

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

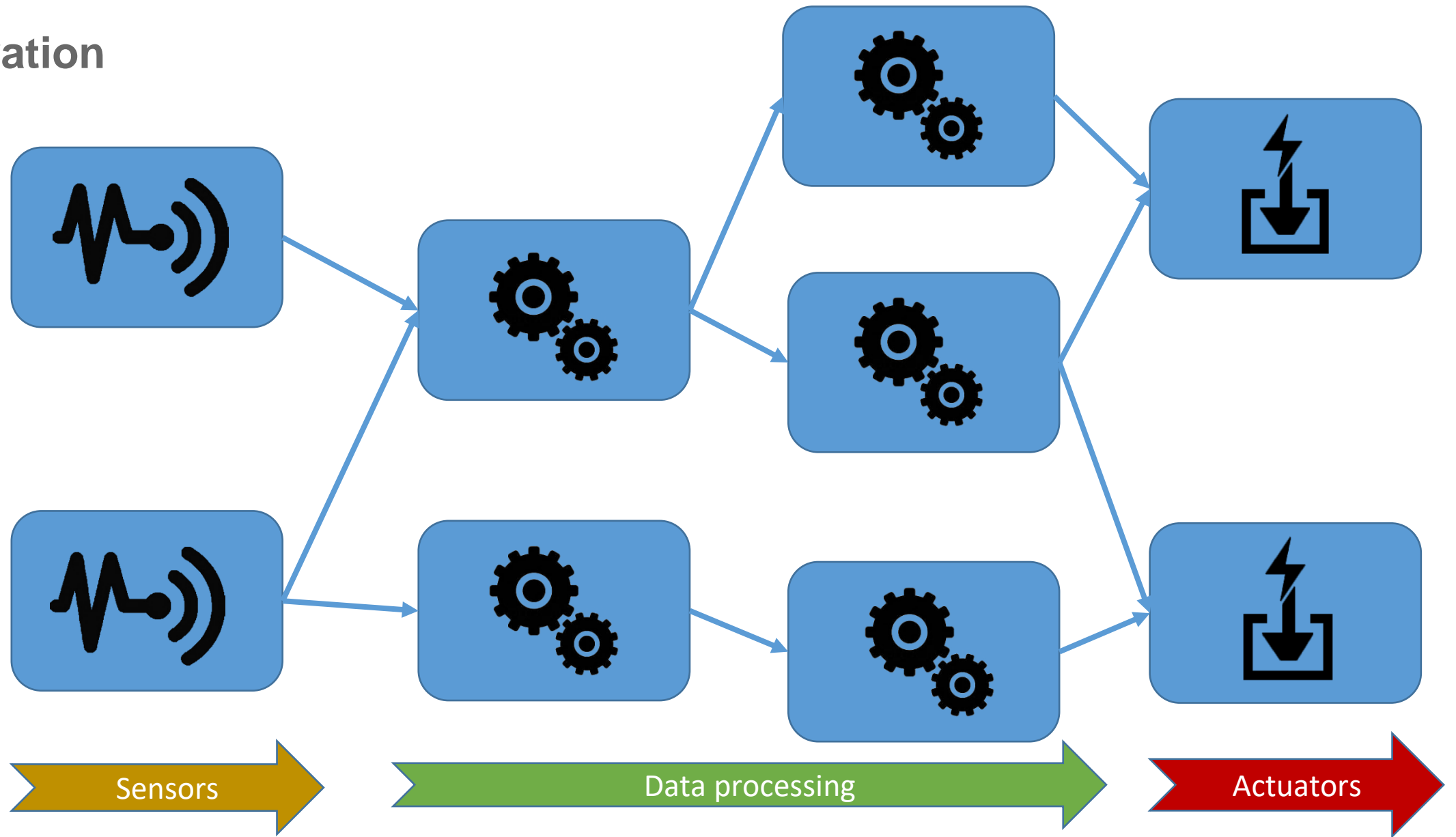
Tobias Franz, Ayush M. Nepal, Zain A. H. Hammadeh, Dr. Olaf Maibaum, Dr. Andreas Gerndt, Daniel Lütke



