PaTaS

Quality Assurance in Model-Driven Software Engineering for Spacecraft

Kilian Hoeflinger, Jan Sommer, Ayush Nepal, Olaf Maibaum, Daniel Lüdtke
DLR - Simulation and Software Technology
Contact: kilian.hoeflinger@dlr.de

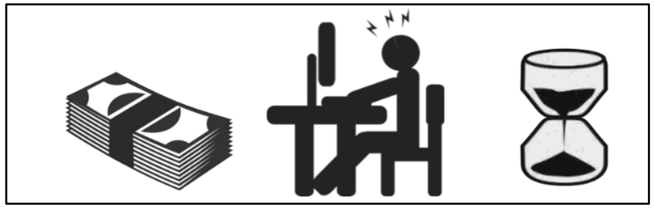
DLR esa

Motivation and Outline of the Study

Motivation

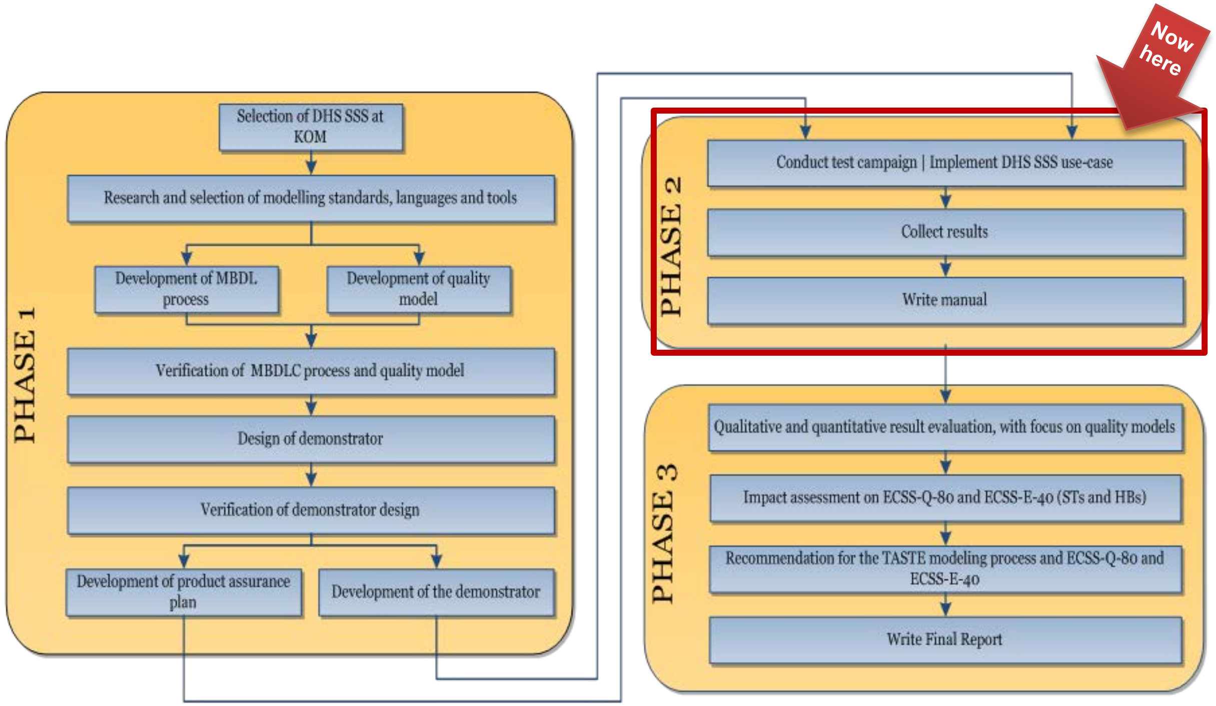
- Improve S/W PA for model-driven development by measuring model quality with model metrics
- Early evaluation/detection of:
  - Flaws in specification
  - Functional requirements
  - Non-functional requirements (Maintainability, Reusability etc.)

You save…



Outline of the PATAS study

- One year study
- Development of product quality model with software and model metrics
- Implementation of an end-to-end model-driven software engineering lifecycle demonstrator, based on TASTE [6]
- Evaluation of the demonstrator with mission-critical parts of the onboard S/W of a satellite mission, being modelled and subsequently coded
- Improvement of model-driven S/W PA at ESA



Workflow of PATAS study

PaTaS

MBSD Lifecycle Demonstrator Design

Workflow

1. Define computation independent PUS, communication data and communication test model
2. Refine platform independent model in TASTE Interface View
3. Generate code skeletons from TASTE Deployment View
4. Test-driven implementation of OBSW

