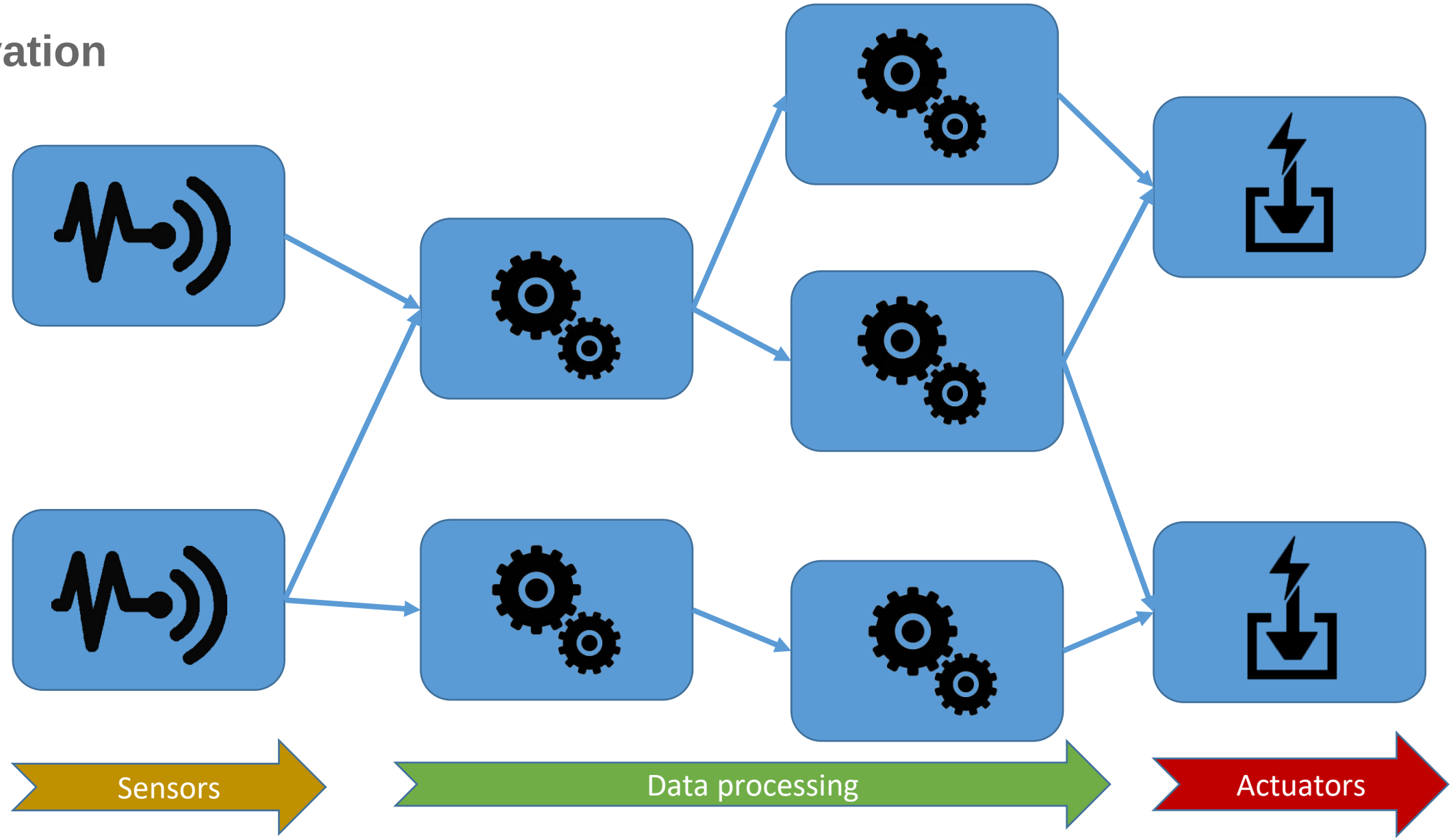
Tasking Modeling Language: A toolset for model-based engineering of data-driven software systems

Tobias Franz, Ayush M. Nepal, Zain A. H. Hammadeh, Dr. Olaf Maibaum, Dr. Andreas Gerndt, Daniel Lüdtke

