all the other groups are visualised via miniature views.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/bb/World State-View

4.4. Interoperability standards

This section gives an overview on standards investigated due to their relevance for CRISMA.

The standards are relevant for different aspects of the architecture:

- **Data Integration**
  For data integration two aspects need to be considered: How to process data (syntactic interoperability) and how to represent the meaning of data (semantic interoperability)
- **Simulation Model Integration**
  How to control model execution, but also how to describe models, parameters and data.
- **Meta-Information**
  Standards used to describe data and Simulation Models. While this is required in general it is especially important for Cascade Events where it is necessary to

decide if the output from one simulation model is usable as input for the next model. This topic relates to all other standards.

- **Communication**
  Standards specifying the transport of data, including notifications.
- **User interfaces and data presentation**
  Since there is the requirement for modular user interfaces, allowing parts to be embedded in other solutions, the most relevant standard for CRISMA user interfaces will be HTML5 and JavaScript.
- **Authentication and Authorisation**
  Standards used control the access to information or services.

4.4.1. Open geospatial consortium and opengis standards

There are a many standards from OGC (Open Geospatial Consortium) relevant for Data- and Simulation Model Integration. These standards describing data encodings, meta-information and service interfaces are well established in the GIS community and beyond.

All OGC standards documents can be found at http://www.opengeospatial.org.

CRISMA relevance: The OGC standards are relevant for CRISMA since many of the data sources and Simulation Models that will be integrated into CRISMA are already accessible in an OGC compliant way. Moreover, there are explicit user requirements asking for the usage of OGC standards.

4.4.1.1. SWE common data model encoding standard

"The Sensor Web Enablement (SWE) Common Data Model Encoding Standard defines low level data models for exchanging sensor related data between nodes of the OGC® Sensor Web Enablement (SWE) framework. These models allow applications and/or servers to structure, encode and transmit sensor datasets in a self-describing and semantically enabled way." (SWEdata, 2011)

CRISMA relevance: Describes basics for data encoding, used by OGC services.

4.4.1.2. SWE service model implementation standard

"This standard currently defines eight packages with data types for common use across OGC Sensor Web Enablement (SWE) services. Five of these packages define operation request and response types. The packages are:

1) *Contents – Defines data types that can be used in specific services that provide (access to) sensors;*
2) *Notification – Defines the data types that support provision of metadata about the notification capabilities of a service as well as the definition and encoding of SWES events;*
3) *Common - Defines data types common to other packages;*
4) *Common Codes –Defines commonly used lists of codes with special semantics;*
5) *DescribeSensor – Defines the request and response types of an operation used to retrieve metadata about a given sensor;*
6) *UpdateSensorDescription –Defines the request and response types of an operation used to modify the description of a given sensor;*
7) *InsertSensor – Defines the request and response types of an operation used to insert a new sensor instance at a service;*

8) *DeleteSensor – Defines the request and response types of an operation used to remove a sensor from a service.*

These packages use data types specified in other standards. Those data types are normatively referenced herein, instead of being repeated in this standard." (SWEservices, 2011).

CRISMA relevance: Describes the basics for other OGC model integration standards.

4.4.1.3. Sensor model language encoding standard

"The OpenGIS® Sensor Model Language Encoding Standard (SensorML) specifies models and XML encoding that provide a framework within which the geometric, dynamic, and observational characteristics of sensors and sensor systems can be defined. There are many different sensor types, from simple visual thermometers to complex electron microscopes and earth observing satellites. These can all be supported through the definition of atomic process models and process chains. Within SensorML, all processes and components are encoded as application schema of the Feature model in the Geographic Markup Language (GML)" (SensorML, 2007).

CRISMA relevance: This standard can be used to describe Simulation Models in an OGC compliant way. The usage of this standard depends on the Software Components actually used.

4.4.1.4. Web map service

"The OpenGIS® Web Map Service Interface Standard (WMS) provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc) that can be displayed in a browser application. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined or not." (WMS, 2006)

Note: WMS 1.3 is also described in ISO/DIS 19128.

CRISMA relevance: WMS provides maps as rastered images.

4.4.1.5. Web coverage service

"The OGC Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space/time-varying phenomena. This document specifies the WCS core; every implementation of a WCS shall adhere to this standard. This standard thus defines only basic requirements. Extensions to the core will define extensions to meet additional requirements, such as the response encoding. Indeed, additional extensions are required in order to completely specify a WCS for implementation. A WCS provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients." (WCS, 2010)

CRISMA relevance: This service can deliver the same data as the WMS, but in a format allowing further processing. This might be used for the chaining of Simulation Models to process Cascade Events.

4.4.1.6. Web coverage processing service

“The OpenGIS® Web Coverage Processing Service Interface Standard (WCPS) defines a protocol-independent language for the extraction, processing, and analysis of multi-dimensional gridded coverages representing sensor, image, or statistics data.

Services implementing this language provide access to original or derived sets of geospatial coverage information, in forms that are useful for client-side rendering, input into scientific models, and other client applications.” (WCPS, 2009)

CRISMA relevance: Might be useful not only for chaining of Simulation Models but also to visualize and compare data.

4.4.1.7. Web feature service

“The Web Feature Service (WFS) offers direct fine-grained access to geographic information at the feature and feature property level. Web feature services allow clients to only retrieve or modify the data they are seeking, rather than retrieving a file that contains the data they are seeking and possibly much more. That data can then be used for a wide variety of purposes, including purposes other than their producers' intended ones.

In the taxonomy of services defined in ISO 19119, the WFS is primarily a feature access service but also includes elements of a feature type service, a coordinate conversion/transformation service and geographic format conversion service.” (WFS, 2010)

Note: WFS 2.0 is also described in ISO/DIS 19142

CRISMA relevance: WFS can be used to access structured data including access to simulation results. Unlike WCS this service is not limited to gridded data.

4.4.1.8. Sensor observation service

“The OpenGIS® Sensor Observation Service (SOS) has been designed to provide access to sensor observations. In practical terms “sensor observations” mean data provided by sensors (e.g. weather measurements) as well as model results (e.g. weather forecast maps). So sensor can be either a real measurement device (Temperature sensor, weather observation satellite) or a simulation (Weather forecast model). Many sensors repeatedly observe the same phenomenon, so the result is typically a timeseries of observations, each associated to a time stamp. In contrast to WMS, WFS and WCS there are not many open source implementations available.” (SOS, 2012).

CRISMA relevance: Access to time series data from sensors and simulations. Will be of limited use because of a small usage area and lack of support in popular Software Components.

4.4.1.9. Observations & measurements

“The OGC Observations & Measurements (O&M) standard specifies an XML implementation for the OGC and ISO Observations and Measurements (O&M) conceptual model (OGC Observations and Measurements v2.0 also published as ISO/DIS 19156), including a schema for Sampling Features. This encoding is an essential dependency for the OGC Sensor Observation Service (SOS) Interface Standard. More specifically, this

standard defines XML schemas for observations, and for features involved in sampling when making observations.

These provide document models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities." (O&M, 2011)

<u>CRISMA relevance</u>: One of the standards for data encoding that can be used within OGC services (part of SWE Common).

4.4.1.10. Web processing service

"The OpenGIS® Web Processing Service (WPS) Interface Standard provides rules for standardizing how inputs and outputs (requests and responses) for geospatial processing services, such as polygon overlay. The standard also defines how a client can request the execution of a process, and how the output from the process is handled. It defines an interface that facilitates the publishing of geospatial processes and clients' discovery of and binding to those processes. The data required by the WPS can be delivered across a network or they can be available at the server." (WPS, 2007)

<u>CRISMA relevance</u>: This is the preferred service to be used for Simulation Model Integration (see section 4.2.4). This standard is already used by many already existing Simulation Models that are planned to be integrated into CRISMA and was also used for the prototyping activities.

4.4.1.11. Geography markup language

"The Geography Markup Language (GML) is an XML grammar for expressing geographical features. GML serves as a modelling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. As with most XML based grammars, there are two parts to the grammar – the schema that describes the document and the instance document that contains the actual data. A GML document is described using a GML Schema." (GML, 2012).

<u>CRISMA relevance</u>: This is one of the basic GIS related standards, relevant because most data processed within CRISMA have a spatial reference.

4.4.1.12. ISO 19115

"ISO 19115 is a standard of the International Organization for Standardization (ISO). The standard is part of the ISO geographic information suite of standards (19100 series). ISO 19115 and its parts defines how to describe geographical information and associated services, including contents, spatial-temporal purchases, data quality, access and rights to use.

The objective of this International Standard is to provide a clear procedure for the description of digital geographic datasets so that users will be able to determine whether the data in a holding will be of use to them and how to access the data. By establishing a common set of metadata terminology, definitions and extension procedures, this standard will promote the proper use and effective retrieval of geographic data.

This International Standard defines metadata elements, provides a schema and establishes a common set of metadata terminology, definitions, and extension procedures.

This International Standard defines the schema required for describing geographic information and services. It provides information about the identification, the extent, the quality, the spatial and temporal schema, spatial reference, and distribution of digital geographic data.

This International Standard defines:

- *mandatory and conditional metadata sections, metadata entities, and metadata elements;*
- *The minimum set of metadata required to serve the full range of metadata applications (data discovery, determining data fitness for use, data access, data transfer, and use of digital data);*
- *optional metadata elements – to allow for a more extensive standard description of geographic data, if required;*
- *a method for extending metadata to fit specialized needs."*

(ISO_19115, 2003)

CRISMA relevance: This international standard describes basic concepts used by many other standards concerning Control and Communication Information that might be used in CRISMA.

4.4.1.13. ISO 19119

"The geographic services architecture specified in this International Standard has been developed to meet the following purposes:

- *provide an abstract framework to allow coordinated development of specific services;*
- *enable interoperable data services through interface standardization;*
- *support development of a service catalogue through the definition of service metadata;*
- *allow separation of data instances and service instances;*
- *enable use of one provider's service on another provider's data;*
- *define an abstract framework which can be implemented in multiple ways."*

(ISO_19119, 2003)

CRISMA relevance: International standards about service specifications. Relevant as the CRISMA Framework Architecture is based on a service oriented architecture (SOA).

4.4.2. W3C standards

"The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards. Led by Web inventor Tim Berners-Lee and CEO Jeffrey Jaffe, W3C's mission is to lead the Web to its full potential". (W3C, 2012)

CRISMA relevance: Most of the standards used in WWW have been developed by the W3C. For CRISMA this includes not only Web-Clients but also the communication within the service oriented architecture, either local or over a (public or private) network.

4.4.2.1. *Extensible markup language*

XML is not only one standard. Different aspects are specified by different W3C standards. The most relevant are Extensible Markup Language (XML), Namespaces in XML and W3C XML Schema Definition Language (XSD). (XML, 2010)

CRISMA relevance: XML is one of the basic methods for data encoding for storage and process communication. All the OGC standards are primarily based on XML.

4.4.2.2. *Web design and applications*

"This W3C standard set contains HTML and CSS that are the basics for every web page, extended by graphics format, JavaScript APIs and Canvas. All this is described in W3C standards." (Web Design, 2013).

CRISMA relevance: The set of standards around HTML is highly relevant as this will be the basic technology behind the GUI related Building Blocks.

4.4.3. Other standards

"In computer network engineering, a Request for Comments (RFC) is a memorandum published by the Internet Engineering Task Force (IETF) describing methods, behaviours, research, or innovations applicable to the working of the Internet and Internet-connected systems." (RFC, 2013)

"The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual." (IETF, 2013)

More information can be found at http://www.ietf.org/.

CRISMA relevance: The RFC's defines some basics for the internet which are used for communication in CRISMA – even in a local installation.

4.4.3.1. *RFC 1945 – hypertext transfer protocol 1.0*

RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0 describes the common usage of the HTTP protocol in the early years of the Web. (RFC_1945, 1996).

CRISMA relevance: The first version, still in use, but superseded by an optimized update. The standard describes all basic concepts and operations.

4.4.3.2. *RFC 2616 – hypertext transfer protocol 1.1*

RFC 2616: Hypertext Transfer Protocol – HTTP/1.1 describes the HTTP 1.1 protocol. Quoting from the RFC: "*The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers.*" (RFC_2616, 1999)

CRISMA relevance: Nearly all the communication over the internet uses this as a basis. It is therefore a basis for all the higher level communication standards used in CRISMA.

4.4.3.3. RFC 4918 – web distributed authoring and versioning

"Web Distributed Authoring and Versioning (WebDAV) consists of a set of methods, headers, and content-types ancillary to HTTP/1.1 for the management of resource properties, creation and management of resource collections, URL namespace manipulation, and resource locking (collision avoidance). RFC 2518 was published in February 1999, and this specification obsoletes RFC 2518 with minor revisions mostly due to interoperability experience." (RFC_4918, 1999)

CRISMA relevance: This protocol is foreseen to be used where OGC services, which are mainly targeted to geo data, don't fit.

4.4.3.4. Dublin core

"The Dublin Core metadata terms are a set of vocabulary terms which can be used to describe resources for the purposes of discovery. The terms can be used to describe a full range of web resources (video, images, web pages, etc.), physical resources such as books and objects like artworks." (Dublin Core, 2013)

CRISMA relevance: This is a very basic and therefore relatively easy to provide set of descriptions that might be useful to allow a user to search for information.

4.4.3.5. JavaScript object notation

"JSON (JavaScript Object Notation) is a text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for many languages. The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML." (JSON, 2013)

CRISMA relevance: JSON as lightweight and more flexible alternative to XML is often much easier to use. Especially when exchanging data with an HTML / JavaScript GUI as the one foreseen for CRISMA. JSON was chosen as default encoding standard for CCIMs as described in section 5.2.3.

4.4.3.6. Representational state transfer

"Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant web service design model. The term representational state transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. Fielding is one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification versions 1.0 and 1.1." (REST, 2013)

CRISMA relevance: Even as there is no formal specification for REST it is a widely used style for implementing services. Easy to implement and use, often combined with a lightweight JSON data encoding it is ideal for services that need not to provide any of the formal service interfaces.

4.4.3.7. OpenId

"OpenId is a standard for user identification and often used for single sign on. The user is authenticated by one trusted party (the OpenID provider). Applications don't need to know the users credentials, so there is no need for a user to have a password or identify himself in different ways for each service but only once at his OpenID provider." (OpenID, 2007)

CRISMA relevance: OpenId is the preferred way for single sign on.

4.4.3.8. OAuth

"The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in RFC 5849." (OAuth, 2012) This specification is the same as RFC 4749.

CRISMA relevance: OAuth might be useful to enable Building Blocks to access other Building Blocks on behalf of a user without the need to store and transmit usernames and passwords. The problems with OAuth are interoperability issues between different implementations.

4.4.3.9. OpenMI

"OpenMI is a standard promoted by the OpenMI Association, it is an environment aimed to define a Software Component interface for the interoperation between simulators (Open MI has been designed with deterministic Simulation Models) in a water domain. Model components that comply with this standard can, without any programming, be configured to exchange data during computation (at run-time). This means that combined systems can be created, based on OpenMI-compliant models from different providers, thus enabling the modeller to use those models that are best suited to a particular project. The standard supports semantic connectivity, where the involved models mutually depend on calculation results from each other. Like in HLA a simulator must be OpenMI compliant to interoperate, but differently from it the OpenMI environment provides utilities to support wrapping of legacy code and to configure and deploy components which are not part of the OpenMI standard decreasing in this way the effort of integration with other components; the architecture imposes only few rules on how the wrapper should be implemented. In addition any software language can be used for computational cores, but may require wrapping in a .NET language or Java." (Adinolfi, 2009)

CRISMA relevance: Recommendations: The default Simulation Model Integration Approach of the CRISMA Framework is based on OGC WPS as explained in section 3.14. However, certain concepts of OpenMI like the data adapters became part of the architecture. OpenMI specification and concepts may also further influence the specification of application-specific Control and Communication Information Models for the integration of legacy Simulation Models.

4.4.3.10. NGSI content management

"NGSI Context Management by Open Mobile Alliance (OMA) specifies two interfaces (NGSI-9 and NGSI-10) to a publish / subscribe system. NGSI-9 specifies operations (register, discovery, subscribe, notify) on context. Context is a description of what values

might exist, there units and how they are grouped together. NGSI-10 specifies operations (update, query, subscribe, notify) on actual value changes within one or more contexts." (NGSI, 2010)

CRISMA relevance: While this is just one of many standards in the field of notification services this interfaces are offered by the FI-WARE Orion Context broker used within the CRISMA Framework (see section 5.1.1.3).

5. Implementation

The Implementation Architecture is a realisation of the Functional Architecture which has been specified in the previous chapters Mission, Concepts and Realisation (Figure 71) according to the MRCI schema introduced in section 1.7.2. In reference to section 1.7.3, the main purpose of the Implementation Architecture is to map the Functional Building Blocks to concrete Software Components, to specify a generic Core Control and Communication Information Model (Core CCIM) that is usable by any CRISMA Federation, to provide a template for the specification of individual Application Architectures and to formulate rules and guidelines for the development of Software Components (Figure 71).

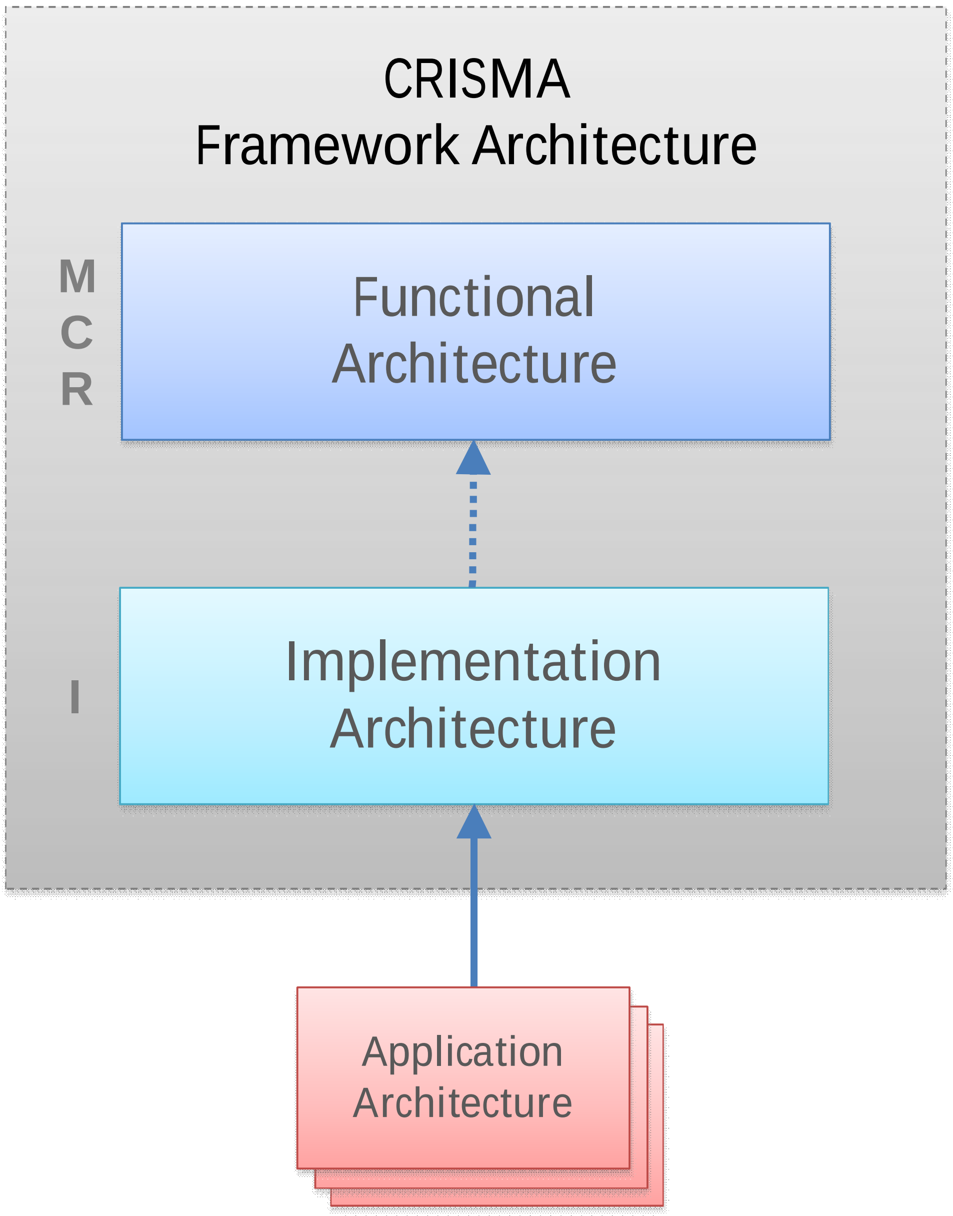


Figure 69: Distinct Parts of the CRISMA Framework Architecture.

The second iteration of the CRISMA Framework Architecture provides a consolidated and complete view on the Implementation Architecture which is a result of the combined effort of WP32 and WP34 (Figure 70). The specification of the Implementation Architecture went hand in hand with the development of the CRISMA Framework and its Building Blocks.

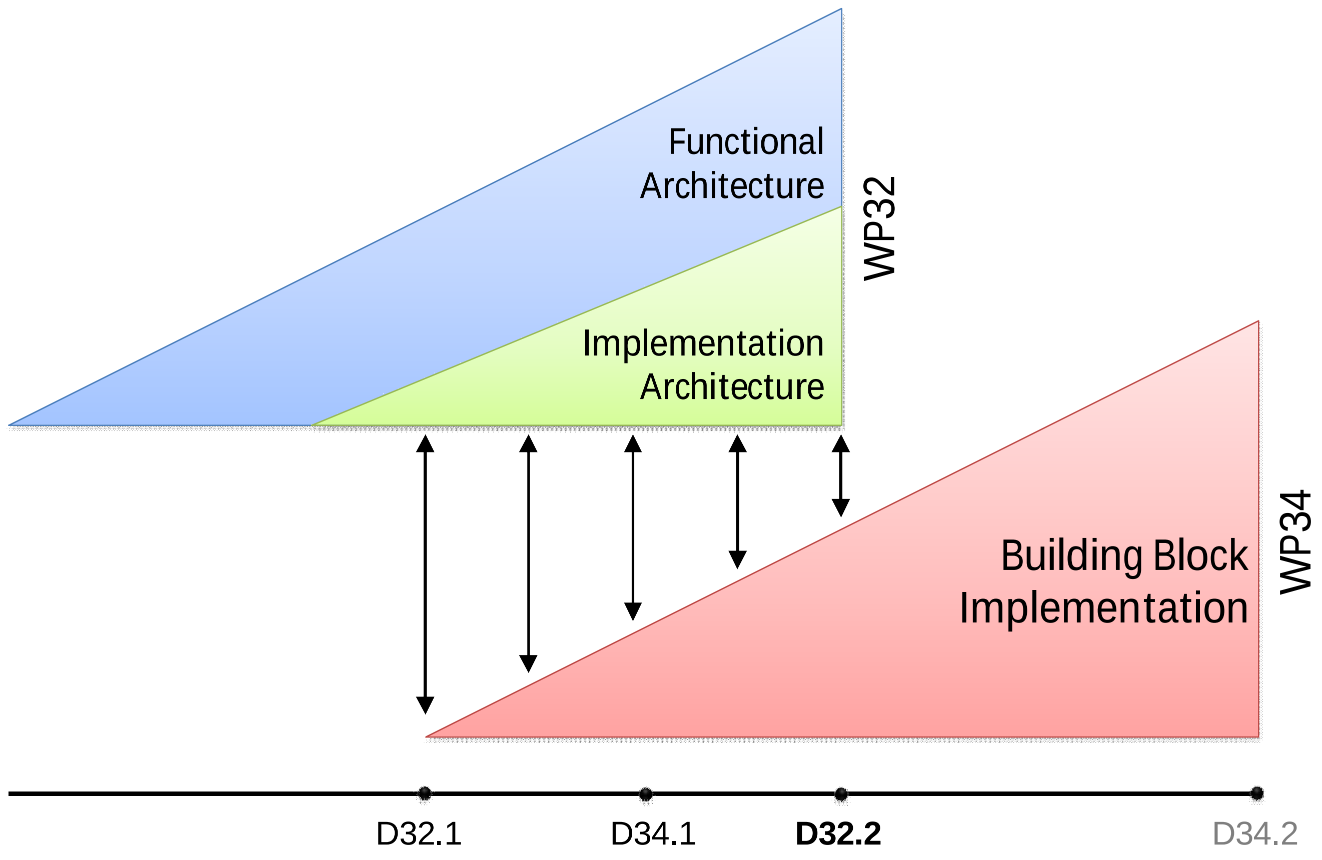


Figure 70: Evolution of the Implementation Architecture.

While the Functional Architecture establishes the basic concepts of the CRISMA Framework and identifies the core Building Blocks, it does neither provide formal specifications nor is it concerned with detailed software design. Software design has been carried out jointly in T3.2.3 “ICMS Implementation Architecture V1”, T3.4.1 “ICMS Building Block Implementation V1” and the respective implementation workpackages of SP4. The implementation is based on an iterative development approach; therefore the Implementation Architecture evolved as the development progressed.

Also shown in Figure 70, the development and partially also the software design process in WP34 continues beyond the end of WP32. Although all Software Components that constitute the CRISMA Framework have been properly identified and described, some updates and improvements are likely to happen. Those updates will be documented in the CRISMA Catalogue as well as in the upcoming deliverable D34.2 “ICMS Building Blocks V2” of WP34.

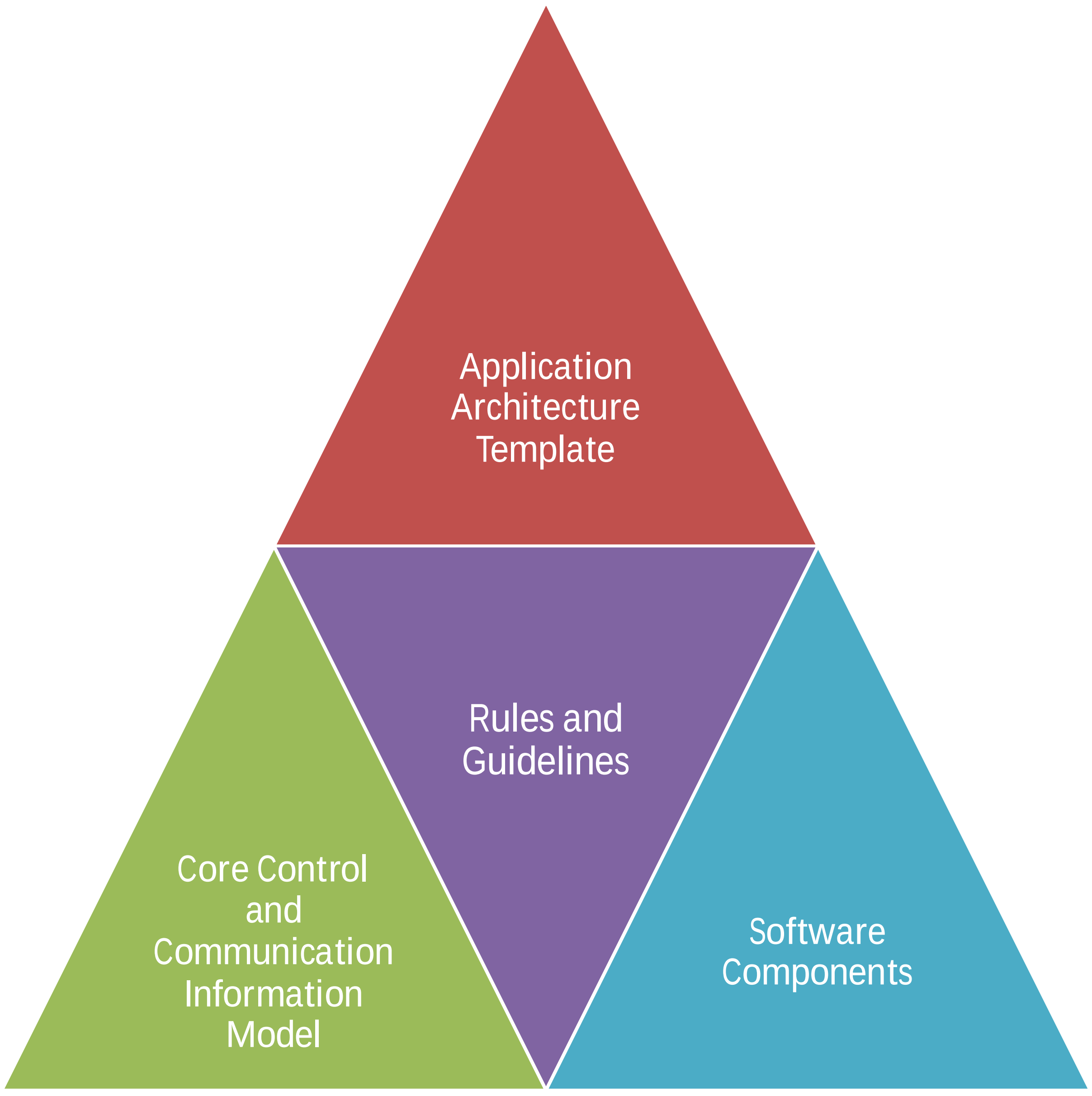

Application Architecture Template
Rules and Guidelines
Core Control and Communication Information Model
Software Components

Figure 71: Implementation Architecture.

Figure 71 recalls the four cornerstones of the final version of the Implementation Architecture which are described in the subsequent chapters: Software Components (section 5.1), the Core CCIM (section 5.1.3.15), the Application Architecture Template (section 5.3) and Rules and Guidelines (section 5.4).

5.1. Software components

According to the high level concepts of the Functional and Implementation Architecture (section 1.7.3), a Software Component is a concrete realisation of a Functional Building Block (Figure 72). The architecture anticipates also the possibility that a Functional Building Block is implemented in several different ways and different Software Components for the implementation of a Functional Building Block do exist.

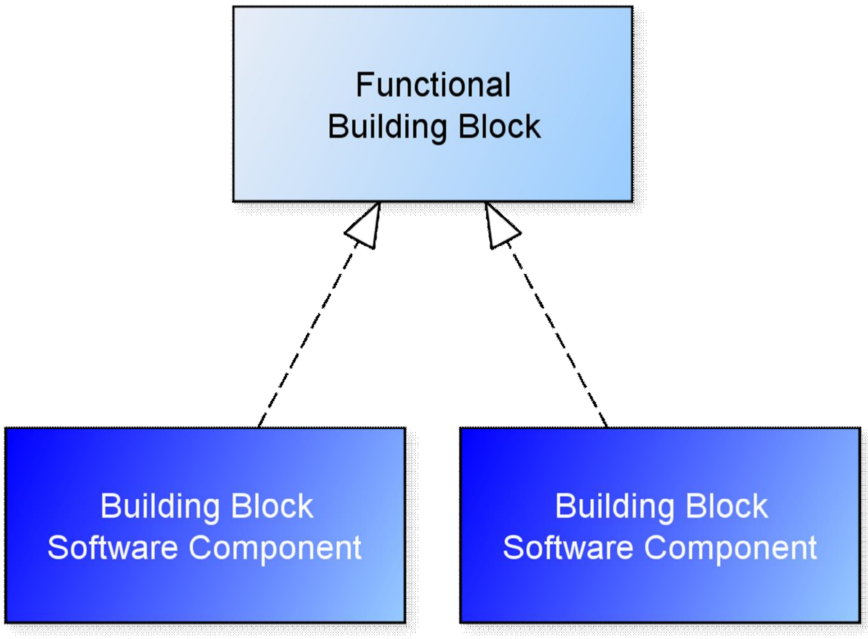


Figure 72: Building Blocks and Software Components.

Realisation in this context does not only encompass providing the functionalities specified in the Functional Building Block Description (chapter 4), but also to ensure that the resulting Software Component complies to the architectural concepts and technological and non-functional requirements as specified in D31.2 “Technology, Concepts & Technical Requirements Report V2” (D31.2, 2013).

Several candidates for Software Components suitable for the implementation of the Functional Building Blocks described in chapter 4 have been identified in the first iteration of the Architecture specification (D32.1, 2013). A Software Candidate is an existing piece of software that is extended, adapted or simply used as-is to realise a specific Functional Building Block. On the one hand, the focus has been on open source software and on the other hand on proprietary software developed and owned by CRISMA partners. Software Components owned by CRISMA partners have been selected because they can easily be adapted to fulfil CRISMA functional and technical requirements. The result of this investigation has been taken up by T3.4.1 “ICMS Building Block Implementation V1” to develop extensions and modifications of the Software Candidates. The results have been reported in the D34.2 “ICMS Building Blocks V2” (D34.1, 2013) and CRISMA Catalogue.