Knowledge for Tomorrow

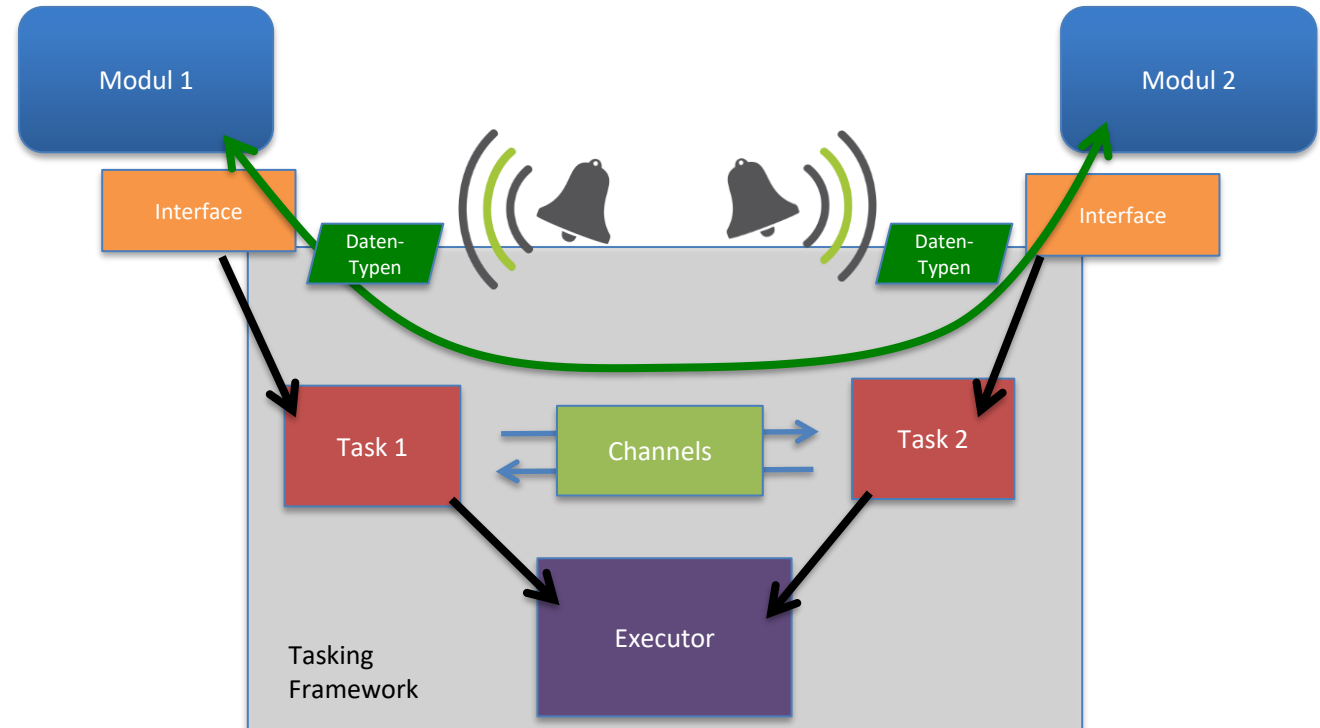


# Motivation



# 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>

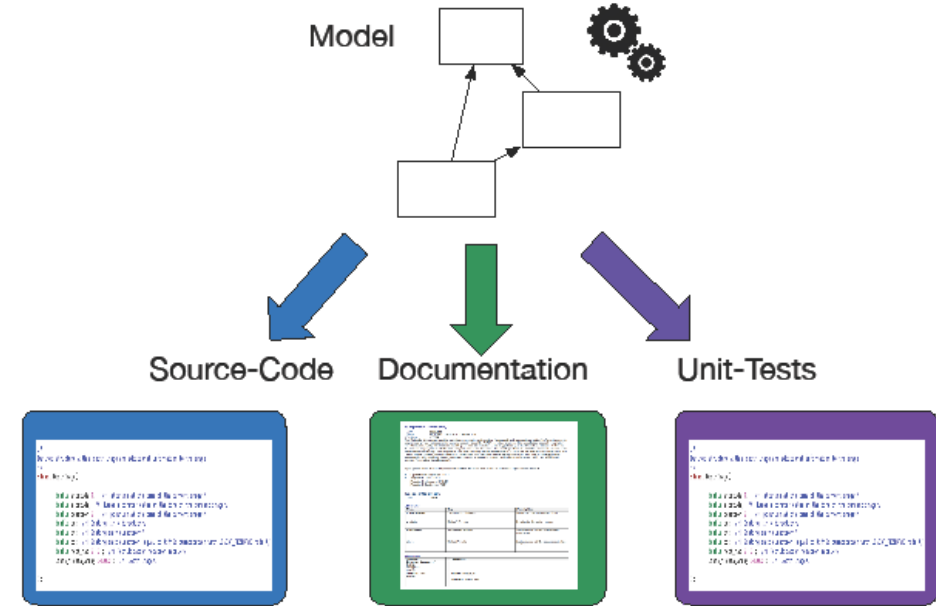
Z. A. H. Hammadeh, T. Franz, O. Maibaum, A. Gerndt, and D. Ludtke, "Event-driven multi-threading execution platform for real-time on-board software systems," in *15th annual workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pp. 29–34, Juli 2019