Motivation

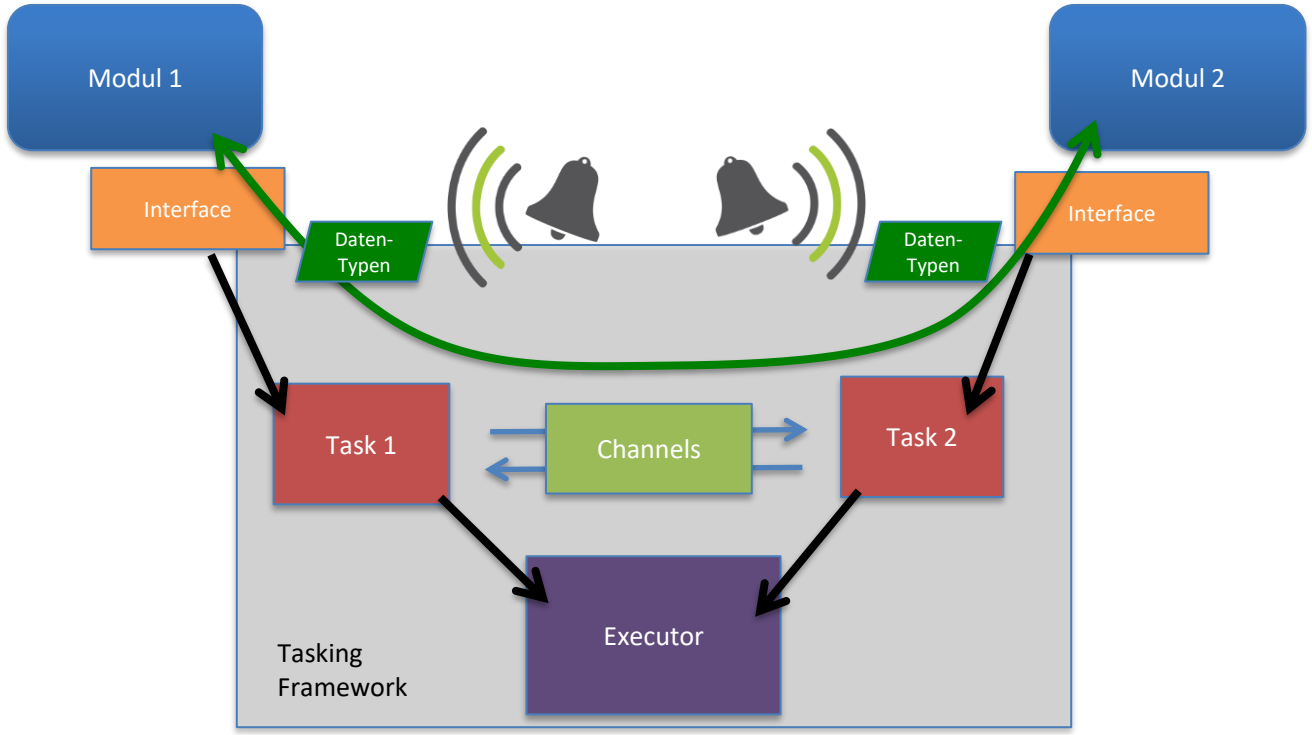


Motivation and Requirements

- Improve development process (Productivity & Prototyping)
  - Component interfaces
  - Data flow

- Event- & data flow driven
  - Execution parameters
  - Event control

- Customization to project specific requirements

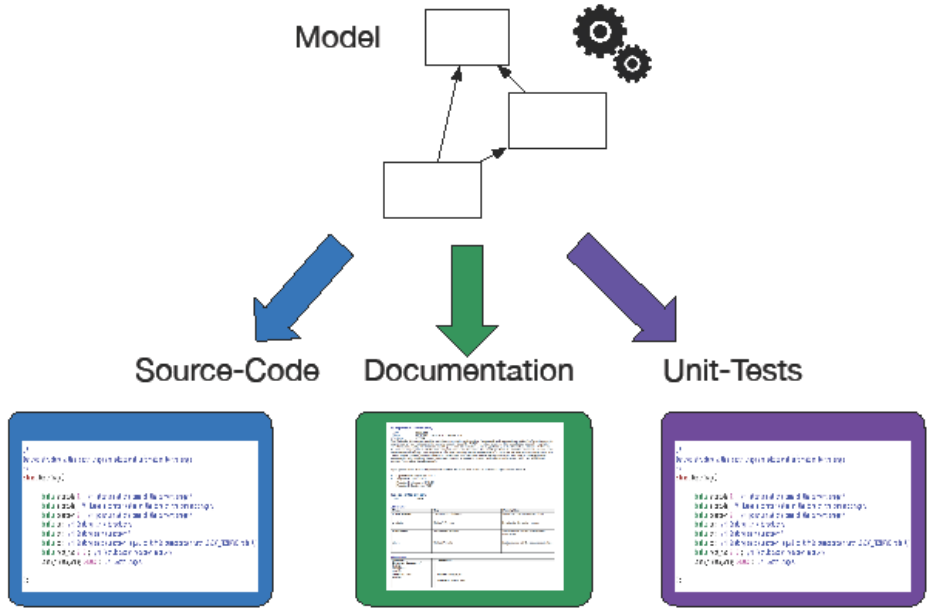


Tasking Framework: Github https://github.com/DLR-SC/tasking-framework

Z. A. H. Hammadeh, T. Franz, O. Maibaum, A. Gerndt, and D. Ludtke, “Event-driven multi-threading execution platform for real-time on-board software systems,” in *15th annual workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pp. 29–34, Juli 2019

⇨ Default framework; but not limited to it

Model-driven Development



- System specification in a central model
- Generation of project artifacts (Source code, documentation, configuration…)
- Improved productivity
  - Short-term
  - Long-term

C. Atkinson and T. Kuhne, “Model-driven development: a metamodeling foundation,” *IEEE software*, vol. 20, no. 5, pp. 36–41, 2003.