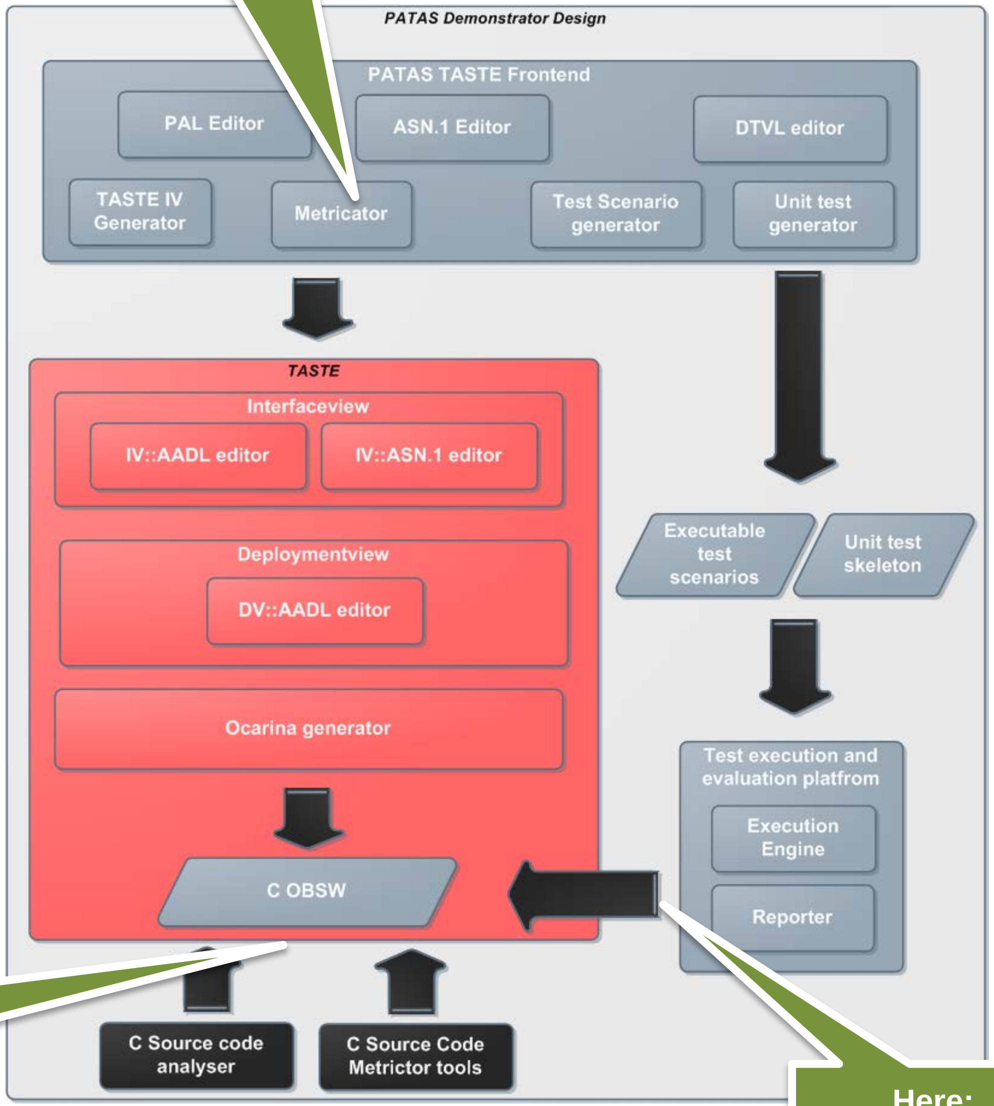
Applied standards and methodologies

- ECSS PUS [9], OMG Model-driven Architecture standard [7], Model-based testing taxonomy [8], TASTE inherent standards [10]

Use case

- Parts of ACS, ONS and CDH of an actual small satellite mission of DLR
- Targeting lab quality (x86), no flight H/W
- Project lifecycle from S/W-PDR to S/W-CDR

Here: Collecting Model Metrics



PaTaS demonstrator design

Here: Collecting S/W metrics

Here: Collecting functionality reports

DLR esa

Model Metrics
Overview

ID	Model Metric Name	Applicable Sub-characteristic
MM-01*	Adherence to Modelling Conventions	Modularity, Completeness, Self-descriptiveness, Conciseness, Balance, Correctness
MM-02	**Interaction Diagram Coverage**	**Completeness, Balance**
MM-03*	Model Type Instance Weight	Complexity, Balance
MM-04*	Model Coupling	Modularity, Complexity, Balance
MM-05*	Model Type Instances per Use Case	Modularity, Complexity, Balance, Conciseness
MM-06*	Use Cases per Model Type Instance	Modularity, Complexity, Balance, Conciseness
MM-07*	Lines of model code	Complexity, Balance, Self-descriptiveness
MM-08*	Model comment frequency	Complexity, Balance, Self-descriptiveness
MM-09*	Low of Functional Cohesion	Modularity, Complexity, Balance
MM-10	**Module Fan-in / Fan-out**	**Modularity, Balance**

PaTaS model metrics overview

*Detailed description can be found in the appendix of this presentation

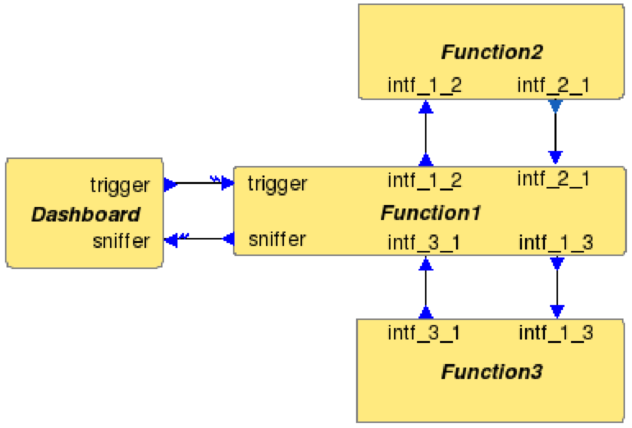
Model Metrics
Interaction Diagram Coverage MM-02

Description
- Evaluation of coverage by counting of model type instances of a system model, used in a behavioral test model [2]

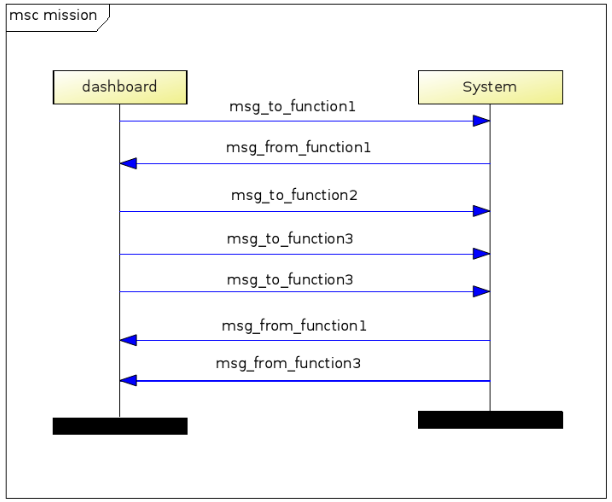
Purpose
- Supports requirements implementation coverage and structural coverage S/W metric with test case generation
- Provides support to increase the fault tolerance of the S/W, by showing the coverage of fault handling components
- A high *IDC* value can indicate low functional cohesion
- *IDC* = 0 raises the question about the general purpose of the component

A	B	C	D
>=1	>=1	>=1	>=1

*IDC threshold per criticality level**



Small TASTE IV example system



TASTE MSC example

Model Type Instance	IDC Value
Function1	3
Function2	1
Function3	3

IDC results for above example

*Threshold might change as study is still ongoing and all results are not yet available

PaTaS

Model Metrics
Interaction Diagram Coverage MM-02

All Mission Scenarios	Used Model Type Instance	Usage value
PUS Application	ACS	208
PUS Service	ACS-Service-1	125
PUS SubService	s1-1-acceptance	52
PUS SubService	s1-2-acceptance-failure	12
PUS SubService	s1-3-execution-started	9
PUS SubService	s1-4-execution-started-failure	2
PUS SubService	s1-7-execution-complete	39
PUS SubService	s1-8-execution-complete-failure	11
PUS Service	ACS-Service-2	36
PUS SubService	s2-2-parameter-load-command	12
PUS SubService	s2-5-parameter-dump-command	12
PUS SubService	s2-6-parameter-dump-report	12
PUS Service	ACS-Service-3	15
PUS SubService	s3-2-defining-new-diagnostics-parameter-reports	1
PUS SubService	s3-4-clear-diagnostics-parameter-report-definitions	1
PUS SubService	s3-7-enable-diagnostics-parameter-report-generation	1
PUS SubService	s3-8-disable-diagnostics-parameter-report-generation	1
PUS SubService	s3-11-request-diagnostic-parameter-report-definitions	1
PUS SubService	s3-18-select-periodic-diagnostic-parameter-report-generation-mode	1
PUS SubService	s3-20-select-filtered-diagnostic-parameter-report-generation-mode	1
PUS SubService	s3-129-select-triggered-diagnostic-parameter-report-generation-mode	1
PUS SubService	s3-2-aocs-diagnostic-report	1
PUS SubService	s3-26-aocs-diagnostic-data-report	1
PUS SubService	s3-12-diagnostic-parameter-report-definitions-report	1
PUS SubService	s3-25-aocs-housekeeping-report	4
PUS Service	ACS-Service-8	32
PUS SubService	s8-1-aocs-tc-set-pwr	8
PUS SubService	s8-1-aocs-tc-unlock-pwr	8
PUS SubService	s8-1-aocs-tc-reset-tm-budget	1