While with the delivery of D32.2 “ICMS Architecture Document V2” most specifications of Functional Building Blocks can be considered as finalised (Figure 73), the implementation of Software Components and the related technical implementation specifications are subject to continuous improvement during the second development phase of WP34 (T3.4.2 “ICMS Building Block Implementation V2”). Therefore, this section of the Implementation Architecture can only provide a snapshot of the current status of Software Component descriptions at the end of the second specification phase of WP32 (Figure 73).

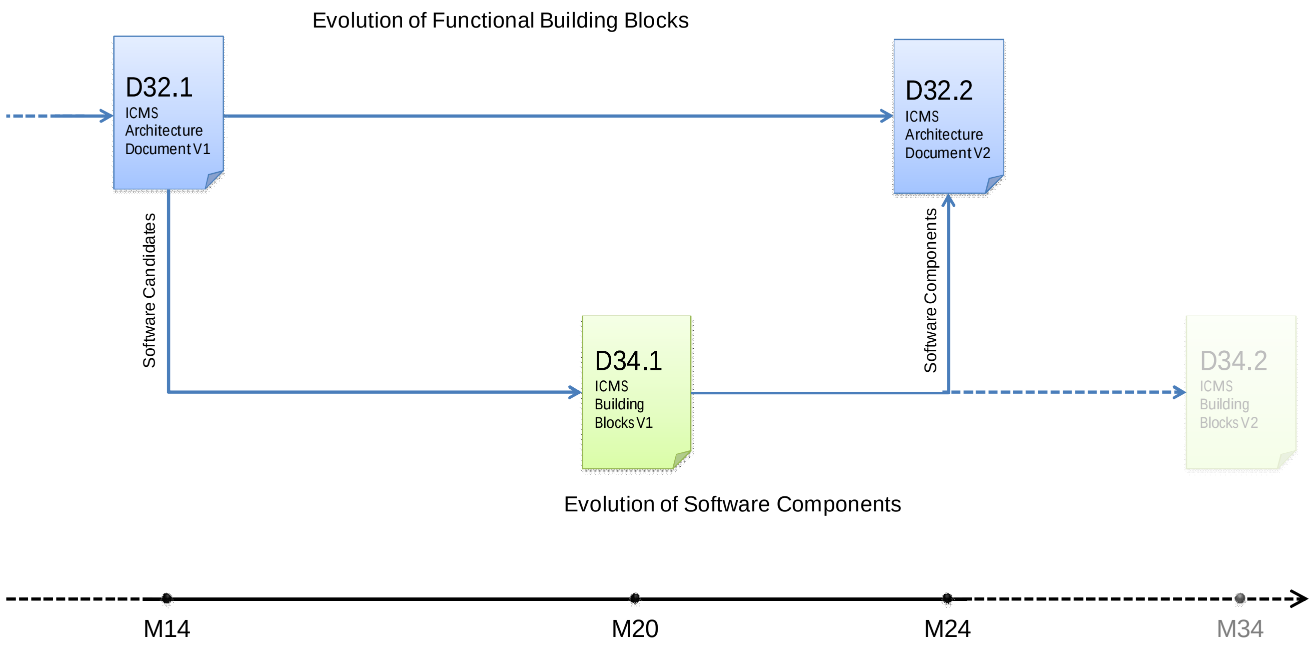

Evolution of Functional Building Blocks
D32.1
ICMS Architecture Document V1
D32.2
ICMS Architecture Document V2
Software Candidates
Software Components
D34.1
ICMS Building Blocks V1
D34.2
ICMS Building Blocks V2
Evolution of Software Components
M14
M20
M24
M34

Figure 73: Evolution of Building Blocks and Software Components.

As already mentioned in the introduction to this chapter, a separate deliverable, D34.2 “ICMS Building Blocks V2”, as well as the CRISMA Catalogue will report on the final status of Software Components. In consequence, this section follows the same approach as the Functional Building Blocks descriptions of chapter 4 and provides only a brief summary of the Software Components along with a reference to the most up to data descriptions in the CRISMA Catalogue.

Table 2 gives a complete overview on Building Blocks and related Software Components that are described in this document.

Table 2: Building Blocks and related Software Components.

Building Block Group	**Name of the Functional Building Block**	**Name of the Building Block Software Component**	**Type of the Software Component**
Simulation Management	Simulation Model Integration	PyWPS	Generic Web Processing Service
	Simulation Model Interaction Widget	Simulation Model Interaction Wirecloud Widget	Web Processing Service Client
	Cascade Events Configuration and Interaction View	*To be defined in Task 3.4.2*	*To be defined in Task 3.4.2*
World State Management	World State View	Scenario Evolution Java Widget	Java GUI Component
		Scenario Evolution Wirecloud Widget	Wirecloud Widget (HTML+JS)
		Scenario List Java Widget	Java GUI Component
		Scenario List Wirecloud Widget	Wirecloud Widget (HTML+JS)
		World State Tree Java Widget	Java GUI Component
		World State Tree Wirecloud Widget	Wirecloud Widget (HTML+JS)

Building Block Group	Name of the Functional Building Block	Name of the Building Block Software Component	Type of the Software Component
		World State Java Widget	Java GUI Component
		World State Java Wirecloud	Wirecloud Widget (HTML+JS)
World State Data Management	GIS Widget	cismap	Java GUI Component
		OpenLayers	JavaScript Framework
		Leaflet.js	JavaScript Framework
	Object of Interest World State Repository	Object of Interest World State Storage Service	OGC Web Processing Service
	Data Integration	GeoServer	OGC Web Map Service and OGC Web Feature Service
		MapServer	OGC Web Map Service and OGC Web Feature Service
Preparedness Planning	Preparedness Plan	Insta Preparedness Planner	Rich Internet Application
Indicator Management	Indicators Building Block	PyWPS	Web Processing Service
Economic Impact	Economic Impact Calculation View	Economic Impact Calculation Setup Wirecloud Widget	Wirecloud Widget (HTML+JS)
		Economic Impact Calculation Results Wirecloud Widget	Wirecloud Widget (HTML+JS)
Resource Management	Object of Interest Management	Object of Interest Management	Wirecloud Widget (HTML+JS)
	Resource Management Training Dispatch and Monitor View	Resource Management Training Dispatch and Monitor Wirecloud Widget	Wirecloud Widget (HTML+JS)
	Resource Management Training Simulation Scenario Setup View	Resource Management Training Simulation Scenario Setup Wirecloud Widget	Wirecloud Widget (HTML+JS)
	Resource Management Training Indicators and Statistic View	Resource Management Training Indicators and Statistic Wirecloud Widget	Wirecloud Widget (HTML+JS)
	Resource Management Tactical Training View	NICE Situator Training System	Desktop Application
	Agent-Oriented Simulation Platform	Dynamic Map Agents	Rich Internet Application
World State and Scenario Analysis	Multi Criteria Analysis and Decision Support View	*To be defined in Task 3.4.2*	*To be defined in Task 3.4.2*
	Scenario Analysis and Comparison View	*To be defined in Task 3.4.2*	*To be defined in Task 3.4.2*
	Integrated Planning View	*To be defined in Task 3.4.2*	*To be defined in Task 3.4.2*

The following subsections provide a brief description of the Software Components listed in Table 2 and a relation to their Building Block.

5.1.1. Infrastructure software components

Figure 74 shows the Infrastructure Software Components that have been identified and described for the Implementation Architecture of the CRISMA Framework.

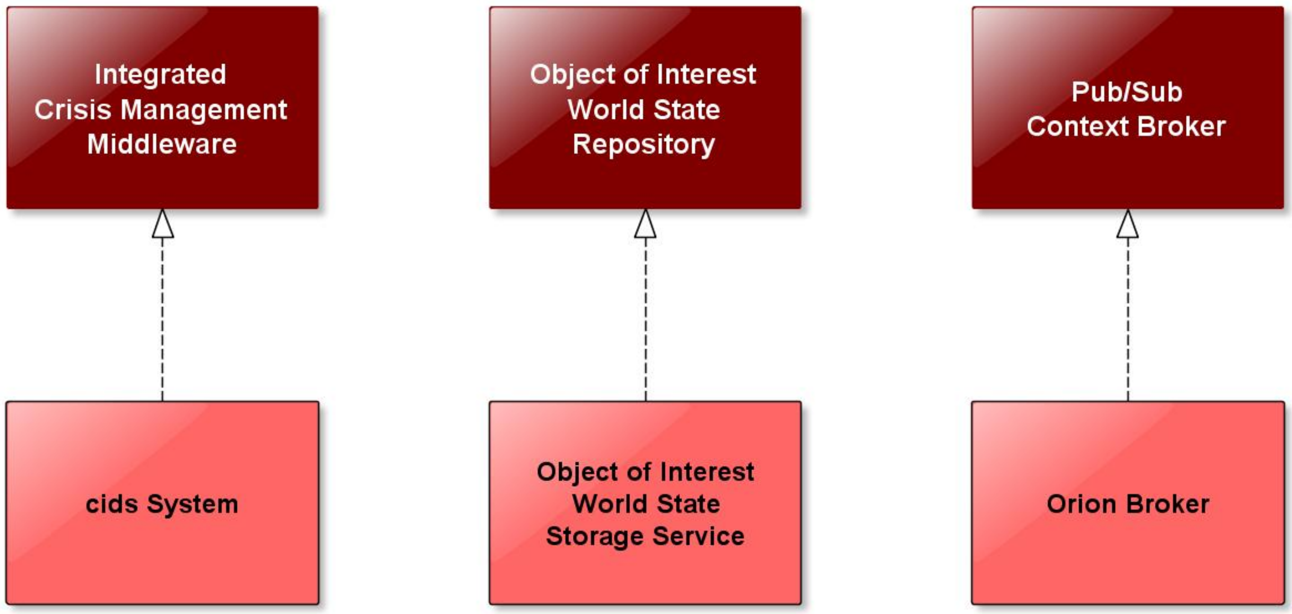

Integrated Crisis Management Middleware
Object of Interest World State Repository
Pub/Sub Context Broker
cids System
Object of Interest World State Storage Service
Orion Broker

Figure 74: Infrastructure Software Components and related Building Blocks.

5.1.1.1. Cids system

Related Building Block: Integrated Crisis Management Middleware (section 4.1.1)

The ICMM is implemented by CRISMA partner CIS on basis of the cids system, an open source product owned by CIS. The cids system is licensed under GNU Lesser General Public License (LGPL, 2007) and can be used free of charge during and beyond the project lifetime.

The cids product suite consists of a set of services, applications, software components, management tools, development tools, and application programming interfaces (APIs) for the management, integration, and development of heterogeneous information systems with a special focus on interactive geo-spatial systems.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/cids-System

5.1.1.2. Object of interest world state repository

Related Building Block: Object of Interest World State Repository (section 4.1.2)

The OOI-WSR is implemented by CRISMA partner NICE as new Software Component. It is licensed under GNU Lesser General Public License (LGPL, 2007) and can be used free of charge during and beyond the project lifetime. The OOI-WSR is developed entirely from scratch in phase one of WP34.

OOI-WSR is a Resource Management related module that enables archiving, querying and manipulation of OOI World State data. This module serves as a Repository service for OOI data that can be consumed or manipulated by other Integrated Components, CRISMA Federates and resource management models.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/OOI-World-State-Repository

5.1.1.3. Orion context broker

Related Building Block: Publish/Subscribe Context Broker (section 4.1.3)

The Software Component selected for implementing the Publish/Subscribe Context Broker Infrastructure Building Block of the CRISMA Framework is the Orion Broker, a Generic Enabler in the Future Internet Public Private Partnership (FI-PPP) Project and was implemented by the FI-Ware partner Telefonica[8]. The licensing conditions are not clearly stated but the usage is free of charge for partners (i.e. AIT). The software will most likely be distributed as Open Source in the future.

Using the Orion Context Broker one is able to register context elements and manage them through updates and queries. In CRISMA for instance a “World State” or a “resource” can be seen as a typical context element. Subscriptions can be used to track changes of context elements with respect to some predefined conditions.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Orion-Context-Broker

5.1.2. Integration software components

Figure 75 shows the Integration Software Components that have been identified and described for the Implementation Architecture of the CRISMA Framework.

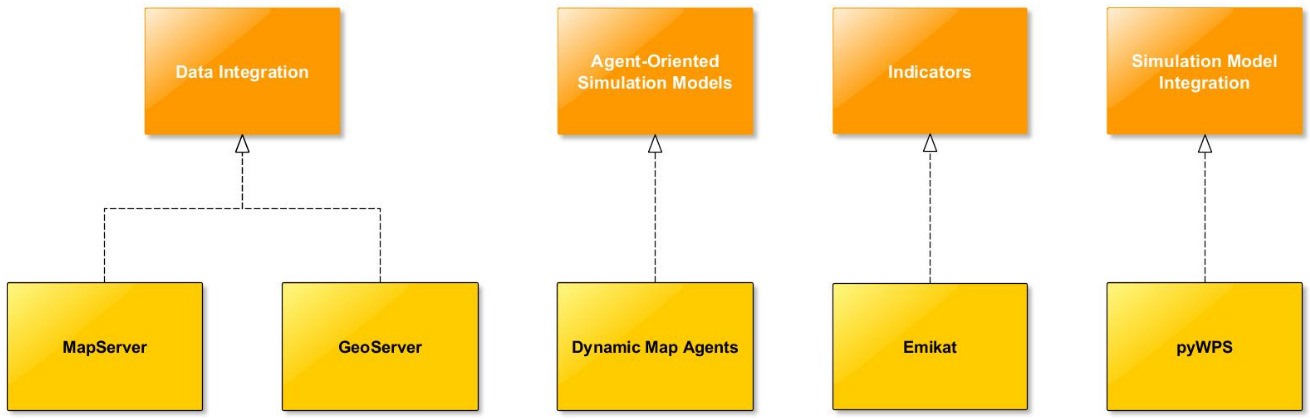


Figure 75: Integration Software Components and related Building Blocks.

[8] http://www.tid.es

5.1.2.1. Agent-oriented simulation models

Related Building Block: Agent-Oriented Simulation Models (section 4.2.1)

The Agent-Oriented Simulation Models is implemented by CRISMA partner TTU on basis of their own Dynamic Map Agents Software.

Dynamic Map Agents is a self-organizing system of networking information and map agents. The multi-layer map is constructed and updated dynamically, on the basis of reciprocal interactions of its actors – the map emerges from activities of its agents (incl. users). The map could be coupled with specific information layers adjusted to particular users and collections of objects-of-interest (e.g. situational information items or (pro-)active information providers) linked with it.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Dynamic-Map-Agents

5.1.2.2. MapServer

Related Building Block: Data Integration Building Block (section 4.2.1)

There exist several Software Components, both commercial and open source products that cover nearly the entire functionality of the Data Integration Building Block. Currently, MapServer is one of the open source products that have been selected for the implementation of the Data Integration Building Block.

"*MapServer is an Open Source geographic data rendering engine written in C. Beyond browsing GIS data, MapServer allows you to create "geographic image maps", that is, maps that can direct users to content. For example, the Minnesota DNR Recreation Compass provides users with more than 10,000 web pages, reports and maps via a single application. The same application serves as a "map engine" for other portions of the site, providing spatial context where needed.*" (http://mapserver.org/about.html)

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/MapServer

5.1.2.3. GeoServer

Related Building Block: Data Integration Building Block (section 4.2.1)

There exist several Software Components, both commercial and open source products that cover nearly the entire functionality of the Data Integration Building Block. Currently, GeoServer is one of the open source products that have been selected for the implementation of the Data Integration Building Block.

"*GeoServer is an open source software server written in Java that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. GeoServer is the reference implementation of the Open Geospatial Consortium (OGC) Web Feature Service (WFS) and Web Coverage Service (WCS) standards, as well as a high performance certified compliant Web Map Service (WMS). GeoServer forms a core component of the Geospatial Web*" (http://geoserver.org/display/GEOS/Welcome)

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/GeoServer

5.1.2.4. PyWPS

Related Building Block: Simulation Model Integration (section 4.2.4)

The Software Component technically appropriate to realise the functionalities of the Simulation Model integration Building Block is PyWPS (Python Web Processing Service). PyWPS is an implementation of the Web processing Service (WPS, 2007) standard from Open Geospatial Consortium. Within CRISMA, the PyWPS is (can be) used to make legacy models and other software available within the CRISMA Framework. In order to do so, one has to extend the basic PyWPS classes and introduce invocations of the own software.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/PyWPS

5.1.3. User interaction software components

Figure 76 shows the User Interaction Software Components that have been identified and described for the Implementation Architecture of the CRISMA Framework.

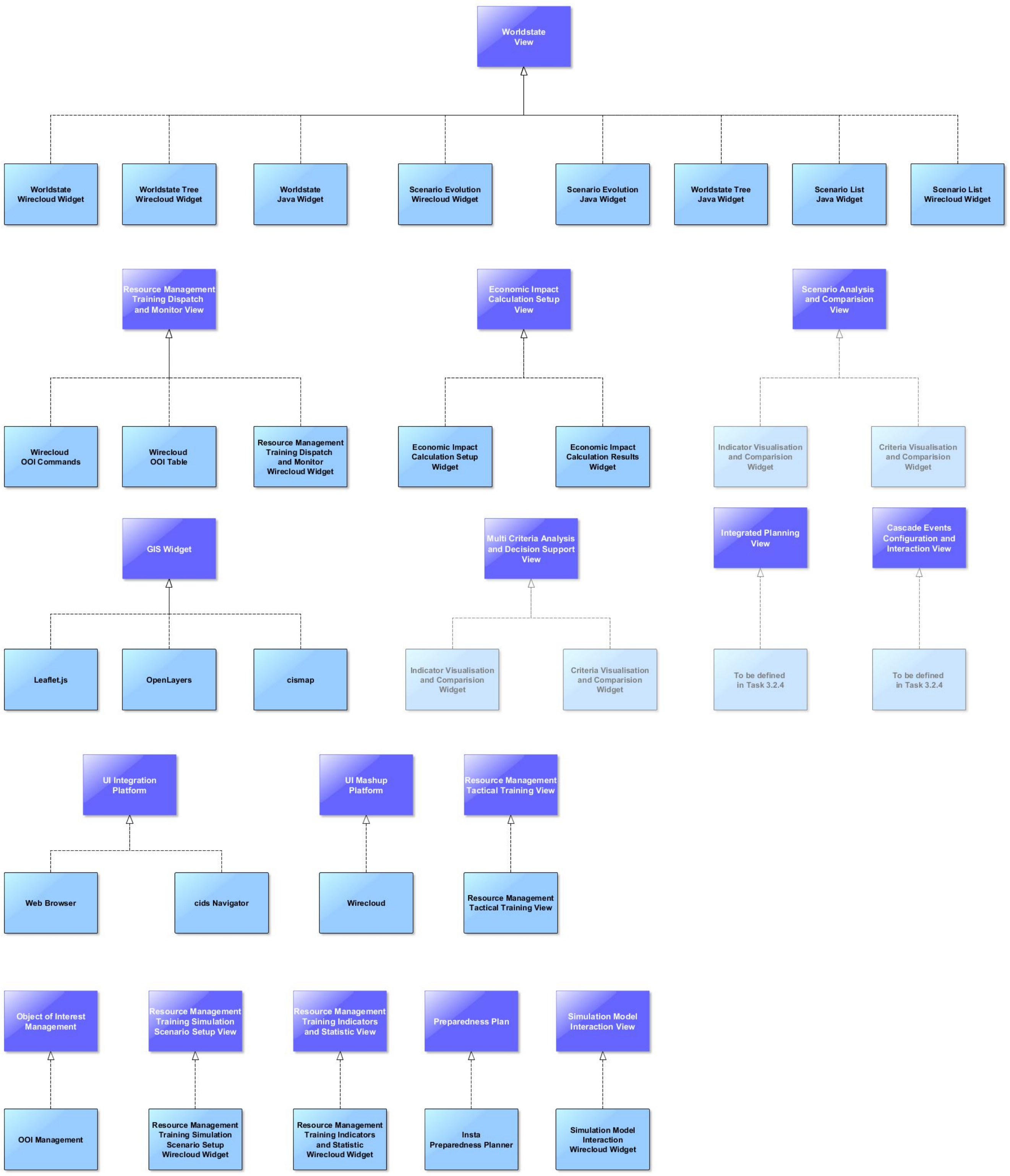

Worldstate View
Worldstate Wirecloud Widget
Worldstate Tree Wirecloud Widget
Worldstate Java Widget
Scenario Evolution Wirecloud Widget
Scenario Evolution Java Widget
Worldstate Tree Java Widget
Scenario List Java Widget
Scenario List Wirecloud Widget
Resource Management Training Dispatch and Monitor View
Economic Impact Calculation Setup View
Scenario Analysis and Comparision View
Wirecloud OOI Commands
Wirecloud OOI Table
Resource Management Training Dispatch and Monitor Wirecloud Widget
Economic Impact Calculation Setup Widget
Economic Impact Calculation Results Widget
Indicator Visualisation and Comparision Widget
Criteria Visualisation and Comparision Widget
GIS Widget
Multi Criteria Analysis and Decision Support View
Integrated Planning View
Cascade Events Configuration and Interaction View
Leaflet.js
OpenLayers
cismap
Indicator Visualisation and Comparision Widget
Criteria Visualisation and Comparision Widget
To be defined in Task 3.2.4
To be defined in Task 3.2.4
UI Integration Platform
UI Mashup Platform
Resource Management Tactical Training View
Web Browser
cids Navigator
Wirecloud
Resource Management Tactical Training View
Object of Interest Management
Resource Management Training Simulation Scenario Setup View
Resource Management Training Indicators and Statistic View
Preparedness Plan
Simulation Model Interaction View
OOI Management
Resource Management Training Simulation Scenario Setup Wirecloud Widget
Resource Management Training Indicators and Statistic Wirecloud Widget
Insta Preparedness Planner
Simulation Model Interaction Wirecloud Widget

Figure 76: User Interaction Software Components and related Building Blocks.

5.1.3.1. Economic impact calculation results widget

Related Building Block: Economic Impact Calculation View (section 4.3.1)

The Economic Impact Calculation Results Widget is a new software component developed from scratch and based on a Microsoft Excel prototype developed by VTT.

The development in HTML and JavaScript follows the technological requirements for software developments of new User Interaction Building Blocks (Mashable Composite UI Module).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Economic-Impact-Calculation-Results-Widget

5.1.3.2. Economic impact calculation setup widget

Related Building Block: Economic Impact Calculation View (section 4.3.1)

The Economic Impact Calculation Setup Widget is a new software component developed from scratch in V2 of WP34 and based on a Microsoft Excel prototype developed by VTT.

The development in HTML and JavaScript follows the technological requirements for software developments of new User Interaction Building Blocks (Composite UI Module).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Economic-Impact-Calculation-Results-Widget

5.1.3.3. Cismap

Related Building Block: GIS Widget (section 4.3.3)

There exist several Software Components, both commercial and open source products that cover large parts of the functionality of the GIS Widget Block. Currently, cismap is one of the open source products that have been selected for the implementation of the GIS Widget Building Block.

Cismap is a java-based Software Component for the GIS Widget and is implemented by CRISMA partner CIS on basis of the open source software cismap. Cismap is licensed under the GNU Lesser General Public License (LGPL, 2007) and can be used free of charge during and beyond the project lifetime.

Cismap is a map application specialised to access OGC compliant services like WFS (WFS, 2010) and WMS (WMS, 2006) for information from internal and external geospatial data sets. Thereby, it supports also powerful visualisation and editing functionalities for OGC compliant geospatial information (e.g. GML).

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/cismap

5.1.3.4. Leaflet.js

Related Building Block: GIS Widget (section 4.3.3)

There exist several Software Components, both commercial and open source products that cover large parts of the functionality of the GIS Widget Block. Currently, Leaflet.js is one of the open source products that have been selected for the implementation of the GIS Widget Building Block.

"*Leaflet is a modern open-source JavaScript library for mobile-friendly interactive maps. It is developed by Vladimir Agafonkin with a team of dedicated contributors. Weighing just* *about 31 KB of JS, it has all the features most developers ever need for online maps*." (http://leafletjs.com)

Leaflet implements basic functionalities requested by the GIS Widget Building Block. Manipulation of GIS data is currently not the scope of this lightweight mapping component. However, as the first phase of WP34 does not require more than the most basic controls Leaflet suits the needs of the first phase and is ready for integration out of the box.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Leaflet.js

5.1.3.5. OpenLayers

Related Building Block: GIS Widget (section 4.3.3)

There exist several Software Components, both commercial and open source products that cover large parts of the functionality of the GIS Widget Block. Currently, OpenLayers is one of the open source products that have been selected for the implementation of the GIS Widget Building Block.

OpenLayers is a JavaScript-based map application specialised to access OGC compliant services like WFS and WMS for information from internal and external geospatial data sets. Thereby, it provides powerful visualisation and editing functionalities for OGC compliant geospatial information (e.g. GML). OpenLayers is a powerful mapping component implementation that has grown over the years and thus provides a rather good variety of supported standards and tools.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/OpenLayers

5.1.3.6. Object of interest management user interface component

Related Building Block: Object of Interest Management (section 4.3.4)

The OOI Management UI Component is implemented by CRISMA partner NICE, it is a new software component that is developed from scratch. The OOI Management UI Component is licensed under the GNU Lesser General Public License (LGPL) and can be used free of charge during and beyond the project lifetime.

It is developed as standalone Web Application that can be hosted in a standard Webserver or as Wirecloud widgets (Mashable Composite UI Module). It is implemented using popular web technologies like Angular.js MVVM Platform, Twitter Bootstrap and OpenLayers.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/OOI-Management-UI-Component

5.1.3.7. Preparedness plan

Related Building Block: Preparedness Plan (section 4.3.7)

The Preparedness Plan Building Block is implemented by CRISMA partner INSTA on basis of Insta Preparedness Planner Software. The Insta Preparedness Planner is a commercial software and can be used free of charge during the project lifetime.

Insta Preparedness Planner is a decision support mechanism, based on existing situation assessment functionality of the Insta Response product. Insta Response is a networked, integrated solution, which provides situation picture, resource management, integrated communications, computer-aided situation assessment and dispatching and field support for emergency response centres and other authorities, in the command centres, on the field and for administrative users as well.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Insta-Preparedness-Planner

5.1.3.8. Resource management tactical training

Related Building Block: Resource Management Tactical Training (section 4.3.8)

The Resource Management Tactical Training Building Block is implemented by CRISMA partner NICE on basis of the NICE Situator Training System. The NICE Situator is licensed under a proprietary license by CRISMA partner NICE and can be used free of charge during the project lifetime.

Designed for the unique needs of Tactical Training, NICE Security Solutions for Training and Simulation combine a powerful set of tools and technologies that merge all cameras, sensors, communication systems, data sources and operating procedures into a single unified platform to secure the entire transportation infrastructure. In the process, public safety and security agencies not only gain valuable insight into everyday operational issues, they can substantially optimize them for measurable and valuable improvements and cost savings.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Situator-Training-System

5.1.3.9. Resource management training dispatch and monitor wirecloud view

Related Building Block: Resource Management Training Dispatch and Monitor View (section 4.3.9)

The Resource Management Training Dispatch and Monitor View Building Block is implemented by CRISMA partner AIT as Mashup Application composed of different Mashable Composite UI Modules (Wirecloud Widgets). It is licensed under an Open Source licensing scheme and can be used free of charge during and beyond the project lifetime.

This Software Component uses the JavaScript library OpenLayers, which is a freely available library displaying map data (from sources such as Google, Bing, WMS, OpenStreetMap, or any other, depending on the configuration) available under the FreeBSD license.

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/components/wirecloud-ooi-gis-map
https://crisma-cat.ait.ac.at/components/wirecloud-worldstate-picker

https://crisma-cat.ait.ac.at/components/wirecloud-ooi-table
https://crisma-cat.ait.ac.at/components/wirecloud-simulation-picker
https://crisma-cat.ait.ac.at/components/wirecloud-ooi-commands

5.1.3.10. Resource management training indicators and statistics wirecloud view

Related Building Block: Resource Management Training Indicators and Statistics View (section 4.3.10)

The Resource Management Training Indicators and Statistics View is implemented by CRISMA partner AIT as Mashup Application composed of different Mashable Composite UI Modules (Wirecloud Widgets). It is licensed under an Open Source licensing scheme and can be used free of charge during and beyond the project lifetime.

The Software Component is mainly based on jqPlot (http://www.jqplot.com/), which is a JavaScript and jQuery-based library available under MIT License (MIT, 1998) and GNU General Public License (GPL, 2007). The dual-licensing allows for more flexibility concerning how and where this library can be used. This charting library makes use of HTML5's canvas features and thus works in any HTML5 compatible environment where JavaScript is enabled (e.g. WebKit-based browsers) and should also extend to contemporary mobile devices.

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/components/wirecloud-charts
https://crisma-cat.ait.ac.at/components/wirecloud-worldstate-picker
https://crisma-cat.ait.ac.at/components/wirecloud-simulation-picker

5.1.3.11. Resource management training simulation scenario setup wirecloud view

Related Building Block: Resource Management Training Simulation Scenario Setup View (section 4.3.11)

The Resource Management Training Simulation Scenario Setup View Building Block is implemented by CRISMA partner AIT as Mashup Application composed of different Mashable Composite UI Modules (Wirecloud Widgets). It is licensed under an Open Source licensing scheme and can be used free of charge during and beyond the project lifetime.

This Building Block uses the JavaScript library OpenLayers, which is a freely available library displaying map data (from sources such as Google, Bing, WMS, OpenStreetMap, or any other, depending on the configuration) available under the FreeBSD license (BSD, 1998).

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/components/wirecloud-ooi-table
https://crisma-cat.ait.ac.at/components/wirecloud-simulation-picker

5.1.3.12. Simulation model interaction widget

Related Building Block: Simulation Model Interaction View (section 4.3.13)

The Simulation Model Interaction Widget Building Block is a Composite UI Module that lets end users interact with the various Simulation Models exposed by a Simulation Model

Integration Building Block. It is responsible for collecting Simulation Control Parameters (SCP), launching the respective simulation model, showing status information and retrieving simulation model.

Since the current model integration approach described in the CRISMA Framework Architecture is based on OGC Web Processing Service, the Simulation Model Interaction Widget is essentially a CRISMA-aware WPS client.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Simulation%20Model%20Interaction%20Widget

5.1.3.13. Wirecloud

Related Building Block: UI Mashup Platform (section 4.3.14)

The Software Component selected to realise the UI Mashup Platform is the Wirecloud application mashup platform. Wirecloud is developed by CoNWeT Lab (University of Madrid) licensed under GNU Affero General Public License (AGPL, 2007). It can therefore be used free of charge during and beyond the project lifetime. Widgets running on Wirecloud are not affected by the licensing terms by a special clause.

Wirecloud is a web-based mashup platform that allows combining individual, usually general purpose components (“widgets”) into a web application. The software features a catalogue of widgets that can be published for other users of the system. Additionally, this catalogue allows also the publication of entire mashup applications that can be re-used by others.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/Wirecloud-Application-Mashup-Platform

5.1.3.14. Cids navigator

Related Building Block: UI Integration Platform (section 4.3.15)

There exist several pieces of software that are able to provide the key functionality of the UI Integration Platform such as any modern Browser (Firefox, Opera, etc.) or platforms like JavaFX.

Hence there is virtually no development effort needed when a CRISMA Application is also at the same time a Mashup Application. However the situation is different when the user interfaces of a CRISMA Application are only partially realized as Mashup Application. In this case, HTML/JavaScript widgets may have to interact with other types of existing legacy GUI code.

A Software Component for the realization of an UI Integration Platform that is able to integrate other Composite UI Modules with existing Java GUIs is the cids navigator. The cids navigator was used as one CRISMA Reference Application to demonstrate the capabilities of the CRISMA Framework V1.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/cids-Navigator

5.1.3.15. Scenario evolution widget (java and JavaScript)

Related Building Block: World State View (section 4.3.16)

A java-based prototype of the Scenario Evolution Widget was implemented by CRISMA partner CIS. The development of the Scenario Evolution Java Widget has been discontinued in favour of the Scenario Evolution Wirecloud Widget which satisfies all technological requirements of Mashable Composite UI Modules. The Scenario Evolution Widgets are licensed under the GNU Lesser General Public License (LGPL, 2007) and can be used free charge during and beyond the project lifetime

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/component/Scenario-Evolution-Widget-%28Java%29
https://crisma-cat.ait.ac.at/component/Scenario-Evolution-Widget-%28JavaScript%29

5.1.3.16. Scenario list widget (java and JavaScript)

Related Building Block: World State View (section 4.3.16)

A java-based prototype of the Scenario List Widget was implemented by CRISMA partner CIS. The development of the Scenario List Java Widget has been discontinued in favour of the Scenario List Wirecloud Widget which satisfies all technological requirements of Mashable Composite UI Modules. The Scenario List Widgets are licensed under the GNU Lesser General Public License (LGPL, 2007) and can be used free charge during and beyond the project lifetime

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/component/Scenario-List-Widget-%28Java%29
https://crisma-cat.ait.ac.at/component/Scenario-List-Widget-%28JavaScript%29

5.1.3.17. World state tree widget (java and JavaScript)

Related Building Block: World State View (section 4.3.16)

A java-based prototype of the World State Tree Widget was implemented by CRISMA partner CIS.

