Tim Weilkiens

Variant Modeling with SysML

www.model-based-systems-engineering.com

- MBSE4U Booklet Series -

Variant Modeling with SysML

- MBSE4U Booklet Series -

Tim Weilkiens

This book is for sale at http://leanpub.com/vamos

This version was published on 2016-04-12

ISBN 978-3-9817875-4-2

/MBSE4U

MBSE4U - Tim Weilkiens is a publishing organization for books about MBSE. They are intended to be regularly updated to align the content with the highly dynamic systems engineering domain.

Contents

About MBSE4U

MBSE4U - Tim Weilkiens is my publishing organization for MBSE books. The main focus is eBooks that are regularly updated to follow the dynamic changes in the MBSE community and the markets. Printed books are also available.

MBSE4U has published

- Tim Weilkiens. SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML. http://leanpub.com/sysmod. 2015. (ISBN 978-3-9817875-1-1 (PDF), 978-3-9817875-2-8 (ePub), 978-3-9817875-3-5 (MOBI), 978-3981787504 (Print))
- Tim Weilkiens. Variant Modeling with SysML. http://leanpub.com/vamos. 2016.
- Tim Weilkiens, The New Engineering Game. http://leanpub.com/the-new-engineering-game. Planned 3Q 2016.
- Tim Weilkiens, MBSE Craftsmanship. http://leanpub.com/mbse-craftsmanship. Planned 1Q 2017.

About me

I am a managing director of the German consulting company oose, a consultant and trainer, and active member of the OMG and INCOSE community. I have written sections of the initial SysML specification and I am still active in the ongoing work on SysML. I am involved in many MBSE activities and you can meet me on several conferences about MBSE and related topics.

As a consultant I have advised a lot of companies in different domains. The insights into their challenges are one source of my experience that I share in my books and presentations.

I have written many books about modeling including *Systems Engineering with SysML* (Morgan Kaufmann, 2008) and *Model-Based System Architecture* (Wiley, 2015). I am the editor of the pragmatic and independent MBSE methodology SYSMOD – the Systems Modeling Toolbox (We15).

You can contact me at tim@mbse4u.com and read my blog about MBSE at www.model-based-systems-engineering.com.

History and Outlook

This chapter gives a brief overview about the version history of the variant modeling approach and looks forward to future plans.

Version History

- 1.0 Initial comprehensive publication of VAMOS.

Outlook

- More analysis of other variant modeling approaches
- Provide more support of the profile for common SysML modeling tools.
- Explicit description of the alignment with other variant modeling approaches.
- Provide a guideline and template for report generation

Preface

From the early beginning of SysML I consult organizations from different domains to apply model-based systems engineering (MBSE) methodologies. Very often one of the first questions after the basics were understood is how to model variants with SysML. There is a high and increasing demand for variant modeling. The SysML does not provide first class model elements for variant concepts. However general SysML model elements like the *Generalization* relationship could be used to build a system model with variants.

I have developed a method how to model variants with SysML and have presented it in different publications, conference talks, or blogs. The original source of the concepts provided in this book is the MBSE methodology SYSMOD (We15). I first described the variant modeling method in the book *Systems Engineering with SysML/UML* (We08, We14). This book is a more comprehensive and recent description and is now the official source of the variant modeling method. Because it is much easier to talk about something that has a name I call the VAriant MOdeling method for SysML *VAMOS*.

The concept of VAMOS is not a complete new creation of mine. It is more a new configuration and adaption of existing concepts for the usage with SysML. Chapter 4 covers some other variant modeling concepts.

A disadvantage of classical books is the low frequency of updates. That was my main motivation to publish this book about variant modeling with SysML with my own publishing organization MBSE4U. I continuously update the method based on feedbacks and experiences of industrial projects and changes in the context like other new variant methods or changes of SysML.

A disadvantage of self-published books is the missing quality gate of a traditional publisher. There is no copy-editor who, for example, proves the correct usage of the English language - in particular if the author is not a native speaker like me - or if the line of arguments makes sense for the readers.

I appreciate any feedback on the book. Be it on the content or on my English skills. You can reach me by email: tim@mbse4u.com.

Thanks to Keith Smith who gave me a lot of feedback about SYSMOD, FAS and VAMOS.

I thank my colleagues at my company oose, in particular Axel Scheithauer and Stephan Roth for long profound discussions about MBSE.