➡ 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.

**➔** NASA's 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/Pia\\_19808-main\\_tight\\_crop-monday.jpg](https://upload.wikimedia.org/wikipedia/commons/3/30/Pia_19808-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



taste

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 Proceedings*, 2013

UML customization versus domain-specific languages

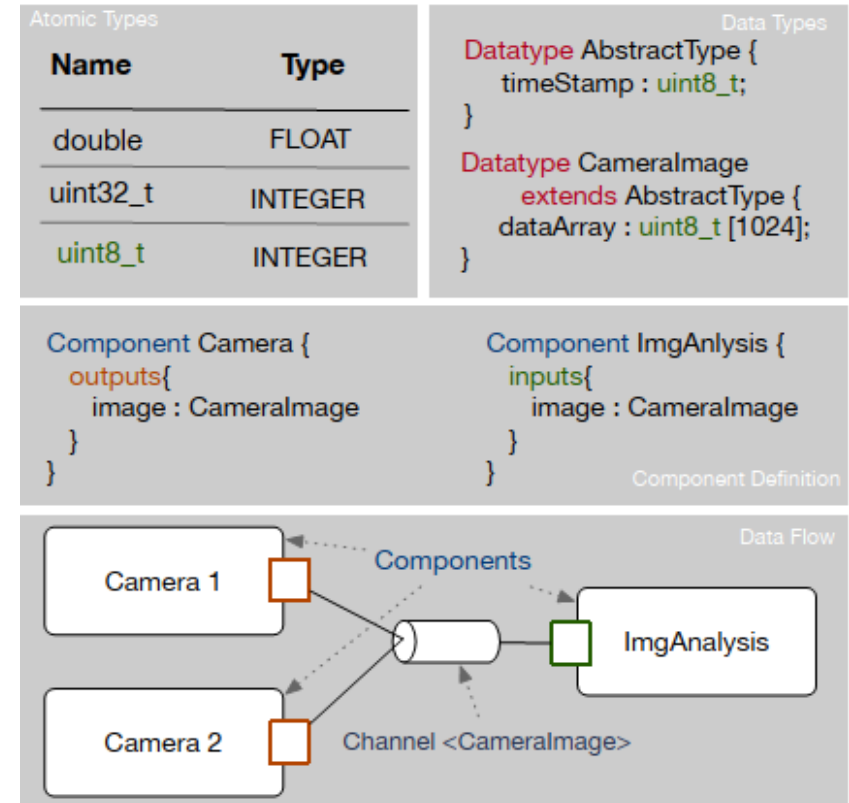
- Difficulties with SysML and UML for domain experts
- High learning effort and ambiguity
- Posed question: "whether it is better to develop a new DSL [...] iteratively, or start with a general-purpose modeling language and restricting it until it fits the specific use"

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

Section for: referenceframes - ReferenceFrameDefinition

Name	
↳ NED	

Basic Definitions

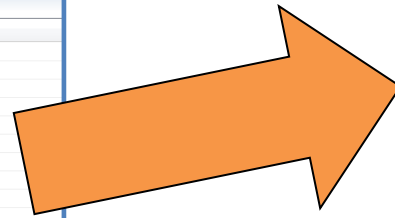
Add ReferenceFrameDefinition Remove ReferenceFrameDefinition Drill-Down Export to Excel