➡ NASAs Curiosity verwendete ca. 75% generierten Code

G. J. Holzmann, “Landing a Spacecraft on Mars,” *IEEE Software.*, vol. 30, no. 2, pp. 83–86, 2013

Quelle:https://upload.wikimedia.org/wikipedia/commons/3/30/Pia19808-main_tight_crop-monday.jpg

Related Work

TASTE : A real-time software engineering tool-chain Overview, status, and future

- Modeling with
  - AADL: architecture
  - ASN.1: data types
- Generation of
  - Source-Code
  - Documentation
  - Binaries

M. Perrotin, E. Conquet, J. Delange, and A. Schiele, “TASTE : A real-time software engineering tool-chain Overview , status , and future,” in *SDL 2011: Integrating System and Software Modeling*, Springer-Verlag Berlin Heidelberg, 2011, pp. 26–37

Domain-Specific Languages and Diagram Customization for a Concurrent Engineering Environment

- Systems Modeling Language (SysML) for interface modeling
- High learning effort
- Creation of a DSL on the basis of SysML
- Customizations with UML Profiles

B. Cole, G. Dubos, P. Banazadeh, J. Reh, K. Case, Y. F. Wang, S. Jones, and F. Picha, “Domain-Specific Languages and Diagram Customization for a Concurrent Engineering Environment,” *IEEE Aerospace Conference Procedings*, 2013

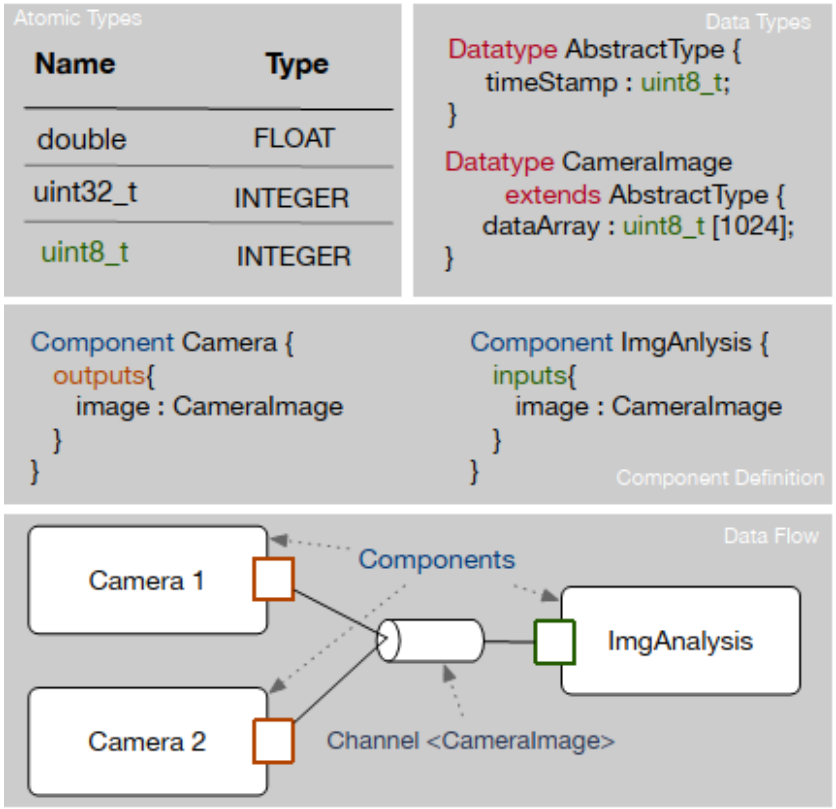
UML customization versus domain-specific languages

- Difficulties with SysML and UML for domain experts
- High learning effort and ambiguity
- Posed question: “whether it is better to develop a new DSL […] iteratively, or start with a general-purpose modeling language and restricting it until it fits the specific use”

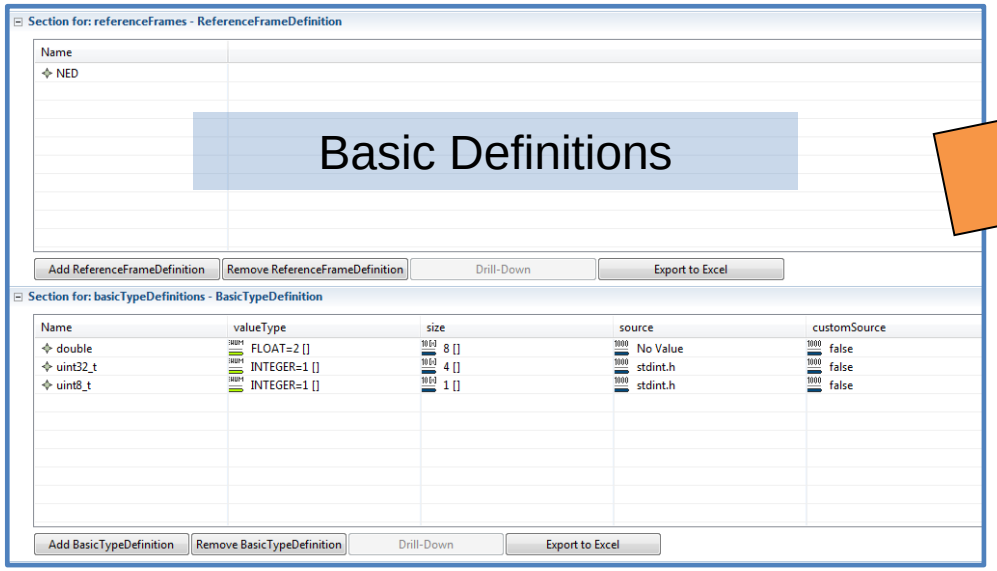
J. Gray and B. Rumpe, “Uml customization versus domain-specific languages,” 2018.