*Excerpt of IDC metric result of ACS**

Increased usage of service 1

Reveals fault tolerance test case coverage

All model type instances are used at least once

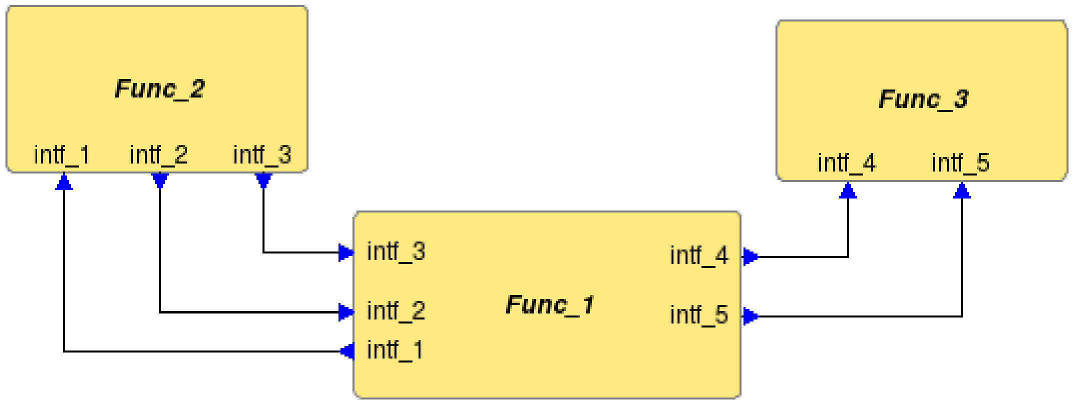
* All results are preliminary and represent the state at the 22. August 2017

DLR esa

Model Metrics
Module Fan-in / Fan-out MM-10

Description
- Fan-in: local flows into a model type instance [4]
- Fan-out: local flows out of the specific model type instance
- Expressiveness can be further improved when combined with other metrics, e.g. Model Type Instance Weight

Purpose
- High *FIN or FOUT* indicates high complexity of the system and monolithic design, making it hard to maintain and reuse
- Complexity of a procedure depends on the complexity of the control flow in the procedure and of the procedure's connection



Small TASTE IV example function

Model Type Instance	FIN Value
Func_1	2
Func_2	1
Func_3	2

FIN result

Model Type Instance	FOUT Value
Func_1	3
Func_2	2
Func_3	0

FOUT result

A	B	C	D
<20	<25	<25	<30

*Module FIN / FOUT threshold per criticality level**

*Threshold might change as study is still ongoing and all results are not yet available

Model Metrics
Module Fan-in / Fan-out MM-10

PUS_Application	Module Fan-In
ACS	11.0
ONS	15.0
CDH	8.0

PUS_Application	Module Fan-Out
ACS	21.0
ONS	32.0
CDH	3.0

PUS_Service	Module Fan-In
ACS-Service-1	0.0
ACS-Service-2	2.0
ACS-Service-3	8.0
ACS-Service-8	11.0
ONS-Service-1	0.0
ONS-Service-3	0.0
ONS-Service-8	32.0
ONS-Service-150	0.0
ONS-Service-5	0.0
CDH-Service-1	0.0
CDH-Service-3	0.0
CDH-Service-8	3.0

PUS_Service	Module Fan-Out
ACS-Service-1	6.0
ACS-Service-2	1.0
ACS-Service-3	4.0
ACS-Service-8	0.0
ONS-Service-1	6.0
ONS-Service-3	2.0
ONS-Service-8	3.0
ONS-Service-150	3.0
ONS-Service-5	1.0
CDH-Service-1	6.0
CDH-Service-3	1.0
CDH-Service-8	1.0

*Result of Fan-in and Fan-out metrics**

Reveals Service 1 only has TM capabilities

Combination reveals high complexity

Combination reveals high complexity

Reveals overall data flow direction and correlating message sizes

PUS_Application	Module Type Instances Weight
ACS	392.0
ONS	326.0
CDH	72.0

PUS_Service	Module Type Instances Weight
ACS-Service-1	33.0
ACS-Service-2	15.0
ACS-Service-3	308.0
ACS-Service-8	36.0
ONS-Service-1	33.0
ONS-Service-3	72.0
ONS-Service-8	172.0
ONS-Service-150	44.0
ONS-Service-5	5.0
CDH-Service-1	33.0
CDH-Service-3	3.0
CDH-Service-8	36.0

*Result of Model Type Instance Weight metric**

* All results are preliminary and represent the state at the 22. August 2017

Quality Model and Mapping of Metrics

- Quality Model is a factor-criteria-metrics model
- **Mapping formulae for model to S/W metrics**
  - ***Nested*** - A software metric is nested in a model metric, determining and subsequent handling special points of interest
  - ***Complementary*** – Combination of model and S/W metric to derive a quality verdict
  - ***Independent*** – Model and S/W metric are alone standing
  - Further formulae possible
- Also combinations of model and model metrics or S/W and S/W metrics possible

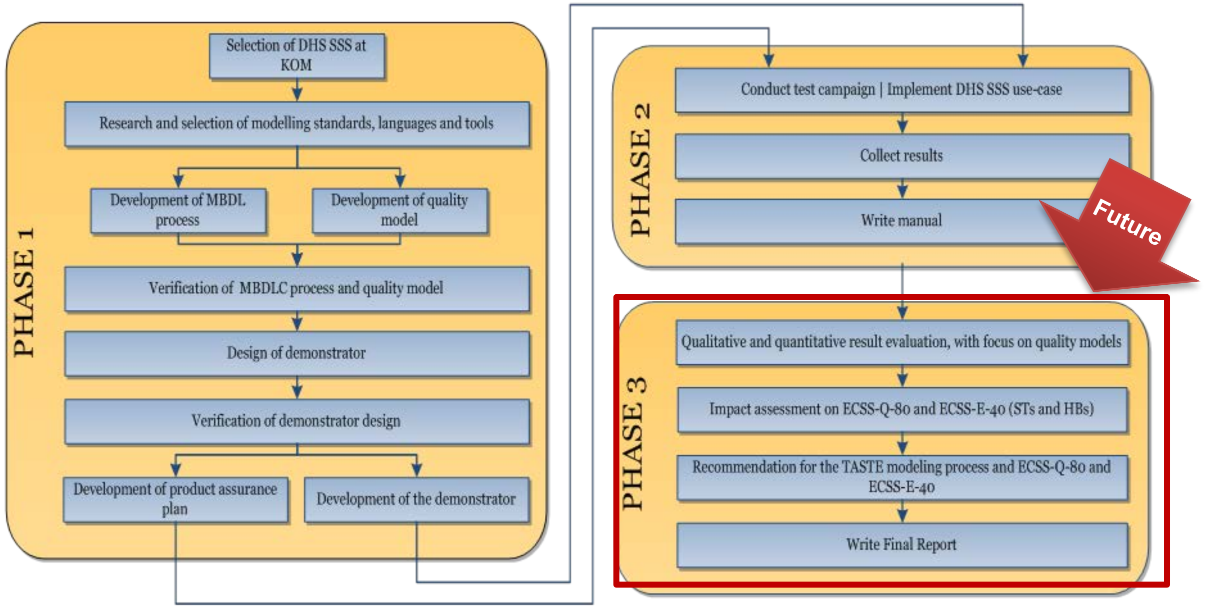
Req. ID	Characteristic	Sub-characteristic	Model Metric (ID)				Software Metric (ID)				Mapping Formulae
			A	B	C	D	A	B	C	D	
REU-01	Reusability	Modularity	Model Coupling (MM-04)				Cyclomatic Complexity (SWM-04)				Nested
			<10	<15	<20	<25	<10	<10	<15	<20	
MAN-06	Maintainability	Modularity	Model Type Instances per Use Case (MM-05)				Modularity Size Profile (SWM-06)				Complementary
			<9	<11	<13	<15	<5	<5	<5	<7	

