2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)

Describing and Reusing Warfighter Processes and Products: An Agile Training Framework

Jeff Waters, Joanne Pilcher, Bruce Plutchak, Eric Voncolln, Daniel Grady and Ritesh Patel
Space and Naval Warfare Systems Center Pacific (SSC Pacific) San Diego, CA

***Abstract*—This position paper describes a framework, i.e. a set of design and architecture recommendations, for achieving agile training. The approach for the design is to be process and data driven, focused on reusability, and borrowing basic principles derived from web-based architectures, semantic processing, user-centered design, composability, complexity management, machine-understandability, scalability, gaming and open linked data. The fundamental features of the framework are open, easily understood, easily implemented, and tool-agnostic. With such a framework defined, the training community could collaborate to build out the more extensive cloud content, extend the capability and ensure that the benefits of agile training are achieved, namely more focused and faster training on shared processes anytime, anywhere at reduced cost and without a large support staff.**

Keywords—Agile, Training, Decision making, command and control, applications, resource allocation and management, web services, standards

I. Introduction: What is Agile Training?

This paper describes a prototype Agile Training Framework (ATF). The ATF focus is on agility and strong familiarity and expertise with processes. The goal for an ATF user is to not only understand the process, but to actually walk through it, producing the expected agile-version of the products at each step, and collaborating in small groups or large groups as needed to improve overall familiarity and facility with the processes. The ATF does support linking to products or resources produced by specific systems, but the ATF does not require the setup of specific equipment or systems. Instead, the ATF provides generic interfaces for the basic capabilities (e.g. map, timeline, product viewer/creator). The ATF runs in a browser, so it does not require installation of any client software. See Figure 1. Let's begin with some definitions.

What is an "Agile Training Framework"? As opposed to general education, which is focused more on the "what" of knowledge, training is focused more on the "how", i.e. the process. [1] An ideal agile training framework would support a full range of training needs, from beginner to expert, from individual to small group to large group training, from walk-through to simulation to operational environment, from our starting day to our retirement.

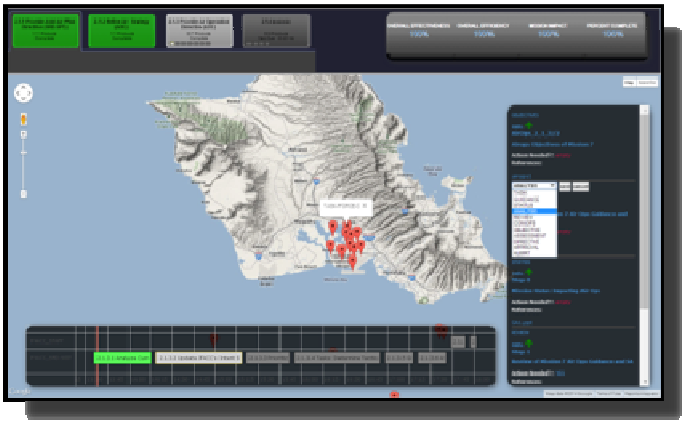

Fig. 1: A Prototype ATF

Agility implies flexibility, speed, ease of use, lightness, scalability, and quick adaptability to new and unforeseen conditions. [2] We'll see in the rest of this paper how "agile training" is enabled in terms of a framework; however, in terms of the end result, the agile training capability should easily allow any individual needing or desiring training to learn a process, walk-through it, simulate it, practice on it, join a larger community, support operational use, measure performance, revisit, and hone skills. The ATF should be available anytime from any location and be usable without travel, equipment or support staff setup. [3]

A framework, in this context, is a set of design principles with a support structure, such as an extensible, expandable reference implementation, which provides a foundation for building more capability consistent with those principles. [4] For example, one of the design principles of the ATF is that process input/output products should be progressively organized (i.e. hierarchically organized with a relatively fixed set of top-level generic types, such as Observation, Report, Request, Approval). So, the ATF provides a set of progressively organized data type representations ready-to-use for building an application consistent with that principle. A framework, once defined, also implies a community of contributors who will use the framework to build out the larger system over a period of time. [5] Although the ATF supports foundational principles and can guide community development to ensure agility, the community itself - the

trainers, the trainees, the subject matter experts, the data modelers, the process modelers, the simulation system developers, the command and control system developers, private industry, government agencies, academic institutions, coalition partners – all will build the specific components of capability that will be integrated and interoperate via this framework.