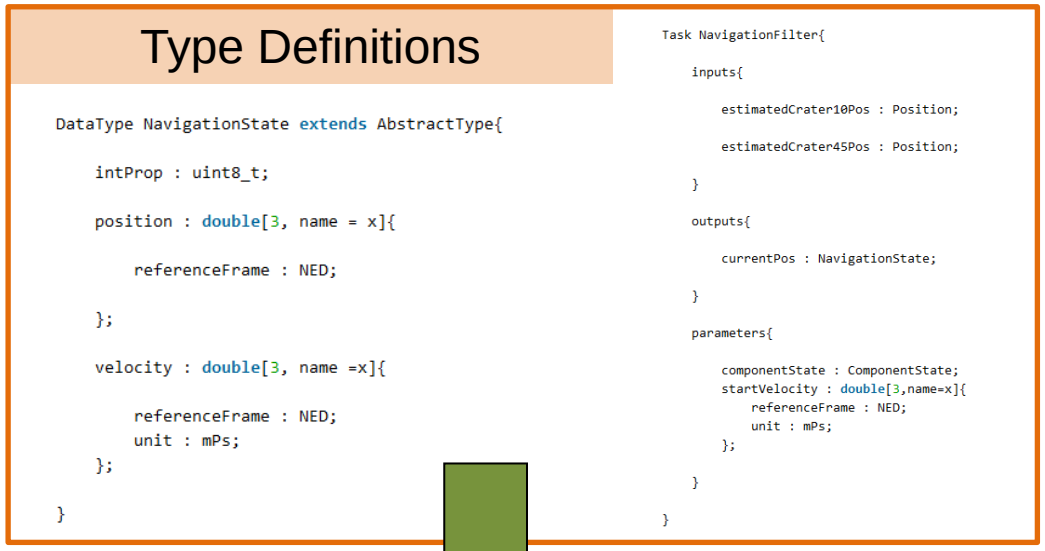
Concept

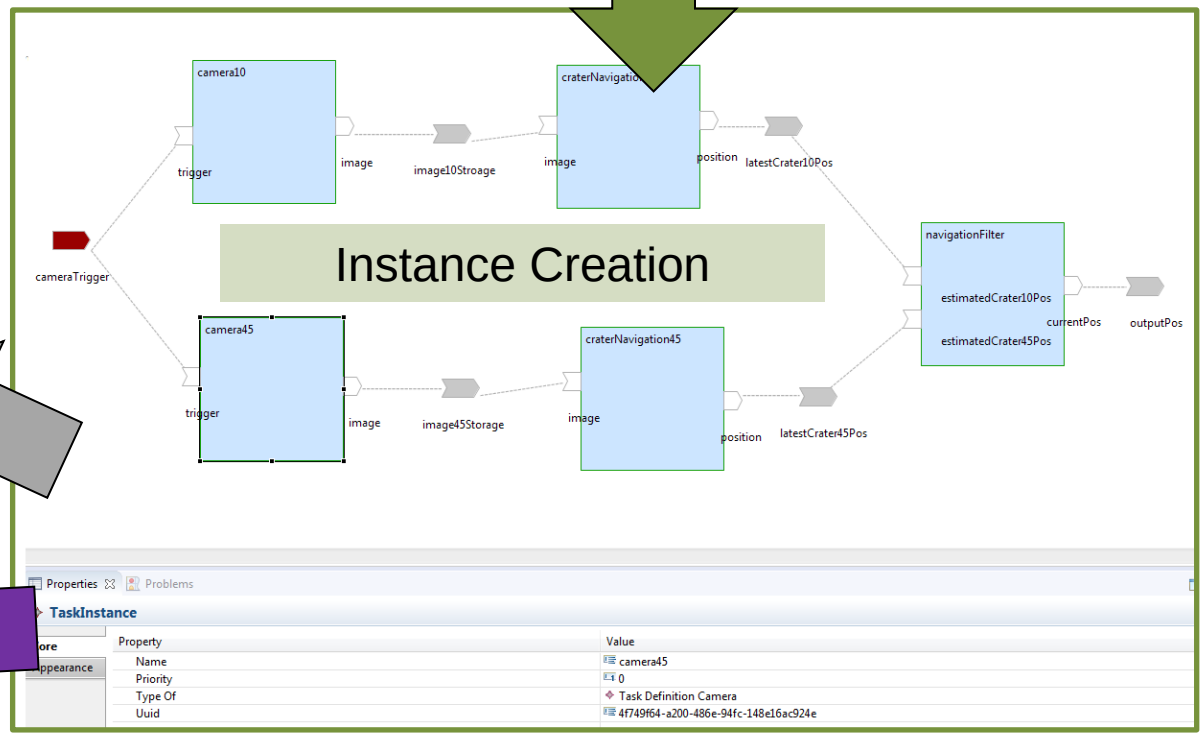
- Developing a DSL from scratch: only elements that
  - we need
  - generators support
- Different ways of modeling
  - Tables / List
    - Simple
    - Works only for simple content
  - Diagrams
    - Intuitive
    - Provide overview
    - Improves visual understanding
    - More modeling work: additional layout
  - Textual language
    - Supports most complex model aspects
    - Highest learning effort: complex grammar support