Excerpt of an example quality model with model and software metrics

Conclusion

Remarks

- ***Finding optimal thresholds*** for model metrics takes further evaluation/usage
- The ***balance*** of the model metric result is already a ***strong indicator***
- ***Model metrics have to be tailored*** under consideration of the used standards and modelling methods/tools
- ***Model metrics*** shall also ***measure requirements/constraints coverage***
- ***Automatic evaluation*** mostly requires the usage/definition of a computation independent model, targeting the problem domain
- ***Single-view model metrics are not meaningful*** when conducting model-driven development, as the source code can also be evaluated with existing tools

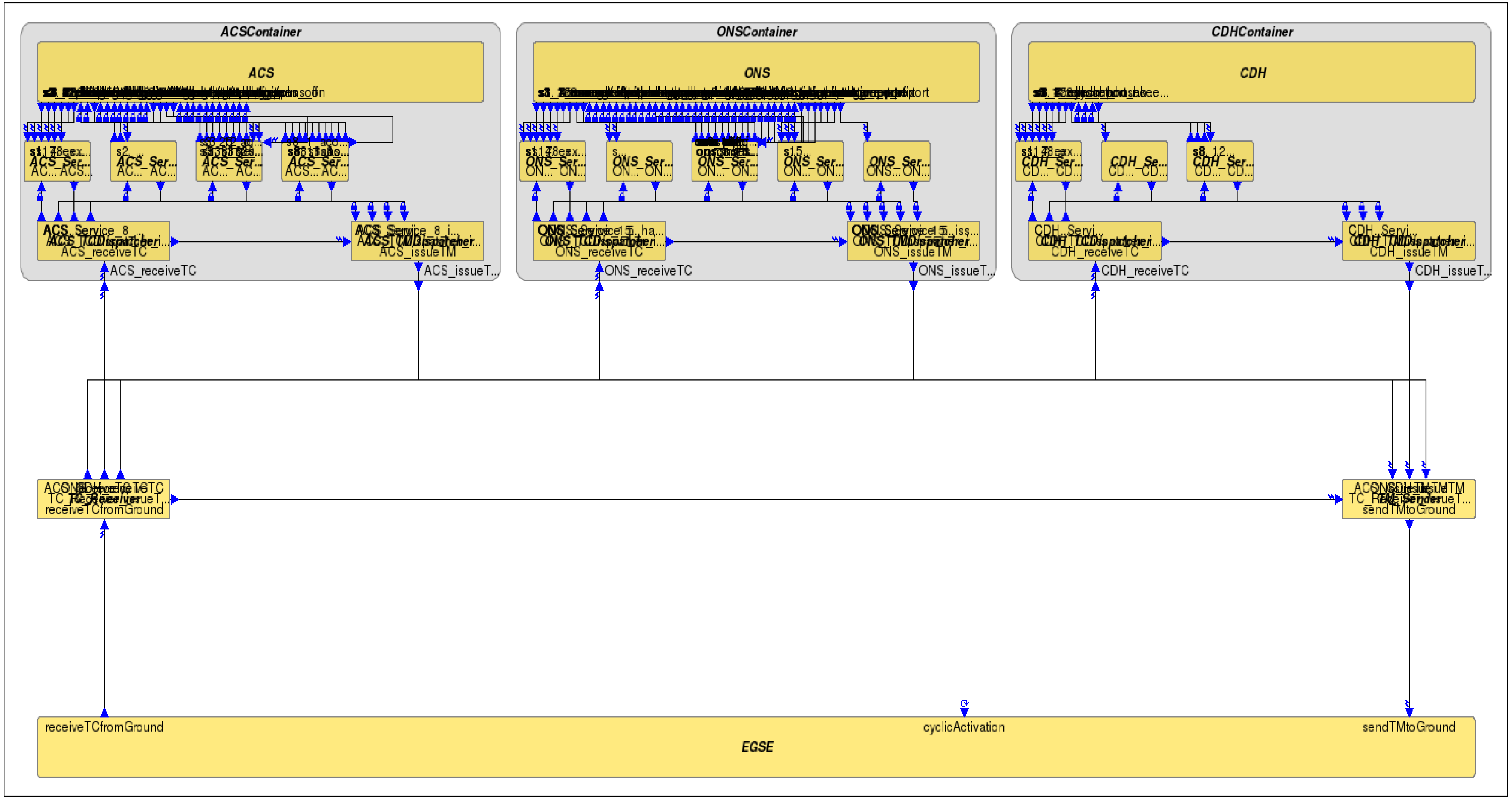


Workflow of PATAS study

Further Steps

- Try to ***define a generic model metric notation*** to describe (model) metrics
- ***Investigate the collected use-case data*** to find beneficial model and S/W metric combinations
- Write ***final recommendations*** for ECSS standards