A final introductory word: One of the purposes of this paper is to suggest that, although the reader may be familiar with many of these principles and techniques, and they may seem simple, that the reader take a second look and consider adopting them as embodied here for the purposes of agile training. [6] We present here the principles and techniques, how they achieve the desired goal, and how they facilitate the adopters' goals and business models. [7]

II. PROCESSES, PRODUCTS AND PRINCIPLES

Let's start the ATF story by considering the underlying principles and any significant implications derived from teaching people "how" to do something. Since the "how" is fundamental to training, we should consider a process-driven approach. [8] In this sense, process-driven simply means that we should consider what a process is and what foundational principles we can extract from that definition to help guide agile training.

What is a "process"? A process can be thought of as a series of "steps" for accomplishing a goal. [9] A "step" in this setting is a "task", or an "action", something someone or something needs to do. [10] For a task to be completed, something needs to have been accomplished, and that accomplishment can be considered an output "product" of the task. [11] Each task may have one or more input or output products. We all execute processes in our daily lives and work; however, the processes are not always obvious or well defined. A well-defined task will have clearly defined tangible input and output products. [12] For some tasks, e.g. management, where the output product may seem at first intangible, the output product can take tangible form as a status report or other type of report.

Although processes can be defined and used for many purposes, such as automation, our purpose is training. This training purpose provides scope for the types of processes we should consider. For example, if a process is so well defined that it can be automated, and no human is going to perform it, then we don't need to train on it. [13] Similarly, if a "process" is too poorly defined, then apprenticeship might be the only way to effectively train someone. [14] Although the ATF can support these processes, the focus is to encourage defining and documenting processes with steps that have clearly defined input and output products and to support processes for training where the human component is still vital, such as higher-level processes of management, assessment and decision-making, sometimes referred to as command and control. [15]

What other significant implications can we draw from a process-driven approach to training? Previous work on net-centric interoperability and composability has produced a list of principles for us to consider. [16] These principles include: (a) machine-understandability; [17] (b) human-understandability; [18] (c) scalability; [19] (d) simplicity; [20] (e) modularity; [21] (f) composability; [22] (g) linkability; [23] (h) extensibility; [24] (i) visibility; [25] and (j) accessibility [26]. Fortunately, a few design techniques help greatly to enable and enforce these principles. One of the primary umbrella principles from the web environment is Representational State Transfer, known familiarly as REST. [27] Let's consider REST and how it helps enable a number of these principles of data representation for interoperability and composability.

What is REST? REST is an architectural style that is the basis for the current world wide web and is best characterized by these principles: (a) Every significant component, i.e. resource, of your application has a unique id (URI); [28] (b) Every component has a web-friendly representation (e.g. HTML for humans, JSON for systems); [29] (c) A limited, well-defined interface, such as HTTP, is used to Create, Retrieve, Update and Delete (CRUD) components; [30] (d) No application state is maintained on the server, i.e. everything you need to proceed is provided in the URL or the body of the response; [31] (e) executing a traditional application "service" is accomplished by defining the representation of the output product of the service so the client can use the CRUD interface to create, retrieve, update or delete that product [32]. By using REST, all your significant resources become visible, accessible, composable, modular, scalable, and support higher level knowledge management.

Machine Understandability: A few principles, if followed, are simple to explain and go a long way to enable machine understandability. One of these principles is to use URIs to provide unique names for terms, whether those terms refer to real things or concepts, so that we can be clear and begin to define the meaning of the terms. [33] Words in languages can often mean different things to different people and the definitions that we find in dictionaries are not adequate for machine-understandability. Semantic standards, such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL), use URIs to identify terms and to link terms via relationships. Although the use of URLs was also advocated in the discussion of REST, we here note this additional important reason for using URLs, to ensure terms are more machine understandable.

Another simple principle to aid machine understandability is to avoid defining elements as strings, or any other form of free text. [34] Free text is difficult for a machine to understand. Although string pattern matching and natural language processing are powerful techniques, essential elements can and should be better defined in simpler ways. More standardized terms and definitions for most text fields can be determined by visiting with the practitioners of the given process. Those terms can then be referenced by unique

identifiers, e.g. URLs, and then terms can be linked to their definitions, relationships, or other useful information supporting drill-down, inferencing, and querying.