The development of the World State Tree Java Widget has been discontinued in favour of the World State Tree Wirecloud Widget which satisfies all technological requirements of Mashable Composite UI Modules. The World State Tree Widgets are licensed under the GNU Lesser General Public License (LGPL, 2007) and can be used free charge during and beyond the project lifetime

CRISMA Catalogue URLs:
https://crisma-cat.ait.ac.at/component/World State-Tree-Widget-%28Java%29
https://crisma-cat.ait.ac.at/component/World State-Tree-Widget-%28JavaScript%29

5.1.3.18. World state widget (java and JavaScript)

Related Building Block: World State View (section 4.3.16)

A java-based prototype of the World State Widget was implemented by CRISMA partner CIS. The development of the World State Java Widget has been discontinued in favour of the World State Wirecloud Widget which satisfies all technological requirements of

Mashable Composite UI Modules. The World State Widgets are licensed under the GNU Lesser General Public License (LGPL, 2007) and can be used free charge during and beyond the project lifetime.

CRISMA Catalogue URL:
https://crisma-cat.ait.ac.at/component/World State-Widget-%28Java%29

5.2. Core control and communication information model

This part of the Implementation Architecture provides a specification of the Core Control and Communication Information Model (Core CCIM) of the CRISMA Framework. This information model is the basis for all application specific information models (Application CCIMs). The Core CCIM defines the information classes of the Conceptual Business Logic of CRISMA on an abstract and generic level.

The main purpose of the Core CCIM is to allow the management of World States and their Transitions as defined by the Conceptual Business Logic in a uniform manner across all CRISMA Federations. Thus, it leverages the development of generic Building Blocks that can be used in any CRISMA Federation with minimal or no configuration and development effort.

5.2.1. Classes of the core control and communication information model

The Core CCIM defines a minimal set of classes and relationships needed to realise the Conceptual Business Logic of the CRISMA Framework. The Core CCIM can be seen as the least common denominator on which all CRISMA Federations (Applications) have to agree.

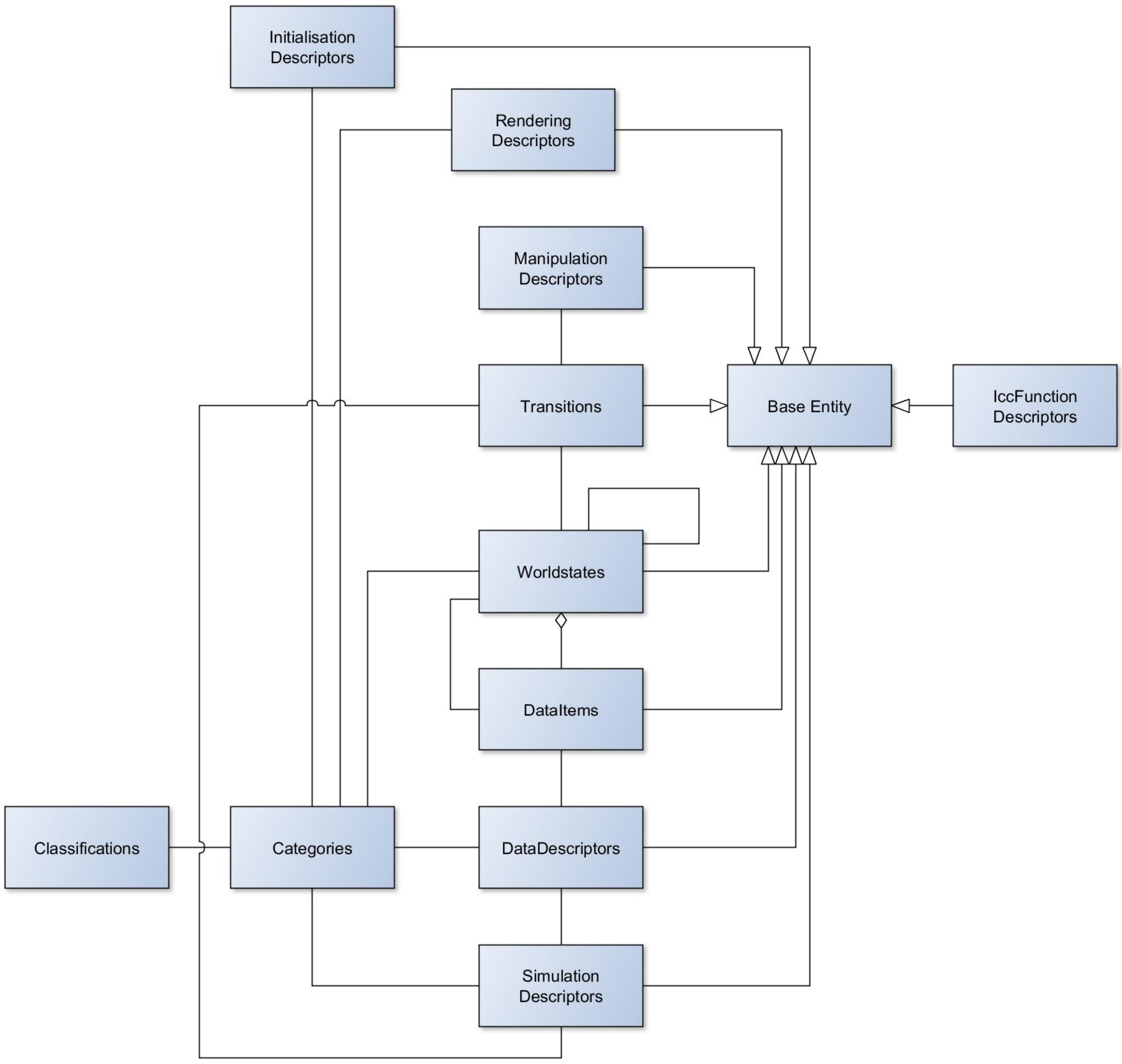


Figure 77: Core CCIM Classes and Relationships.

The Core CCIM can be subdivided into two parts: A static part and a dynamic part. The static part has to be defined during the design phase of a CRISMA Application and represents invariant domain information of this particular CRISMA Application. Once defined, it is not supposed to change.

This part of the CCIM is also called the domain definition of the CRISMA Application. The CCIM Classes of the domain definition are:

- **Categories**
  A class representing an arbitrary categories that can be used to categorise scenarios, simulations, etc.
- **Classifications**
  A class that represents a classification of categories, e.g. scenario types, crisis management phases, etc.

- **DataDescriptors**
  A DataDescriptor provides invariant meta-information about an arbitrary dataset (World State data, simulation control parameters, ...).
- **IccFunctionDescriptors**
  IccFunctionDescriptors provide Information about the ICC Functions that are executed after each World State Transition. ICC Functions Descriptors are optional since most functions will be realised by the application logic of a specific Building Block Implementation. Indicator Functions for example are implemented by the Indicator Building Block Software Component.
- **InitialisationDescriptors**
  Initialisation descriptors describe user interfaces and business logic (e.g. Composite UI Modules, Setup and Configuration Building Blocks) that are needed to initialize the set of Simulation Models, the initial World State, descriptors, etc.
- **ManipulationDescriptors**
  ManipulationDescriptors refer to World State Transitions and describe the user interfaces and business logic (e.g. Composite UI Modules, World States Widget Building Block and associated World State Views) to manipulate the changeable parameters of a World State.
- **RenderingDescriptors**
  A Rendering Descriptor describes a User interfaces to visualize the World State (World States Widget Building Block and related Viewer Widgets).
- **SimulationDescriptors**
  A SimulationDescriptor provides information about a Federated Simulation (a simulation consisting of one or more Simulation Models wrapped as one WPS service). It holds references (Generic Reference) to concrete descriptions on how to execute the simulation (Simulation Model Integration Building Block) and by which user interface (Simulation Model Interaction Building Block) to control (e.g. parameterise, monitor) the simulation.

The dynamic part of the core CCIM represents the changeable parameters of the world modelled as World States and Transitions. It is represented by the following classes:

- **DataItems**
  This class represent an actual data set (instance), e.g. World State data, ICC data, simulation control parameters, etc.
- **Transitions**
  This class represents a concrete World State Transition. It contains information on the performed manipulation or simulation, the actual Simulation Control Parameters and further meta-information (e.g. which parts of the World State were changed).
- **World States**
  This class represents a concrete World State. A snapshot of the World or World State consists of all data related to a specific crisis simulation experiment.

Please note that **TransitionPoints** are not explicitly modelled in the Core CCIM since they are implicitly defined of by the Simulation Cases which are part of the application logic of the respective CRISMA Application. Thus, the application logic is responsible to create appropriate Transitions that are valid in the scope of a specific Simulation Case Graph (see section 3.1.4). A validation of scenarios (sequences of Transitions) as instances of a Simulation Case Graph on the information model level is therefore not supported.

In summary, the Core CCIM leverages interoperability between Federates and Federations within the whole CRISMA System (the sum of all CRISMA Federations) while extensions leverage interoperability between Federates within one CRISMA Federation. Further details and examples on the intended usage of the Core CCIM are given in the Application Architecture Template in APPENDIX (B). A technical specification of the Core CCIM in UML can be found in APPENDIX (A).

5.2.2. Relation to conceptual business logic

As explained in the previous section, the Core CCIM maps the Conceptual Business Logic (see section 3.1) to classes of a modular information model.

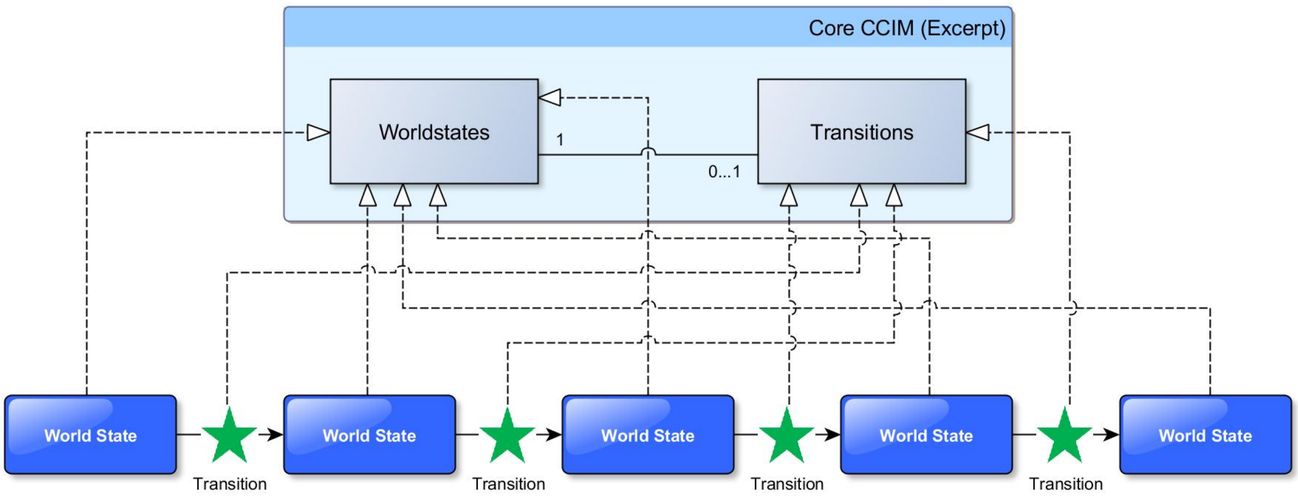


Figure 78: Scenario expressed with help of the Core CCIM.

Figure 78 shows how a Simulated Crisis Management Scenario (see section 3.1) can be expressed by instances of Core CCIM Classes and Figure 79 provides a more detailed view on the relationships of the Core CCIM to the Conceptual Business Logic.

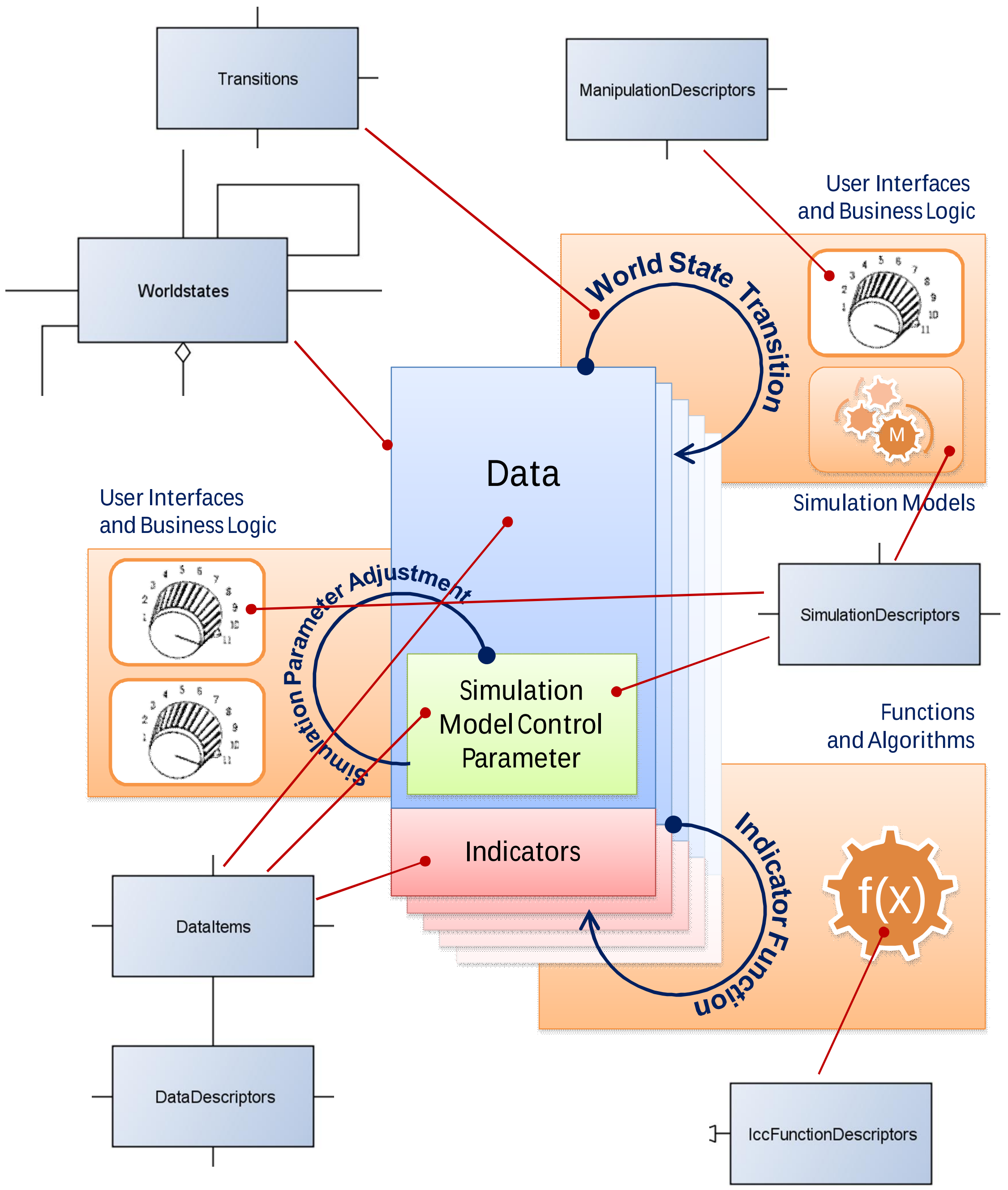


Figure 79: Mapping the Core CCIM to the Conceptual Business Logic (detailed).

5.2.3. Technical realisation

Technically, the Core CCIM is implemented as part of a generic and object-oriented relational database model of the Software Component that implements the ICMM Building Block (section 5.1.1.1). Thus, implementing a CCIM is a configuration task and does not require any changes to the ICMM itself. Figure 80 shows an excerpt of the Core CCIM implemented as database model of the ICMM.

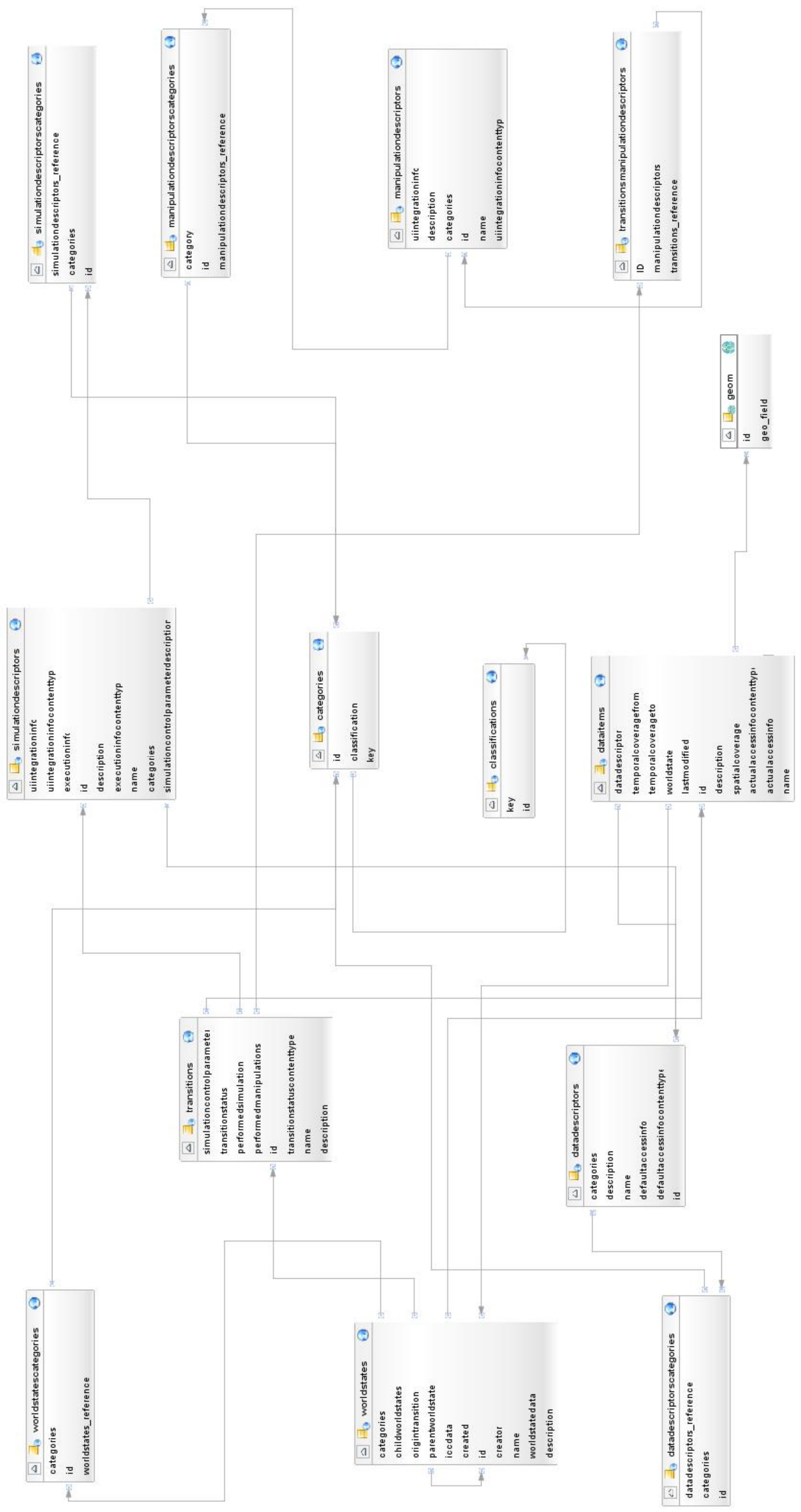


Figure 80: Core CCIM Database Model (excerpt).

According to the rules for the specification of information models (section 5.4.4) a CCIM modelled in UML has to be encoded in Java Script Object Notation Format (JSON).

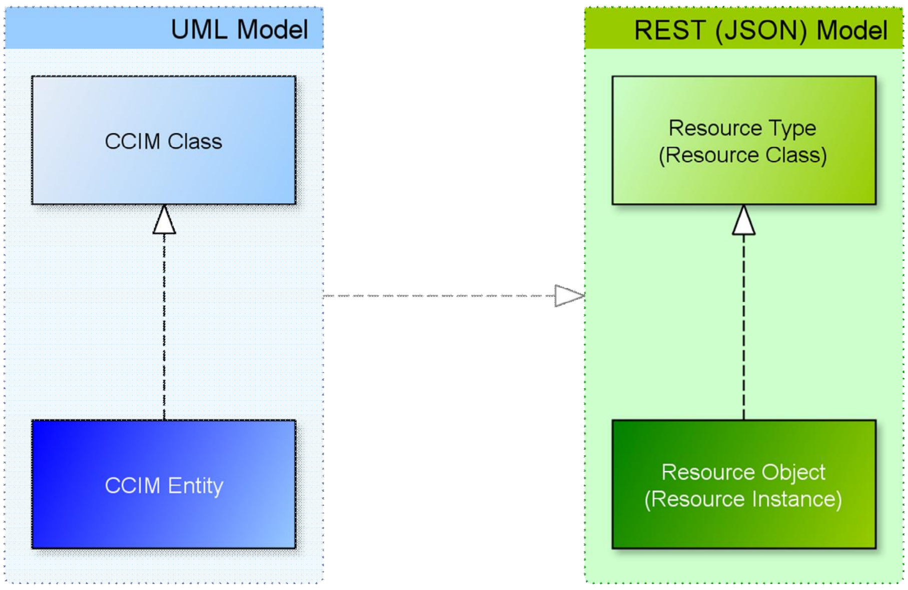
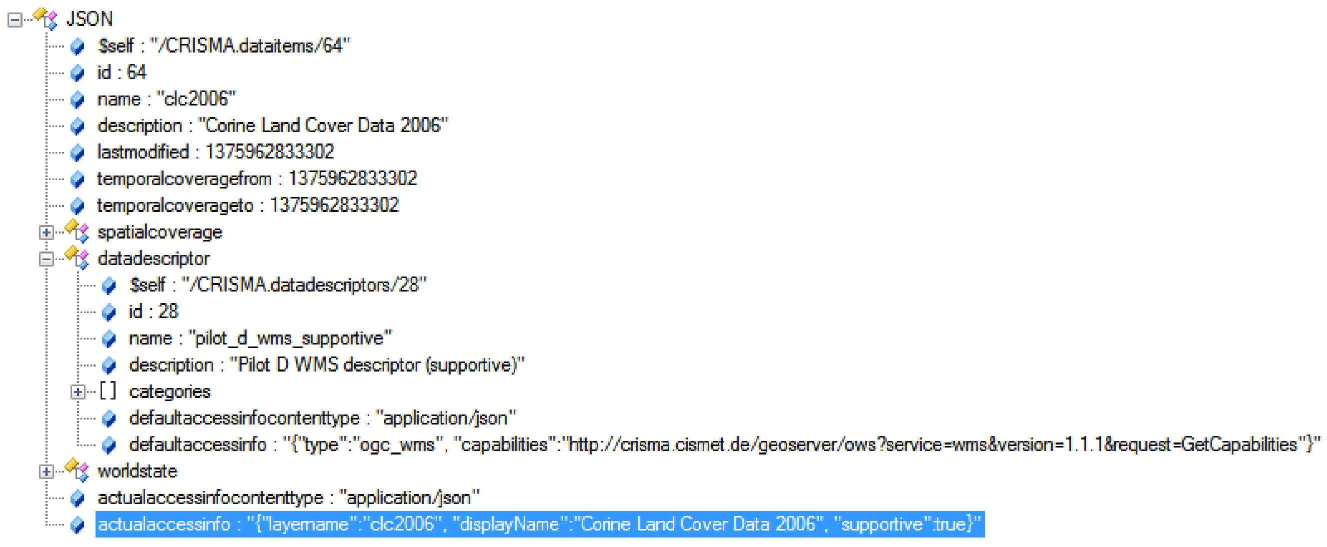


Figure 81: Mapping UML Models to REST Models.

The relation between the UML and JSON models is shown in Figure 81. On the level of the UML Model, an instance of a CCIM Class is called CCIM Entity. When the UML Model is encoded in JSON, the CCIM Class becomes a Resource Type or Resource Class and the CCIM Entity a Resource Object or Resource Instance.

The actual CCIM Resource Instances can be accessed via the REST APIs of the ICMM. Figure 82 shows an example of such a Resource Instance.



```
JSON
    $self : "/CRISMA.dataitems/64"
    id : 64
    name : "clc2006"
    description : "Corine Land Cover Data 2006"
    lastmodified : 1375962833302
    temporalcoveragefrom : 1375962833302
    temporalcoverageto : 1375962833302
    spatialcoverage
    datadescriptor
        $self : "/CRISMA.datadescriptors/28"
        id : 28
        name : "pilot_d_wms_supportive"
        description : "Pilot D WMS descriptor (supportive)"
        [ ] categories
        defaultaccessinfocontenttype : "application/json"
        defaultaccessinfo : "{"type":"ogc_wms", "capabilities":"http://crisma.cismet.de/geoserver/ows?service=wms&version=1.1.1&request=GetCapabilities"}"
    worldstate
    actualaccessinfocontenttype : "application/json"
    actualaccessinfo : "{"layername":"clc2006", "displayName":"Corine Land Cover Data 2006", "supportive":true}"
```

Figure 82: DataItems Instance JSON.

5.2.4. Generic extension mechanisms

Since CRISMA Applications (Federations) are very heterogeneous in nature, the Core CCIM is optimised for genericity and reusability and provides only rudimentary meta-information about World States, Transitions, etc.

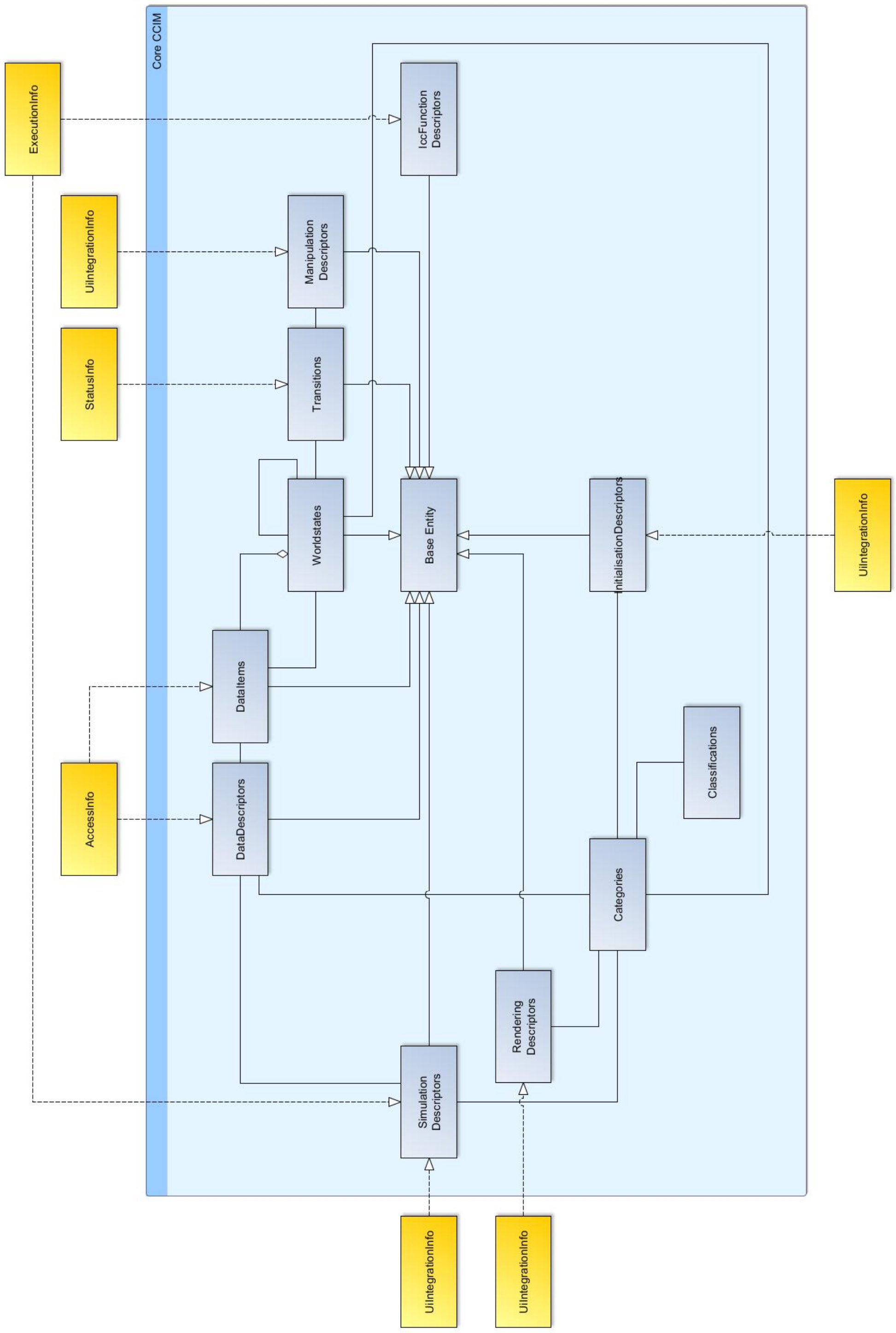


Figure 83: Core CCIM with extension points for Application CCIMs.

According to this, it does not define a concrete meta-information schema for data access, the description of manipulation and simulations, etc. Therefore, each CRISMA Federation has to define some federation-specific extensions of the Core CCIM.

For this purpose, the Core CCIM provides two extension mechanisms which are described in the subsequent section: Extension by inheritance and predefined Extension Points. Figure 83 below gives an overview on the generic extension points for Application CCIMs.

5.2.4.1. *Extension by inheritance*

The first possibility for extending the Core CCIM is to directly inherit from Core CCIM Classes (Figure 84). In this way the extended CCIM Classes will become directly part of the ICMM data model (see section 5.2.3). The main advantage of this approach is that entities based on such classes can benefit from the built-in search, access control and validation functionalities of the ICMM.

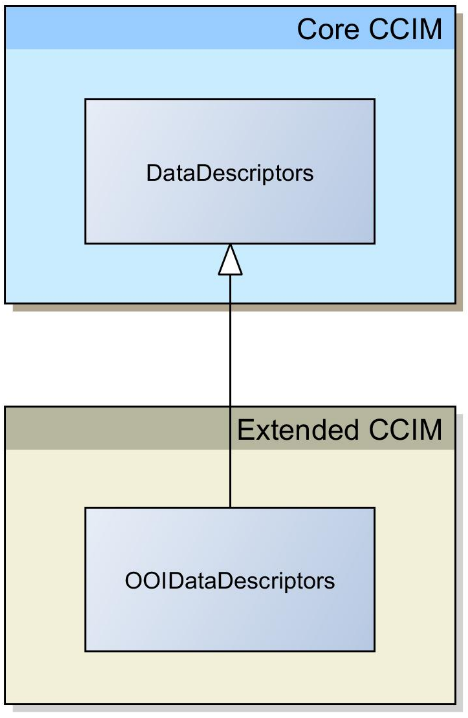


Figure 84: Extension by inheritance.

One disadvantage of this approach is that a data model has to be designed and installed into an ICMM instance. Although this is an administrative task which is supported by graphical tools, it requires a certain amount of expert knowledge about the ICMM internals.

5.2.4.2. *Extension Points*

A second, more convenient and easy to use mechanism for the extension of the Core CCIM and CCIMs in general are predefined Extension Points. Extension Points are realised in the Core CCIM by so called Generic References.

A Generic References in a CCIM is a reference to an entity which is not modelled as class of the enclosing CCIM. Such a Generic Reference contains in general domain specific information that may be used by (application specific) components. The actual content type of the Generic Reference is determined by a dedicated content type attribute. Thus, a Generic Reference may point e.g. to a JSON or XML document or even a link to a JSON or XML document.

Interestingly, Generic References may point to arbitrary JSON objects including JSON objects that comply with a CCIM Class. With help of this mechanism it is possible to establish relations to CCIM Classes that are neither part of the Core CCIM nor known during the design time of the Core CCIM. Therefore, although not required, it is nevertheless good practice to model such entities that should be represented by Generic References also compliant to CCIM Classes.