Workflow

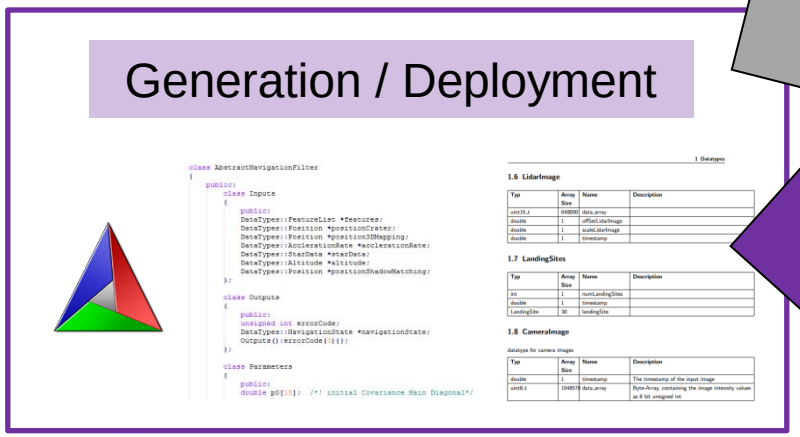




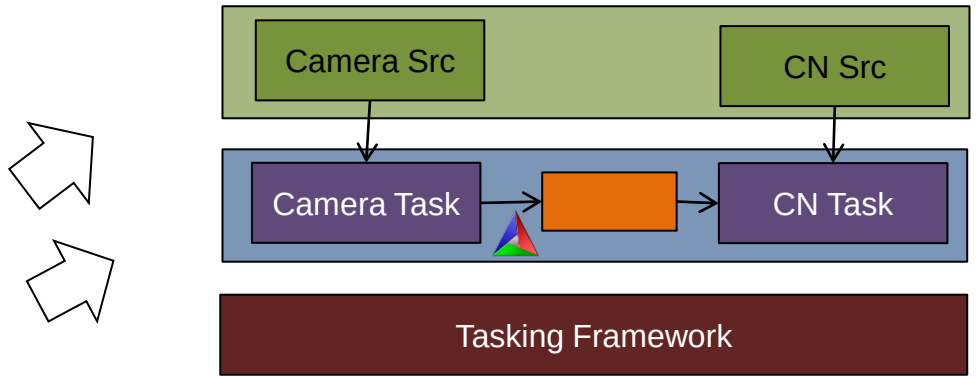


A. Kovalov, T. Franz, H. Watolla, V. Vishav, A. Gerndt, and D. Ludtke, “Model-based reconfiguration planning for a distributed on-board computer,” in *Proceedings of the 12th System Analysis and Modelling Conference*, pp. 55–62, 2020

SCOSA Node-Mapping



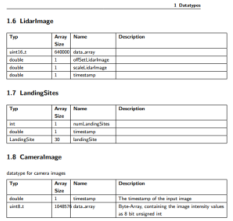
Artifact Generation



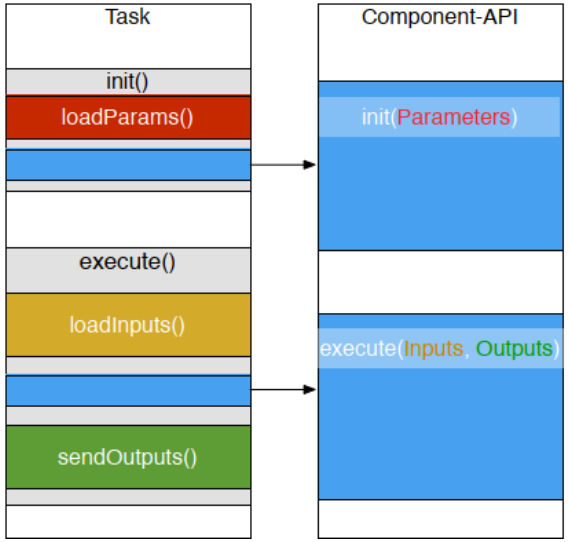
Component Layer — Stubs generated

Interaction Layer — generated

Execution Platform

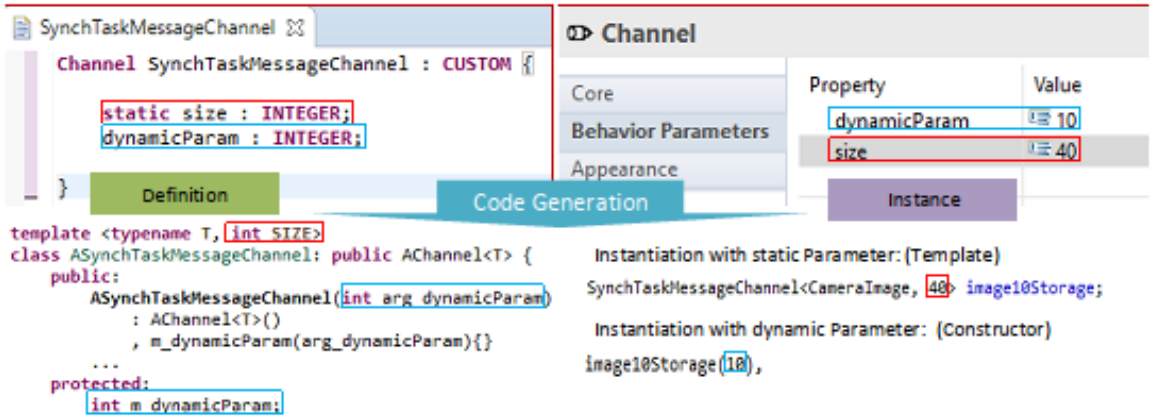


Documentation



- Templates for source code and documentation

- Levels of Extensibility
  - Model Extensions
  - Template Customization
  - Generation Gap Pattern