Another principle for ensuring that processes are machine understandable is to ensure that each step in the process has well-defined input and output products. [35] Even if the product is simply a status report, clearly-defined tangible products can be used by a machine for many purposes, including monitoring process status, linking processes together via their input/output products, alerting others, forwarding messages, inferencing, and supporting innovative and dynamic changes to processes (e.g. a process or sub-process can be reassigned or changed or skipped as long as the needed product is produced).

Human Understandability: For processes to be learned in the form of training, they should be as humanly understandable as possible. Human understandability is aided by products and processes that are organized in a progressive fashion, i.e. organized hierarchically where parent nodes are logical groupings of the child nodes, such that ideally no more than 7 to 10 steps (or tasks) are under any give node. [36] This logical grouping can be applied to both products and processes. This type of organization makes processes much more amenable to human understandability.

Simplicity: Modular components, performing simple functions whose design includes a standardized linking ability, can be used to manage the complexity of larger systems. This principle is followed in the ATF. As noted, all processes are defined in terms of progressively-organized steps. Since each product is identified by a URL, it's possible to establish links between the steps using the input/output products as the links. [37] URL filters can be used that specify the type of input product desired with any desired range limitations, such as a product from a specific author role or a specific geographic area. [38] By this indirect method, process steps may be dynamically linked to form new processes or to reveal undocumented existing processes. By using the products as the link between the steps, we have the full-range of flexibility from hard-wired to broad scope filters.

Scalability: The ATF is designed to be scalable both in terms of its structure and its content. Scalability means the system is capable of expanding gracefully as the numbers of users and the need for services increases. There is a technical side to scalability which solutions like REST attempt to address, but there is also a substantive content perspective which addresses who is going to put in all of the content. So a key design feature to enable substantive scalability is to make the system accessible to the widest number of users and to empower those users to provide the content easily. The primary advantage to empowering users is that there are potentially thousands of users, e.g. in the ATF case, all those who want or need training, whereas the number of support staff is limited.

Extensibility: The ATF supports extension of its content, as well as expansion, through the extendable progressive organization of the content. Any user can extend any of the processes or datatypes (e.g. Observation, Report, Request, Approval) and then save the new process or datatype back to the cloud repository for reuse by others. Users can add new standard "answers" to pull-down lists for data field "questions", such as why was the request denied or what is the status level. In all of these ways, the user is empowered to not only contribute content, but to tailor or extend the entire system as needed to support their processes and terminology. This extensibility is a key enabler of agility and is well used in the ATF.

User Interface: For each step in a workflow, a user needs an interface appropriate for that purpose. Since each workflow step in this agile framework produces a product, and since the products are progressively typed, share the same basic core data and links, and are expanded in a standard way as property/values, the user interface can be built from reusable widgets, one per *generic* product type. [42] More specialized widgets, one per *specialized* product type, can be fleshed out by the community. The simple but structured representation of the data products and processes enables a structured user interface with reusable components. In this way, even when a new process is implemented in the framework, a user interface to drive the training on that process can be created automatically.

III. THE ATF PROTOTYPE

Based on the principles noted above, a relatively simple, generic set of data products have been proposed for use in the prototype ATF for proof-of-concept. Each product representation includes the set of core common elements listed in Figure 2 and then each product adds on any additional elements required for that particular data type.

The high-level types, relevant for decision-making, have a number of elements and links in common which support the basic principles. The types are simple, standardized, and combined in sequential or hierarchical ways to construct more complex structures such as progressively-organized processes or data products. To the extent each type of data product varies, individual unique elements are added as appropriate. The selection of data products and the amount of detail represented is intended to address the goals of the framework effort, which is to serve as a proof-of-concept prototype that can be expanded by the community as needed.

A. Agile Web-based Data Repository

The ATF prototype uses MongoDB, an open-source, document-oriented, NoSQL database as a process and product data repository. MongoDB stores JSON-style documents with dynamic schemas allowing significant flexibility for front-end developers. MongoDB's features provide significant flexibility and agility which can be leveraged effectively to provide a real-time, run anywhere data access capability appropriate for enabling a cloud solution for the agile training framework.

The current data format for information exchange from client to data repository is JSON. JavaScript Object Notation

(JSON), including GeoJSON, is a text-based, open standard, language-independent, lightweight data interchange format, easy to read and write, and well-used and supported by modern web and cloud services.