---

Section for: basicTypeDefinitions - BasicTypeDefinition

Name	valueType	size	source	customSource
↳ double	↳ FLOAT=2 []	32.0 8 []	3200 No Value	3200 false
↳ uint32_t	↳ INTEGER=1 []	32.0 4 []	3200 stdint.h	3200 false
↳ uint8_t	↳ INTEGER=1 []	32.0 1 []	3200 stdint.h	3200 false

Add BasicTypeDefinition Remove BasicTypeDefinition Drill-Down Export to Excel



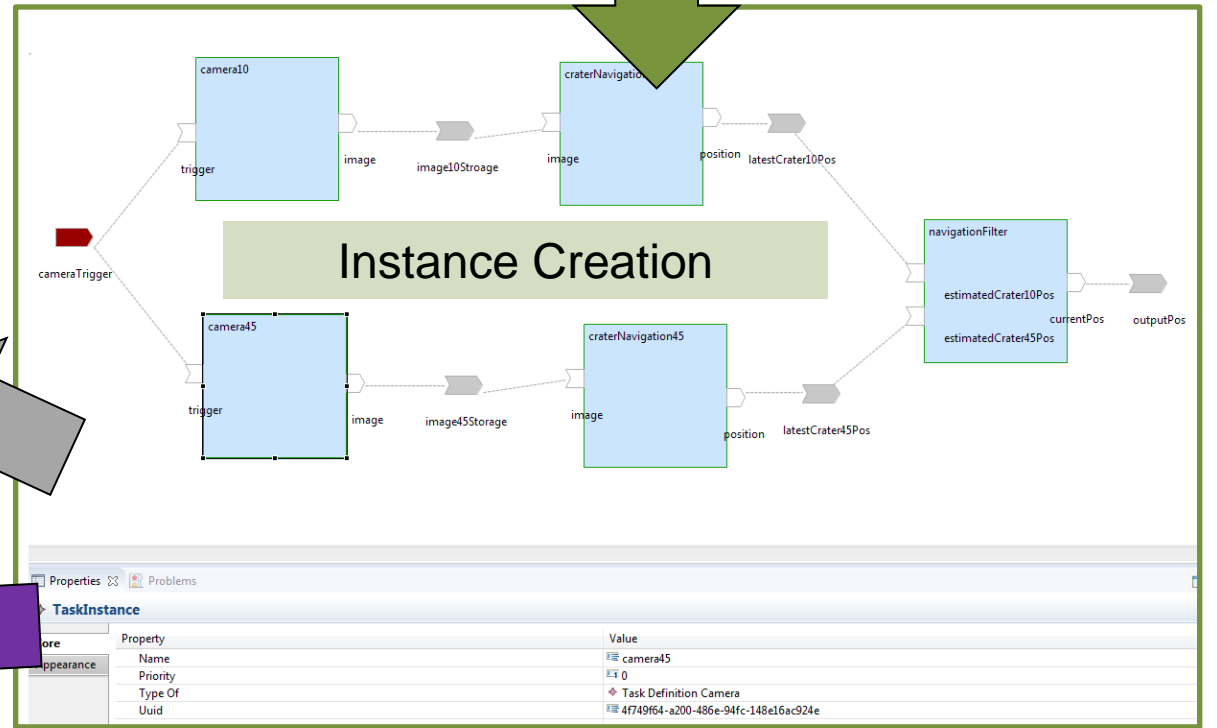
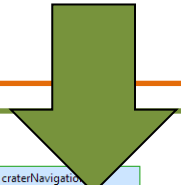
## Type Definitions

```

DataType NavigationState extends AbstractType{
    intProp : uint8_t;
    position : double[3, name = x]{
        referenceFrame : NED;
    };
    velocity : double[3, name =x]{
        referenceFrame : NED;
        unit : mPs;
    };
}
    
```

```

Task NavigationFilter{
    inputs{
        estimatedCrater10Pos : Position;
        estimatedCrater45Pos : Position;
    }
    outputs{
        currentPos : NavigationState;
    }
    parameters{
        componentState : ComponentState;
        startVelocity : double[3,name=x]{
            referenceFrame : NED;
            unit : mPs;
        };
    }
}
    
```