Thank you for your Attention



Screenshots of TASTE Interface View use-case model

PaTaS

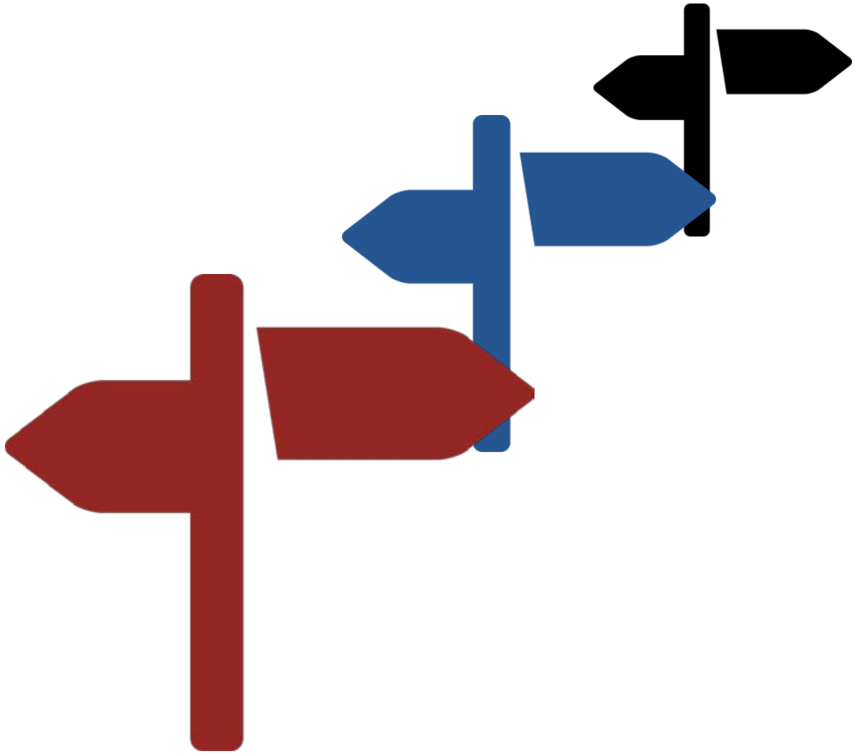
References

[1]	S. R. Chidamber and C. F. Kemerer, “A Metrics Suite for Object Oriented Design,” IEEE Trans. Softw. Eng., vol. 20, no. 6, pp. 476-493, #jun# 1994.
[2]	C. F. J. Lange and M. R. V. Chaudron, “Managing Model Quality in UML-Based Software Development,” in Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice, Washington, DC, USA, 2005.
[3]	J. Muskens, M. Chaudron and C. Lange, “Investigations in applying metrics to multi-view architecture models,” in Proceedings. 30th Euromicro Conference, 2004., 2004.
[4]	S. Henry and D. Kafura, “Software Structure Metrics Based on Information Flow,” IEEE Transactions on Software Engineering, Vols. SE-7, no. 5, pp. 510-518, Sept 1981.
[5]	B. a. L. C. F. J. a. D. S. a. C. M. R. V. Du Bois, “A Qualitative Investigation of UML Modeling Conventions,” in Models in Software Engineering: Workshops and Symposia at MoDELS 2006, Genoa, Italy, October 1-6, 2006, Reports and Revised Selected Papers, T. K{\"u}hne, Ed., Berlin, Heidelberg, Springer Berlin Heidelberg, 2007, pp. 91-100.
[6]	M. a. C. E. a. D. J. a. S. A. a. T. T. Perrotin, “TASTE: A Real-Time Software Engineering Tool-Chain Overview, Status, and Future,” in SDL 2011: Integrating System and Software Modeling: 15th International SDL Forum Toulouse, France, July 5-7, 2011. Revised Papers, I. a. O. I. Ober, Ed., Berlin, Heidelberg, Springer Berlin Heidelberg, 2012, pp. 26-37.
[7]	J. M. Joaquin Miller, MDA Guide Version, 2003.
[8]	J. Zander, Model-based testing for embedded systems, Boca Raton, FL: CRC Press, 2012.
[9]	ECSS E-70-41C APRIL 2016, SPACE ENGINEERING - TELEMETRY AND TELECOMMAND PACKET UTILIZATION (HTTP://WWW.ECSS.NL)
[10]	TASTE framework (https://taste.tuxfamily.org)

Appendix

Content

- Adherence to Modelling Conventions MM-01
- Model Type Instance Weight MM-03
- Model Coupling MM-04
- Model Type Instances per Use Case MM-05
- Use Cases per Model Type Instance MM-06
- Lines of model code MM-07
- Model comment frequency MM-08
- Low of Functional Cohesion MM-09
- Overview of used S/W metrics



Model Metrics
Adherence to Modelling Conventions MM-01

Description
- Guidelines for the model, like naming conventions, consistency rules etc. [5]
- Such conventions are equivalent to coding guidelines
- Have to be adapted to the modelling tools, as some conventions are fulfilled by default (e.g. each message corresponds to method in TASTE)
- Difficult to get tool-support for automatic evaluation

Purpose
- Increases maintainability as well as reusability
- Especially good for graphical modelling languages, as it creates overview

A	B	C	D
100%	100%	100%	100%

*Adherence to Modelling Conventions threshold per criticality level**

Sub Characteristic	ID	Convention	Checked [Date]
Conciseness	1	Every model type instance has to have a unique name.	TBD
Conciseness	2	The name of model type instance should explain its functionality.	TBD
Balance	3	All use-cases should cover a similar amount of functionality.	TBD
Completeness	4	All model type instances that interact with other model type instances shall be covered by at least one sequence diagram.	TBD
Completeness	5	Each use case must be described by at least one sequence diagram.	TBD
Consistency	6	Each message must correspond to a method (operation).	TBD
…	…	…	…

Example model convention list

*Threshold might change as study is still ongoing and all results are not yet available

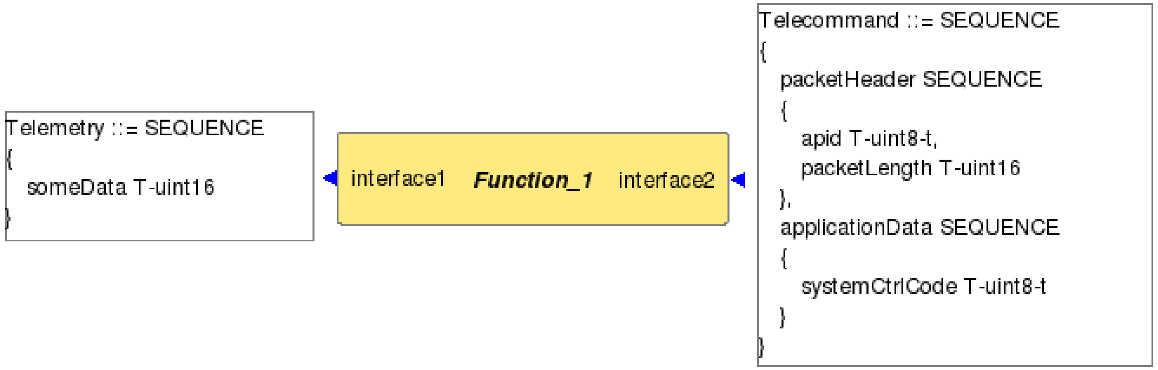
Model Metrics
Model Type Instance Weight MM-03

Description

- Accumulation of all model type instances, “owned” by a model type instance, considering a model type specific weight factor, determined by any indicator of complexity [1]

Purpose

- A high *MTIW* value, indicates complexity, which complicates testing, maintaining and reusing
- Threshold value depends on the used indicator to determine complexity of the “owned” model type instances
- Could be improved when considering also the range of an ASN.1 datatype