I thank NoMagic for their support. I have created the SysML diagrams in this book with their modeling tool Cameo Systems Modeler.

Finally I would like to thank you for buying this book. The money is well spent. Now you have a description how to model variants with SysML. And I have some money to finance the infrastructure to provide MBSE goodies to the MBSE community like my blog www.model-based-systems-engineering.com.

If you need MBSE training or consulting services feel free to contact me. My company - the consultancy oose - provides MBSE trainings and coaching, for example to introduce MBSE in your organization.

Tim Weilkiens, April 2016.

1. Introduction

All along products exist in different variants. In recent years organizations face more and more the challenge to provide a huge set of product variants. The industry moves from the phase of mass production to a new phase of mass customization.

There are many reasons to manage variants in a model: A product line, a customized product or different designs for trade studies. Typically a single variant of a system affects only a few parts of the system. It is a slight derivation from the initial system. However it is not possible to quantify the number or level of detail that is allowed to vary to be still a variant of a system and not a new system.

You cannot measure abstraction and you cannot give an objective metric for the deviation of variants. The benefit must be larger than the effort to manage a complex variant model.

A car as well as an aircraft could be a variant of a transportation system. However in most cases it makes no sense in practice to handle a car and an aircraft as variants of a transportation system with all the appropriate relationships in a single system model. The common parts of a car and an aircraft are too abstract.

The description of variants is a sophisticated task. It is already challenging to create a good description of a single system. Every variation adds another dimension to a multi-dimensional system model. For example, the engine could be a variation of a car system with three possible variants: diesel, electric, or hybrid engine. Next variation could be the chassis with the variants small, deluxe, cabriolet. Now you can combine the variants, for example a car with diesel engine and a small chassis or a car with a hybrid engine and a deluxe chassis, and so on. Any additional variation increases the dimension and the number of potential combinations.

The OMG Systems Modeling Language (SysML)[1] (Sy15) is a general-purpose modeling language for systems engineering applications. It is mainly used for requirements and system architecture specifications. Both are strongly affected by variants. Therefore it is highly valuable to use SysML for the specification of variants.

The book implies knowledge about SysML. If you need more information about SysML I recommend my book *Systems Engineering with SysML/UML* published by Morgan Kaufman (We08, We14) or for experienced readers the SysML specification itself (Sy15).

In the following chapter I give some definitions of variant terms. The next chapter describes the concept for VAriant MOdeling with SysML (VAMOS).

To think out of the box chapter 4 covers other variant modeling concepts. Finally chapter 5 presents the definitions of the VAMOS stereotypes.

The appendix provides additional examples of VAMOS: a forest fire detection and a virtual museum tour system.

[1] OMG SysML is a trademark of the Object Management Group.

2. Variant Modeling Concepts

In this chapter I give the definitions of variability terms how I use them in the context of variant modeling to get a solid fundament. The terms and concepts are conform with common variant concepts presented in publications about variant modeling, for example the Orthogonal Variability Model (OVM) (Po05) (see chapter 4).

Figure 2.1[1] shows the conceptual model of the VAMOS terms. The stereotype *«domainBlock»* represents conceptual elements. It is part of the SYSMOD profile (Wei15). A conceptual model[2] is a helpful, comprehensive, concise and clear description of domain terms.

On the top level I differentiate between the core, the variants, and the configurations.

The core contains all elements that are used in all system configurations. A **core element** is an element of the core and independent of any variant element.

A **variation point** marks a core element of the system as a docking point for a variant element.

A **variant element** only occurs in some configurations and is part of exactly one variant. The variants and the core are orthogonal concepts. The core is independent of the variants. The variants could be stored in a separate physical model.

[1] The colored boxes in the diagram are no SysML model elements. They are geometric figures to emphasize the separate variant concepts. Most modeling tools provide those graphical adornments of SysML diagrams.

[2] In SYSMOD the conceptual model is named domain knowledge model.

A **variant** is a complete set of variant elements that varies the system according to a variation. A variant is also known as a feature of the system.