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 12<sup>th</sup> System Analysis and Modelling Conference*, pp. 55–62, 2020

SCOSA  
Node-Mapping

Generation / Deployment

```

class AbstractNavigationFilter
{
public:
    class Inputs
    {
    public:
        DataType::FocusClass *Focuser;
        DataType::Position *positionCrater;
        DataType::Position *position3DMapping;
        DataType::AbstractPosition *referenceFrame;
        DataType::AbstractData *startData;
        DataType::AbstractData *latestData;
        DataType::Position *position3DMatch;
    };
    class Outputs
    {
    public:
        unsigned int *errorCode;
        DataType::AbstractPosition *NavigationData;
        OutputCode *errorCode();
    };
    class Parameters
    {
    public:
        double *g(); /* Initial Coverage Ratio Disposal! */
    };
}
        
```

1.6 LidarImage

Type	Array	Name	Description
uint8_t	320000	data.array	
double	1	offset.lidarImage	
double	1	scale.lidarImage	
double	1	timestamp	

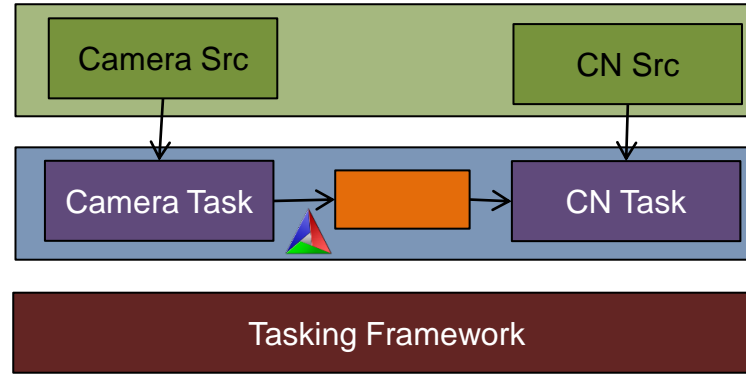
1.7 LandingSites

Type	Array	Name	Description
int	1	count.landingSites	
double	1	timestamp	
LandingSite	30	landingSites	

1.8 CameraImage

Type	Array	Name	Description
double	1	timestamp	The timestamp of the input image
uint8_t	1048576	data.array	Raw Array, containing the image intensity values as 8 bit unsigned int.

# Artifact Generation



Component Layer Stubs generated

Interaction Layer generated

Execution Platform

- Templates for source code and documentation

1.4 Lidelange

Type	Area	Name	Description
Unit	Area	lidelange	
Unit	Area	lidelange	
Unit	Area	lidelange	
Unit	Area	lidelange	

1.7 LandingSite

Type	Area	Name	Description
Unit	Area	landingSite	
Unit	Area	landingSite	
Unit	Area	landingSite	

1.8 ControlRange

Type	Area	Name	Description
Unit	Area	controlRange	
Unit	Area	controlRange	
Unit	Area	controlRange	

Documentation

- Levels of Extensibility

- Model Extensions
- Template Customization
- Generation Gap Pattern

```

class SynchTaskMessageChannel {
    Channel SynchTaskMessageChannel : CUSTOM {
        static size : INTEGER;
        dynamicParam : INTEGER;
    }
}
                
```

Property	Value
dynamicParam	10
size	40

```

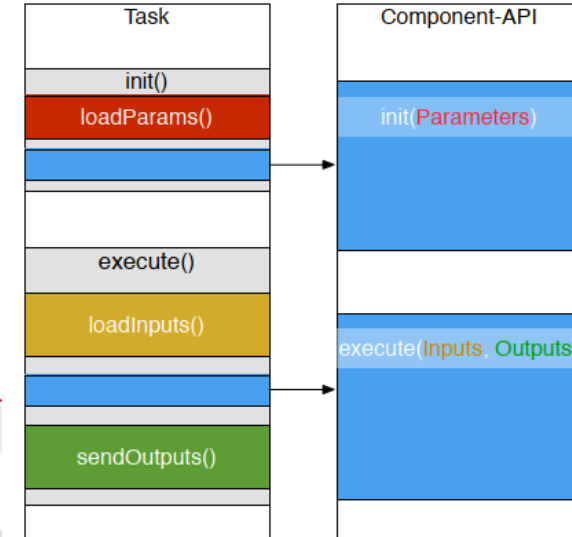
template <typename T, int SIZE>
class ASynchTaskMessageChannel: public AChannel<T> {
public:
    ASynchTaskMessageChannel(int arg_dynamicParam)
        : AChannel<T>()
        , m_dynamicParam(arg_dynamicParam){}
    ...
protected:
    int m_dynamicParam;
}
                
```