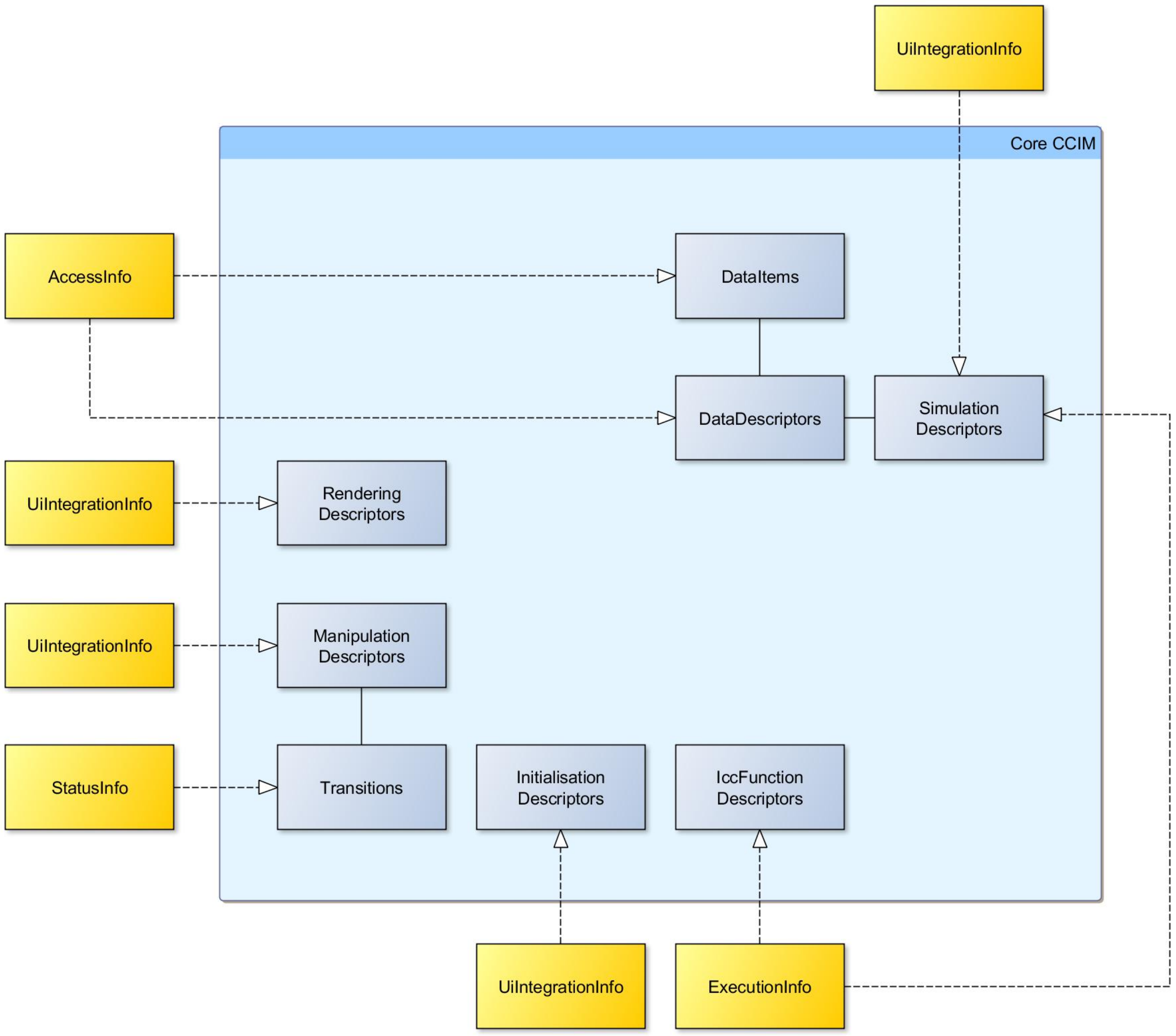


Figure 85: Extension Points of the Core CCIM.

In any case, it should be clear that it is always up to the designer of a concrete CRISMA Application to define the actual structure and content of Generic References. Therefore, a Generic Reference which may be suitable for a particular CRISMA Application may not be usable in another Application.

Figure 85 shows the Extension Points of the Core CCIM, shown as yellow boxes. Each of those Generic References is modelled in the Core CCIM as two String attributes: The actual content of the reference and the content type of the reference.

The following types of Generic References are defined in the Core CCIM:

- **AccessInfo** is the extension Point of the DataDescriptors and DataItems Classes. It can be divided into defaultAccessInfo (relevant for DataDescriptors) and actualAccessInfo (relevant for DataItems).

  - **defaultAccessInfo** describes the actual data source, e.g. a link to an external meta data document describing the data set or other types of access information and credentials (e.g. WFS URI). It represents invariant meta-information that stays the same for all DataItems that are described by a DataDescriptor.
  - **actualAccessInfo** is the actual information on how to access the data. In rare cases or for very small and simple datasets (e.g. ICC Data) it may contain also the data itself. In general, it provides information to indentify an individual data item or a specific version of a data item (e.g. a specific request parameter, a filename, etc.).
- **UIIntegrationInfo** is the Extension Point of the Rendering-, Manipulation-, Simulation- and InitialisationDescriptors. It provides the actual information on how to instantiate and integrate the respective user interface (Composite UI Module). Depending on the CRISMA Application it may contain an URI that points to a Composite UI Module instance or more detailed information.
- **ExecutionInfo** is the Extension Point of Simulation- and IccFunctionDescriptors. It provides information on how to execute a federated simulation or an ICC Function. In general, it contains information on the WPS that provides the simulation (Simulation Model Integration Building Block).The concrete details depend on the types of simulations available in a particular CRISMA Application.
- **StatusInfo** is the Extension Point of the Transition Class. It provides detailed Transition status information. It may be represented by an arbitrary CRISMA JSON object that is defined for a specific Application CCIM and may be injected dynamically, e.g. by the Simulation Model Integration or Interaction Building Blocks (sections 4.2.4and 4.3.12). It may also point to the raw Transition status, e.g. a log file, a link to a website with status messages, a WPS URL, etc.

Several domain-independent Generic References modelled as JSON entities are provided in the Application Architecture Template in APPENDIX (B).

5.3. *Application architecture template*

The Application Architecture Template is a template for the technical specification of an individual Application Architecture and thus acts as blueprint for the actual implementation of a CRISMA Application.

While this section of the Implementation Architecture provides only a general description of the Application Architecture Template and its related concepts, the actual template can be found in APPENDIX (B).

An Application Architecture provides a specification of application-specific Simulation Cases in accordance to the Integrated System Viewpoint of the Conceptual Business Logic of the CRISMA Framework Architecture (section 3.1). Thereby, it describes the Simulation Cases which are based on Pilot Use Cases (D23.1, 2013) and that shall be supported by the CRISMA Application in terms of individual World States and Transitions. Thus, it allows the development of a Crisis Management Simulation System as CRISMA Application with help of the concepts and Building Blocks of the CRISMA Framework.

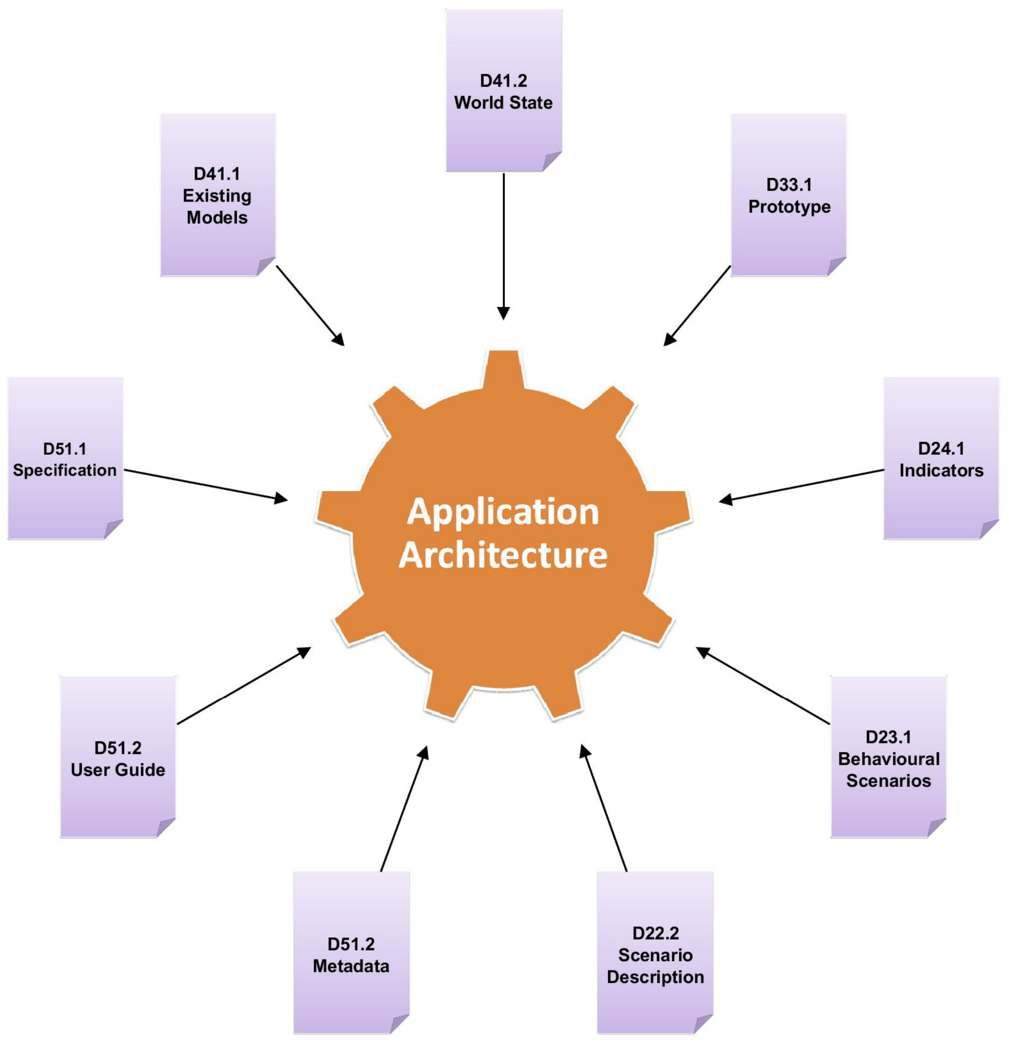


Figure 86: Input to Application Architecture.

As shown in Figure 86, specifying an Application Architecture according to the template requires input from many different sources. Although most information for the development of a CRISMA Application is currently available in the CRISMA project, this information is scattered across many different documents and described in such a way that it is not directly usable for the development.

Thus, the main intention of the Application Architecture is to leverage the development of the CRISMA Framework and CRISMA Application in SP3 and SP5 by providing a homogenous and consolidated view on the relevant material produced so far in the CRISMA Project.

The Application Architecture Template provided in APPENDIX (B) of this document consists of the following two main parts:

- **Initialisation and configuration information**
  Initialisation and configuration information has to be collected once for a particular or CRISMA Application. It consists of a specification of all Data Slots, Simulation Models and ICC Elements that are available for the Simulation Cases of the Pilot Use Case as well as all possible Transition Points themselves. Those elements are described in detail in the following:

  - **World State Data Slot Descriptions**
    All World State Data Slots of the whole Crisis Management Use Case (i.e. Pilot use Case) have to be described in accordance to the Conceptual Business Logic and the Core CCIM. The specification of World State Data Slot is separated from the specification of Simulation Cases since a Pilot Use Case usually consists of different Simulation Cases and not all World State Data Slots may be relevant for each Simulation Case.
  - **ICC Descriptions**
    Like World State Data Slot Descriptions, ICC (Indicators, Criteria and Costs) Descriptions are also defined once for the complete Pilot Use Case. The specification of an Indicator Element is as bit different than the specification of Data Slot, since the ICC Element describes in general a simple numerical value rather than a complete data source.
  - **Simulation Model Descriptions**
    As third part of the initialisation and configuration information, a technical description of the Simulation Models has to be provided. While such a specification could be arbitrary complex, only the most import information like input and output Data Slots and simulation control parameters is collected by the template.
  - **Transition Point Descriptions**
    The last part of the initialisation and configuration information contains all possible Transition Points available for the Simulation Cases and how they affect the World State. It specifies furthermore a Transition Point Graph that defines (coarsely) in which order Transitions can be executed.

- **Simulation Cases**
  This part of the template describes one Simulation Case of the overall Pilot Use Case. It has to be repeated for each distinct Simulation Case. It consists of the following elements:
  - **General Information**
    This section provides some general information about the Simulation Case. This information may be used as descriptive meta-information in the CRISMA Application or the CRISMA Catalogue.
  - **Initial World State Definitions**
    This section provides initial (static) definitions of the Data Slots that play a role in the World State of the current Simulation Case.
  - **Simulation Case Specification**
    This section provides a description of the distinct steps of the Simulation Case (Decisions, Analyses) performed by a decision maker in the CRISMA Application and relates them to Transitions.

5.4. *Rules and guidelines*

The last part of the Implementation Architecture defines the guidelines, rules and recommendations that have to be followed or considered when developing, adapting or integrating Software Components for the CRISMA Framework and when developing CRISMA Applications, respectively. This includes e.g. how APIs are specified and implemented or which technologies are used for the development of Composite UI Modules, how Composite UI Modules communicate and interact, etc. Several general rules have already been formulated on a conceptual level in the Functional Architecture section (chapter 3). The intention of this section is to provide a formalisation of those rules under consideration of technological and implementation specific aspects.

5.4.1. Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (RFC2119, 1997).

The terms "JSON", "JSON text", "JSON value", "member", "element", "object", "array", "number", "string", "boolean", "true", "false", and "null" in this document are to be interpreted as defined in RFC 4627 (RFC4627, 2006).

5.4.2. General rules for federates

The following chapter provides a summary and a formalisation of rules for CRISMA Federates that were introduced by the concept chapter of the Functional Architecture (chapter 3).

5.4.2.1. Rules for federates

Those are the basic rules for CRISMA Federates that are needed to guarantee and support interaction, interoperability and integration within CRISMA Federations and CRISMA Applications.

1. A CRISMA Federate must be aware of the APIs of the CRISMA Middleware Infrastructure (ICMM) and the related Core Control and Communication Information Models (CCIM).
2. A CRISMA Federate may but does not have to expose a dedicated RESTful CRISMA API that is aligned to the APIs of the ICMM.
3. Control and Communication Information that changes the status of a World State Transition shall not be exchanged between Federates directly. It must be transferred solely using the APIs provided by the Middleware Infrastructure (ICMM) to ensure that the Middleware Infrastructure is always aware of the current status of a World State Transition.
4. The mandatory communication protocol used in communication with the ICMM is the hypertext transfer protocol (HTTP).
5. CRISMA Federates may exchange data with non-CRISMA components (Integrated Components).
6. The preferred protocol for the exchange of data is HTTP.
7. The preferred method for direct data transfer is based upon open standards and services of the Open Geospatial Consortium (OGC).
8. Integrated Components are not aware of the ICMM API but may take part in an interaction of CRISMA Federates.
9. A CRISMA Federation must be composed of CRISMA Federates only.
10. An existing software component that wants to join a CRISMA Federation and thus become a CRISMA Federate has to be aware of the ICMM API and thus

must be able to exchange Control and Communication Information with the ICMM.

11. A CRISMA Application may be composed of CRISMA Federates, Integrated Components, Integrated Simulation Models and Integrated Systems.

5.4.2.2. Rules for building blocks

In addition to the general rules for Federates, CRISMA Building Blocks have to adhere to the following rules:

1. CRISMA Building Blocks are Federates that are provided by the CRISMA Framework developed in SP3 of the CRISMA Project.
2. CRISMA Framework Building Blocks must be domain and location independent and thus be usable in any CRISMA Application regardless of the concrete Simulation Case or crisis management domain covered by the application.
3. CRISMA Framework Building Blocks must be composable and configurable, and thus adaptable to the needs and requirements of the CRISMA Application (Pilot Applications, respectively) to be built.

5.4.2.3. Rules for federated simulations and simulation models

The following rules apply to Federated Simulations and Federated or Integrated Simulation Models.

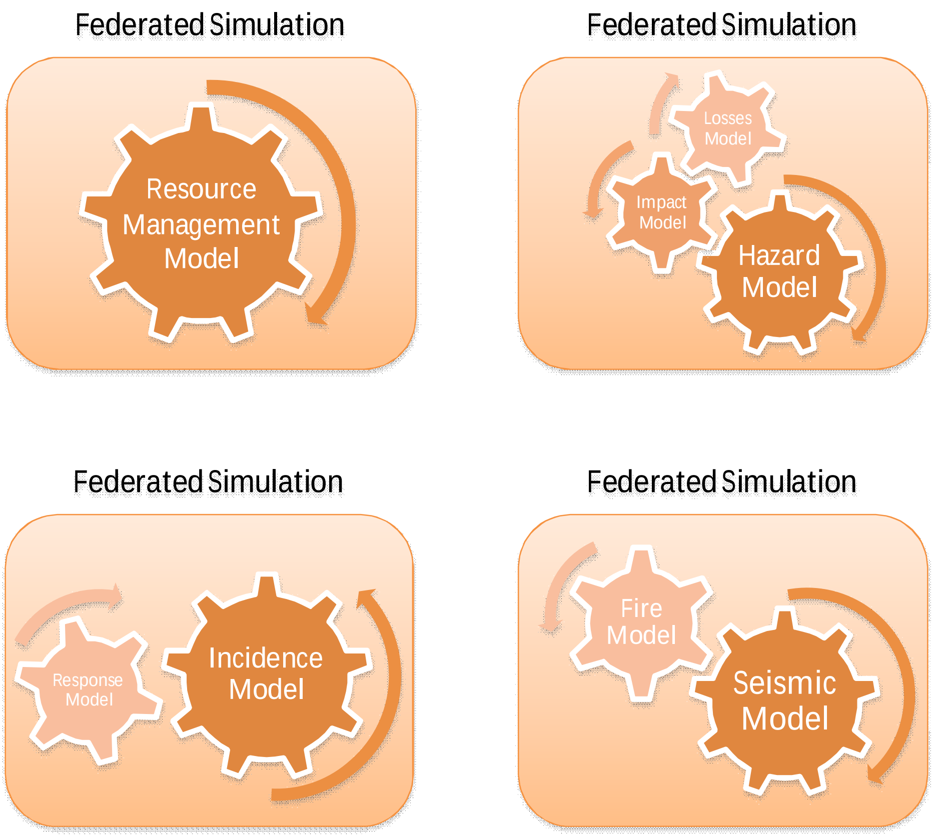


Figure 87: Examples for Federated Simulations.

1. According to the Simulation Model Integration Approach of the architecture (section 3.14), any **federated** Simulation Model must be provided as OGC Web Processing Service (WPS).
2. According to the Data Integration Approach of the architecture (section 3.13) input and output data of federated Simulation Models should be provided via OGC services such as SOS, WFS and WMS.
3. A Federated Simulation may be composed of one or more Federated or Integrated Simulations models but appears itself as Federated Simulation Model and thus is accessible via one OGC WPS interface.
4. Any Simulation Model shall be able to provide a human readable description of its purpose.
5. Any Simulation Model shall be able to provide information about the supported, required and optional parameters that have to be available in order to perform a new model run. This includes information on how this data can be provided, e.g. if it can be transmitted prior to the actual execution request and where it shall reside, if it shall be part of the actual execution request, if the actual execution request may contain links to the actual data. Additionally it must state what formats and encodings are supported for each parameter as well as being able to provide validity ranges if applicable.
6. Any Simulation Model shall be able to provide meta-information about its result data. This includes information on supported data formats and means for retrieving the data
7. Any Simulation Model shall be able to provide information about the status of a specific execution (simulation run). Requesters may be informed at any time whether an execution is scheduled, running, paused, failed, or finished.
8. Any Simulation Model shall be able to either indicate the expected execution time for a certain run and the remaining execution time or at least provide an indication of the actual status.
9. Any Simulation Model shall be able to provide detailed error messages e.g. in case of a failed execution.

5.4.3. Rules for building block implementations

The following rules cover the adaptation of existing software components and the implementation of new Building Blocks from scratch.

5.4.3.1. Rules for software candidates

Software Candidates are Software Components that are selected and adapted for the realisation of a certain CRISMA Framework Building Block.

1. A candidate Software Component for a Building Block Implementation must be made CRISMA-aware which means that it can exchanges events and Control and Communication Information with the CRISMA Middleware Infrastructure (ICMM).
2. The user interface(s) of a Software Component that is a candidate for Building Block implementation should be exposed as Composite UI Modules that can either be integrated into other GUIs (with help of the UI Integration Platform

Building Block) or put together with other Composite UI Modules to a rich internet application (with help of the UI Mashup Platform Building Block).

3. If for technical reasons the user interface(s) of the candidate Software Component cannot be provided as Composite UI Modules a mechanism must be provided that allows the integration of other Composite UI Modules into the native GUI of the Software Component (with help of the UI Integration Platform Building Block).
4. A candidate Software Component should offer its programmatically accessible functionality (functionality that can be accessed by other software components) through a well-defined interface that must be realised as HTTP RESTful API.

5.4.3.2. Rules for building block implementation

Those are general rules for the implementation of Building Blocks of the CRISMA Framework (see chapter 4). A concrete implementation of a Building Block is called Software Component (see section 5.1).

1. Software Components implementing a Building Block of the CRISMA Framework shall be generic but nevertheless adaptable to needs of a specific CRISMA Application. Thus, they should provide convenient configuration mechanisms and well-defined extension points.
2. Software Components implementing a Building Block of the CRISMA Framework must implement the functionalities that are marked as mandatory in the functional descriptions and specifications of the respective Functional Building Blocks.
3. Software Components implementing a Building Block of the CRISMA Framework should implement the functionalities that are marked as optional in the functional descriptions and specifications of the respective Functional Building Blocks.
4. Software Components implementing a Building Block of the CRISMA Framework that are implemented from scratch and thus not based on an existing Software Candidates must provide their user interface(s) as composite UI Modules.

5.4.4. Rules for information models

Those are rules for the specification of Control and Communication Information Models (CCIMs).

1. All CCIMs must be modelled and formally specified in Unified Modelling Language (UML). The UML specification of CCIMs serves as formal documentation for the Implementation Architecture as well as reference for API developers.
2. The actual CCIM data (CCIM Entities) that is exposed through a RESTful API has to be encoded in the JavaScript Object Notation (JSON) (RFC4627, 2006) data format.
3. The UML to JSON mapping is not supported by mapping tools but has to be performed by API developers during the specification and design of the RESTful API. Apart from a few rules, this mapping is subject to interpretation of the API developer.

4. The RESTful APIs as well as the REST (JSON) models that are based on the CCIM UML models must be formally specified using the Swagger API Documentation Toolkit as described in Rules and Best Practices for API Specifications (5.4.5).
5. Besides the mandatory JSON encoding other optional representations of CCIM Entities, e.g. in XML, may be supported by a specific implementation of an REST API.

5.4.4.1. Rules for control and communication information models

The following rules and recommendations for CCIM modelling apply both to UML models as well as to JSON models. As already mentioned before, the JSON models are another representation of the UML models for a specific data encoding format. A CCIM class defines the schema of a CCIM Entity, thus a CCIM Entity is an instance of a CCIM Class.

1. A CCIM may consist of several CCIM Classes.
2. All CCIM classes should be derived from one base class which defines the common properties id, domain, name and description.
3. Properties of a CCIM class must consist of atomic types (integer, string, boolean, …), arrays and other CCIM Classes only.
4. Collections of CCIM entities must be represented by simple arrays, associative arrays (maps) are not supported.
5. Complex relationship like many to many relationships or cyclic relationship between classes should be avoided.
6. camelCase should be used when naming properties, underscores shall be omitted.
7. When defining CCIM Classes the plural form shall be used, e.g. World State**s** instead of World State.
8. The ISO 8601 standard should be used to encode date and time. UTC should be used for dates.

5.4.4.2. Rules for JSON models

There are also a few rules and recommendations which are specific to the JSON representation (REST Model) of CCIMS Entities. Since this JSON representation is related to the resource oriented architecture (ROA) and REST APIs, respectively, the terminology of the ROA is used to describe classes and instances. A CCIM Class in the context of a REST (JSON) Model is therefore called Resource Type or Resource Class and a CCIM Entity is called Resource Object, Resource Instance or just Resource.

The following rules and recommendations apply to REST (JSON) Models:

1. Supplemental properties which are not part of the originating UML Model but are added to a Resource Type of the REST Model must be prefixed with a dollar sign, e.g. *$ref*, *$self*, etc. The properties shall be in the following order: First come *$ref* or *$self* properties, then any other $-prefixed properties and at last all properties defined by the originating UML Model in the order they were specified in the UML Model.

2. Many-to-Many relationships between two Resource Types (classes) should be represented by a dedicated Resource Type that models those relationships.
3. In addition to the properties defined by a CCIM Class, a resource instance has an additional and mandatory *$self* property which contains the relative URI of the resource instance (self reference). The contents of the *$self* property must match the API path component of the location HTTP header file, e.g. location=http://someserver.com:8118/crsima/ICMM/v2/classes/earthquake.World State/31337, $self="/classes/earthquake.World State/31337"
4. References to other resources have to be expressed with help of the *$ref* property which contains an URI (RFC3986), that identifies the location of the JSON object being referenced. The *$ref* property is also a relative URI.
5. Collection resources (collections of complex resources) have to be represented by an array of JSON values that either contain resource instances or references to resource instances. Links to resource instances have to be represented by *$ref* properties. Additionally, a collection resource object has to define the properties *$offset*, *$limit*, *$first*, *$previous*, *$next* and *$last* to navigate through paginated collection resources.
    a. $offset defines the actual offset in a paginated collection
    b. $limit defines the maximum number of distinct entries in a paginated collection
    c. $first provides a relative URI to the first page of a paginated collection, e.g. "$first": "/my.persons?offest=0&limit=100". $first must always be specified.
    d. $last provides a relative URI to the last page of a paginated collection, e.g. "$last": "/my.persons?offest=300&limit=100". $last must always be specified.
    e. $prev provides a relative URI to the previous page of a paginated collection. $prev must not be specified if the current page is identical to $first.
    f. $next provides a relative URI to the next page of a paginated collection. $next must not be specified if the current page is identical to $last.

5.4.5. Rules for RESTful API specifications

Those are rules for the specification and implementation of RESTful APIs.

For the formal specification of REST APIs the Swagger API Documentation Toolkit shall be used: "*Swagger is a specification and complete framework implementation for describing, producing, consuming, and visualizing RESTful web services. With Swagger's declarative resource specification, clients can understand and consume services without knowledge of server implementation or access to the server code. The Swagger UI framework allows both developers and non-developers to interact with the API in a sandbox UI that gives clear insight into how the API responds to parameters and options.*"

https://developers.helloreverb.com/swagger/

REST APIs documented and visualised with the Swagger Framework can be easily tested by client developers and even by end users or pilot partners who want to learn about the technical details and capabilities of the Building Block Software Component exposing the API. Usage of the Swagger Framework is mandatory for all new REST APIs developed for the CRISMA Framework.

There are many best practices to be followed when designing and implementing REST APIs. Describing them all in detail would go beyond the scope of this document. Nevertheless, those are the most important rules and recommendations for the specification of RESTful APIs in CRISMA:

1. Each resource has a global lifetime unique ID. Resource types (classes) which represent the schema of a resource are uniquely indentified by their canonical name which consists of the name of the application domain (e.g. earthquake, laquila, nordicpilot, etc.) and the name of the resource class (e.g. World States) separated by a dot (e.g. earthquake.World States). Resource instances (objects) are uniquely identified by the id of their class (domain. class) and a key that is defined by the underlying service implementation (e.g. table key, UUID, ...). This unique id is also part of the URI of the resource. Therefore it has to be present in the location header of the server response and in *$self* property of the resource object or class.

2. The server response <u>shall</u> always provide the HTTP Header field '*Location*' containing the absolute URI (with host) of the resource. Each resource object returned by the API <u>shall</u> provide a "*$self*" property containing the relative URI of the resource. Further rules for mandatory properties of resource objects are defined by the rules and guidelines for CCIMs (section 5.4.4.1).

3. If the API supports different output formats (content encoding) of the same resource (e.g. JSON, XML, CSV), the resource format that shall be returned by the API <u>shall</u> be determined by the order of HTTP accept headers provided by the client. Alternatively, the API <u>should</u> support resource name extensions to determine the format requested by the client. E.g. /resource.*json* returns the JSON representation of the resource while /resource.*csv* returns the CSV representation of the resource. The <u>mandatory</u> resource format to be supported by any CRISMA API is JSON. Thus, the default behaviour of the API when no accept header and no resource name extensions is provided is to return the JSON representation of the resource.
   The API documentation <u>shall</u> inform about the output formats supported by the API.

4. We distinguish between collection resources and instance resources. Collection and instance resources are referenced by the canonical name of their resource type. /$domain.$class/ <u>shall</u> be used for access to a collection of all resources of this type and /$domain.$class/$key <u>shall</u> be used for access to a specific instance resource, e.g. GET /earthquake.World States/ returns all World State resource objects of the earthquake domain while GET/ earthquake.World States/aabb returns just the World State object 'aabb'.

5. HTTP operations are mapped to CRUD (create, read, update delete) in the following way:
   a. GET <u>shall</u> be used for read operations. The HTTP header of a GET response <u>should</u> optionally contain a *Link* header field with relation type "describedBy" that points to the type definition (class) of a resource instance, e.g. <http://example.com/ICMM/v2/classes/earthquake.World States>; rel="describedBy".
   b. DELETE <u>shall</u> be used for delete operations.
   c. PUT and POST <u>shall</u> be used for update and create operations.

i. PUT can be used for create whereby the unique id of the resource must be known by the client beforehand (e.g. PUT /earthquake.World States/aabbc).

ii. POST can be used for create whereby the unique id of the resource does not have to be known beforehand (e.g. POST /earthquake.World States/).The id of the newly created resource instance must be part of the server response (location header field).

iii. PUT and POST can be used for **full update** operations. In case of a full update the complete resource object must be send to the server regardless how many properties have changed. Update operations have to be executed against the URI of a particular resource instance, e.g. /earthquake.World States/aabbc.

iv. PUT and POST can also be used for **partial updates**. A partial update may send a JSON object containing only changed resource properties.

6. Deleting specific properties of a resource is considered as a special case of a partial update. It has to be realised by a PUT or POST request whereby the properties to be deleted are set to NULL.
7. GET, DELETE and PUT are idempotent operations, thus repeated calls shall result in the same server state, especially if PUT is used for full or partial updates.
8. A full or partial update or a create issued via PUT or POST method may return the complete modified or created resource object because the server could alter timestamps, version numbers, etc. of the resource object; therefore always the freshest instance of the modified resource should be made available to the client. This behaviour can be controlled by the optional query parameter *requestResultingInstance {true|false}*.
The API documentation shall inform about the default behaviour when no *requestResultingInstance* parameter is provided.
9. For all GET, POST, PUT, DELTE operations, the server shall always return

   d. correct and verbose HTTP Status codes in the HTTP header, e.g. 201 created;

   e. the location header which contains the full URI of the resource; and

   f. the correct media type, e.g. *application/json* for JSON resources.
10. APIs should be versioned and backwards compatible to previous versions. Major versions should become part of the API URI (e.g. /api/v2/). Optionally, the client may append the requested version to the media-type property in the HTTP header, e.g. application/json;application&v=1.
The API documentation shall provide information about the current version and backwards compatibility to previous versions.
11. When the API is queried to return complex resource objects (that are resources containing nested resources), the complex resource object may contain only the references of the nested resources. A resource reference is defined as "*$ref*" property that contains the relative URI of the resource. It is up to the client to resolve those linked resources. The same behaviour applies also to collection resources, e.g. a request to /earthquake.World States/ may return the stubs of

all World State objects of the earthquake domain.
The API documentation shall inform about the default behaviour whether complete or lightweight complex resource objects are returned.

12. A client <u>may</u> also control the behaviour of the API whether complete complex resource objects or lightweight complex resource objects consisting of references to nested resource objects shall be provided. This behaviour can be controlled by the <u>optional</u> *expand* and *level* query parameters which are appended to the GET resource request.

    g. The expand parameter has as value a list of comma separated resource property names. If a property points to a complex resource, the resource shall be expanded by the server.

    h. The level parameter has as value a positive integer that determines the lowest level to which resources are expanded.

    i. Expand and level parameters can be combined.

    j. If the expand parameter is provided and the level parameter is omitted, resources matching the expand properties are expanded to the lowest level.

    k. If the level parameter is provided and the expand parameter is omitted, any resources are expanded up to the level specified by the level parameter.

    l. If the expand parameter is omitted and the level parameter is set to -1, the API shall return a fully expanded resource.

The API documentation <u>shall</u> inform whether those parameters are supported or not and what the default behaviour of the API regarding resource expansion is when those parameters are not supported or not provided.