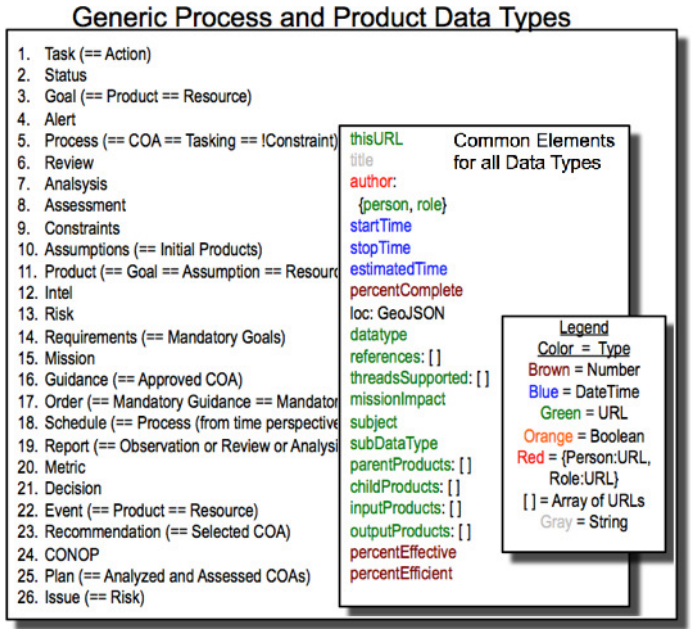

Fig. 2: Generic Data Types and Their Common Elements

B. *Future Development: Scenaro Worlds*

Agile training is designed to support rapid transition from individual to small group to large group training, and the fastest transition to large groups, following the online gaming example, is to maintain online scenario worlds. These virtual worlds can be designed from two perspectives. First, planners have expectations for the types of worlds in which they will need to conduct their operations. For example, one might envision a Disaster Relief World which supports modeling of a large scale disaster, such as a hurricane or tsunami. Such a world would pose many significant, but foreseeable, barriers to effective operations, and pose significant, but foreseeable, needs, e.g. evacuations, shelter, water, medical care. Second, planners have knowledge of the major defined mission processes upon which all participants should be trained. Planners could map these mission processes to the core 7 or 8 scenario worlds, and tailor the worlds accordingly, to ensure all core processes are represented in these worlds. Since the planners and world builders know about the agile training core products and processes, they can build specific aspects of the scenario worlds to simulate those processes and support any online user who wants to join and participate.

Advantages of scenario worlds include: (a) a great home for advanced modelers and scenario builders to create reusable, detailed models and scenarios for broad reuse; (b) available online 7/24, 365, and accessible remotely from anywhere in the world; (c) enable users to join or rejoin for any amount of time; (d) competition against other live users; (e) enables collection of metrics to document, assess and motivate; and (f) turns the one-time expense of building scenarios into sound investment through reuse.

IV. Conclusion

The purpose of the Agile Training Framework is to enable and empower local control of training needs 7/24 365 in a flexible cloud-based, browser-based, easy-to-use, process and data-driven interface. The interface design and its reference implementation will serve as a working proof-of-concept of scalable, flexible, dynamic, cloud-based training on the continuum of process and products from coalition, joint, service, command and local units. The ATF is designed to be extensible and evolvable by the training community. The result of the work will be to improve mission information clarity, sharing and interoperability to reduce mission errors, improve speed of situational awareness, and assessment across the enterprise, and to increase the number of missions that can be managed by a single operator.

Principles & References

[1] Training is Learning and Practicing Process; Training is focused on the "how" of doing something, i.e. the process. See Edwards, J. "A Process View of Knowledge Management: It Aint What you do, it's the way That you do it." *Electronic Journal of Knowledge Management* V. 9, Issue 4 www.ejkm.com/issue/download.html?idArticle=301

[2] Agility is required in modern warfare; Agility implies flexibility, speed, ease of use, scalability and quick adaptability to new circumstances and unforeseen conditions. See Dekker, Anthony H. "Measuring the Agility of Networked Military Forces [online]." *Journal of Battlefield Technology*, Vol. 9, No. 1, Mar 2006: 19-24. <http://search.informit.com.au/documentSummary;dn=111183921111700;res=IELENG> ISSN: 1440-5113.