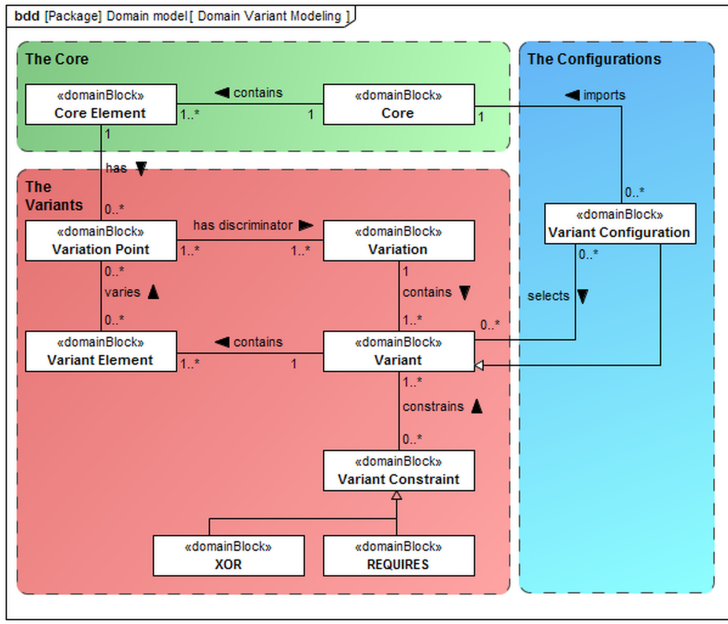

Figure 2.1: Variant Modeling Domain

A **variation** is the discriminator for variants. For example, diesel, electric, and hybrid engines are variants of the variation engine kind.

A variant could again include variations (figure 2.2). The structure of these variations is the same as for the top level variations. This recursive structure makes the variant modeling concept scalable for any size of a system.

The core is a concept relative to the variations on the same level. A variant that again includes variations contains also core elements relative to these variations. But they are variant elements relative

to the variations of the upper level.

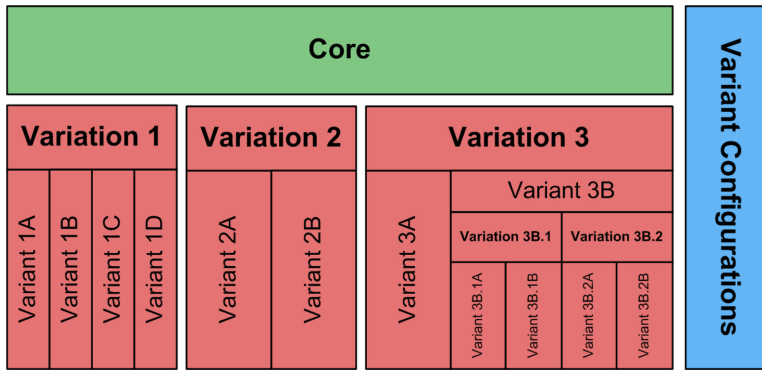

Figure 2.2: Recursive Structure of a Variant Model

A **variant configuration** is a valid set of variants and the core, for example, a car with a hybrid engine and a deluxe chassis. A variant configuration is also a special variant and part of a variation. In our example the variant configuration *Hybrid engine with Deluxe chassis* is part of the variation *eco editions*.

A **variant constraint** specifies rules for a valid set of variants. Two common variant constraints are predefined. XOR is used to exclude a variant if a specific variant is selected. REQUIRES specifies that other variants are required if a specific variant is selected.

The SysML profile for VAMOS presented in chapter 5 implements the concepts.

3. Variant Modeling with SysML (VAMOS)

SysML does not provide explicit built-in language constructs to model variants. Nevertheless SysML is useful to create a model with variability aspects. The VAMOS method presented in this chapter is one option how to model variants with SysML. It uses the profile mechanism of SysML to extend the language with a concept for variant modeling. The chapter VAMOS Stereotypes describes the stereotypes for the variant modeling concepts. The stereotypes extend SysML to model the concepts of variants, variations, variation points, variant elements, variant configurations and variant constraints presented in chapter 2. The VAMOS Stereotypes are part of the SYSMOD profile (We15), although they could be used independently of SYSMOD.

Typically modeling in general has three separate concerns: the method, the language, and the tool (figure 3.1).