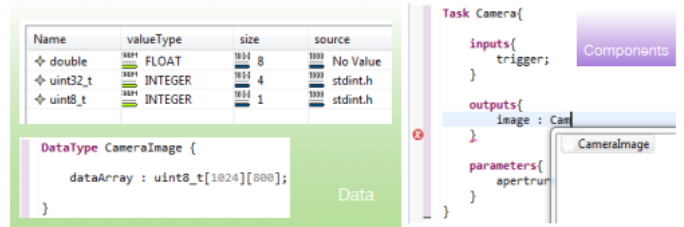


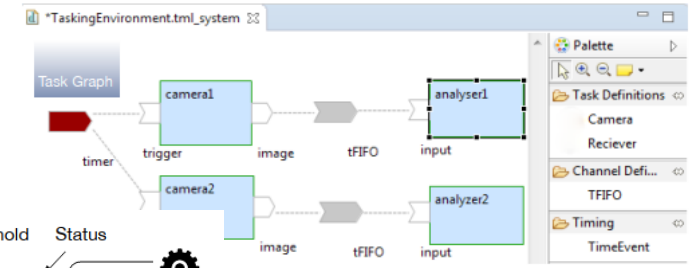
Evaluation

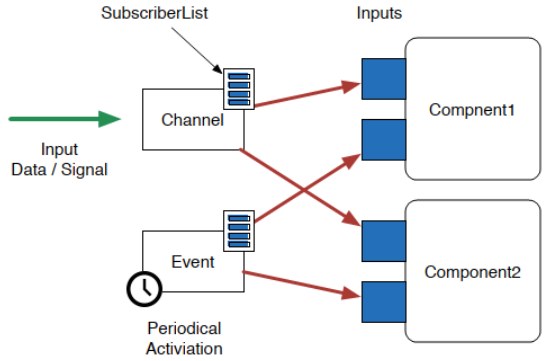
Requirements

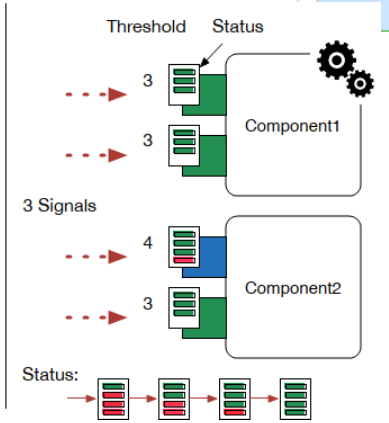
Slide 3

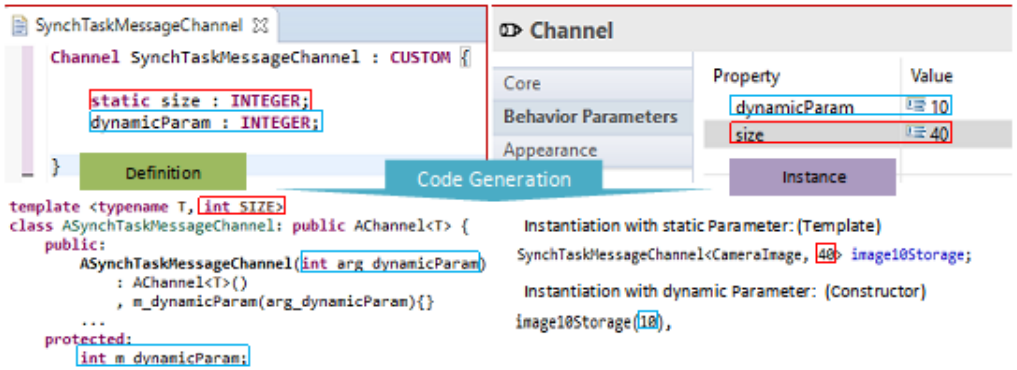
- Improve development process (Productivity & Prototyping)
  - Component interfaces
  - Data flow
- Event- & data flow driven
  - Execution parameters
  - Event control
- Customization to project specific requirements