13. The client <u>may</u> also request the API to return partial resource objects. Therefore, the client appends the *fields* request parameter to the GET resource request and specifies in a comma separated list only those properties (*fields*) of the resource the client is interested in. Please note, that this mechanisms is only supported for top-level resource objects and the reduction is not applied on nested resources. The *fields* request parameter can be used in combination with the expand request parameter, though. This mechanism <u>shall not</u> be applied to the *$self* field which is mandatory for all resource objects.
    The API documentation <u>shall</u> inform whether the API supports the provision of partial resource objects.

14. The representation of resource objects containing null values <u>should</u> be controllable by the *omitNullProperties* parameter that can be used in a GET request. If the parameter is set to true, properties that have a null value <u>shall not</u> be present in the returned resource object.
    The API documentation <u>shall</u> inform whether this parameter is supported or not.

15. The amount of resource objects returned by queries to collection resources <u>should</u> be controllable by the *offset* and *limit* query parameters. Thereby, *limit* specifies the maximum number of resources returned by one GET resource request to the collection resources and *offset* the offset in the entire list resources available in the particular collection. In addition to the array of resource objects or resource links collection resources <u>shall</u> contain a set of specific fields which provide links to navigate through the collection. (see also rules and guidelines for CCIMs, section 5.4.4.1).

The API documentation shall inform whether those parameters are supported or not and what their default values are.

16. Error messages should be as descriptive as possible and should provide as much information as possible to trace back the error. Besides the correct HTTP status codes (404, 409, 500, ...) in the HTTP header, in case of an error the API shall also return a specific error resource object which contains detailed information on the cause of the error. Thereby, the error message must not contain security sensitive information like password, etc.

5.4.6. Rules for user interface integration and development

The following rules apply to the development of Composite UI Modules and the related UI Integration and UI Mashup Platform.

1. Each Composite UI Module shall be realised as distinct HTML Widget which consists of a static part (HTML/CSS + other static resources) defining layout and structure of the user interface and a dynamic part (JavaScript) providing dynamic content and interaction functionalities.
2. Composite UI Modules must be implemented in HTML5, CSS3 and JavaScript (ECMA 262).
3. Composite UI Modules must not rely on any type of external plug-ins like Oracle Java, Adobe Flash, Microsoft Silverlight, and others.
4. Composite UI Modules shall be realised as purely client-side and server-agnostic solution that does not require any specific server-side infrastructure (e.g. enterprise application server, servlet container, portal server, ...) apart from the Infrastructure Building Blocks provided by the CRISMA Framework (e.g. ICMM, UI Mashup Platform for Mashable Composite UI Modules, etc.).
5. Composite UI Modules should not rely on proprietary APIs or services that pose the risk of exposing security and privacy sensitive information to third parties.
6. Composite UI Modules shall expose their programmatically accessible functionality via a JavaScript API.
7. As any other Federates, Composite UI Modules shall support bidirectional communication with the ICMM. Specifically, Composite UI Modules must be able to receive action requests or event notifications dispatched by the ICMM and the Pub/Sub context Broker Building Block, respectively. Thereby, appropriate transport techniques like WebSockets, Server-sent Events or long-polling shall be supported by Composite UI Module.
8. A Composite UI Module must provide a self-description that informs about its capabilities. Specifically, this self-description has to be provided as JSON-encoded CCIM and has to contain general descriptive and technical meta-information as well as a description of the Composite UI Modules' JavaScript API.
9. The UI Integration Platform must be able to render and execute Composite UI Modules. Thus, it must support HTML5, CSS3 and JavaScript.
10. A Mashable Composite UI Module must be able to interact with the UI Mashup Platform or other Mashable Composite UI Modules through the Mashup JavaScript API which is provided by the UI Mashup Platform.

11. The UI Mashup Platform shall support to interactively compose Mashable Composite UI Modules and Integrated Widgets to rich internet applications.
12. Mashable Composite UI Modules should be based upon Composite UI Modules which can be used independently of a particular UI Mashup Platform in a CRISMA Application.
13. Communication and Interaction between Mashable Composite UI Modules among each other and between the UI Mashup Platform should be based upon functions exposed by the JavaScript API of wrapped Composite UI Modules.
14. The UI Integration Platform should be able to interact with Composite UI Modules via their JavaScript API.
15. The UI Integration Platform should be able to interact with Mashable Composite UI Modules that are assembled in a UI Mashup Platform. This interaction should either be performed via the JavaScript API of the Composite UI Modules that have been wrapped into Mashable Composite UI Modules or via APIs of the respective UI Mashup Platform.
16. The business logic of Composite UI Modules that need to perform demanding and long-running processes should be encapsulated in a dedicated server-side component.

5.4.7. Rules for JavaScript APIs

As described in the previous chapter, Composite UI Modules shall expose a JavaScript API that can be accessed by the UI Integration Platform, possibly also by other Composite UI Modules. There are currently no specific rules for the design of such an API apart from the general rule that the Google JavaScript Language Rules and the JavaScript Style Guide[9] (ref) shall be followed when developing JavaScript APIs and Composite UI Modules in general. Especially the rules for documentation of functions with JSDoc[10] and exception handling are important.

[9] https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml
[10] https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml?showone=Comments#Comments

6. Conclusions

The WP32 team had the task of describing the cohesive element of the CRISMA project – the CRISMA Framework Architecture and the Building Blocks pertinent to this framework.

In order to provide a solid ground for the architecture, the initial WP32 work was to agree on a consistent set of questions that need to be answered before designing the architecture (section 1.7.1), assuring each of the questions is answered in a way acceptable to the whole consortium and transforming the answers into high-level architectural design decisions (D32.1, 2013). These decisions were then materialized in the technology-independent ("functional") and technology-dependent ("implementation") architecture, following the methodology described in section 1.7.2.

As a result, this document offers comprehensive answers to "Mission" (Section 2), "Concepts" (Section 3), "Realisation" (Section 4) and "Implementation" (Section 5) of the CRISMA Framework and will be useful within the project on several levels:

- First, the inherent traceability introduced by this process makes it easy to follow the line of argumentation from requirements to Building Blocks and understand the consequences of any changes that may be requested by various stakeholders in the future.
- Second, the discussion of the (consequences of) various architectural concepts presented in section 3 makes it easier for application builders to define their Application Architecture – possibly exposing gaps which can be mended in the second development cycle.
- Third, the document presents a solid base for implementation and integration work in SP3 and SP4.
- Finally, the Implementation Architecture of the CRISMA Framework guides and supports the development of Pilot Applications in SP5.

Some important findings of the architecture team are summarized below:

- The CRISMA Framework shall facilitate building of the crisis-management planning and training applications. It will be implemented as a software framework which offers a set of defined functionalities represented by Software Components (Building Blocks - BBs).
- Aspects of transferability and integration with legacy systems have been incorporated into the architectural design process and thus also into the resulting software framework.
- At the heart of the framework are ICMM "middleware" components essentially managing Control and Communication Information on available world State Data Slots and Simulation Models, Transitions, etc. The ICMM provides furthermore common auxiliary functionality like access control, versioning, discovery, etc.
- Any Software Component that is capable to interact with the Middleware Infrastructure is considered a CRISMA Federate. The Federates developed in the scope of SP3 are called "Building Blocks" and, as a rule, expose a well-defined RESTful API aligned with the ICMM APIs.
- Any other component, model or system that is not aware of the ICMM APIs but takes part in an interaction of CRISMA Federates is called "Integrated Component". Integrated components aren't bound by the rules of the CRISMA Framework Architecture.

- All Federates must exchange the Control and Communication Information through the ICMM. However, they can exchange the data directly (for instance, the data resulting from one Simulation Model can be directly used by next model in processing chain).
- In addition to the integration at service level, the CRISMA Framework Architecture foresees integration on the GUI level, by means of functional GUI widgets (Composite UI Modules) linked to Building Blocks. These widgets can be combined in ad-hoc web mashup applications as needed. The result can be either used stand-alone or embedded in existing applications.
- Functionalities provided by CRISMA Building Blocks include support for the integration of data and Simulation Models required by CRISMA applications, interaction of users with these models, as well as the visualisation and support for evaluation of the effects of specific mitigation measures (Crisis Management Scenario Analysis).
- The architecture does not directly support exchange of data between two models during the model run (“tight binding”). If needed, the tightly bound models can be realized independent of CRISMA (e.g. using OpenMI), and exposed as a single “composite” model within the framework. As a consequence, the architecture also does not support elaborate interaction of models during the run.
- The “decision making” related to possible “what if”-scenarios within the CRISMA framework corresponds to changing of the parameters exposed by the Simulation Case. This can be performed at specific Transition Points. Users will be able to navigate a simulated crisis scenario (over time), try out different crisis management actions and eventually compare the results. On top of this an indicator based decision support concept is defined that even ranks the outcome of a specific action according to a number of criteria (Multi-Criteria-Analysis).

The feedback on D32.1 architecture document and lessons learned from the first development cycle have been taken into account. The document is well-organised and provides essential information for the follow-up activities in related CRISMA work packages, especially at the level of the implementation and integration of the respective Software Components in WPs 34 and 35 and the specification and implementation of the Pilot Applications in SP5.

Nevertheless, the realisation (chapter 4) and implementation (chapter 5) parts of the architecture should not be understood as “cast in stone”. To a large extent, the current specification of Functional Building Blocks and related Software Components is an offer for developers and integrators in sub projects three, four and five and will be adapted and extended in the future through the CRISMA Catalogue.

7. References

AGPL. 2007. GNU Affero General Public License, Version 3. Free Software Foundation, Inc. 19 November 2007. https://www.gnu.org/licenses/agpl-3.0.html.

Beyer, U., Cohnitz, S., Stachowiak, J., Usov, A., Rome, E., Beyel, C. & Börding, J. 2010. DIESIS – Design of an Interoperable European Federated Simulation network for critical InfraStructures, Deliverable D4.1b Final Architectural Design, Collaborative Project, Specifically Targeted Research Project (STReP), FP7 RI Grant Agreement N° 212830.

BSD 1998. The BSD 2-Clause License. http://opensource.org/licenses/bsd-license.php.

Carson, J.S. 2005. Introduction to Modeling and Simulation. In Proceedings of the 2005 Winter Simulation Conference, 28/02/2012.

D11.1. 2012. Frings, S. (Ed.). Initial consolidation report of the internal workshops. Deliverable D11.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D21.1. 2012. Aubrecht, C. (Ed.). Technology Inventory for Crisis Management Tools & Models. Deliverable D21.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D23.1. 2011. Leone, M. (Ed.). Requirements and Use Cases. Deliverable D23.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D25.1. 2014. Engelbach, W. (Ed.). Updated technology inventory, requirements, scenarios, criteria and key performance indicators. Deliverable D25.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D31.1. 2012. Taveter, K. (Ed.). Technology, Concepts & Technical Requirements Report V1. Deliverable D31.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D31.2. 2013. Taveter, K. (Ed.). Technology, Concepts & Technical Requirements Report V2. Deliverable D32.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D32.1. 2013. Dihé, P. (Ed.). ICMS Functional Architecture Document v1. Deliverable D32.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D34.1. 2013. Dihé, P. (Ed.). ICMS Building Blocks V1. Deliverable D34.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D35.1. 2014. Kutschera, P. (Ed.). ICMS Framework V1. Deliverable D35.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D42.1. 2013. Garcia-Aristizabal, A. (Ed.). Multi-risk analysis; scenarios of cascading effects; concept model of dynamic scenario assessment. Deliverable D42.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D43.1. 2013. Zuccaro, G. (Ed.). Version 1 of Dynamic vulnerability functions, Systemic vulnerability, and Social vulnerability . Deliverable D43.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D44.1. 2013. Broas, P. (Ed.). Version 1 of Model for decision-making assessment, Economic impacts and consequences, and Simulation. Deliverable D44.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D51.1. 2013. Erlich, M., Cabal, A. (Eds.). Site Status. Deliverable D51.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D51.2. 2013. Erlich, M., Cabal, A. (Eds.). Pilot user guide. Deliverable D51.2 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

D63.1. 2013. Stevenot, B. (Ed.). Report on market analysis. Deliverable D63.1 of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

David, O., Carlson, J.R., Leavesley, G., Ascough II, J., Geter, F., Rojas, K. & Ahuja, L. 2010. Object Modeling System v3.0 Developer and User Handbook, Aug 10, 2010.

Denzer, R., Güttler, R., Schlobinski, S. & Williams, J. 2003. A Decision Support System for Marine Applications. Environmental Informatics Group, Saarbrücken, Germany, Marine Tech South, Southampton, United Kingdom.

DoW. 2013. Heikkilä, A-M (Ed.). GRANT Agreement Part B (Description of Work) of the European Integrated Project CRISMA, FP7-SECURITY- 284552.

Dublin Core. 2013. Dublin Core. https://en.wikipedia.org/wiki/Dublin_Core.

ECMA. 2011. ECMAScript® Language Specification. Ecma International, Geneva. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

FI-WARE. 2011. FUTURE INTERNET Core Platform. http://www.fi-ware.eu/.

GML. 2012. Geography Markup Language. http://www.opengeospatial.org/standards/gml.

IEEE. 2010. IEEE 1516-2010 IEEE Standard for Modeling And Simulation (M&S) High Level Architecture (HLA). New York, NY: Institute of Electrical and Electronics Engineers.

IEEE. 2011. IEEE 1516E-2011 IEEE Standard for Modeling And Simulation (M&S) High Level Architecture (HLA). New York, NY: Institute of Electrical and Electronics Engineers.

IETF. 2013. Internet Engineering Task Force. http://www.ietf.org/.

ISO/IEC. 1998. ISO/IEC 10746-1:1998 (E). Information technology – Open Distributed Processing – Reference model.

ISO_19115. 2003. ISO 19115:2003 GEOGRAPHIC INFORMATION - METADATA. http://www.isotc211.org/Outreach/ISO_TC_211_Standards_Guide.pdf.

ISO_19119. 2003. ISO 19119:2005 GEOGRAPHIC INFORMATION – SERVICES . http://www.isotc211.org/Outreach/ISO_TC_211_Standards_Guide.pdf.

Lee, Y.T. 1999. Information modelling from design to implementation. National Institute of Standards and Technology.

Moore, R. 2010. OpenMI Standard 2 Specification. The OpenMI Association Technical Committee (OATC), 30/11/2010.

NGSI. 2010. NGSI Context Management. Candidate Version 1.0 – 03 Aug 2010. Open Mobile Alliance. http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf

NetCDF. 2011. OGC network Common Data Form (netCDF) standards suite. http://www.opengeospatial.org/standards/netcdf

O&M. 2011. Observations and Measurements. http://www.opengeospatial.org/standards/om.

OAuth. 2012. OAuth 2.0 Specification - RFC 6749. http://tools.ietf.org/html/rfc6749.

OGC. 2013. The Open Geospatial Consortium (OGC).http://www.opengeospatial.org/ogc.

OpenID. 2007. OpenID Authentication 2.0. http://openid.net/specs/openid-authentication-2_0.html.

Paulheim, H. 2010. Integration of User Interface Components based on Ontologies and Rules. SAP Research Center Darmstadt, Germany.

Plantenga, T. & Friedman-Hill, E. 2010. Integrated Modelling, Mapping, and Simulation (IMMS) Framework for Planning Exercises, Sandia National Laboratories.

Powell, E. & Noseworthy, J. 2012. The Test and Training Enabling Architecture (TENA), SAIC, Inc, US Government Contracts 1435-04-01-CT-31085, DASG60-02-D-0006-40, and DASG60-02-D-0006-122.

Probst, F. & Paulheim, H. 2010. Application Integration on the User Interface Level: an Ontology-Based Approach. SAP Research Center Darmstadt.

REST. 2013. Representational state transfer.
https://en.wikipedia.org/wiki/Representational_state_transfer.

RFC. 2013. Request for Comments. https://en.wikipedia.org/wiki/Request_for_Comments.

RFC_1945. 1996. Hypertext Transfer Protocol – HTTP/1.0. http://www.ietf.org/rfc/rfc1945.txt.

RFC_2119. 1997. Key words for use in RFCs to Indicate Requirement Levels.
https://www.ietf.org/rfc/rfc2119.txt.

RFC_2616. 1999. Hypertext Transfer Protocol – HTTP/1.1. http://www.ietf.org/rfc/rfc2616.txt.

RFC_3986. 2005. Uniform Resource Identifier (URI): Generic Syntax.
http://www.ietf.org/rfc/rfc3986.txt.

RFC_4627. 2006. The application/json Media Type for JavaScript Object Notation (JSON).
https://www.ietf.org/rfc/rfc4627.txt.

RFC_4918. 1999. HTTP Extensions for Web Distributed Authoring and Versioning - WebDAV.
https://tools.ietf.org/html/rfc4918.

RM-OA. 2007. Usländer, T. (Ed.). Reference Model for the ORCHESTRA Architecture Version 2 (Rev. 2.1). OGC Best Practices Document 07-097, 2007.

Rome, E., Bologna, S., Gelenbe, E., Luiijf, E. & Masucci, V. 2009. DIESIS: An Interoperable European Federated Simulation Network for Critical Infrastructures. In Proceedings of the 2009 SISO European Simulation Interoperability Workshop (ESIW'09), Istanbul, Turkey, July 13–16, pp. 139–146. San Diego, CA: Simulation Councils, Inc.

SANY-SA. 2009. SANY Consortium, Specification of the Sensor Service Architecture V3, SANY public deliverable, http://www.sany-ip.eu, November 2009.

Schmidt, J., Matcham, I., Reese, S., King, A., Bell, R., Henderson, R., Smart, G., Cousins, J., Smith, W. & Heron, D. 2011. Quantitative multi-risk analysis for natural hazards: a framework for multi-risk modelling. Springer Science+Business Media B.V. 2011.

Semantic Web. 2013. Semantic Web. World Wide Web Consortium.
http://www.w3.org/standards/semanticweb/

SensorML. 2007. Open Geospatial Consortium. Sensor Model Language.
http://www.opengeospatial.org/standards/sensorml

SOA-RM. 2006. OASIS Reference Model for Service Oriented Architecture 1.0. Committee Specification 1, 2 August 2006.

SOS. 2012. Sensor Observation Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/sos.

SPS. 2011. Sensor Planning Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/sps.

Stachowiak, H. 1973. Allgemeine Modelltheorie. Wien-New York.

SWEdata. 2011. SWE Common Data Model Encoding Standard. Open Geospatial Consortium. http://www.opengeospatial.org/standards/swecommon.

SWEservice. 2011. SWE Service Model Implementation Standard. Open Geospatial Consortium. http://www.opengeospatial.org/standards/swes.

TOGAF. 2011. The Open Group Architecture Framework, Open Group Standard, Version 9.1, Enterprise Edition.

UML. 2013. Unified Modelling Language. http://en.wikipedia.org/wiki/Unified_Modeling_Language.

W3C. 2012. World Wide Web Consortium. http://www.w3.org/Consortium/.

WCPS. 2009. Web Coverage Processing Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/wcps.

WCS. 2010. Web Coverage Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/wcs.

Web Design. 2013. Web Design and Applications. World Wide Web Consortium. http://www.w3.org/standards/webdesign/.

WFS. 2010. Web Feature Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/wfs.

WMS. 2006. Web Map Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/wms.

WNS. 2003. Web Notification Service. Open Geospatial Consortium. http://portal.opengeospatial.org/files/?artifact_id=1367.

WPS. 2007. Web Processing Service. Open Geospatial Consortium. http://www.opengeospatial.org/standards/wps.

WSN. 2006. Web Services Notification. Open Geospatial Consortium. https://www.oasis-open.org/standards#wsnv1.3.

XML. 2010. XML Technology. World Wide Web Consortium. http://www.w3.org/standards/xml/.

APPENDIX (A) Core CCIM formal specification

This appendix provides a complete overview of all element details of the Core Control and Communication Information Model of the CRISMA Framework Architecture. The following class diagram gives a complete overview of the current Core CCIM of CRISMA (Figure 88).

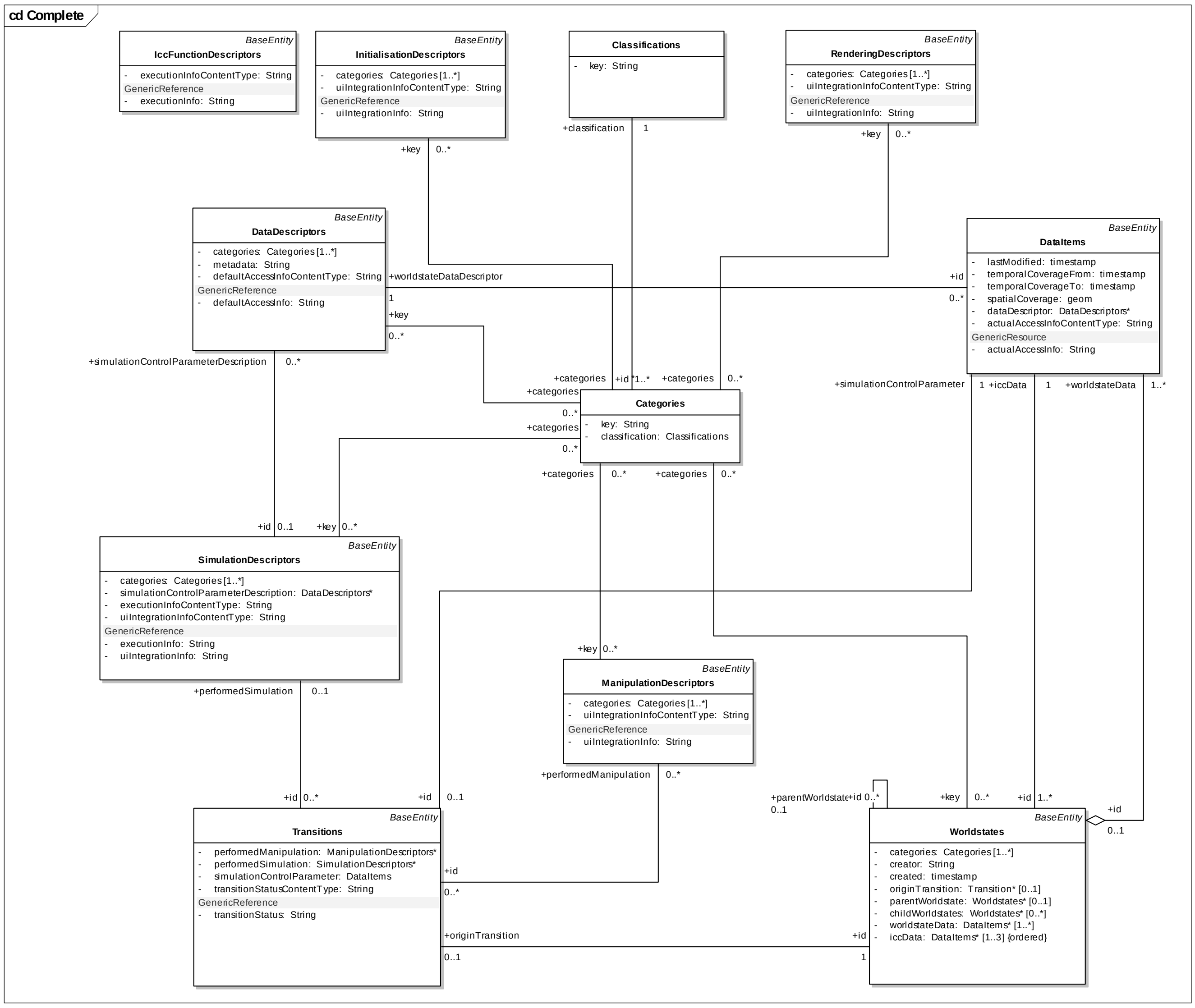

cd Complete
BaseEntity
IccFunctionDescriptors
- executionInfoContentType: String
GenericReference
- executionInfo: String
BaseEntity
InitialisationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String
Classifications
- key: String
BaseEntity
RenderingDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String
BaseEntity
DataDescriptors
- categories: Categories [1..*]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String
Categories
- key: String
- classification: Classifications
BaseEntity
SimulationDescriptors
- categories: Categories [1..*]
- simulationControlParameterDescription: DataDescriptors*
- executionInfoContentType: String
- uiIntegrationInfoContentType: String
GenericReference
- executionInfo: String
- uiIntegrationInfo: String
BaseEntity
ManipulationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String
BaseEntity
Transitions
- performedManipulation: ManipulationDescriptors*
- performedSimulation: SimulationDescriptors*
- simulationControlParameter: DataItems
- transitionStatusContentType: String
GenericReference
- transitionStatus: String
BaseEntity
Worldstates
- categories: Categories [1..*]
- creator: String
- created: timestamp
- originTransition: Transition* [0..1]
- parentWorldstate: Worldstates* [0..1]
- childWorldstates: Worldstates* [0..*]
- worldstateData: DataItems* [1..*]
- iccData: DataItems* [1..3] {ordered}
+key 0..*
+classification 1
+key 0..*
+worldstateDataDescriptor
1
+key
0..*
+id
0..*
+simulationControlParameterDescription 0..*
+categories +id 1..* +categories 0..*
+categories
0..*
+categories
0..*
+categories 0..* +categories 0..*
+simulationControlParameter 1 +iccData 1 +worldstateData 1..*
+id 0..1 +key 0..*
+key 0..*
+performedSimulation 0..1
+performedManipulation 0..*
+id 0..* +id 0..1
+parentWorldstate +id 0..* 0..1
+key 0..* +id 1..*
+id 0..1
+id
0..*
+originTransition
0..1
+id
1

Figure 88: Core CCIM Class Diagram with Relationships.

BaseEntity

A Base Class that defines common properties needed by all classes. All classes should be derived from this class.

Table 3: Specification of Base Entity Attributes.

Attribute	Notes
id String **Private**	**Unique id of the entity.**
domain String **Private**	**Name of the domain the entity belongs to.**
name String **Private**	**Optional name of the entity.**
description String **Private**	**Optional description of the entity.**

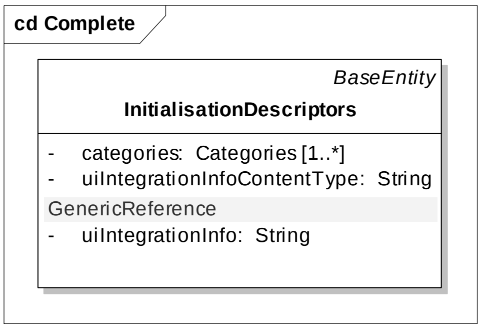

cd Complete
BaseEntity
InitialisationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String

Figure 89: BaseEntity Class Diagram.

Categories

A class representing an arbitrary category that can be used to categorise scenarios, simulations, etc. Categories are furthermore used to categorise World States, data items and descriptors, transitions, etc. They can be used for search and filtering, e.g. to filter for all data descriptors that represent World State data, for all transitions that represent a manipulation, etc.

Table 4: Specification of Categories Attributes.

Attribute	Notes
key String **Private**	**A static key of the category that can be referenced in applications. In contrast to the name property, it will not be changed (e.g. depending on the language of the application).**
classification Classifications **Private**	**Further classification of the category, e.g. scenario types, crisis management phases, etc.**

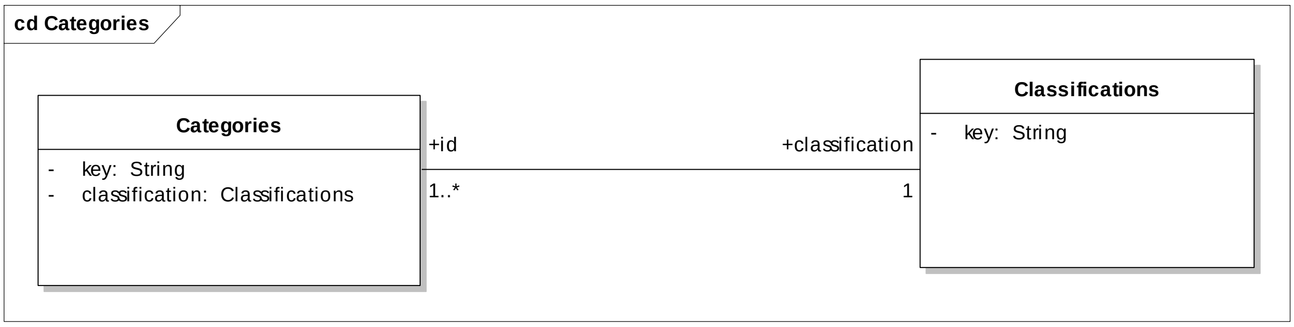


Figure 90: Categories Class Diagram with Relationships.

Classifications

A class that represents a classification of categories, e.g. scenario types, crisis management phases, etc.

Table 5: Specification of Classifications Attributes.

Attribute	**Notes**
key String **Private**	**A static key that can be referenced in applications. In contrast to the name property, it will not be changed (e.g. depending on the language of the application).**

DataDescriptors

A DataDescriptor provides invariant meta-information about an arbitrary dataset (World State Data, Simulation Control Parameters, ICC Data). A DataItem is associated with a DataDescriptor. The DataDescriptor contains thereby static information like the URI of a data source (Data Integration Building Block – data access service, e.g. OGC WFS) while the DataItem provides information to identify an individual DataItem or a specific version of a DataItem (e.g. a specific request parameter, a filename, etc.). A DataDescriptor can also be considered as describing the type or source of the data and the DataItem as representing a concrete data set of this type.

Table 6: Specification of DataDescriptors Attributes.

Attribute	**Notes**
categories Categories **Private Collection [1..*]**	**Categories that describe the Data Descriptors, e.g. World State Data, Simulation Control Parameter data, etc.**
metadata String **Private**	**This field contains either a link to an external INSPIRE metadata document describing the data set (e.g. the URL of an XLS file, a metadata catalogue entry, a website, etc.) or the INSPIRE metadata itself (XML, JSON, TEXT, ...). This is descriptive information about the data source, thus it is not required that the contents of this field are machine-processable. Nevertheless, it is recommended to provide the INSPIRE metadata in JSON format so that it can easily be processed by web-based clients.**
defaultAccessInfoContentType String **Private**	**Content Type of the specific access info for this Data Descriptor.**
defaultAccessInfo String **Private GenericReference**	**A GenericReference that describes the technical access information of the actual data source, e.g. URI and credentials (e.g. WFS URI). This is invariant meta-information that stays the same for all DataItems that are described with this DataDescriptor.**

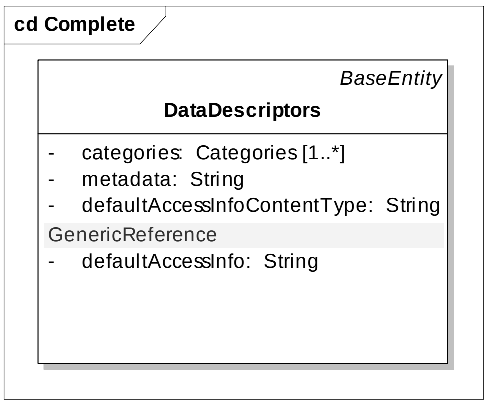

cd Complete
BaseEntity
DataDescriptors
- categories: Categories [1..*]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String

Figure 91: DataDescriptors Class Diagram.

DataItems

This class represents an actual data set (instance), e.g. World State Data, ICC Data, Simulation Control Parameters, etc.

Table 7: Specification of DataItems Attributes.

Attribute	Notes
lastModified timestamp **Private**	**Last modification timestamp of the data item.**
temporalCoverageFrom timestamp **Private**	**The temporal coverage of the data (start date).**
temporalCoverageTo timestamp **Private**	**The temporal coverage of the data (end date).**
spatialCoverage geom **Private**	**The spatial coverage (geometry) of the data.**
dataDescriptor DataDescriptors **Private**	**Reference to a DataDescriptor object containing invariant control and meta-information about the World State Data, e.g. access information (URI) for a Data Integration Building Block.**
actualAccessInfoContentType String **Private**	**Content Type of the actual access info of the DataItem.**
actualAccessInfo String **Private GenericResource**	**This is the actual information on how to access the data or for very small and simple datasets (e.g. ICC Data) it may contain also the data itself. In general it provides information to indentify an individual DataItem or a specific version of a DataItem (e.g. a specific request parameter, a filename, etc.). It is also related to a DataDescriptor whereby the DataDescriptor may provide invariant and general access information (base URI, database name, table name, ...) and the actual access information to indentify a data set may consist of request parameters, table names, filenames, etc. Like any GenericReference, actual access information may be described by an Application CCIM.**

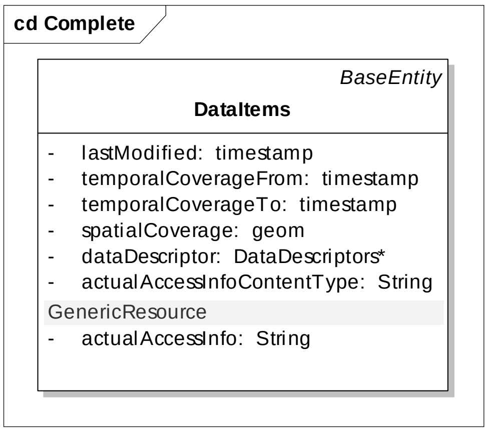

cd Complete
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String

Figure 92: DataItems Class Diagram.