Instantiation with static Parameter: (Template)

```
SynchTaskMessageChannel<CameraImage, 40> image10Storage;
```

Instantiation with dynamic Parameter: (Constructor)

```
image10Storage(10),
```





# 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

The screenshot displays the TML development environment with several key components:

- Task Graph:** A central diagram showing two camera components (camera1, camera2) connected to FIFO buffers (tFIFO) and then to analyzer components (analyser1, analyser2). A timer component provides a 'trigger' signal to the cameras.
- Component Details:**
  - Task Camera:** Shows input/output definitions and a data type:
 

```

                    Name      valueType  size  source
                    double   float     8     No Value
                    uint32_t  integer   4     stdint.h
                    uint8_t   integer   1     stdint.h

                    DataType CameraImage {
                    dataArray : uint8_t[1024][800];
                    }
                    
```
  - Channel:** Shows properties for a custom channel:
 

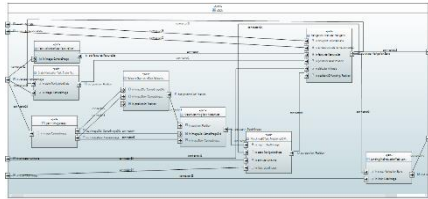
```

                    Channel SynchronTaskMessageChannel : CUSTOM {
                    static size : INTEGER;
                    dynamicParam : INTEGER;
                    }
                    
```
  - Code Generation:** Shows the C++ template code for the channel:
 

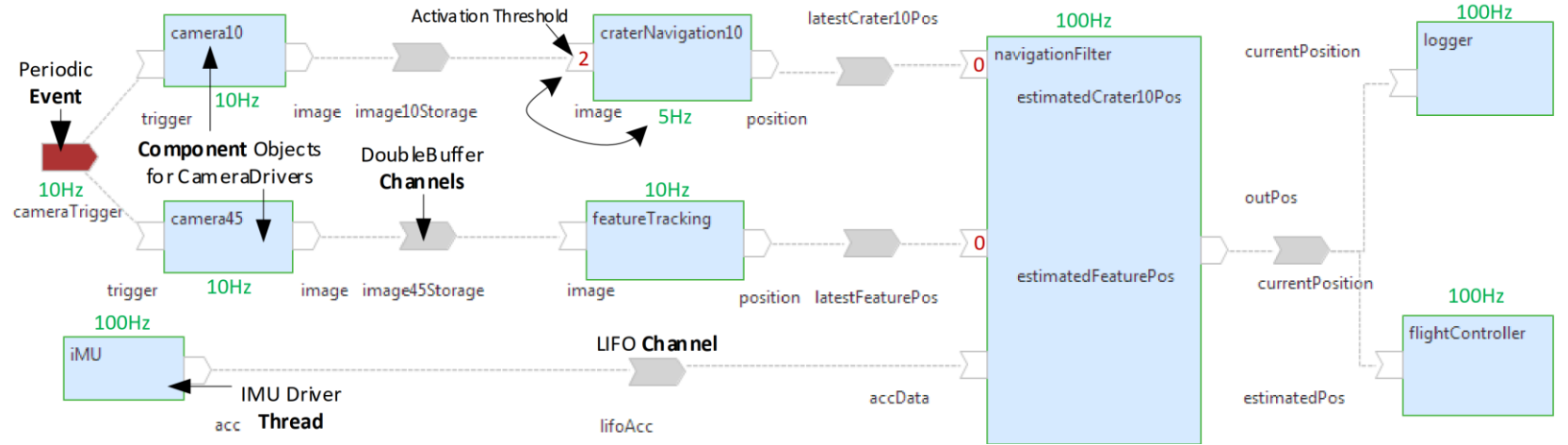
```

                    template <typename T, int SIZE>
                    class ASynchronTaskMessageChannel: public AChannel<T> {
                    public:
                    ASynchronTaskMessageChannel(int arg_dynamicParam)
                    : AChannel<T>()
                    , m_dynamicParam(arg_dynamicParam){}
                    ...
                    protected:
                    int m_dynamicParam;
                    
```
- Diagrammatic Views:**
  - SubscriberList:** Shows a 'Channel' and an 'Event' (with 'Periodical Activation') connected to 'Component1' and 'Component2'.
  - Threshold/Status:** Shows signal levels (3, 4, 3) and status indicators for 'Component1' and 'Component2'.