Small TASTE IV example function with correlating ASN.1 interface parameters

Specific model element	Weight-factor ω_k
Sequence/Choice (ASN.1)	2
Simple Datatype (ASN.1)	1

Applied weight–factor and formula

Model Type	A	B	C	D
PUS Application	<250	<350	<450	<550
PUS Service	<50	<70	<90	<110

*MTIW threshold per criticality level**

Interfaces	*MTIW* value of Function_1
Interface1	2+1 = 3
Interface2	2+(2+1+1)+(2+1) = 9
Total	12

MTIW result

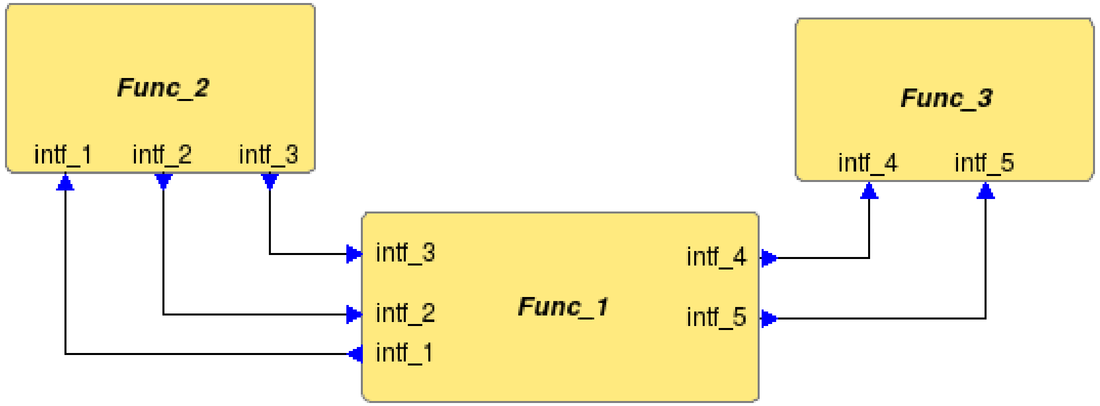
*Threshold might change as study is still ongoing and all results are not yet available

PaTaS

Model Metrics
Model Coupling MM-04

Description
- Coupling of a model type instance is determined by the count of other coupled model type instances [1]
- A coupling weight can be introduced in case communication can be differentiated in the model

Purpose
- Evaluation of complexity, reveals complexity hot spots for later software implementation
- High coupling results in monolithic misbalanced model/software, hindering re-usage and effective maintenance, due to side effects among components



Small TASTE IV example function

A	B	C	D
<10	<15	<20	<25

*Model Coupling threshold per criticality level**

Func_1	
Connected Model Type Instance	Model Coupling Value
Func_2	3
Func_3	2
Total	5

Model Coupling result investigating Func_1

*Threshold might change as study is still ongoing and all results are not yet available

DLR esa

PaTaS

Model Metrics
Model Type Instances per Use Case MM-05

Description

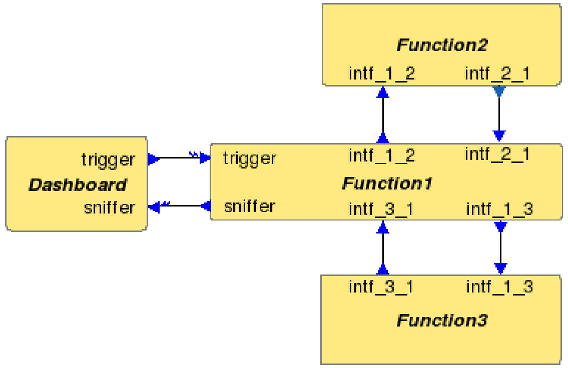
- Amount of model type instances per use case has to be counted. Here, a use case is the implementation of a test for a software requirement [3]

Purpose

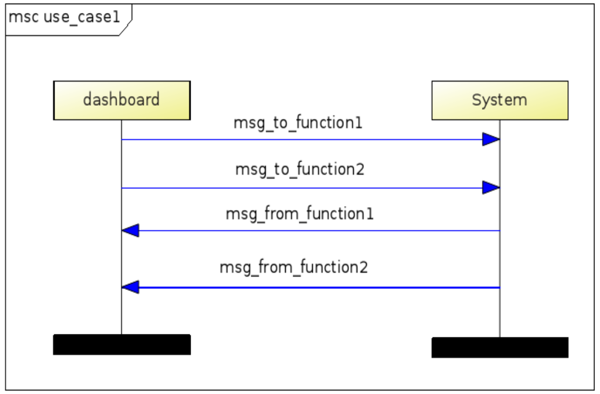
- Determines complexity, modularity, balance and conciseness of the system
- In case of a high *MTIpUC* count, a change in the requirement has a great impact on the system design and implementation; And it indicates low functional cohesion, as functionality is spread over many model elements.
- It also determines how well balanced and detailed the requirements are and how good the specification fits to the requirements

A	B	C	D
<9	<11	<13	<15

*MTIpUC threshold per criticality level**



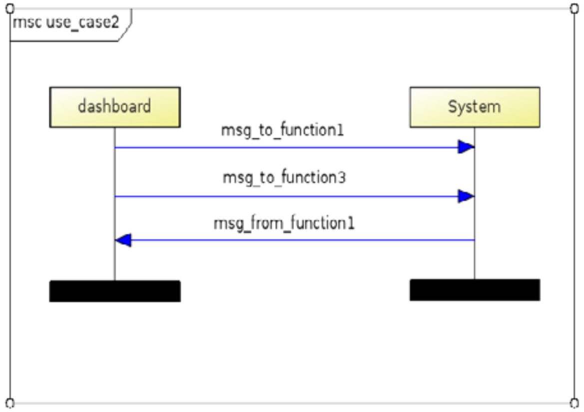
Small TASTE IV example system



TASTE MSC use_case1

Use Case	MTIpUC Value
use_case1	2
use_case2	2

MTIpUC result



TASTE MSC use_case2

DLR esa

*Threshold might change as study is still ongoing and all results are not yet available

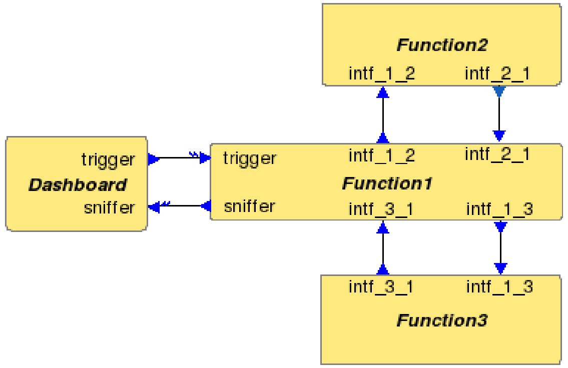
Model Metrics
Use Cases per Model Type Instance MM-06

Description
- Counting the amount of use cases per model type instance. Here, a use case is the implementation of a test for a software requirement [3]