ExternalResource

This **metaclass** represents an external resource that is referenced by a GenericReference. The External Resource may be defined by another application specific CCIM. Thus, External Resource is a defined extension point of the Core CCIM. The actual content of the External Resource is described by a contentType property.

GenericReference

This **stereotype** represents a generic reference to an external resource, e.g. a JSON or XML document, a link to a JSON or XML document, etc. In general domain specific information that may be used by (application specific) components. The content of a generic reference is identified by a content type attribute. Generic References of type CCIM Reference describe a reference to an arbitrary CCIM JSON object (CCIM entity). The CCIM Reference is used to establish relations to CCIM entities that are not known during the design time of the Core CCIM. Generic References provide an extension point where application specific CCIMs can be plugged-in. CCIM Reference is not an instantiable Class but a stereotype of a string attribute. The value of the attribute contains either a complete CCIM JSON object or a link to a CCIM JSON object (an object with one $ref property).

IccFunctionDescriptors

IccFunctionDescriptors provide Information about the ICC Functions that are executed after each World State transition.

Table 8: Specification of IccFunctionDescriptors Attributes.

Attribute	Notes
executionInfoContentType String **Private**	**Content Type of the specific execution info for this IccFunctionDescriptor.**
executionInfo String **Private GenericReference**	**Information on how to execute the ICC Function, may include an execute command, a service URL, an API call, etc. The concrete details depend on the types of ICC functions available in a particular CRISMA Application.**

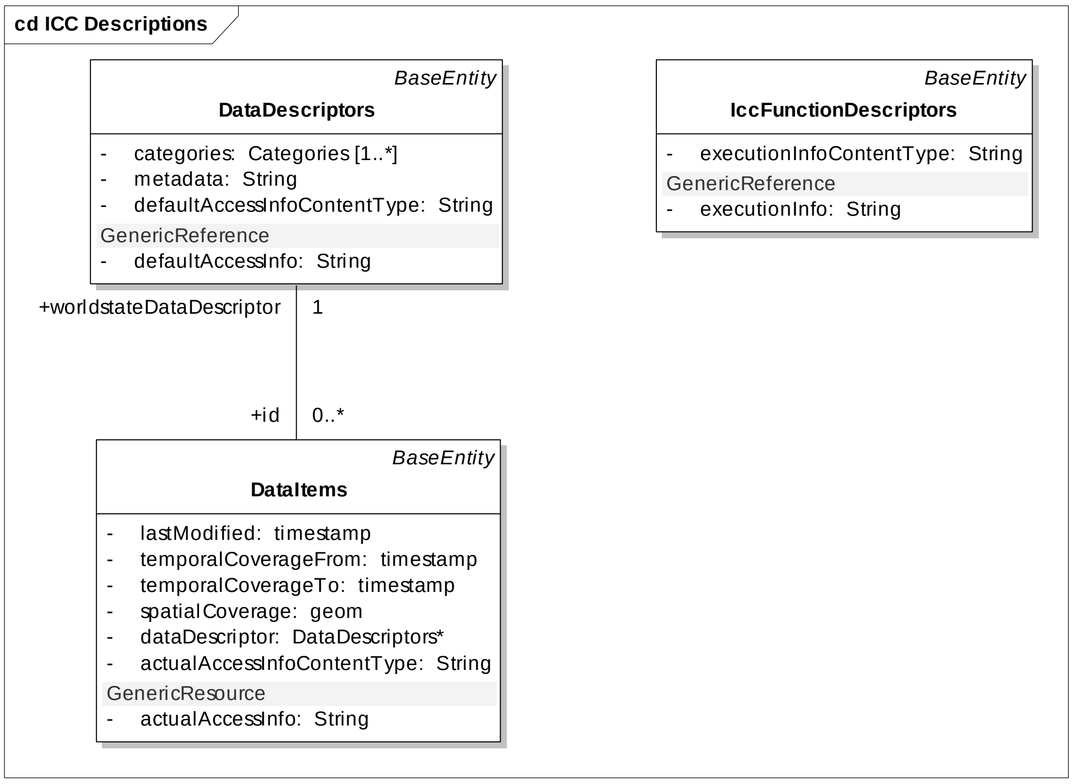

cd ICC Descriptions
BaseEntity
DataDescriptors
- categories: Categories [1..*]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
BaseEntity
IccFunctionDescriptors
- executionInfoContentType: String
GenericReference
- executionInfo: String
+worldstateDataDescriptor
1
+id
0..*
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String

Figure 93: IccFunctionDescriptors Class Diagram with Relationships.

InitialisationDescriptors

Initialisation descriptors describe user interfaces and business logic (e.g. Composite UI Modules, Setup and Configuration Building Blocks) that are needed to initialize the set of simulation models, the initial World State, descriptors, etc. This refers to the initial setup and configuration of the CRISMA Application.

Table 9: Specification of InitialisationDescriptors Attributes.

Attribute	Notes
categories Categories **Private Collection [1..*]**	**Categories that describe the Initialisation Descriptors.**
uiIntegrationInfoContentType String **Private**	**Content Type of the specific UI integration info for this Initialisation Descriptor.**
uiIntegrationInfo String **Private GenericReference**	**This is the actual information on how to instantiate and integrate the respective initialisation user interface (Composite UI Module), depending on the CRISMA Application, this field may contain an URI that points to a Composite UI Module instance..**

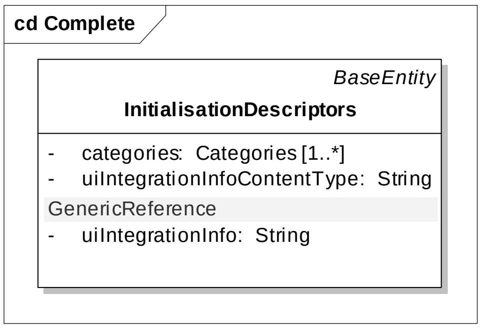

cd Complete
BaseEntity
InitialisationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String

Figure 94: InitialisationDescriptors Class Diagram.

ManipulationDescriptors

ManipulationDescriptors refer to World State Transitions and describe the user interfaces and business logic (e.g. Composite UI Modules, World States Widget Building View and associated Widgets) to manipulate the changeable parameters of a World State.

Table 10: Specification of ManipulationDescriptors Attributes.

Attribute	Notes
categories Categories **Private Collection [1..*]**	**Categories that describe the Initialisation Descriptors.**
uiIntegrationInfoContentType String **Private**	**Content Type of the specific UI integration info for this Initialisation Descriptor.**
uiIntegrationInfo String **Private GenericReference**	**This is the actual information on how to instantiate and integrate the respective initialisation user interface (Composite UI Module), depending on the CRISMA Application, this field may contain an URI that points to a Composite UI Module instance.**

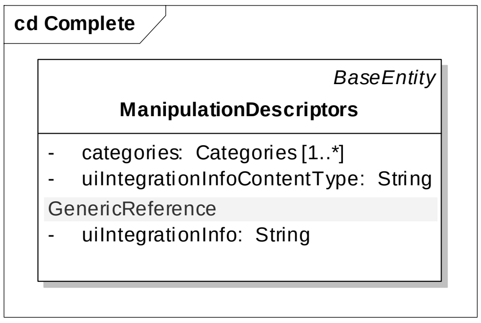

cd Complete
BaseEntity
ManipulationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String

Figure 95: ManipulationDescriptors Class Diagram.

RenderingDescriptors

A Rendering Descriptor describes a User Interfaces to visualize the World State (World State View Building Block and related Widgets).

Table 11: Specification of RenderingDescriptors Attributes.

Attribute	Notes
categories Categories **Private Collection [1..*]**	**Categories that describe the Rendering Descriptors**
uiIntegrationInfoContentType String	**Content Type of the UI integration info for the Rendering**

Attribute	Notes
Private	**Descriptor.**
uiIntegrationInfo String **Private GenericReference**	**This is the actual information on how to instantiate and integrate the respective (World State) user interface (Composite UI Module, World States Widget Building Block). Depending on the CRISMA Application, this field may contain an URI that points to a Composite UI Module.**

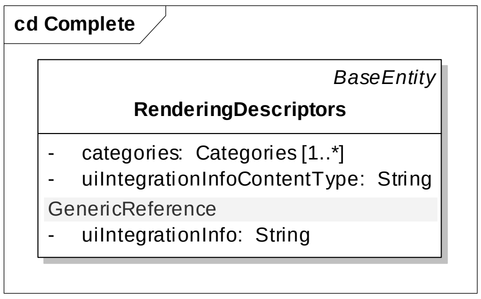

cd Complete
BaseEntity
RenderingDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String

Figure 96: RenderingDescriptorsClass Diagram.

SimulationDescriptors

A SimulationDescriptor provides information about a Federated Simulation (a simulation consisting of one or more simulation models wrapped as one WPS service). It holds references (Generic Reference) to concrete descriptions on how to execute the simulation (Simulation Model Integration Building Block) and by which user interface (Simulation Model Interaction Building Block) to control (e.g. parameterise, monitor) the simulation.

Table 12: Specification of SimulationDescriptors Attributes.

Attribute	Notes
categories Categories **Private Collection [1..*]**	**Categories that describe the Simulation Descriptors, e.g. the type of the simulation (impact scenario, ...)**
simulationControlParameterDescription DataDescriptors **Private**	**A description of the simulation model control parameters.**
executionInfoContentType String **Private**	**Content Type of the specific execution info for this Simulation Descriptor.**
uiIntegrationInfoContentType String **Private**	**Content Type of the specific UI integration info for this Simulation Descriptor.**
executionInfo String **Private GenericReference**	**Information on how to execute the federated simulation. In general it will contain information on the WPS that provides the simulation (Simulation Model Integration Building Block).The concrete details depend on the types of simulations available in a particular CRISMA Application.**
uiIntegrationInfo String **Private GenericReference**	**This is the actual information on how to instantiate and integrate the respective simulation control user interface (Composite UI Module, Simulation Model Interaction Building Block). Depending on the CRISMA Application, this field may contain an URI that points to a Composite UI Module instance which is able to configure a simulation, monitor its progress, etc..**

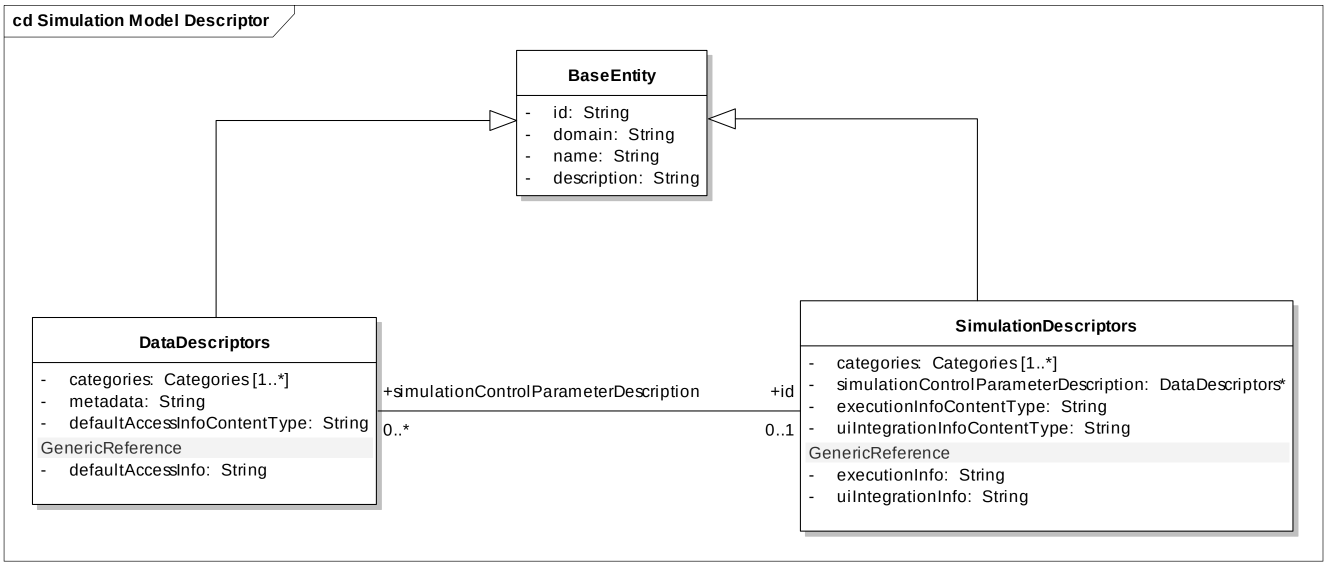

cd Simulation Model Descriptor
BaseEntity
- id: String
- domain: String
- name: String
- description: String
DataDescriptors
- categories: Categories [1..*]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
+simulationControlParameterDescription
0..*
+id
0..1
SimulationDescriptors
- categories: Categories [1..*]
- simulationControlParameterDescription: DataDescriptors*
- executionInfoContentType: String
- uiIntegrationInfoContentType: String
GenericReference
- executionInfo: String
- uiIntegrationInfo: String

Figure 97: SimulationDescriptors Class Diagram with Relationships.

The class diagram shown in Figure 97 visualises all classes of the Core CCIM that form a Simulation Case, that is, a concrete setup of a CRISIS Management Application.

Transitions

This class represents a concrete World State Transition. It contains information on the performed manipulation or simulation, the actual simulation control parameters and further meta-information (e.g. which parts of the World State were changed).

Table 13: Specification of Transitions Attributes.

Attribute	Notes
performedManipulation ManipulationDescriptors **Private**	**This property will contain information about the type of manipulations that have been performed.**
performedSimulation SimulationDescriptors **Private**	**This property will contain information about the type of simulation that has been performed.**
simulationControlParameter DataItems **Private**	**This property will contain the actual simulation control parameter data.**
transitionStatusContentType String **Private**	**Content Type of the specific transition status of this transition.**
transitionStatus String **Private GenericReference**	**This Generic Reference provides the detailed transition status information. It may be represented by an arbitrary CRISMA JSON object that is defined for a specific Application CCIM and may be injected dynamically, e.g. by the actual simulation execution building block. It may also point to the raw transition status, e.g. a logfile, a link to a website with status messages, a WPS url, etc.**

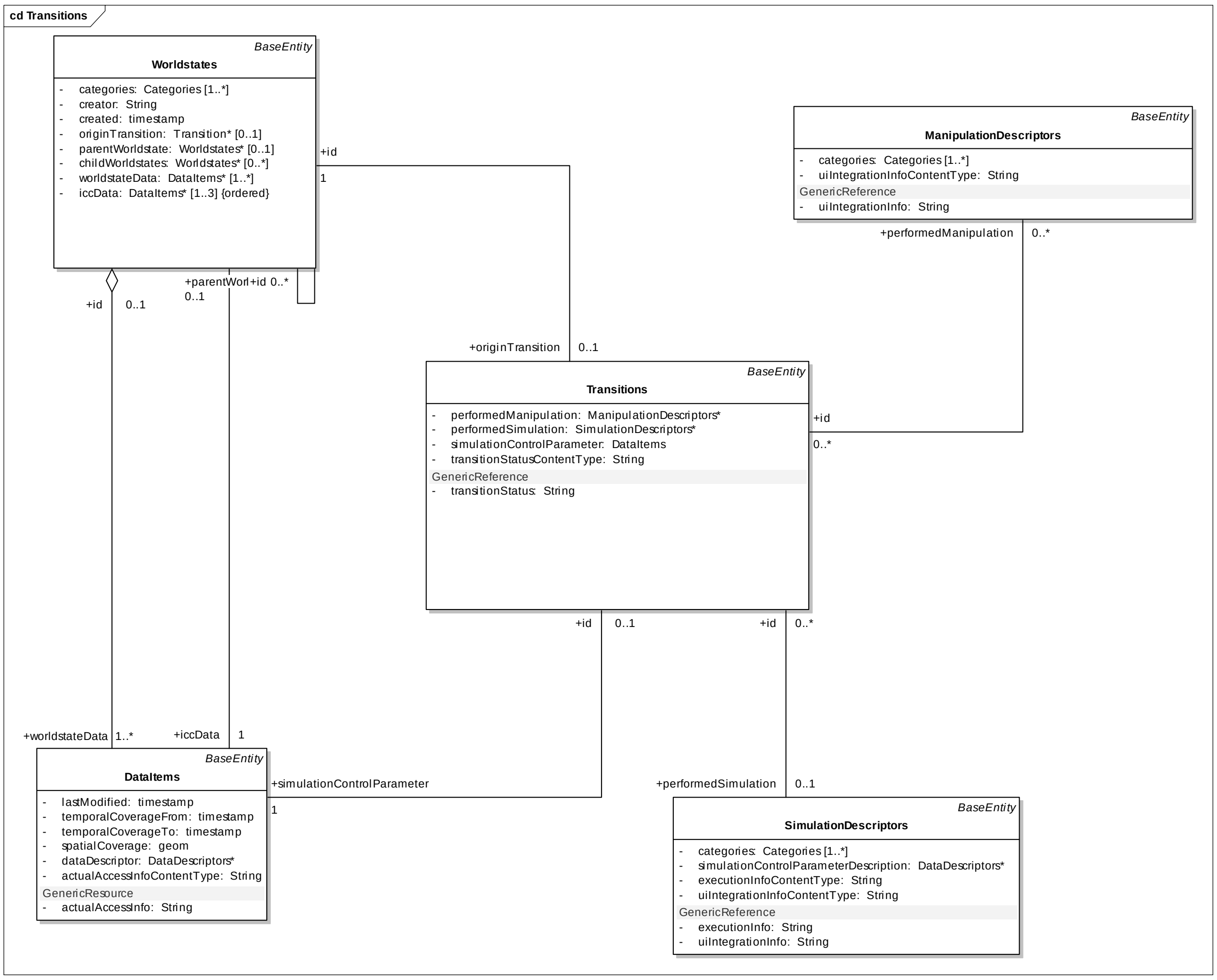

cd Transitions
BaseEntity
Worldstates
- categories: Categories [1..*]
- creator: String
- created: timestamp
- originTransition: Transition* [0..1]
- parentWorldstate: Worldstates* [0..1]
- childWorldstates: Worldstates* [0..*]
- worldstateData: DataItems* [1..*]
- iccData: DataItems* [1..3] {ordered}
+id
1
+parentWorl+id 0..*
0..1
+id
0..1
BaseEntity
ManipulationDescriptors
- categories: Categories [1..*]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String
+performedManipulation
0..*
+originTransition
0..1
BaseEntity
Transitions
- performedManipulation: ManipulationDescriptors*
- performedSimulation: SimulationDescriptors*
- simulationControlParameter: DataItems
- transitionStatusContentType: String
GenericReference
- transitionStatus: String
+id
0..*
+id
0..1
+id
0..*
+worldstateData
1..*
+iccData
1
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String
+simulationControlParameter
1
+performedSimulation
0..1
BaseEntity
SimulationDescriptors
- categories: Categories [1..*]
- simulationControlParameterDescription: DataDescriptors*
- executionInfoContentType: String
- uiIntegrationInfoContentType: String
GenericReference
- executionInfo: String
- uiIntegrationInfo: String

Figure 93: Transitions Class Diagram with Relationships.

WorldStates

This class represents a concrete World State. A snapshot of the World or World State consists of all data related to a specific crisis simulation experiment.

Table 14: Specification of World States Attributes.

Attribute	**Notes**
categories Categories **Private Collection [1..*]**	**Categories of the World State, e.g. to relate the World State to a specific scenario or a crisis management phase.**
creator String **Private**	**User or process that created the World State.**
created timestamp **Private**	**Creation Timestamp of the World State.**
originTransition Transition **Private [0..1]**	**Reference to the transition that created the World State, e.g. a manipulation or a simulation.**
parentWorld State World States **Private [0..1]**	**Reference to the parent World State in the World States tree.**
childWorld States World States **Private Collection [0..*]**	**Reference to all child World States of this World State.**
World StateData DataItems **Private Collection [1..*]**	**An array of data items that represent the actual World State data.**
iccData DataItems **Private [1..3]**	**A three-dimensional array of data items that represents the Indicator, Criteria and Cost data that has been produced by a Indicator Function (in case of Indicatior data) or other services (in case of Cost and Criteria data).**

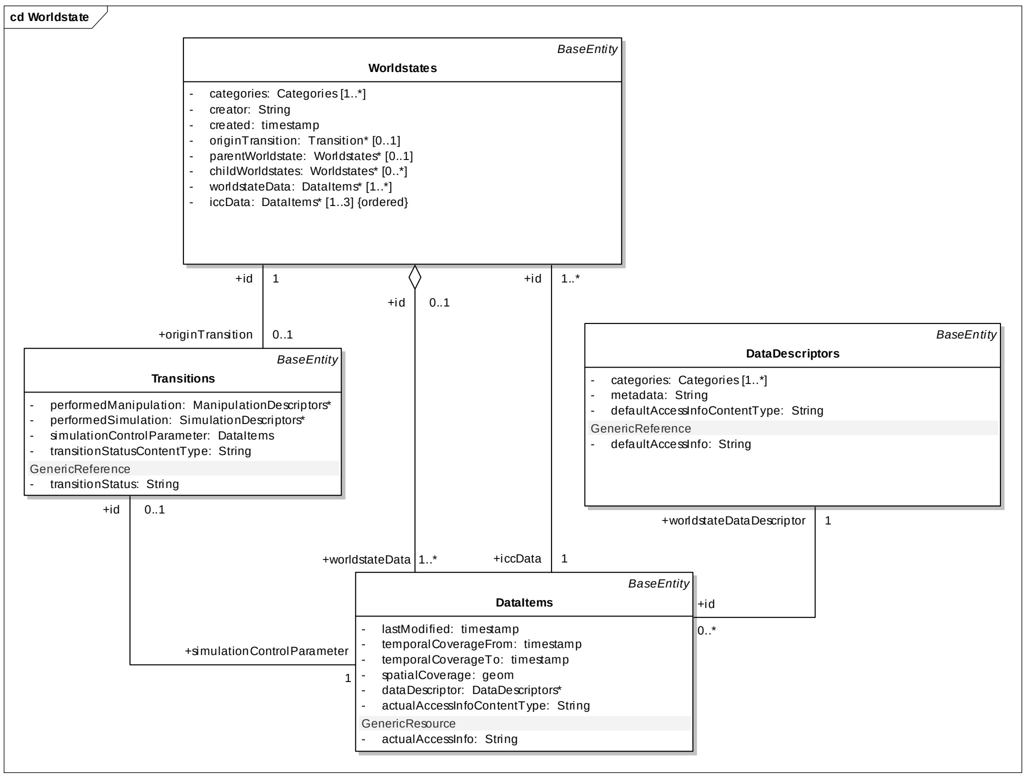

cd Worldstate
BaseEntity
Worldstates
- categories: Categories [1..*]
- creator: String
- created: timestamp
- originTransition: Transition* [0..1]
- parentWorldstate: Worldstates* [0..1]
- childWorldstates: Worldstates* [0..*]
- worldstateData: DataItems* [1..*]
- iccData: DataItems* [1..3] {ordered}
+id
1
+id
0..1
+id
1..*
+originTransition
0..1
BaseEntity
Transitions
- performedManipulation: ManipulationDescriptors*
- performedSimulation: SimulationDescriptors*
- simulationControlParameter: DataItems
- transitionStatusContentType: String
GenericReference
- transitionStatus: String
+id
0..1
BaseEntity
DataDescriptors
- categories: Categories [1..*]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
+worldstateDataDescriptor
1
+worldstateData
1..*
+iccData
1
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String
+id
0..*
+simulationControlParameter
1

Figure 93: World States Class Diagram with Relationships.

APPENDIX (B) Application architecture template

This annex contains the Application Architecture Template of the CRISMA Framework which has been introduced in section 5.3.

The Application Architecture Template is used to describe one particular Crisis Management Scenario. Thereby, World State Data Slots, ICC Elements, Simulation Models and Transition Points are at first specified independently of a particular Simulation Case. This allows not only to reuse the same Transition Points throughout different Simulation Cases but also to consolidate Transition Points from different Pilot Use Cases into reusable Building Blocks of the CRISMA Framework.

The relationship between the different elements of the template which are in most cases represented by tables that have to be filled in by the architect of the CRISMA Application are illustrated in Figure 98.

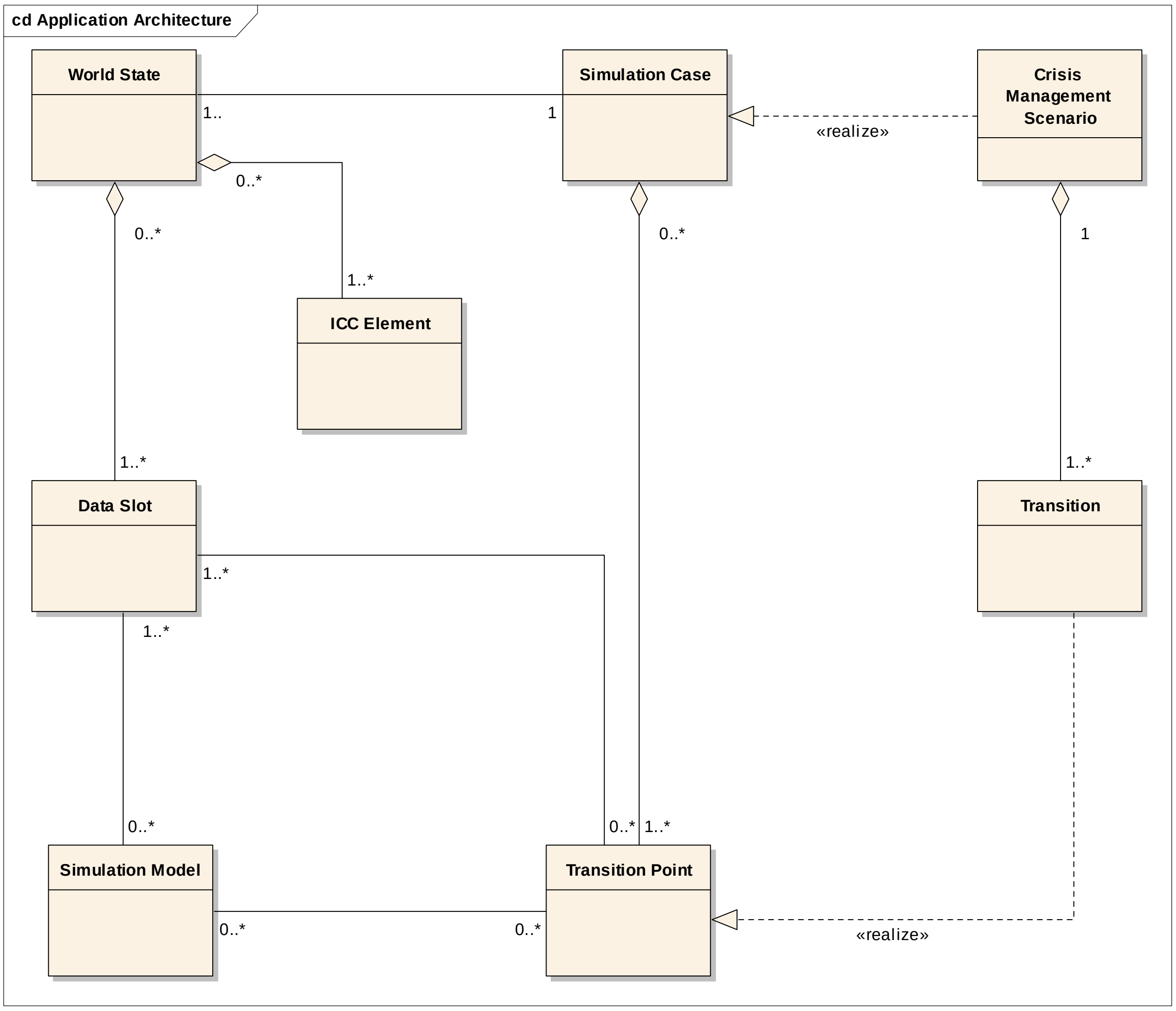


Figure 98: Relationship between Elements of the Template.

The information entered in this template will directly be used to populate the application-specific Control and Communication Information Model (CCIM) which is an essential cornerstone of each CRISMA Application.

Some of the fields of the tables expect the information to be entered in JSON (Java Script Object Notation) format. The reason is that JSON is the default encoding format of the Core CCIM and expected and served by the ICMM. Please note that this technical information is optional as it is not needed for the description of Reference Scenarios. But it is nevertheless required for the actual implementation.

To reduce the effort for providing a first draft specification of the Application Architecture that is both useful for WP25 and WP35, some fields of the tables are optional for a certain purpose of use of the template. Fields with green background contain information that can simply be copied & pasted from other documents. Although this information has eventually to be entered into a CCIM by a developer / integrator of the CRISMA Application, it is not required but it is nevertheless recommended to provide this information. Fields with orange background contain information that is not relevant for the specification of Reference Scenarios (WP25). This mostly technical information is nevertheless highly relevant for the integration of the CIRMSA Framework (WP35) and in most cases required for the development of the CRISMA Application.

Initialisation and configuration information

Initialisation and configuration information has to be collected once for a particular CRISMA Application. It consists of a specification of all Data Slots, Simulation Models and ICC Elements that are available for the Simulation Cases of the Pilot Use Case as well as all possible Transition Points themselves. Since those specifications must also contain information on the relationships between Data Slots (blue), Simulation Models (red) and ICC Elements (green), a diagram as shown in Figure 99 should be provided. Please note, that the diagram in Figure 99 shows a preliminary example of the situation at the Italian Pilot.

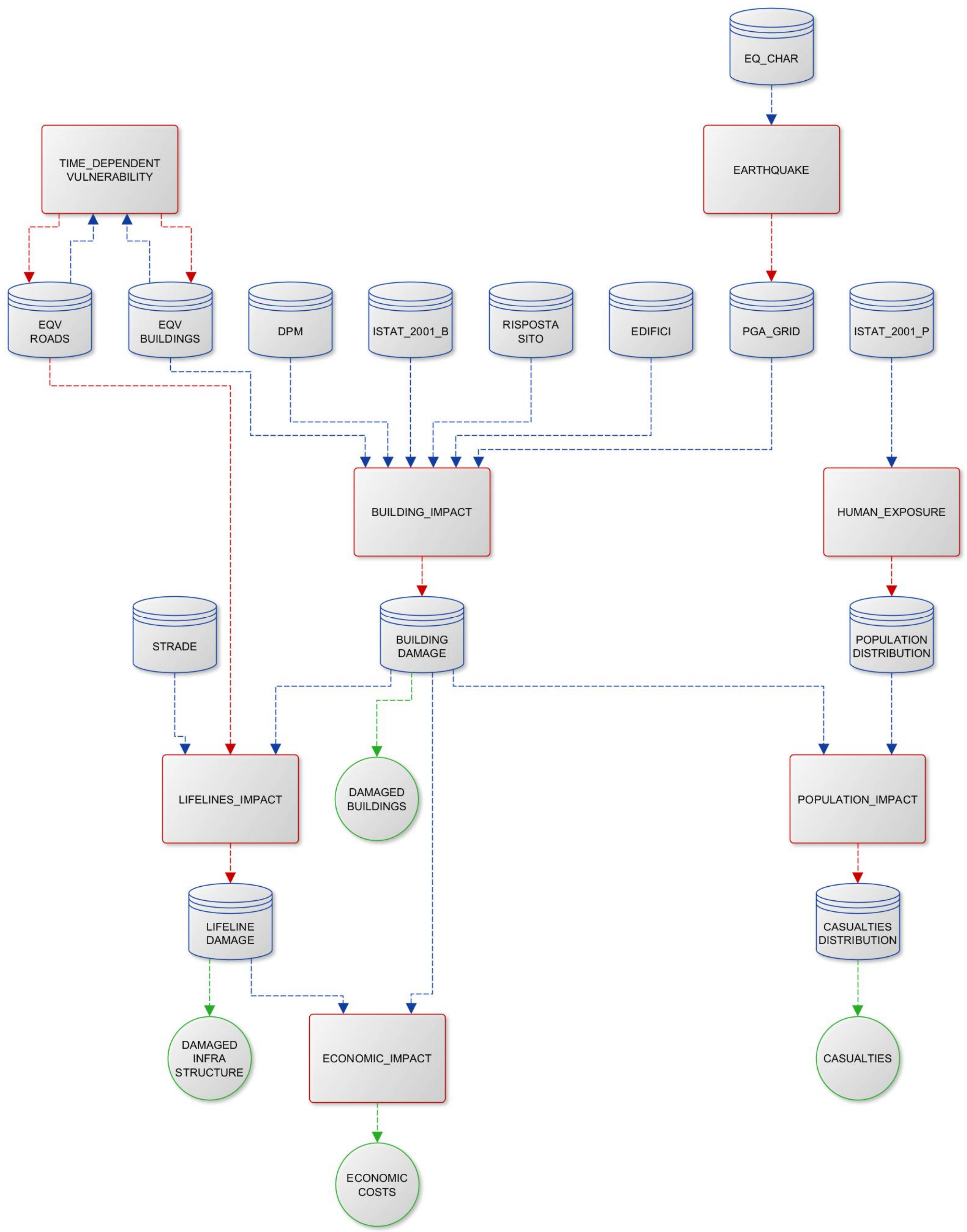


Figure 99: Relationship between Data Slots, Simulation Models and ICC Elements (Example).