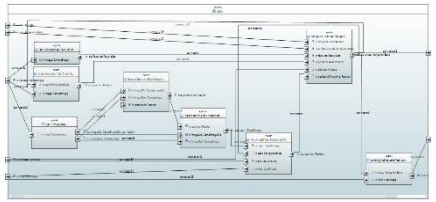




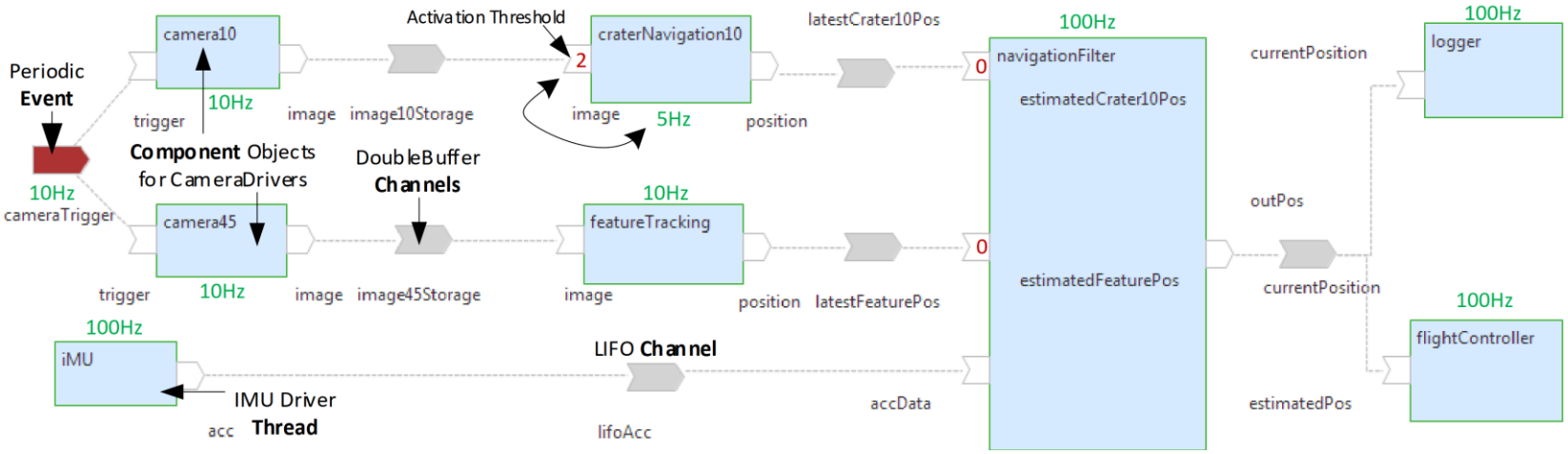




Case Study

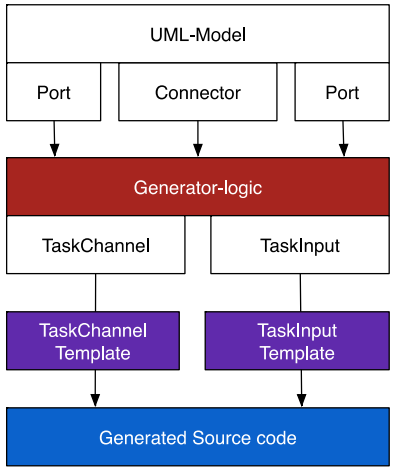


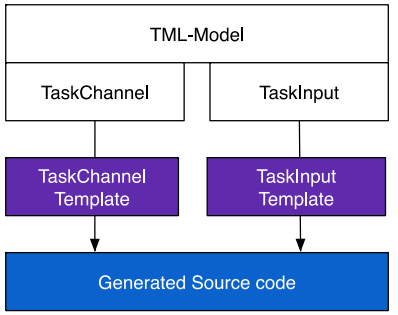
T. Franz, D. Ludtke, O. Maibaum, and A. Gerndt, “Model-based software engineering for an optical navigation system for spacecraft,” *CEAS Space Journal*, vol. 10, no. 2, pp. 147–156, 2018



Comparison with previous SysML solution:

- Viewer model element types in e.g. component / block diagram
  - 4 types in TML vs. > 40 UML/SysML
- Generator does not need to „transform“ the model to domain

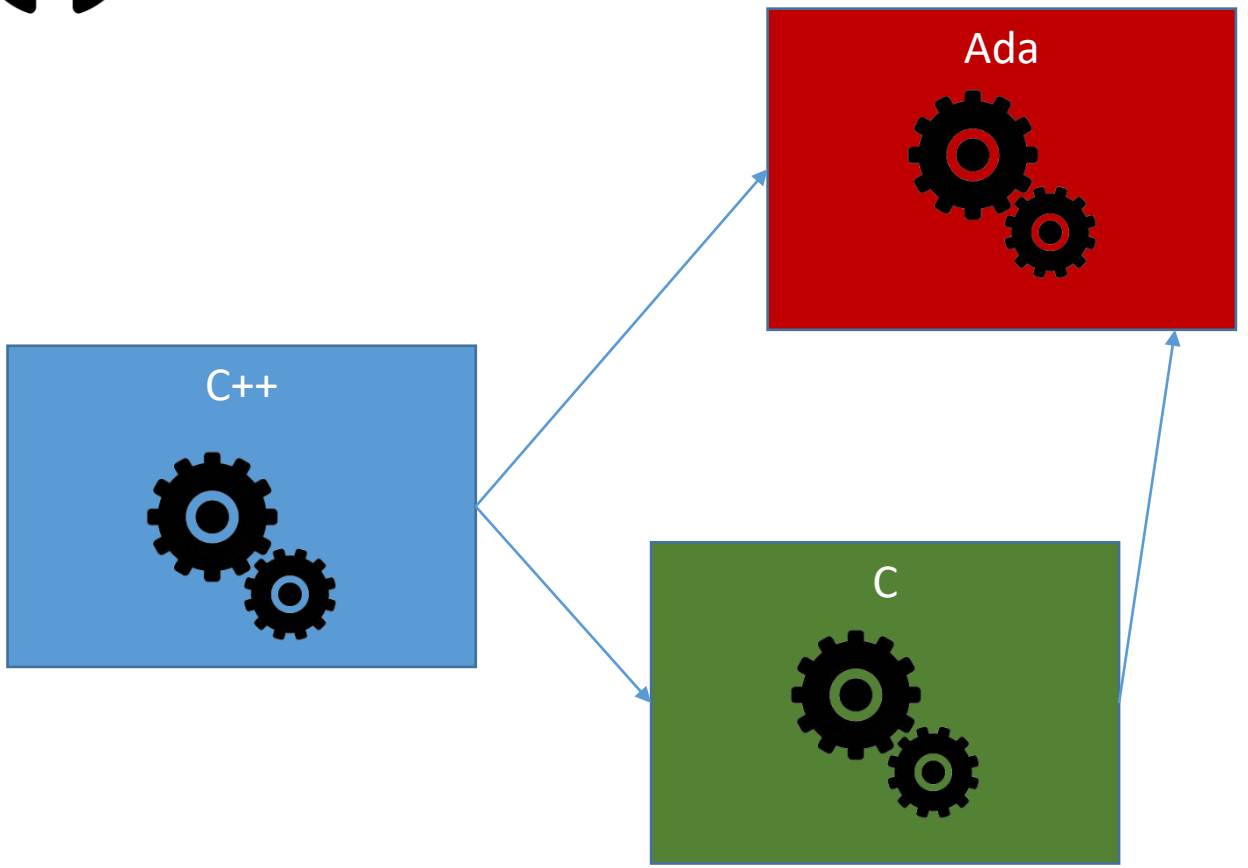




Outlook and future work

- Extension of code generator
- Multilanguage support
- Release on Github
- Integration with TASTE and/or OSRA