# Case Study

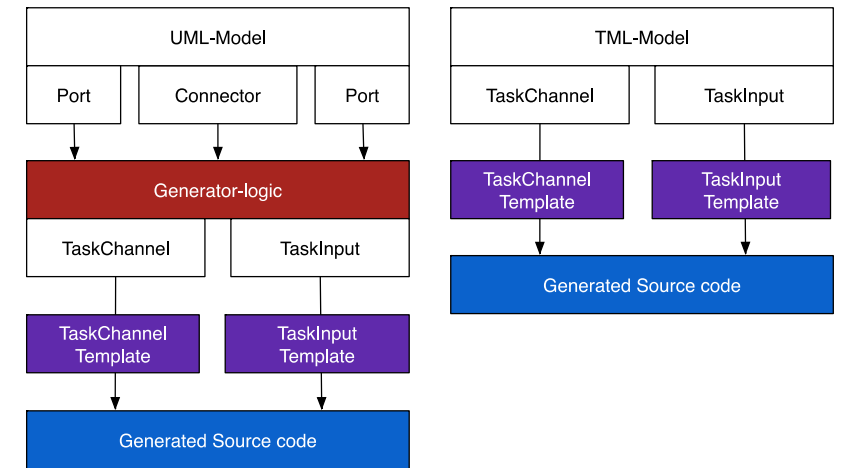


T. Franz, D. Ludtke, O. Maibaum, and A. Gerndt, "Model-based software engineering for an optical navigation system for spacecraft," *CEAS Space Journal*, vol. 10, no. 2, pp. 147–156, 2018