World state data slot descriptions

In this section all World State Data Slots of the whole Crisis Management Use Case (i.e. Pilot Use Case) are described in accordance to the Conceptual Business Logic and the Core CCIM. The specification of World State Data Slots is separated from the specification of Simulation Cases since a Pilot Use Case usually consists of different Simulation Cases and not all World State Data Slots may be relevant for each Simulation Case. Thus it is possible to define individual World States for each Simulation Case without the need to repeat the specification of Data Slots. In particular, the same is also true for Transition Points which are also specified independently of Simulation Cases.

The information about the Data Slots can be taken from the SP5 deliverables D51.1 "Site Status" (D51.1, 2013) and D51.2 "Pilot User Guide" (D51.2, 2013), respectively.

The goal of this section of the Application Architecture Template is to prepare and enhance this information in such a way that it can be used for the description of Reference Scenarios, the integration of the CIRSMA Framework and the development of CRISMA Applications.

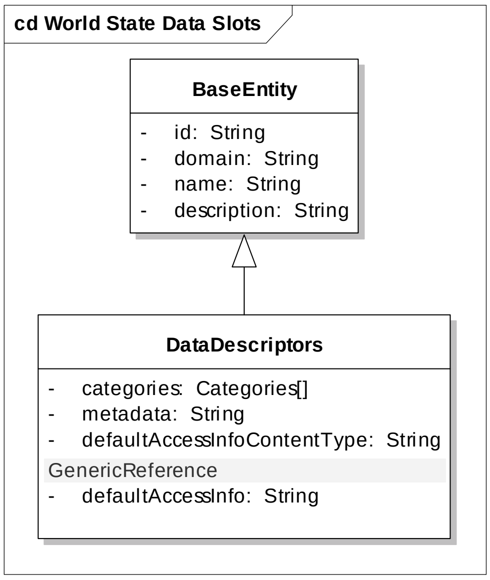


Figure 100: Representation of World State Data Slot Descriptions in the Core CCIM.

The information entered in the tables below map directly to the respective classes of the Core CCIM shown in Figure 100. This means, by collecting the information on Data Slots in the format suggested by the template already a large part of the essential CCIM of the CRISMA Application is provided. It is important to note and essential for the understanding of the World States concepts that a World State Data Slot Description does not represent an actual Data Slot (instance) but just a description of this Data Slot (class). Accordingly, the Core CCIM defines a distinct class to represent instances of Data Slots, Data Items (Figure 101).

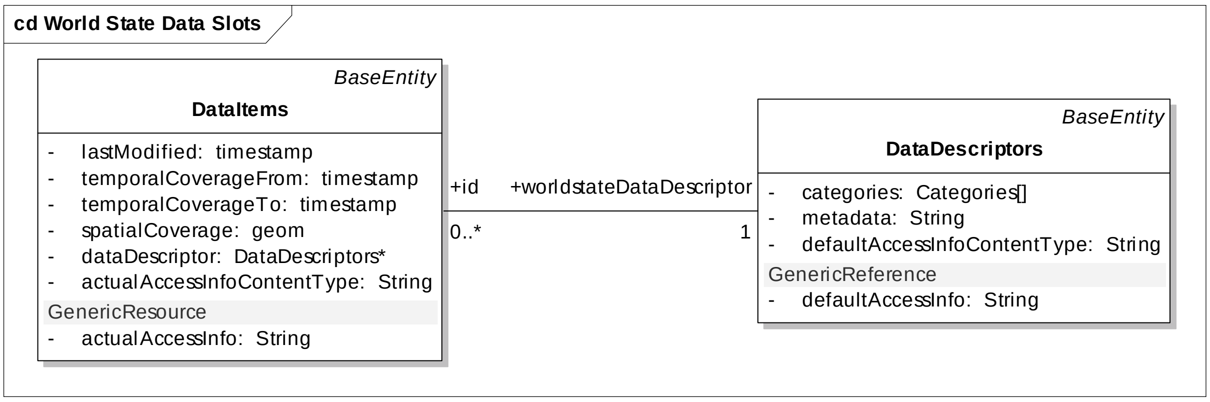

cd World State Data Slots
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String
+id
0..*
+worldstateDataDescriptor
1
BaseEntity
DataDescriptors
- categories: Categories[]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String

Figure 101: Representation of World State Data Slots in the Core CCIM.

For each Data Slot, one of the following tables has to be filled in. If the information which has been collected in Appendix I – IV of D51.2 on the different data to be used in the respective Pilot Use Case is still complete and up to data, table rows with a green background **may** be left blank. Nevertheless, it is highly recommended to copy those fields into the table since it makes the Application Architecture specification a self contained document and supports developers that have to enter this information into the CCIM.

It should also be noted that a Data Slot is a mean for organising and categorising World State Data rather than an accurate representation of a (physical) data source. Therefore one data source or data access service may cover multiple Data Slots. E.g. the same GeoServer (Data Integration Building Block) instance may serve both shake maps and population distribution maps which are modelled as two distinct Data Slots of the World State. In a first attempt to indentify and describe all Data Slots relevant for the World State(s) of a Pilot Use Case the information on pilot data from D51.2 and the Pilot World State Description from D51.1 can be taken into consideration.

For better readability, the fields of the World State Data Slot Descriptions table are explained in the bullet list below while the table itself contains some representative examples from the Italian Pilot.

- **Data Slot #**
  This field contains the consecutive number and a unique id (e.g. name) of the Data Slot.
- Title
  This field contains the title of the Data Slot as it may appear in a CRISMA GUI, e.g. for selecting a map layer, specifying model input data, etc.
- Description
  Provides human readable description of the Data Slot as it may appear in a CRISMA GUI, e.g. as descriptive information. This field may be left blank if a reference to the metadata tables of D51.2 can be given.
- **Categories**
  The categories of the Data Slot. They may be used as labels or tags in a CRISMA GUI or to allow discovery of specific types of Data Slots. Mandatory category types (classifications) and example of possible values are:
  - **Data Slot Category:**
    e.g. “World Description”, “Assets”, “Expert Data Sets”, “Hazard Outputs”, “Impacted Objects”, “Vulnerability Classes Distribution”, “Impact and Losses”, ...

- **Data Category:**
  e.g. “Exposure”, ”Inventory Database”, “Hazard”, “Vulnerability”, “Assets”, “World Description”, “Expert Data”, ….
- **IO Category:**
  e.g. “Model Input”, “Model Output”, “Analysis Input”, “Indicator Input”, ...
- **DataDescriptor Category:**
  always fixed: “World State Data Slot“

- **Metadata**
  Contains detailed metadata describing the Data Slot. If available, INSPIRE metadata encoded in **JSON format** should be provided as this is the default encoding format of the CCIM. For this purpose, the example JSON document presented in the table can be used. Thereby, “property” represents the name of the metadata element and “value” its respective value. In the JSON document presented in the table “value” is populated with the description of the metadata element and has of course to be replaced. Although this field may be left blank if a reference to the metadata tables of D51.2 can be given it must be noted that a transformation of the tabular description to JSON format is recommended when it comes to the implementation of the CRISMA Application and the metadata has to be put into the CCIM of the Application.
- Default Access Information
  This is technical information encoded in JSON format on how to access the actual data source of this Data Slot. Please note that this information does not necessarily indentify the actual data itself since the data may change after each World State Transition. Thus, this field contains only static information about the data source which stays the same for each Transition / change to the World State. As an example consider a database for storing OOIs. The Default Access Information would describe the data base connection (URL) and probably also tables where the individual OOIs are stored (e.g. “Ambulances”, “Hospitals”, etc.) but not ids (rows in the table) of individual OOIs.
  Since a wide variety of data source types and data access mechanisms may be used within one CRISMA Application is not possible to define one universal schema in the Core CCIM that is applicable to all potential data sources. Therefore, the format of Default Access Information is always application-specific and may possibly also be different between Data Slots. Nevertheless, this information is needed for application developers or integrators that need to know how to (technically) get access to the data.
  Furthermore, data of a CRISMA Application may have different representations and different access mechanisms for different purposes. While a WPS-wrapped Simulation Model (Simulation Model Integration Building Block) might need direct access to a database (e.g. a PostGIS database) or to the file system (e.g. Shape Files) to store its results, a client visualising those model results (GIS Widget Building Block) would probably access the same data via a WMS or WFS (Data Integration Building Block). Therefore, different Building Blocks might need different types of access information.
  Please note that this data access information is not relevant for the Reference Scenarios (WP25), it might however be very helpful for Integration of the CRISMA Framework and is required for the development of a CRISMA Application.
  This field **may** be left blank if the data provided by this data source is *not* relevant for World State Transitions, Indicator Functions or managing snapshots of the world (e.g. the data is immutable) and/or the data is not relevant for

development or integration of the CRISMA Application. This might for example be the case when interfaces for data exchange between the data source and Simulation Models are already implemented in the core of the Simulation Models and thus the CRISMA Application or any Building Block it is not required to directly access the data source.
A simple default schema which describes the data access possibilities on the level of Building Blocks or other CRISMA Federates is provided in the World State Data Slot Descriptions table and is furthermore explained in the following:

- **crismaFederateName**
  Name of the CRISMA Federate (Building Block) accessing the data source.
- **crismaFederateRole**
  Role of the CRISMA Federate accessing the data source, allowed values are client or server. A Data Integration Building Block (e.g. GeoServer) might for example act as a server of a non-OGC compliant data source and expose it as WMS, WFS, etc. A GIS Widget Building Block (e.g. OpenLayers) might act as a client of same data source accessing it through the WMS interface of the Data Integration Building Block.
- **accessURI**
  URI of the data source. The actual URI schema depends on the data source type and the role of the Building Block accessing the data source. Referring to the previous example of Data Integration and GIS Widget Building Block, the URI may either point to the file system (directory where Shape files are stored) or to a getMap or getCapabitities request of a WMS. Since the URI identifies only the data source and not the actual data it may contain wildcards that can be substituted e.g. with a concrete filename, a WMS layer name, a bounding box, the IDs of OOIs, etc.
- **accessMethod**
  A data source may support different methods for reading or updating data, for obtaining metadata, etc. The field access method is used to further describe data source specific access methods that can be used by the respective Building Block. Examples for such access methods are getMap and GetCapabilities requests of WMS data sources.

- **Comments**
  Comments must give information on the availability and status of a Data Slot, e.g. whether the data is already available at the pilot site, whether it has already been integrated with a Data Access Building Block, etc. Comments should furthermore provide additional information for developers or integrators, e.g. whether and how Simulation Models have to be coupled with a data sources, etc.

Table 15: World State Data Slot Overview.

ID	Name	Title
1	Example: “POWER_TOWER”	Example:”Electricity distribution power lines towers”
2	...	...
3	....	...
...	...	...

Table 16: World State Data Slot Description Table.

Data Slot #	Example: “POWER_TOWER”
Title	Example: “Electricity distribution power lines towers”
Description	Example: “Points elements that georeference the position of power lines towers.”
Categories	Example: “Data Slot Category: World Description; Data Category: Inventory Database; IO Category: Analysis Input, Model Input; DataDescriptor Category: World State Data Slot
Metadata	This is the structure of the JSON document for describing metadata: `{ "Resource title": "This a characteristic, and often unique, name by which the resource is known.", "Resource abstract": "This is a brief narrative summary of the content of the resource.", "Resource type": "This is the type of resource being described by the metadata.", "Resource locator": "The resource locator defines the link(s) to the resource and/or the link to additional information about the resource.", "Unique resource identifier": "A value uniquely identifying the resource.", "Format of data": null, "Software to read the data format": null, "Topic category": "The topic category is a high-level classification scheme to assist in the grouping and topic-based search of available spatial data resources.", "Keyword value": "The keyword value is a commonly used word, formalised word or phrase used to describe the subject. While the topic category is too coarse for detailed queries, keywords help narrowing a full text search and they allow for structured keyword search.", "Related risk domain ": null, "Geographic bounding box": "This is the extent of the resource in the geographic space, given as a bounding box.", "Temporal extent": "The temporal extent defines the time period covered by the content of the resource.", "Date of publication": "This is the date of publication of the resource when available, or the date of entry into force. There may be more than one date of publication.", "Date of last revision": "This is the date of last revision of the resource, if the resource has been revised. There shall not be more than one date of last revision.", "Date of creation": "This is the date of creation of the resource. There shall not be more than one date of creation.", "Lineage": "This is a statement on process history and/or overall quality of the spatial data set. Where appropriate it may include a statement whether the data set has been validated or quality assured, whether it is the official version (if multiple versions exist), and whether it has legal validity.", "Spatial resolution": "Spatial resolution refers to the level of detail of the data set. It shall be expressed as a set of zero to many resolution distances (typically for gridded data and imagery-derived products) or equivalent scales (typically for maps or map-derived products).", "Conditions applying to access and use": "This metadata element defines the conditions for access and use of spatial data sets and services, and where applicable, corresponding fees as required by Article 5(2)(b) and Article 11(2)(f) of Directive 2007/2/EC.", "Limitations on public access": "When Member States limit public access`

	`to spatial data sets and spatial data services under Article 13 of Directive 2007/2/EC, this metadata element shall provide information on the limitations and the reasons for them."`
	`}`
	
	Next comes an example of a JSON document describing metadata of a data source of Pilot D. This information was taken from respective metadata table in D51.2 and has been converted into JSON.
	
	`{`
	`    "Resource title": "Administrative Region Limits",`
	`    "Resource abstract": "Administrative limits of Abruzzo Region",`
	`    "Resource type": "Spatial data set (dataset)",`
	`    "Resource locator": "https://workspace.vtt.fi/sites/eu_crisma/Shared Documents/Sub-Projects/SP5/WP55/04_PilotD_doc",`
	`    "Unique resource identifier": "aquila.regione",`
	`    "Format of data": "PostGIS geometry table",`
	`    "Software to read the data format": "PostgreSQL, ArcGIS, QGIS, …",`
	`    "Topic category": "3. Boundaries (boundaries)",`
	`    "Keyword value": "geography",`
	`    "Related risk domain ": "Geophysical hazard",`
	`    "Geographic bounding box": "4 values ",`
	`    "Temporal extent": "2 date YY/MM/DD or 2 dates ( time interval) ",`
	`    "Date of publication": "date YY/MM/DD",`
	`    "Date of last revision": "date YY/MM/DD",`
	`    "Date of creation": "date YY/MM/DD",`
	`    "Lineage": "Dummy data sets created for demonstration purposes of CRISMA project",`
	`    "Spatial resolution": null,`
	`    "Conditions applying to access and use": "no conditions apply",`
	`    "Limitations on public access": "PU (public)"`
	`}`

Default Access Information	This is the structure of the JSON document for describing default (static) access information to the data source:
	`defaultAccessInformation =`
	`[{`
	`    "crismaFederateName": "Name of the CRISMA Federate (Building Block) accessing the data source."`
	`    "crismaFederateRole": "Role of the CRISMA Federate, allowed values are client or service.",`
	` "defaultAccessURI": "URI of the data source. The actual URI schema depends on the data source type and the role of the Building Block accessing the data source. Referring to the previous example of Data Integration and GIS Widget Building Block, the URI may either point to the file system (directory where Shape files are stored) or to a getMap or getCapabitities request of WMS. Since the URI identifies only the data source and to the actual data it may contain wildcards that can e.g. be substituted with a concrete filename, WMS layer name, bounding box, IDs of OOIs, etc.",`
	` "accessMethod": "A data source may support different methods for reading or updating data, for obtaining metadata, etc. The field access method can uses to further describe data source specific access methods that can be used by the respective Building Block. Examples for such access methods are getMap and GetCapabilities requests of WMS data sources."`
	`}]`
	
	Example of a JSON document for describing default access information of a data source of Pilot D:

	defaultAccessInformation =
	[{
	    "crismaFederateName": "GIS Widget Building Block (cismap)"
	    "crismaFederateRole": "client",
	    "defaultAccessURI":
	"http://127.0.0.1/geoserver/ows?service=wms&version=1.1.1&request=GetCapabilities",
	    "accessMethod": "getCapabilities"
	},
	{
	    "crismaFederateName": "GIS Widget Building Block (cismap)"
	    "crismaFederateRole": "client",
	    "defaultAccessURI":
	"http://143.225.105.70/geoserver/crismatest/wms?service=WMS&version=1.1.0&request=GetMap&layers=crismatest:regione&styles=&bbox=<boundingBox>&width=<width>&height=<height>&srs=EPSG:32633&format=image/png&transparent=true",
	    "accessMethod": "getMap"
	}]
	
	Note: The part of the getMap URI in red shows a possible usage of wildcards.

Comments	Example: “The data of this Data Slot is available at PLINUS centre and is already served via OGC WMS, not further integration with Data Integration Building Block is needed. The data of this Data Slot is static; it doesn’t change during the analysis or Transition (IO Category: Immutable).”

Instructions: Copy and/or fill in the overview and details tables above for each distinct Data Slot

ICC descriptions

Like World State Data Slot Descriptions, IC (Indicators and Criteria) Descriptions are also defined once for the complete Pilot Use Case. The specification of an Indicator Element is a bit different than the specification of Data Slot, since the IC Element describes in general a simple numerical value rather than a complete data source. Indicators are currently defined in D25.1 “Updated technology inventory, requirements, scenarios, criteria and key performance indicators” (D25.1, 2014).

The goal of the technical specification of IC Elements provided in the tables below is to allow a developer of the Indicators Building Block to implement the respective IC Functions and store the results of the functions in the CCIM. For this purpose, both the algorithm to derive Indicators from the World State as well as the structure of the IC data has to be provided.

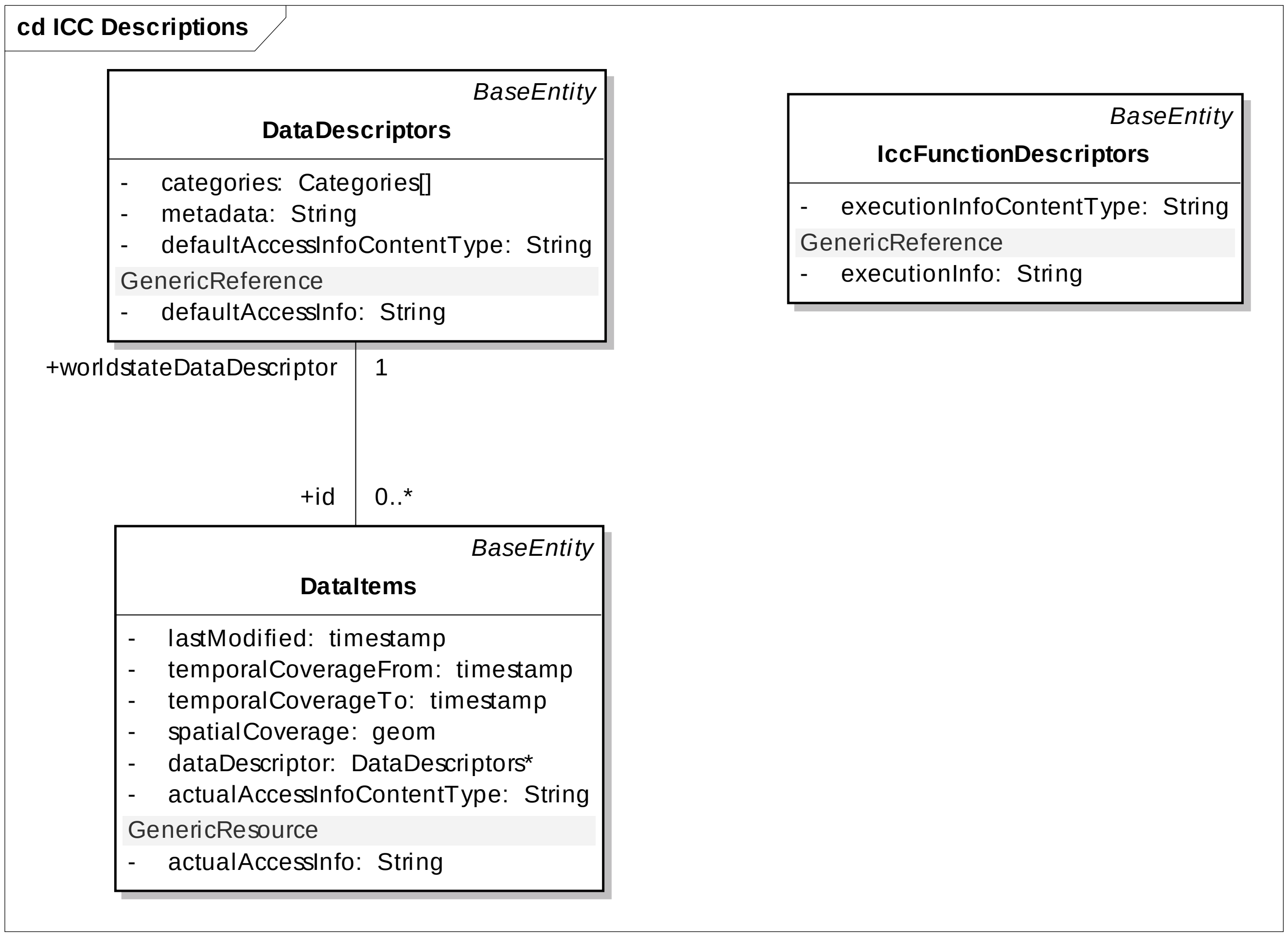

cd ICC Descriptions
BaseEntity
DataDescriptors
- categories: Categories[]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
BaseEntity
IccFunctionDescriptors
- executionInfoContentType: String
GenericReference
- executionInfo: String
+worldstateDataDescriptor
1
+id
0..*
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String

Figure 102: Representation of ICC Descriptions in the Core CCIM.

As shown in Figure 102 in the Core CCIM, the Elements of the ICC Vector of a World State are also represented by the DataDescriptors class and a concrete ICC Vector instance by an array of DataItems classes in the World States class. Thereby, the attribute *defaultAccessInfo* of the DataDescriptors Class maps to the field schema specification and the attribute *executionInfo* to the field ICC Function in Table 18.

For better readability, the fields of the World State Data Slot Descriptions table are explained in the bullet list below while Table 18 contains some representative examples from Pilot D.

- **ICC Element #**
  This field contains the consecutive number and a unique id (e.g. name) of the ICC Element.
- Title
  This field contains the title of the ICC Element as it may appear in a CRISMA GUI. It may be left blank if a reference to the Indicator tables of D24.1 can be given.
- Description
  Provides a human readable description of the ICC Element as it may appear in a CRISMA GUI, e.g. as descriptive information.
- Categories
  The categories of the ICC Element. They may be used as labels or tags in a CRISMA GUI or to allow discovery of specific types of ICC Elements. In the

CCIM mandatory category types (classifications) and example of possible values are:

- o **ICC Category:**
  “Unsafe Viability”, “Evacuate Population”, “Sheltering Options”, “Physical Damage”, “Mitigation Measures”, “Impact and Losses”, ...
- o **IO Category:**
  always fixed for ICC Element: “ICC Output”;
- o **DataDescriptor Category:**
  always fixed for ICC Element: “ICC Element“;

- **Data Slots**
  Comma separated names of the Data Slots that are used as input for the calculation of this ICC Element.
- **ICC Function Algorithm / ICC Function Description**
  This filed should provide a specification of the actual algorithm (ICC Function) to derive the indicator values from the World State Data Slot(s). However, it must be admitted that since there are so many disparate data sources and even more possibilities to calculate related indicators, it is not possible to define one universally valid method or format for specifying executable ICC Functions. Therefore, this field may only provide some hints for the developers of the Indicators Building Blocks rather than an implementable or executable algorithm. (PL/SQL function, JavaScript function, etc.)
  Please note this information is not relevant for the Reference Scenarios (WP25), it is however be very helpful for Integration of the CRISMA Framework (especially for the Indicators Building Block) and required for the development of a CRISMA Application.
- **Default Access Information / Schema Information**
  Like in Default Access Information of the Data Slot Descriptors this field specifies also how the data source that stores the ICC data can be accessed. However, ICC data does not necessarily have to be stored in a dedicated data source. In general, it is represented by a single (e.g. numeric) value or a simple object consisting of several values. Therefore it can directly be put into the CCIM and thus be stored by the ICMM. If this is the case, the field Default Access Information is used differently as in the specification Data Slot Descriptor. Instead of specifying an URI of the ICC data source (which would only point to the ICMM itself), the field shall contain the data type definition provided in **JSON Schema**[11] or alternatively an example JSON Document. JSON Schema is preferred over a JSON example document as it provides also data type information, value ranges, etc.
  Please note that a formal schema specification might not be relevant for the Reference Scenarios (WP25), it is however very helpful for the integration of the CRISMA Framework (WP35) and required for the development of a CRISMA Application. Instead of a formal specification in JSON Schema, for the preparation of Reference Scenarios a simple list of properties and their types would be sufficient.
- **Comments**
  Optional comments that provide e.g. additional information for developers or integrators, e.g. whether the ICC Functions have already been implemented on the level of the data source (as PL/SQL Function), etc.

[11] http://json-schema.org

Table 17: ICC Elements Overview.

ID	Name	Title
1	Example: “DAMAGED_BUILDINGS”	Example: “Damaged Buildings Indicators”
2	...	...
3	....	...
...	...	...

Table 18: ICC Element Description Table.

ICC Element #	Example: “DAMAGED_BUILDINGS”
Title	Example: “Damaged Buildings Indicators”
Description	Example: “This ICC Element covers two Indicators: Number of collapsed buildings and unsafe buildings. See D24.1, Table 12 for more details.”

Categories	Example:
	“ICC Category: Impact and Losses, Impact on Buildings;
	IO Category: ICC Output;
	DataDescriptor Category: ICC Element;”

Data Slots	Example: “BUILDING_DAMAGE”

Default Access Information / Schema Information	Example:
	“This simple ICC Element can be stored directly in the ICMM (as part of the DataDescriptor Element of the CCIM). The JSON Schema of the data to be stored is:
	
	`{`
	`   "properties" : {`
	`      "lostBuildings" : {`
	`         "description" : "Number of collapsed buildings",`
	`         "minimum" : 0,`
	`         "type" : "integer"`
	`      },`
	`      "unsafeBuildings" : {`
	`         "description" : "Number of unsafe buildings ",`
	`         "minimum" : 0,`
	`         "type" : "integer"`
	`      }`
	`   },`
	`   "required" : [ "lostBuildings", "unsafeBuildings" ],`
	`   "title" : "Building Damage Indicators",`
	`   "type" : "object"`
	`} “`
	
	An example JSON Document is (as part of the DataItems Element of the CCIM) stored in the ICMM is:
	`{`
	`   "lostBuildings" :4711,`
	`   "unsafeBuildings" :31337`
	`}`

ICC Function	Example: “lostBuildings: Sum of all damaged buildings with damage class = D in BUILDING_DAMAGE database. unsafeBuildings: Sum of all damaged buildings with damage class < D in BUILDING_DAMAGE database.”
Comments	Example: “Implementation of the ICC Function by Indicators Building Block is not necessary since the algorithm is implemented directly in BUILDING_DAMAGE PostGIS database as PL/SQL Function. Storing of the calculated Indicator values in the ICMM has yet to be implemented.”

Instructions: Copy and/or fill in the overview and details tables above for each distinct ICC Element.

Please note that according to the specification of the Conceptual Business Logic the calculation of the current values of the ICC Elements is initiated after each Transition. Furthermore, it is up to the service that implements the ICC algorithm (e.g. Indicators Building Block) to retrieve and process the current World State Data and to store the calculated Indicator value in the ICMM.

Simulation model descriptions

As third part of the common specification of the Pilot Use Case, a technical description of the Simulation Models has to be provided. While such a specification could be arbitrary complex, only the most import information required for the specification of Transition Points is collected by the template. Those are:

- **Model #**
  This field contains the consecutive number and a unique id (e.g. name) of the Simulation Model.
- Title
  This field contains the title of the Simulation Model as it may appear in a CRISMA GUI.
- Description
  Provides human readable description of the Simulation Model as it may appear in a CRISMA GUI, e.g. as descriptive information. This field may be left blank if a reference to a description or technical specification of the model can be provided (e.g. in the D4.x or D51.x deliverables or the CRISMA Catalogue).If already available, the WPS URL under which the model can be invoked (only relevant for WP35 and SP5) could also be provided.
- Categories
  The categories of the Simulation Model. They may be used as labels or tags in a CRISMA GUI or to allow discovery of specific types of Simulation Models. In the CCIM Mandatory category types (classifications) and example of possible values are:
    - **Model Category:** “Economic”, “Cascading”, “Impact”, “Behaviour”, “Vulnerability”, etc.
- **Input**
  Provides the names of World State Data Slots and other Simulation Models that serve as Input for this Model.
- Output
  Provides the names of World State Data Slots and other Simulation Models that serve as Input for this Model.

- Parameters
  Technical Specification of the Simulation Model control parameters that are not part of a specific Data Slot which serves as model input. Examples are "EQ Magnitude, "Epicentre Coordinates, etc." Similar to the Default Access Information / Schema Information field of the ICC Element Descriptions this field should contain a complete specification of the parameters in JSON Schema Format.
  Please note that a formal specification of Simulation Model Parameters might not be relevant for the Reference Scenarios (WP25), it is however be very helpful for the integration of the CRISMA Framework (WP35) and required for the development of a CRISMA Application.
- **Comments**
  Comments must give information on the availability and status of the Simulation Model, e.g. whether the model is already available at the pilot site, whether it has already been integrated with the Simulation Model Integration Building Block or whether it has yet to be developed and by whom. Comments should furthermore provide additional information for developers or integrators, e.g. how the model can be wrapped technically (e.g. need to invoke a Windows .exe.).
- Optional comments that provide e.g. additional information for developers or integrators, e.g. whether and how Simulation Models have to be coupled with data sources, etc.

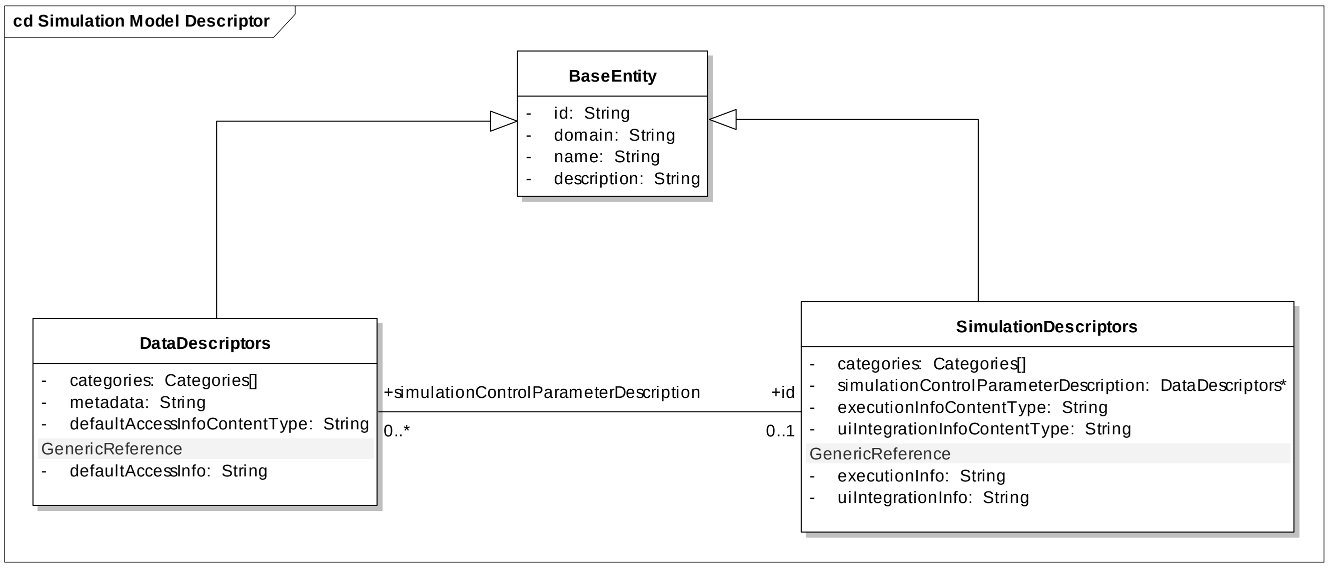

cd Simulation Model Descriptor
BaseEntity
- id: String
- domain: String
- name: String
- description: String
DataDescriptors
- categories: Categories[]
- metadata: String
- defaultAccessInfoContentType: String
GenericReference
- defaultAccessInfo: String
+simulationControlParameterDescription
0..*
+id
0..1
SimulationDescriptors
- categories: Categories[]
- simulationControlParameterDescription: DataDescriptors*
- executionInfoContentType: String
- uiIntegrationInfoContentType: String
GenericReference
- executionInfo: String
- uiIntegrationInfo: String

Figure 103: Representation of Simulation Model Descriptions in the Core CCIM.