[3] Agile Training is the goal – flexible, quick and easy; Agile Training should enable an individual at any time from anywhere to learn a process, walk-through it, simulate it, practice on it, join a larger community to improve, measure performance, and revisit and hone skills. See Gehler, C. "Agile Leaders, Agile Institutions." Carlisle, PA: *Strategic Studies Institute*, US Army War College, 2005. http://www.strategicstudiesinstitute.army.mil/pdffiles/PUB618.pdf

[4] An Agile Training Framework enforces the principles of agile training and makes building capability in alignment with these principles easy; A Framework is a set of design principles coupled with a scaffold infrastructure enabling and optimizing those principles. Users can build capability more easily and more consistently compliant with those principles by using the framework. See Wong, K.Y. and Aspinwall, E., "Knowledge Management Implementation Frameworks: A Review", *Knowledge and Process Management Volume 11 Number 2 pp 93–104 (2004)* http://download.clib.psu.ac.th/datawebclib/e_resource/trial_database/WileyInterScienceCD/pdf/KPM/KPM_5.pdf

[5] The Framework is for the Community; the community can extend or expand the framework to enhance the usability of the framework over time. See Rico, Mariano, David Camacho, and Óscar Corcho. "A contribution-based framework for the creation of semantically-enabled web applications." *Information Sciences* 180.10 (2010): 1850-1864. http://oa.upm.es/5632/2/Corcho_02.pdf

[6] A Framework ensures principles are followed even if they are not well understood; if developers are left unguided, they often ignore recommended implementation solutions because the connection to the underlying principles, and the underlying principles themselves, are not well understood. See [4]

[7] The Agile Training Framework enforces agile training principles; The recommended implementation solutions presented here support these principles and can be easily used.

[8] Agile Training is Process-Driven; An Agile Training Framework (ATF) should take a process-driven approach.

[9] A Process is a series of Steps for accomplishing a goal. Lindsay, Ann, Denise Downs, and Ken Lunn. "Business processes—attempts to find a definition." *Information and software technology* 45.15 (2003): 1015-1019.

[10] A Step in a process is a task, i.e. an action, something that someone or something must do. See [9].

[11] Each completed step results in an output product.

[12] Each Step in a well-defined process must have one or more input and output products.

[13] Processes so fully defined as to be automated don't require human training.

[14] Processes so ill defined as to require "artists" are not amenable to training.

[15] Well-defined Processes, where the human component is vital, are the focus of agile training.

[16] Net-centric literature defines many principles of interoperability and composability. See Kaplan, Jeremy M. "A new conceptual framework for net-centric, enterprise-wide, system-of-systems engineering." *National Defense University Washington DC Center for Technology and National Security Policy*, 2006. http://www.dtic.mil/dtic/tr/fulltext/u2/a453974.pdf

[17] Machine understandability of data and processes allows for machine processing as opposed to using computers to be used as simply fancy telephones, to chat, send email, and VTC. See http://www.w3.org/DesignIssues/LinkedData.html

[18] Human understandability of data and processes ensures we can effectively train a broad and diverse set of ordinary (non-expert) users. See Ottensooser, Avner, et al. "Making sense of business process descriptions: An experimental comparison of graphical and textual notations." *Journal of Systems and Software* 85.3 (2012): 596-606.

[19] Scalability means that the framework empowers users and so can enable and support a vastly growing number of users, data, processes and overall training accomplished.

[20] Keep things simple. See "The Simplicity Principle in Human Concept Learning" *Current Directions in Psychological Science* December 2003 vol. 12 no. 6 227-232.

[21] Modular components are simpler to build and understand, and they fit together in a simple, standardized manner.

[22] Sophisticated systems can be built from composable modular components. See Callebaut, W. "Understanding the Development and Evolution of Natural Complex Systems." (2005 MIT Press).

[23] Linking components is a form of distributed, decentralized, standardized interconnection of modular components. See Wilde, Erik, Florian Michahelles, and Stefan Lüder. "Leveraging the Web Platform for the Web of Things: Position Paper for W3C Workshop on the Web of Things." (2014). *http://dret.net/netdret/docs/wilde-wot2014-w3c.pdf* See also Heath, T., Hepp, M., and Bizer, C. (eds.). "Linked Data – The Story So Far." *http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf*

[24] Extensibility means the entire system doesn't have to be built on day one, but can evolve over time.