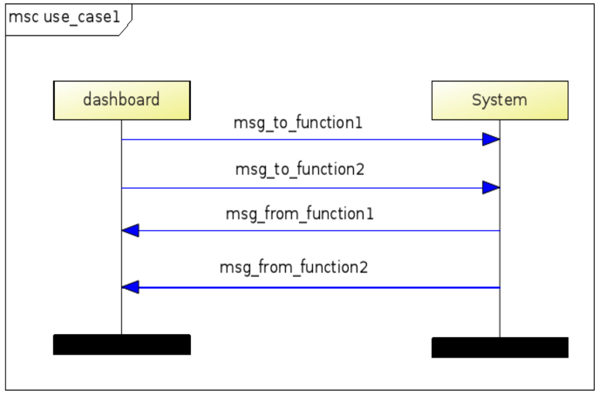
Purpose
- A high *UCpMTI* count, the cohesion of the model type instance might be low and errors in the software implementation might have a broad effect on the overall system
- Helps to focus on heavily used components of the onboard software, which can then be further analyzed manually or with specific software complexity metrics

A	B	C	D
$1 \leq X \leq 10$	$1 \leq X \leq 12$	$1 \leq X \leq 14$	$1 \leq X \leq 16$

*UCpMTI threshold per criticality level**



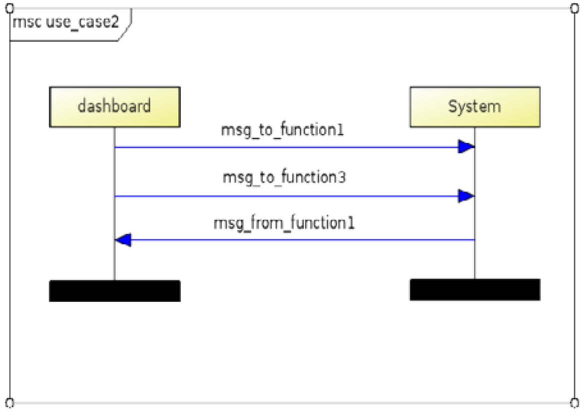
Small TASTE IV example system



TASTE MSC use_case1

Model Type Instance	UCpMTI Value
Function1	2
Function2	1
Function3	1

UCpMTI result



TASTE MSC use_case2

*Threshold might change as study is still ongoing and all results are not yet available

Model Metrics
Lines of model code MM-07

Description

- Counting the number of model lines per model file (excluding comments and blank lines)

Purpose

- Indication of the model complexity, balance and self-descriptiveness
- Too large model files reduce the overview and therefore maintainability and reusability

A	B	C	D
<300	<350	<400	<500

*LOMC threshold per criticality level**

```
mission-scenario sun-sensor-use-case is
    /**
     * The logical power state of a sun sensor shall be set to on by a telecommand.
     */
    use-case setSunSensorStateVirtualON is
        UC-SunSensor.tc-aocs-s8-1-tc-set-pwr
        implies
        UC-SunSensor.tm-acs-1-1-acceptance-sss-on future UC-SunSensor.tm-acs-1-7-execution-complete-sss-on must hold
    end

    /**
     * The logical power state of a sun sensor shall be set to off by a telecommand.
     */
    use-case setSunSensorStateVirtualOFF is
        UC-SunSensor.tc-aocs-s8-1-tc-unlock-pwr
        implies
        UC-SunSensor.tm-acs-1-1-acceptance-sss-off future UC-SunSensor.tm-acs-1-7-execution-complete-sss-off must hold
    end

end

mission-scenario aocs-housekeeping-use-case is
    /**
     * The AOCS shall provide a consistent set of housekeeping data on request from the data handling subsystem.
     */
    use-case getRegularACSHousekeepings is
        always UC-AOCSHousekeeping.tm-acs-3-25-aocs-housekeeping-report must hold
    end

end

mission-scenario ons-housekeeping-use-case is

    /**
     * The ONS shall report selected data in the regular housekeeping data.
     */
    use-case getregularONSHousekeepings is
        UC-ONSHousekeeping.tc-ons-8-1-req-hk-rp implies
        UC-ONSHousekeeping.tm-ons-1-1-acceptance-reg-hk future
        UC-ONSHousekeeping.tm-ons-1-7-execution-complete-reg-hk future
        UC-ONSHousekeeping.tm-ons-3-25-housekeeping-report must hold
    end

    /**
     * The ONS shall deliver extended housekeeping data on request by a telecommand.
     */
    use-case getExtONSHousekeepings is
        UC-ONSHousekeeping.tc-ons-8-1-ons-reg-ext-hk implies
        UC-ONSHousekeeping.tm-ons-1-1-acceptance-reg-ext-hk future
        UC-ONSHousekeeping.tm-ons-1-7-execution-complete-reg-ext-hk future
        UC-ONSHousekeeping.tm-ons-3-128-extended-housekeeping-report must hold
    end

end
```

Example Data Testing and Verification Language model

*Threshold might change as study is still ongoing and all results are not yet available

PaTaS

Model Metrics
Model comment frequency MM-08

Description

- Ratio between number of model comment lines and lines of model code plus number of model comment lines
- In case of model-driven S/W development with readable source code as output, comments could be transferred additionally

Purpose

- To increase the self-descriptiveness, maintainability, reusability of the model

A	B	C	D
>30%	>30%	>30%	>20%

*Model comment frequency threshold per criticality level**

```
/**
 * ONBORAD NAVIGATION SYSTEM MANAGEMENT
 *
 * Description:
 * The On-board Navigation System (ONS) Management
 * handles the estimation of current states of the satellite
 * including its position and velocity. The states can be
 * estimated from either real-time GPS messages or two-lines
 * elements uploaded from ground. In addition, this application
 * provides also the services to manipulate GPS devices and to
 * synchronize between its GPS time and the on-board time
 * (space-craft elapsed time).
 */

application ONS with ID = 640 is

    service ONS-Service-1 with ID = 1 is

        /**
         * Telecommand has been accepted by the service. The execution has not yet started.
         */
        tmMessage s1-1-acceptance with ID = 1 ofType MSG-ONS-SERVICE-1.TM-ons-1-1-acceptance

        /**
         * Telecommand has been rejected by the service. The execution has not been started
         * and no internal data has been altered.
         */
        tmMessage s1-2-acceptance-failure with ID = 2 ofType MSG-ONS-SERVICE-1.TM-ons-1-2-acceptance-failure

        /**
         * The telecommand execution has been successfully started
         */
        tmMessage s1-3-execution-started with ID = 3 ofType MSG-ONS-SERVICE-1.TM-ons-1-3-execution-started

        /**
         * An error occurred during the start of the execution of the telecommand. Internal
         * data may have been changed. The execution was aborted.
         */
        tmMessage s1-4-execution-started-failure with ID = 4 ofType MSG-ONS-SERVICE-1.TM-ons-1-4-execution-started-failure

        /**
         * The telecommand has been successfully executed.
         */
        tmMessage s1-7-execution-complete with ID = 7 ofType MSG-ONS-SERVICE-1.TM-ons-1-7-execution-complete

        /**
         * The telecommand execution ended with an error.
         */
        tmMessage s1-8-execution-complete-failure with ID = 8 ofType MSG-ONS-SERVICE-1.TM-ons-1-8-execution-complete-failure

    end

    service ONS-Service-3 with ID = 3 is

        /**
         * ONS Housekeeping report contains the status of ONS modules and ONS internal counters.
         */
        tmMessage s3-25-ons-housekeeping-report with ID = 25 ofType MSG-ONS-SERVICE-3.TM-ons-3-25-housekeeping-report
```