Figure 103 shows how the information on Simulation Models collected in Table 20 is represented in Core CCIM of the CRISMA Framework. Currently, *executionInfo* (referring to the Simulation Model Integration Building Block) and *uiIntegrationInfo* (referring to the Simulation Model Interaction Building Block) are not collected by this template. This information is nevertheless required for the implementation of a CRISMA Application.

Table 19: Simulation Models Overview.

ID	Name	Title
1	HPDM	Earthquake Model
2	...	...
3	....	...
...	...	...

Table 20: Simulation Model Table.

Model #	**Example: “EARTHQUAKE_MODEL”**
Title	Example: “Earthquake Model” (Hybrid probabilistic-deterministic model)
Description	Expected number of collapsed building per class faced on the roads
Categories	Example: “Model Category: Hazard, Exposure;”
Input	Example: “n/a (see comment)”
Output	Example: “EQ_CHAR”
Parameters	Example: “The Simulation Model has three configurable parameters that have to be provided by the user through a GUI (Simulation Model Interaction Widget). The parameters are: EPICENTER COORDINATES, MAGNITUDE, DEPTH. The formal specification JSON Schema is: `{ "properties" : { "depth" : { "description" : "Depth of the Earthquake", "type" : "integer" }, "epicenter" : { "description" : "Coordinates of the Epicenter", "type" : "text" }, "magnitude" : { "description" : "Magnitude of the Earthquake", "type" : "float" } }, "required" : ["epicenter", "magnitude", "depth"], "title" : "Simulation Model Parameters", "type" : "object" }”`
Comments	Example: “The input of this Simulation Model (identification of fault geometry and the computation of the recurrence periods) is not part of the World State as it is not required for Analysis nor for World State Visualisation.”

Additional, the I/O flows of the Simulation Models (red) to and from Data Slots (blue) and their relation to ICC Elements (green) should be visualised as shown in the diagram in Figure 104.

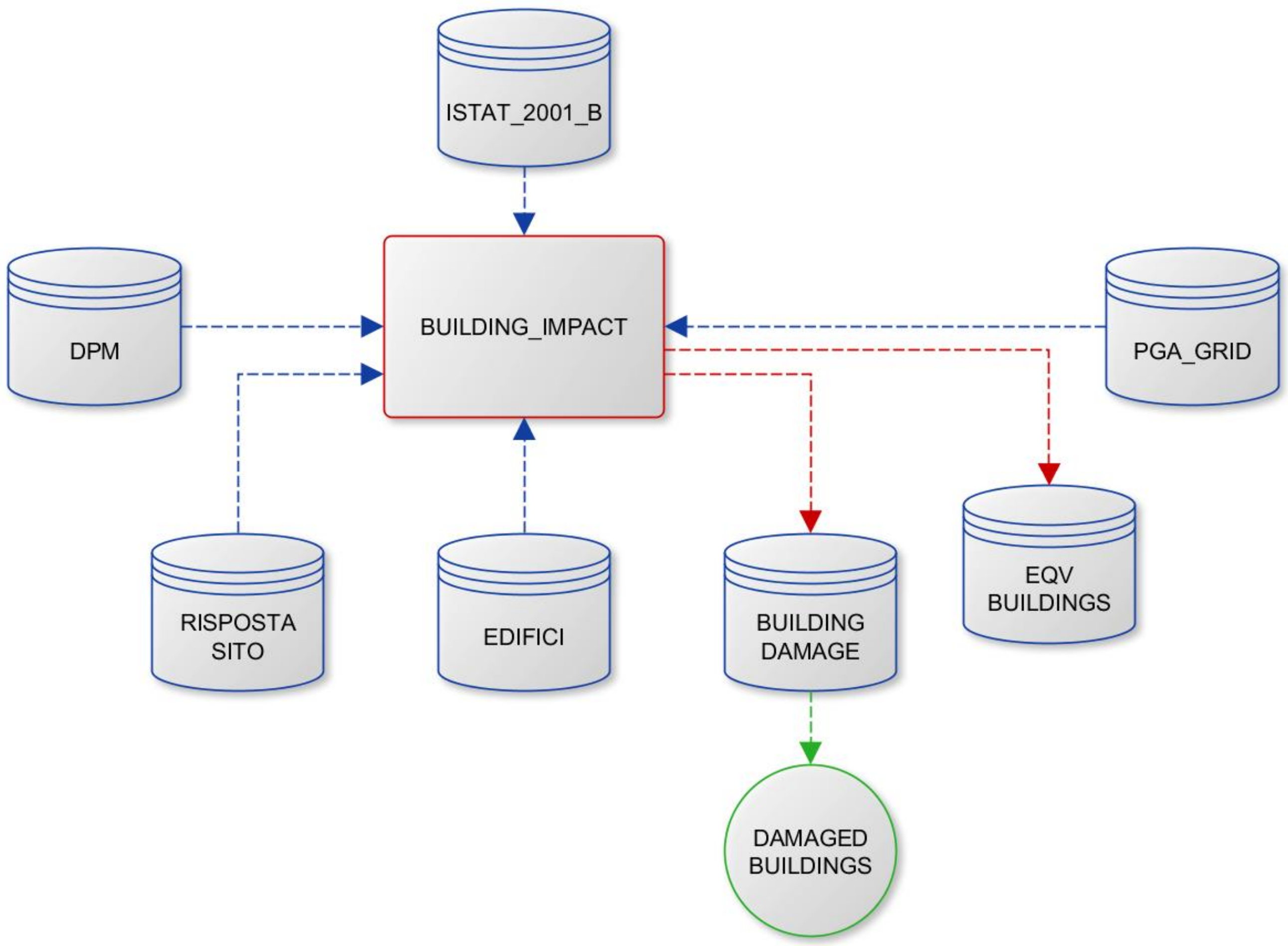


Figure 104: Simulation Model I/O Flows (Example).

Instructions: Copy and/or fill in the overview and details tables above for each distinct Simulation Model and provide diagrams of I/O flows.

Transition point descriptions

The last section of the initial specification describes all possible Transition Points available for the Simulation Cases and how they affect the World State. It specifies furthermore a Transition Point Graph that defines (coarsely) in which order Transitions can be executed.

As shown in Figure 105, in the Core CCIM, TransitionPoints aren't currently explicitly defined. Instead, the information model foresees *SimulationDescriptors* and *ManipulationDescriptors* which provide information on the possible manipulation or simulation performed within a Transition. Additionally, the Core CCIM defines a *Transition* class which represents a concrete Transition (instance of a Transition Point). Concrete Simulation Parameters are represented as *DataItem* of the *Transition* class. The schema of the Simulation Parameters is represented by the *DataDescriptor* class.

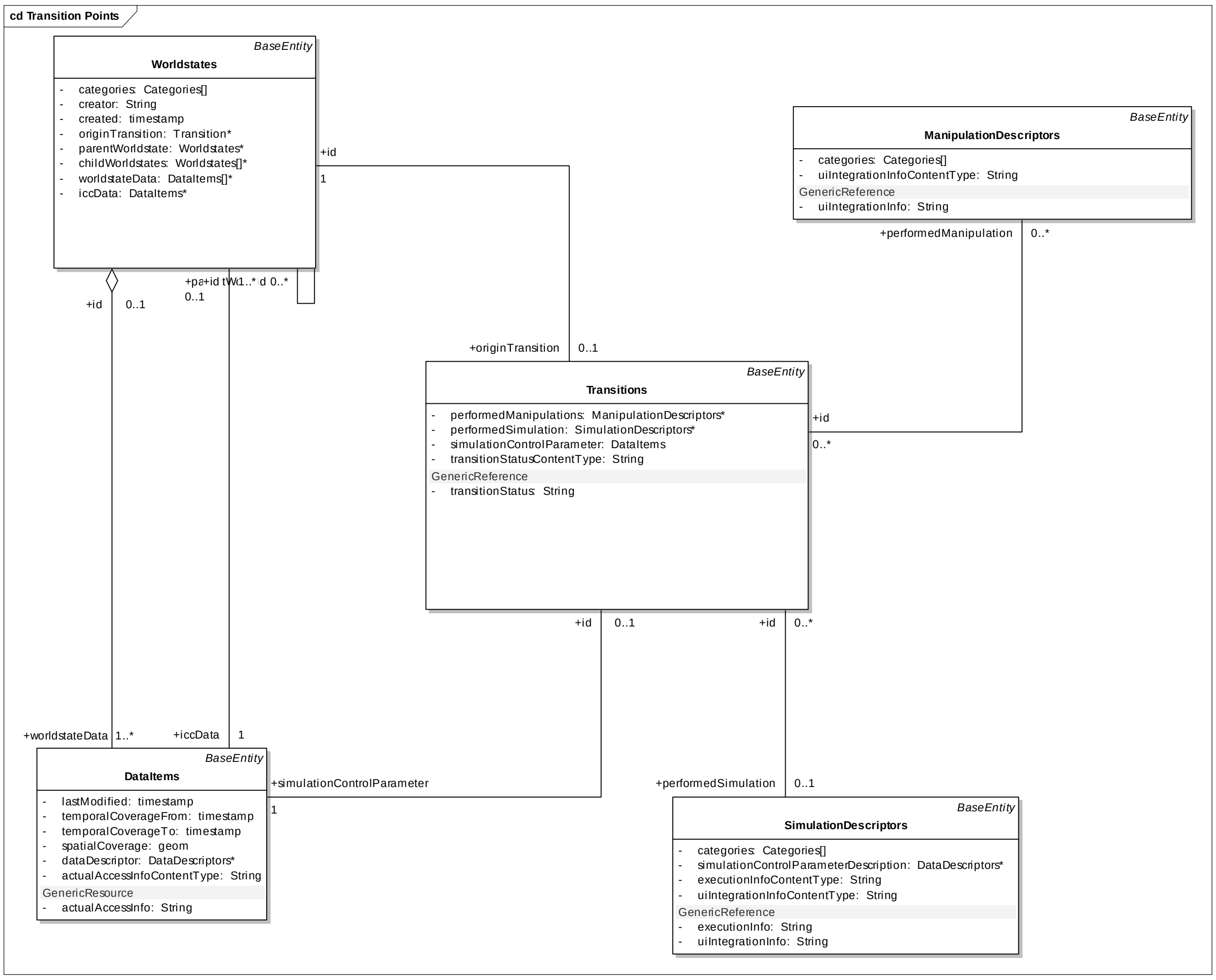

cd Transition Points
BaseEntity
Worldstates
- categories: Categories[]
- creator: String
- created: timestamp
- originTransition: Transition*
- parentWorldstate: Worldstate*
- childWorldstates: Worldstates[]*
- worldstateData: DataItems[]*
- iccData: DataItems*
+id
1
BaseEntity
ManipulationDescriptors
- categories: Categories[]
- uiIntegrationInfoContentType: String
GenericReference
- uiIntegrationInfo: String
+performedManipulation
0..*
+id
0..1
0..1
+originTransition
0..1
BaseEntity
Transitions
- performedManipulations: ManipulationDescriptors*
- performedSimulation: SimulationDescriptors*
- simulationControlParameter: DataItems
- transitionStatusContentType: String
GenericReference
- transitionStatus: String
+id
0..*
+id
0..1
+id
0..*
+worldstateData
1..*
+iccData
1
BaseEntity
DataItems
- lastModified: timestamp
- temporalCoverageFrom: timestamp
- temporalCoverageTo: timestamp
- spatialCoverage: geom
- dataDescriptor: DataDescriptors*
- actualAccessInfoContentType: String
GenericResource
- actualAccessInfo: String
+simulationControlParameter
1
+performedSimulation
0..1
BaseEntity
SimulationDescriptors
- categories: Categories[]
- simulationControlParameterDescription: DataDescriptors*
- executionInfoContentType: String
- uiIntegrationInfoContentType: String
GenericReference
- executionInfo: String
- uiIntegrationInfo: String

Figure 105: Representation of Transitions in the Core CCIM.

The decision which and how many Transition Points should be created depends on how the Crisis Management Scenarios should be analyzed. World States shall represent an actual snapshot of the world (as defined). They shall not represent continuous evolvement e.g. the movement of resources (calculated by some Simulation Model). This shall be handled by the corresponding Simulation Model if it is able (and wants) to visualise this (or store the data for later analysis). However, in such a case it will e.g. represent an individual Data Slot within a result World State and not a series of World States.

The World States are directly connected to the analysis. This means that there shall be a snapshot of the world whenever it is feasible for the actual crisis analysis or when an important decision or event shall be recorded.

<u>**Instructions**</u>**:** Create a Transition Point Graph of the Simulation Case.

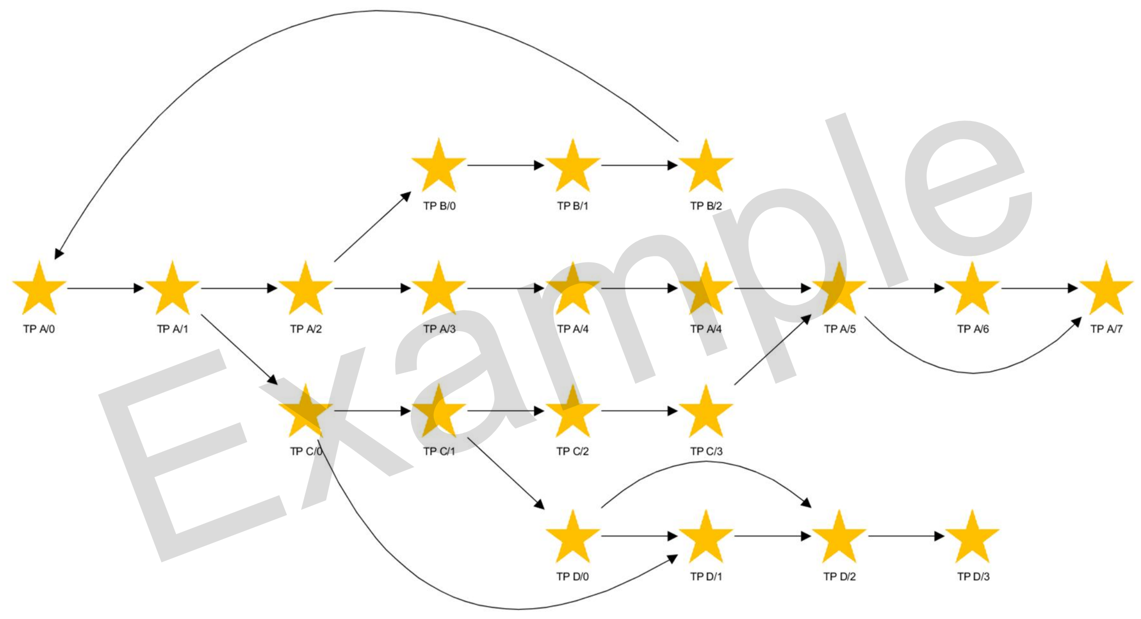


Figure 106: Transition Point Graph (Example).

The Transition Point Graph as well as the description of predecessor and successor Transitions specified in the Transition Point Tables should give only a coarse overview on the meaningful order of Transitions. It does not resent a formal specification that can be mapped 1:1 to the implementation of the CRISMA Application. However, together with the information on crisis evolution (relation to timeline) in the description field of the Transition Point Tables it provides important hints to developers of the application, e.g. how to design the user interaction flows, arrange the order of use interface elements, etc.

Figure 107 shows an example of the (draft) Transition Point Graph of the Italian Pilot. The order of some of these Transitions is directly related to the evolution of a crisis (black arrows). Additionally, there are many alternative paths (blue arrows) that heavily depend on the actual Simulation Case, e.g. whether mitigation options are performed before or after the crisis, whether cascading effects and/or Time-Dependent Vulnerability have to be taken into account, etc.

The difference between the Transition Point Graph (Figure 106) and a Simulation Case Graph (Figure 109) is, that the Transition Point Graph visualises all possible paths of analyses that the CRISMA Application must support while the Simulation Case Graph visualises only the possible paths of analyses that make sense for a particular Simulation Case.

Although this information might not be easy to collect, it is very useful for the overall design of the CRISMA Application as it helps the developer to get an understanding which flows of user interactions *must* be supported.

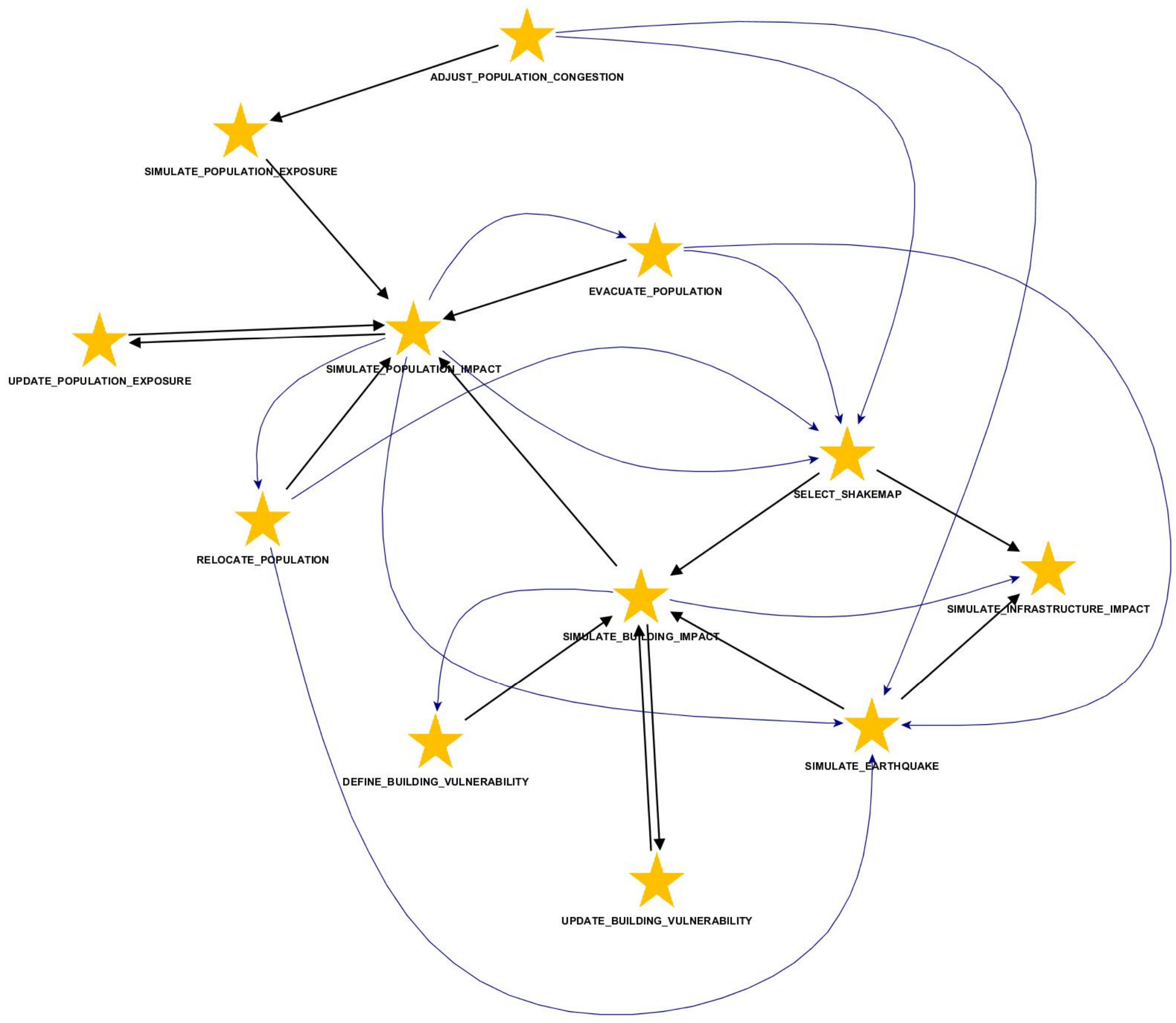

ADJUST_POPULATION_CONGESTION
SIMULATE_POPULATION_EXPOSURE
EVACUATE_POPULATION
UPDATE_POPULATION_EXPOSURE
SIMULATE_POPULATION_IMPACT
SELECT_SHAKEMAP
RELOCATE_POPULATION
SIMULATE_INFRASTRUCTURE_IMPACT
SIMULATE_BUILDING_IMPACT
DEFINE_BUILDING_VULNERABILITY
SIMULATE_EARTHQUAKE
UPDATE_BUILDING_VULNERABILITY

Figure 107: Transition Point Graph (Pilot D).

The individual Transition Points are specified in the following tables.

- **Transition Point #**
  This field contains the consecutive number and a unique id (e.g. name) of the Transition Point.
- Title
  This field contains the title of the Transition Point as it may appear in a CRISMA GUI.
- Description
  Provides a human readable description of the Transition Point, i.e. what is supposed to happen at this specific point in the CRISIS Management Scenario. In case of a Manipulation, it describes of the actions performed by the user of the CRISMA Application. It should also provide information on the position of the Transition Point in the timeline of a Crisis Scenario.
- Categories
  The categories of the Transition. They may be used as labels or tags in a CRISMA GUI or to allow discovery of specific types of Simulation Model. In the

CCIM mandatory category types (classifications) and example of possible values are:

- o **Transition Type Category:** either "Simulation" or "Manipulation";
- o **Transition Category:** "Mitigation", "Incident", "Hazard", "Exposure", etc.

- Predecessor
  The names of possible or reasonable predecessor Transition Points.
- Successor
  The names of possible or reasonable successor Transition Points.
- Data Slots Precondition:
  The Data Slots required by this Transition. Describes the status of the World State Data Slots before the Transition, i.e. which Data Slots are required to initiate the Transition and possible also their current values. Required only if the current status of Data Slots hasn't already been clearly defined by the preceding Transition Points.
- Data Slots Postcondition:
  The Data Slots affected by this Transition. Describes the status of the World State Data Slots after the Transition, i.e. how the Data Slot is changed by the Transition.
- Simulation Models
  Name(s) of Simulation Models if the Transition is a simulation.
- Building Blocks / User Interfaces
  Related Building Block Software Components that are used to perform the Transition and a description of their role during the Transition. A description of the user interface to perform the Transition and optional screenshots or mockups.
  Please note that this information might not be relevant for the Reference Scenarios (WP25), it is however be very helpful for the integration of the CRISMA Framework (WP35) and required for the development of a CRISMA Application.
- Comments
  Optional comments that provide e.g. additional information for developers or integrators.

Table 21: ICC Transition Points Overview.

ID	Name	Title
1	Example: "DEFINE_BUILDING_VULNERABILITY"	Example: "Define Building Vulnerability"
2	...	...
3	....	...
...	...	...

Table 22: Transition Points Description Tables.

Transition Point 2	**Example: “DEFINE_BUILDING_VULNERABILITY”**
Title	Example: “Define Building Vulnerability”
Description	Example: “Vulnerability Classes (EQV_BUILDINGS) of the Buildings can manipulated by the user, e.g. to simulate mitigation measures (increasing building resistance). Thus, for subsequent impact simulations, the changed vulnerability classes are now used.”
Categories	Example: “Transition Type Category: Manipulation; Transition Category: Vulnerability; Building Vulnerability, Mitigation”
Predecessors	Example: “*” (it could be the first Transition performed in a mitigation Simulation Case)
Successors	Example: “SIMULATE_BUILDING_IMPACT”
Data Slots Precondition	Example: “REGIONE, PROVINCE, COMUNE_AQ, GRID, EDIFICI and ISTAT_2001_B are shown in the GUI.”
Data Slots Postcondition	Example: “EQV_BUILDINGS are updated”
Simulation Models	Example: “n/a” (no Simulation Models involved)
Building Blocks / User Interfaces	Example: “The user selects a Grid Cell in the GIS Widget Building Block, a simple dialog pops up where can change resistance (vulnerability class) of the buildings in the selected cell (see Mockup in Figure 108). The Data Integration Building Block takes care to store a new version of the modified EQV_BUILDINGS Data Slot.”
Comments	Example: “Manipulation GUI has to be provided as extension of the GIS Widget Building Block (popup)”

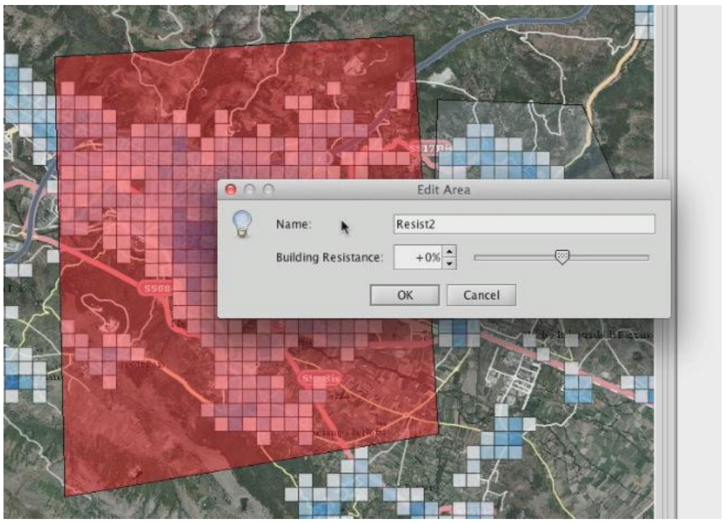


Figure 108: Transition Point GUI Mockup.

Simulation cases

This part of the template describes one Simulation Case of the overall Pilot Use Case. It has to be repeated for each distinct Simulation Case.

General information

This section provides some general information about the Simulation Case. This information may be used as descriptive meta-information in the CRISMA Application or the CRISMA Catalogue.

Name	*Please enter a name*
Category	*Category of the Simulation Case, e.g. "Operational Training", "Long Term Planning", "Incident Evolvement Planning", "Resource Planning", "Command and Control Training ", "Exercise Support View", "Operational Training".*
Description	*Please provide a short description of the Simulation Case. E.g. refer to the respective Pilot Use Case covered by this Simulation Case.*

Initial world state definitions

This section provides initial (static) definitions of the Data Slots that play a role in the World State of the current Simulation Case.

To be able to formally specify a Simulation Case in accordance to the Conceptual Business Logic, at first the initial World State Definition has to be described. The initial World State Definition is an immutable set of World State Data Slots. Immutable means, that a Transition may change the contents of a Data Slot but not the structure of the World State Definition. Thus, a Transition Point defines how the actual data of a Data Slot is changed by the Transition.

World State Data Slot Descriptions as well as ICC Elements that are relevant for this Simulation Case are referenced in the World State Description Table (Table 23).

Besides the unique names of the Data Slot and ICC Elements, additional information on the initial values or further comments for the developer of the respective CRISMA Application should be provided.

Table 23: World State Description Table.

World State of Simulation Case #		
Data Slots		
#	**Name**	**Initial Values / Comments**
ICC Vector		
#	**Name**	**Comments**

Simulation case specification

This section provides a description of the distinct steps of the Simulation Case (Decisions, Analyses) performed by a decision maker in the CRISMA Application and relates them to Transitions. If, for some reasons, the changes to the World State in a particular step of the Simulation Case are different than initially defined in the associated Transition, this has to be explicitly stated in the description of the Simulation Case Description Table (Table 24).

For better readability, the fields of the Simulation Case Description table are explained in the bullet list below while the table itself contains a representative example from the Italian Pilot.

- #
  The number of the step of the Simulation Case performed by the user in the GUI of the CRISMA Application.
- Transition Points
  The name(s) of the Transition Points related to this step of the Simulation Case.
- Description
  A detailed description of the actions and possible analyses that are performed by the use of the CRISMA Application in this particular step of the Simulation Case.

Table 24: Simulation Case Description Table.

#	**Transition Points**	**Detailed Description**
1	Example: "SELECT_SHAKEMAP **or** SIMULATE_ EARTHQUAKE"	Example: "The user chooses the kind of hazard input and the event characterization (e.g. shake maps **or** values of epicentre, magnitude and depth if shake maps are not available; event characterization includes time- e.g. daytime, night time)."
2		
3		
4		

The steps of the Simulation Case specified in Table 24 should be visualised in a Simulation Case Graph (Figure 109) taking also the temporal aspect of the Simulation Case into account.

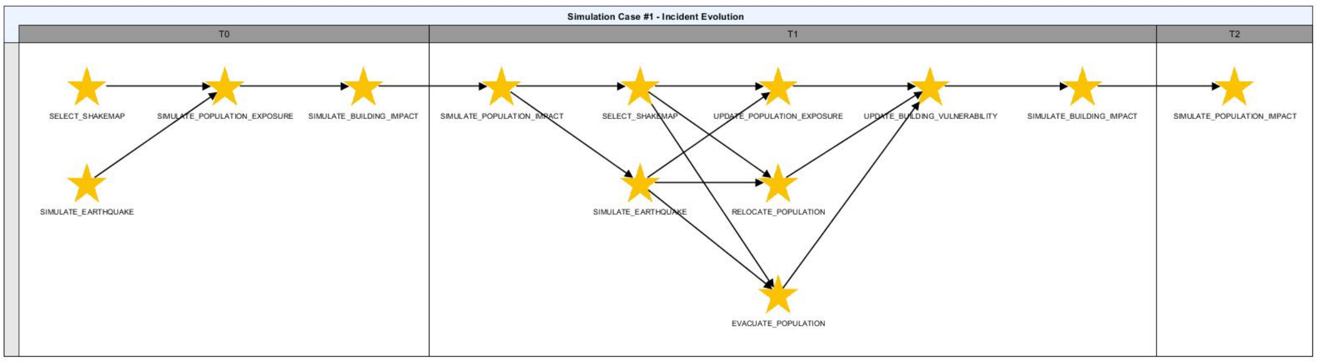

Simulation Case #1 - Incident Evolution
T0
T1
T2
SELECT_SHAKEMAP
SIMULATE_POPULATION_EXPOSURE
SIMULATE_BUILDING_IMPACT
SIMULATE_POPULATION_IMPACT
SELECT_SHAKEMAP
UPDATE_POPULATION_EXPOSURE
UPDATE_BUILDING_VULNERABILITY
SIMULATE_BUILDING_IMPACT
SIMULATE_POPULATION_IMPACT
SIMULATE_EARTHQUAKE
SIMULATE_EARTHQUAKE
RELOCATE_POPULATION
EVACUATE_POPULATION

Figure 109: Simulation Case Graph (Example).

Furthermore, examples of concrete Crisis Management Scenarios based on the previously defined Simulation Case should be provided. Figure 110 shows two Crisis Management Scenarios with different alternatives chosen.

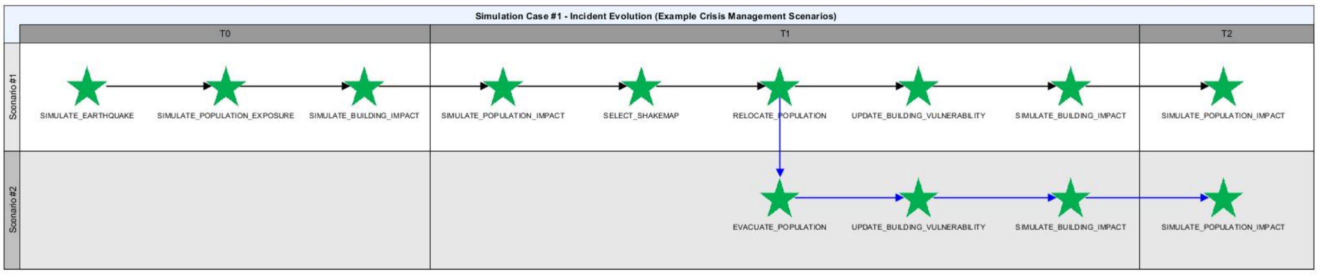

Simulation Case #1 - Incident Evolution (Example Crisis Management Scenarios)
T0
T1
T2
Scenario #1
Scenario #2
SIMULATE_EARTHQUAKE
SIMULATE_POPULATION_EXPOSURE
SIMULATE_BUILDING_IMPACT
SIMULATE_POPULATION_IMPACT
SELECT_SHAKEMAP
RELOCATE_POPULATION
UPDATE_BUILDING_VULNERABILITY
SIMULATE_BUILDING_IMPACT
SIMULATE_POPULATION_IMPACT
EVACUATE_POPULATION
UPDATE_BUILDING_VULNERABILITY
SIMULATE_BUILDING_IMPACT
SIMULATE_POPULATION_IMPACT

Figure 110: Example Scenarios.