[25] Visibility means I can see your capabilities, needs, products and status so I have the chance to help you even if we don't know each other. Kaplan, Jeremy M. "A new conceptual framework for net-centric, enterprise-wide, system-of-systems engineering." *National Defense University Washington DC Center for Technology and National Security Policy*, 2006. http://www.dtic.mil/dtic/tr/fulltext/u2/a453974.pdf

[26] Be Accessible; Accessibility means you and I have the ability to interact and help each other without requiring special stove-piped communication channels. See Stenbit, John P. Department of Defense Net-Centric Data Strategy. *DEPARTMENT OF DEFENSE WASHINGTON DC CHIEF INFORMATION OFFICER*, 2003.

[27] Use REST; REST supports key design principles, including modularity, visibility, accessibility, linkability, machine understandability, scalability, simplicity, extensibility and composability. See Xu, Xiwei, et al. "An architectural style for process-intensive web information systems." *Web Information Systems Engineering*–WISE 2010. Springer Berlin Heidelberg, 2010. 534-547. Also See Haupt, F. et al. "A model-driven approach for REST compliant services." *Proceedings of the IEEE International Conference on Web Services*. ICWS 2014. http://design.inf.usi.ch/sites/default/files/seminar-2014-florian-haupt.pdf.

[28] Every significant component (resource) of your application should be addressable by a unique URL; The URL serves as a globally unique identifier, but also enables accessibility, visibility, semantic referencing, linkability, and CRUD services. See Verborgh, Ruben, et al. "The Fallacy of the Multi-API Culture: Conceptual and Practical Benefits of Representational State Transfer (REST)." *Journal of Documentation* (2014).

[29] Every significant component (resource) of your application should have web-friendly representations, including a human-understandable one (e.g. HTML) and a machine-understandable one (e.g. JSON); The representation supports CRUD services, accessibility, visibility and linking. See Gomez-Perez, Asuncion, Mariano Fernández-López, and Oscar Corcho-Garcia. "Ontological engineering." *Computing Reviews* 45.8 (2004): 478-479.

[30] Every significant component (resource) of your application should have one simple standardized programmer interface (e.g. HTTP or HTTPS) supporting only the basic CRUD (create, retrieve, update, delete) functionality.

[31] Do not maintain application state; scale your applications by representing application state in your data model through the use of URL links pointing forward and backward to next and previous steps. See http://spring.io/understanding/HATEOAS.

[32] A CRUD interface requires that all services be converted into data creation and retrieval services which ensures that all significant components of your application, not just data but also services, are accessible, visible, scalable, simple to use, and machine-understandable. See Leonard Richardson, Mike Amundsen, and Sam Ruby. 2013. Restful Web Apis. O'Reilly Media, Inc..

[33] Use URLs as unique identifiers for your application terms and concepts; The use of URLs (or more broadly URIs) supports semantic processing standards such as the Web Ontology Language (OWL) and the Resource Description Framework (RDF) and the Open Linked Data movment, avoids language-unique issues, and enables accessibility and visibility to the terms. See Bizer, Chris, Richard Cyganiak, and Tom Heath. "How to publish linked data on the web." (2007). See also http://cs.ulb.ac.be/public/_media/teaching/infoh509/8-owl.pdf.

[34] Avoid free text in your data representations; Free text is uncontrolled, unclear, and unmanageable.

[35] Ensure each step in a process has well-defined input and output products; Well-defined input and output products can be monitored, machine-processed, linked to and from and serve as a dynamic, scalable mechanism to link steps in processes.

[36] Organize both your data types and your processes progressively, i.e. with a limited number of generic, manageable types or steps at each level of a hierarchy; Progressive organization enables humans to have a manageable amount of information to learn and understand at each level, and enables more information to be added in a manageable way.

[37] Use URL references to the input/output products to link steps in a process; scalability is enabled as well as composability, modularity, and migration to operational realism to enable one step to link dynamically to one or more processes.

[38] Use URL references to types of input/output products and to subsets of the input/output products to tailor how processes are linked; process can be linked by products produced in a given geographic area, by the unit which produced it, or by other properties. See http://www.cambridgesemantics.com/semantic-university/rdf-101.

[39] Ensure scalability by making your system widely accessible; the more users who can access your system, the more users who can reuse the content and share information.