## 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



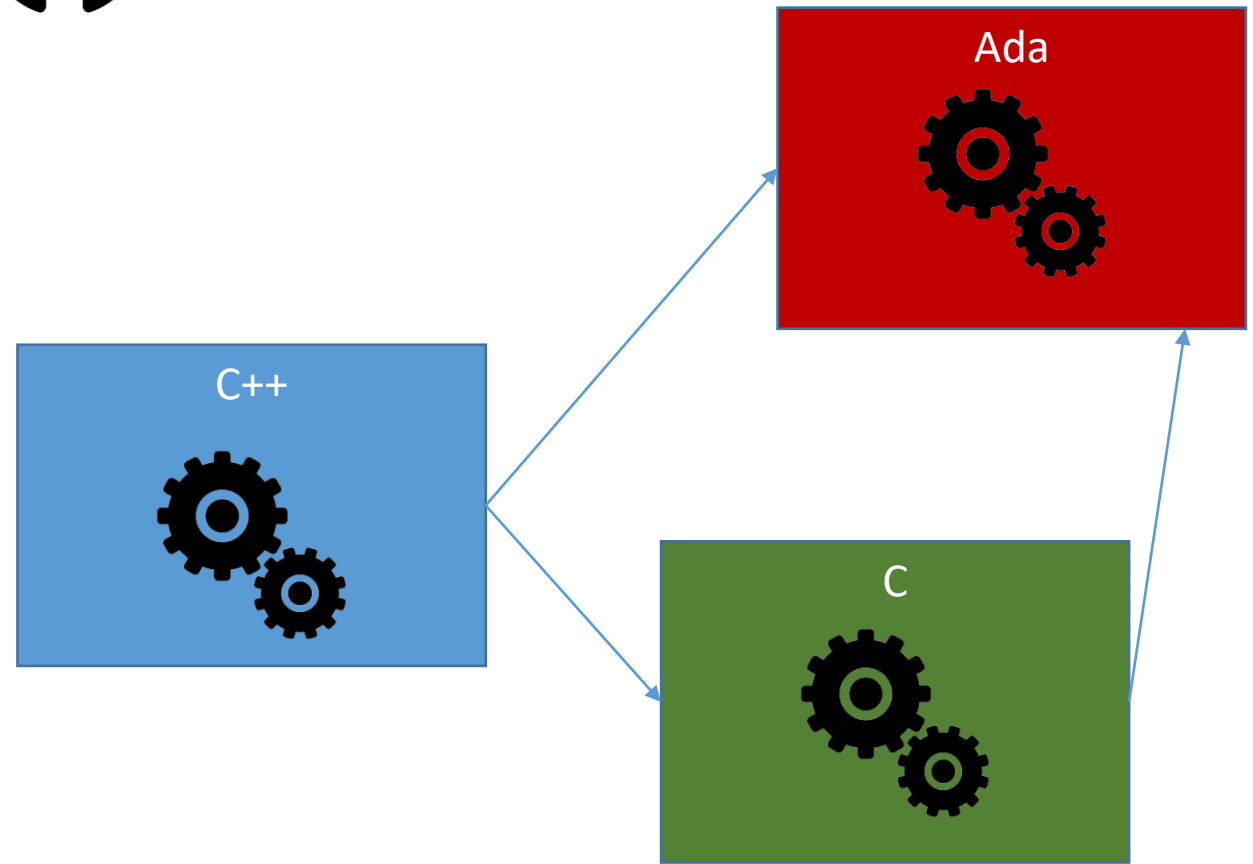
## Outlook and future work

- 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

Name	valueType	size	source
double	IEEE754 FLOAT	8	No Value
uint32_t	INTEGER	4	stdint.h
uint8_t	INTEGER	1	stdint.h

```

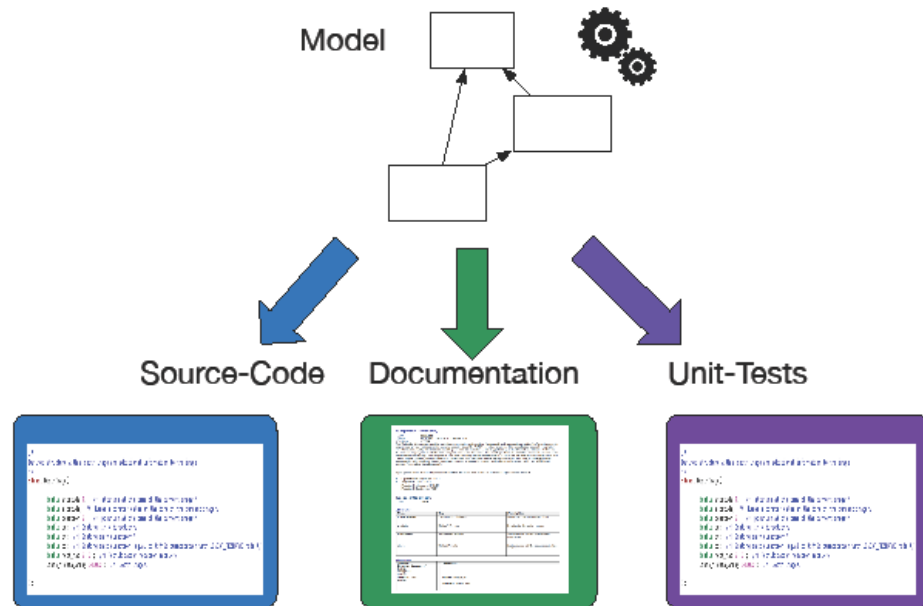
DataType CameraImage {
    dataArray : uint8_t[1024][800];
}

Task Camera{
    inputs{
        trigger;
    }
    outputs{
        image : CameraImage;
    }
    parameters{
        aperture;
    }
}
    
```

**Task Graph**

```

graph LR
    timer -- trigger --> camera1
    timer -- trigger --> camera2
    camera1 -- image --> tfifo1
    camera2 -- image --> tfifo2
    tfifo1 -- tFIFO --> analyzer1
    tfifo2 -- tFIFO --> analyzer2
    
```



Planned Release on Github