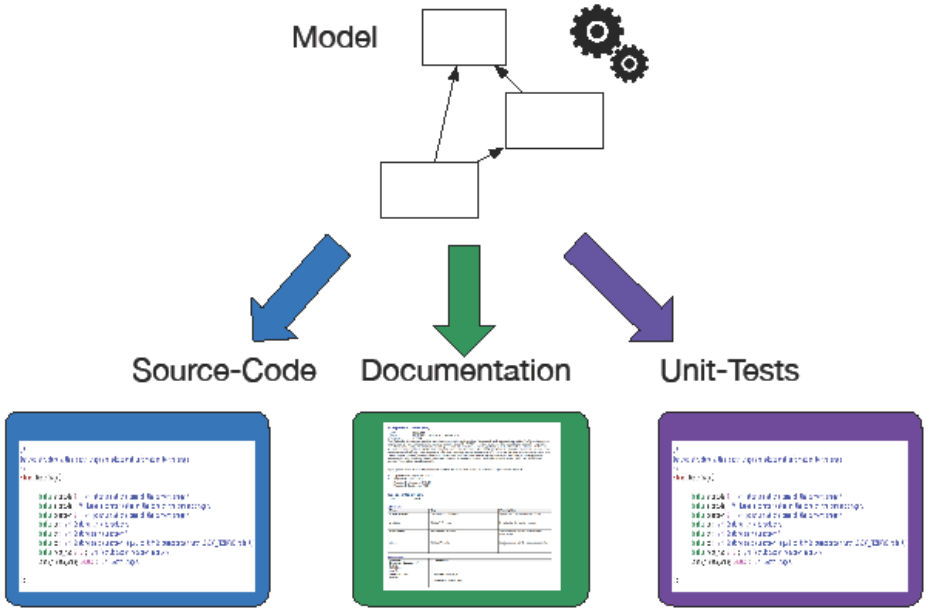
Planned Release on Github

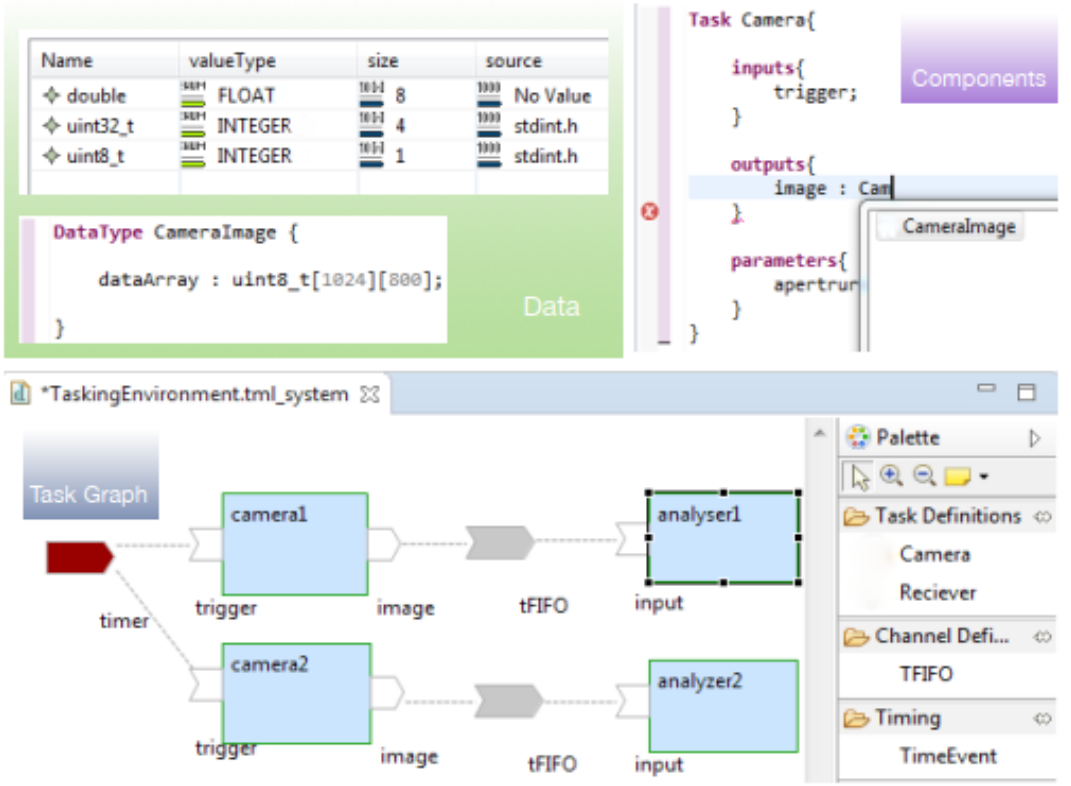


Property	Value
Language	Cpp
Name	Craternavigation
Output	FIFO Channel positions
Priority	100
Stereotype	
TaskSet	Task Set MainTaskSet

Conclusion

- Model-based systems engineering toolset
- Domain-specific languages for data + event driven systems





Planned Release on Github