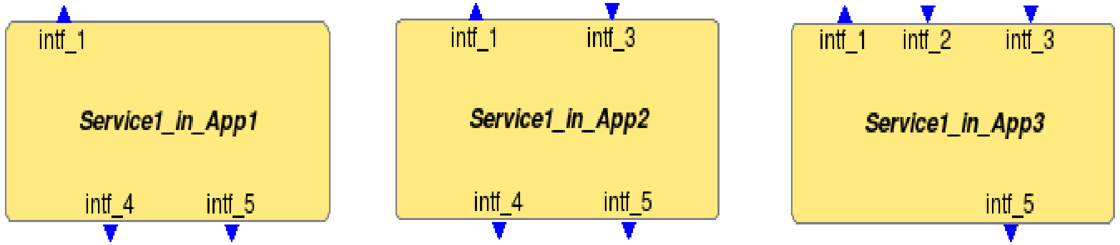
Example PUS model with comments

*Threshold might change as study is still ongoing and all results are not yet available

Model Metrics
Low of Functional Cohesion MM-09

Description
- Determining the similarity of instances of a specific type, by investigating the model type instance's interfaces, through which it is communicating [1]

Purpose
- High similarity of model type instances of a specific type, indicates high functional cohesion of that type
- High functional cohesion increases the maintainability and reusability
- *LoFC* values can range between 1.0 and 2.0



Small TASTE IV example, displaying three service instances

A	B	C	D
$1 \le LoFC_s \le 1.3$	$1 \le LoFC_s \le 1.4$	$1 \le LoFC_s \le 1.5$	$1 \le LoFC_s \le 1.6$

*LoFC threshold per criticality level**

Application #	A	B	X=A/B
1	5	3	1.67
2	5	4	1.25
3	5	4	1.25
Mean (X)			**1.39**

LoFC result

*Threshold might change as study is still ongoing and all results are not yet available

Software Metrics
Overview of the measured software metrics

ID	S/W Metric Name	Affected Sub-characteristic
SWM-01	Adherence to coding standards	Completeness, Correctness
SWM-02	Structural coverage	Completeness, Reliability Evidence
SWM-03	Requirements implementation coverage	Correctness
SWM-04	Cyclomatic Complexity	Modularity
SWM-05	Number of call levels	Balance, Modularity
SWM-06	Modularity size profile	Complexity, Modularity
SWM-07	Code comment density	Complexity

PaTaS software metrics overview

Preliminary Model Metrics Formula

Model Metric	Description of Formula
Low of Functional Cohesion (*LoFC*)	The value shall be low to indicate a low LoFC. $$LoFC_s = \text{Mean}\{\sum_{k=0}^{i} X_k\}$$ With s being a specific model type, with k interfaces and, X = A/B A = number of model type instances of this model type B = number of used interfaces in this specific model type instance Desired value: A=B
Model comment frequency	X= A/B, where: A = number of comment lines in the model; B = Lines of Model Code (*LoMC)* + (number of lines of comments in model) = total number of lines excluding blank lines
Model Type Instance Weight	The weight of the modules is determined by the weight of its contained model type instances. Weight, as a representative of complexity, can be estimated by the specific model type. Therefore needed weight-factors are given, here an ASN.1 example: • Sequence/Choice (ASN.1) = 2 • Simple Datatype (ASN.1) = 1 The computation for the *MTIW* value is: $$MTIW = \sum_{k=1}^{n} \omega_k$$ *n = number of weighted model type instances of a module* ω_k *= weight-factor of model type*

PaTaS

Preliminary Model Metrics Formula
Cont'd

Model Metric	Description of Formula
Module Fan-in	It is necessary to count the used references of a specific model type instance owned by a module. Consider a module *A* and a global set of *i* references to model type instances *R* of a specific model type; Let $\{UR_j\}$ = set of used model type instances of a module A_j. Then, the Fan-in value FIN_x for a specific module A_x *is*: $$FIN_x = \{\sum_{k=0}^{i} R_k \mid R_k = \{\begin{matrix} 1\ if\ R_k \in UR_x \\ else\ 0 \end{matrix}\}\}$$
Module Fan-out	Consider a module *A* and a global set of *i* references to model type instances *R* ; Let $\{ UR_j \}$ = set of referenced model type instances of a specific module A_j. Then, the Fan-out value $FOUT_x$ for a specific module A_x *is*: $$FOUT_x = \{\sum_{k=0}^{i} R_k \mid R_k = \{\begin{matrix} 1\ if\ R_k \in UR_x \\ else\ 0 \end{matrix}\}\}$$
Interaction Diagram Coverage	Consider a model type instance *E* of a specific model type and *n* mission scenario *MS*; Let $\{ UE_j \}$ = set of used model type instances of mission scenario MS_j. There are *n* such sets $\{UE_1\}, \ldots, \{UE_n\}$. Let $P = \{(UE_i, E) \mid UE_i \cap E \neq \emptyset \}$ If all *n* sets $\{ UE_1\},\ldots,\{ UE_n \}$ are $\emptyset$ then let $P = \emptyset$. $$IDC_E = \sum_{k=1}^{n} \lvert P_k \rvert$$ *with n being all mission scenarios;* Informal, this means that the usage of model type instances of a specific type, have to be counted in all mission scenarios.

DLR esa

PaTaS

Preliminary Model Metrics Formula
Cont'd

Model Metric	Description of Formula
Model Type Instances per Use Case (MTIpUC)	Consider a use case U and *n* model type instances E of a specific type of a model; Let *{ UE_j }* = set of used model type instances for a use case U_j. $MTIpUC_u = \{ \sum_{k=1}^{n} E_k \mid E_k = \begin{cases} 1\ if\ E_k \in UE_f \\ else\ 0 \end{cases} \}$ *With u being a specific use case;*
Use Cases per Model Type Instance (UCpMTI)	Consider a model type instance E of a specific type and n use cases U; Let *{ UU_j }* = set of used use cases per model type instance E_j. $UCpMTI_e = \{ \sum_{k=1}^{n} U_k \mid U_k = \begin{cases} 1\ if\ U_k \in UU_e \\ else\ 0 \end{cases} \}$ *with e being a specific model type instance;*
Model Coupling	Coupling value X_A between a model type instance and other instances of a specific type. *X_A = number couplings between the instance and the instances of a specific type*
Lines of Model Code	LoMC = (total number of model lines) – (comment and blank lines)

DLR esa