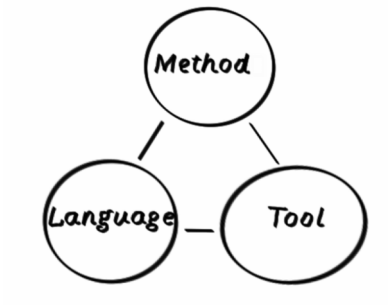

Figure 3.1: The Three Concerns of Modeling

Mapped to the context of VAMOS, VAMOS itself is a method and the VAMOS stereotypes are part of the language. The modeling tool for

VAMOS could be any modeling tool that supports standard SysML.

A great benefit of the VAMOS approach is that there is no need to stress a 3rd party tool for the variant modeling. Certainly a tool with specialized handling of variants has benefits. However the price is another tool including license costs, training, learning yet another modeling language, a tool chain between the SysML tool and the variant tool, a cumbersome engineering environment, and more. It finally depends on your requirements on the variant modeling which approach has the highest value for you. If you have very ambitious requirements, it could makes sense to use a highly specialized tool for variant modeling. Otherwise the value of using VAMOS with a standard SysML modeling tool is higher (figure 3.2).

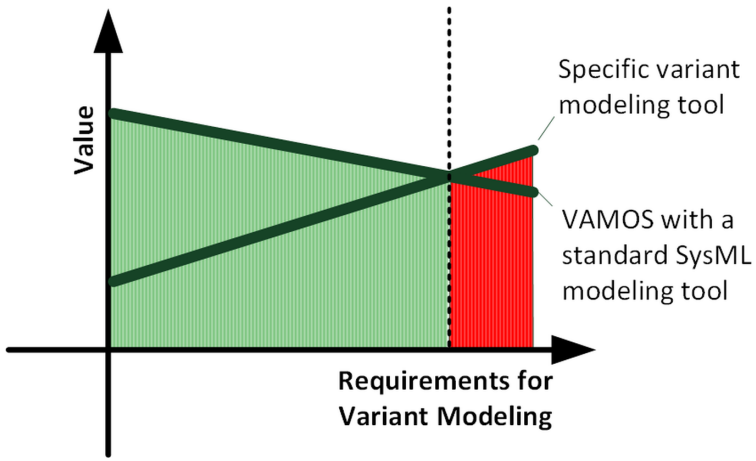

Figure 3.2: Value of VAMOS

Note that the VAMOS line in figure 3.2 stands for VAMOS with a standard SysML modeling tool. If you customize the modeling tool and add specific variant modeling functionality you can jump on the higher line at the right side of the figure. In that case it is not a standard SysML modeling tool anymore, but also a specific variant modeling tool.

3.1 Sample Project

In this chapter I use a special example to demonstrate the concept. It is a cookbook for fruit salads. That way you can focus on the variant modeling method and the usage of SysML and you are not distracted by a technical example. More seriously you find examples of technical systems in the appendix of this book (Virtual Museum Tour, Forest Fire Detection System).

The *Fruit Salad* model describes some fruit salad recipes taken from the cookbook website *allrecipes.com* (Al16). A single recipe is a variant configuration based on core elements and a set of selected variants, for example a special kind of an apple, orange, or dressing.

3.2 Core, Variations, and Variant Configurations

The basic concepts of VAMOS are the core, the variations, and the variant configurations (see also chapter 2). This section gives a general overview. Detailed descriptions of the three concepts are given in following separate sections.

Figure 3.3[1] shows the top level package structure of the *Fruit Salad* model. The model represents the whole configuration space as well as some concrete variant configurations.

[1]The colored boxes in the diagram are no SysML model elements. They are geometric figures to emphasize the separate variant concepts. Most modeling tools provide those graphical adornments of SysML diagrams.

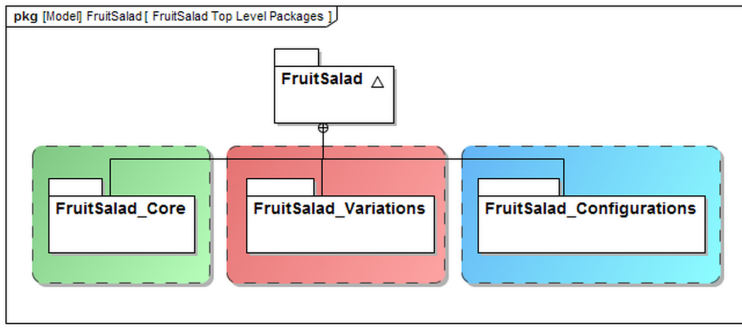

Figure 3.3: Top Level Packages of the *Fruit Salad* Variant Model

On the first level of the model we have three packages that represent the basic concepts:

- The Core package (*FruitSalad_Core*) contains all core elements. The structure of the sub-packages conforms to the system model structure presented in We15. However you can use any package structure here. The structure is independent of the variant modeling approach. Figure 3.4 depicts the top level package structure of the *Core* package. Section 3.3 gives a more detailed description of the core.
- The Variations package (*FruitSalad_Variations*) contains all variations with their variants. The feature tree in figure 3.15 depicts four variations: *Pepper*, *Apple*, *Orange*, and *Dressing*. A variation is a package with stereotype *«variation»*. Each variation package contains the variants according to the variation discriminator. Section 3.5 and section 3.6 give a more detailed description of the variations.
- The Configurations package (*FruitSalad_Configurations*) contains concrete variant configurations, i.e. valid sets of core and variant elements combined to a system or system assembly respectively a fruit salad or fruit salad ingredient. Since a variant configuration is also a special variant (see figure 2.1) we have variation packages on the first level and

the variant configurations as variants on the next level. The variant configuration package includes one variation *Recipes* with two variant configurations: the *Recipes_JuicyFruitSalad* and *Recipes_AppleColeSlaw* (Figure 3.5). Section 3.7 gives a more detailed description of variant configurations.

This package structure clearly separates the three orthogonal variant concepts (figure 3.4).

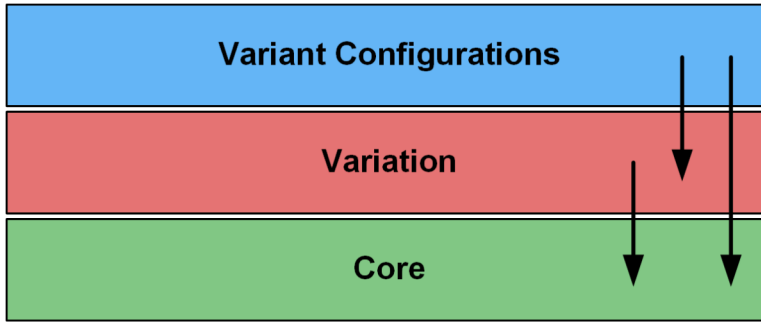

Figure 3.4: Orthogonal aspects

The variant configurations depend on the variant and the core assets. The variation assets depend only on the core assets. And the core assets are independent of the variation assets and variant configuration aspects despite the information about variation points.

4. Variant Stereotypes for SysML

This chapter presents the definitions of the VAMOS stereotypes. They are part of the SYSMOD profile (We15). You can use them independently of the other SYSMOD stereotypes, although I recommend the SYSMOD toolbox to create your system model.

Figure 5.1 depicts all VAMOS stereotypes in a single diagram. In the following each stereotype is described in subchapters in alphabetical order. See also chapter 2 for the semantic of the variant concepts implemented by the VAMOS stereotypes.

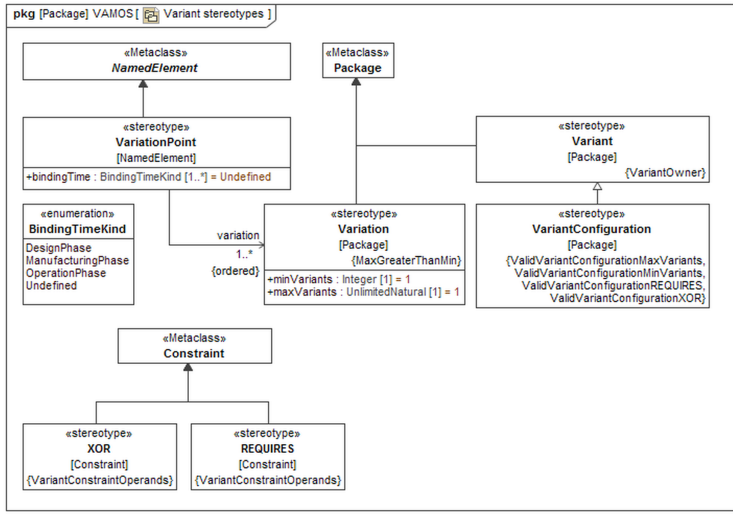

Figure 5.1: VAMOS stereotypes of the SYSMOD profile

4.1 Variant

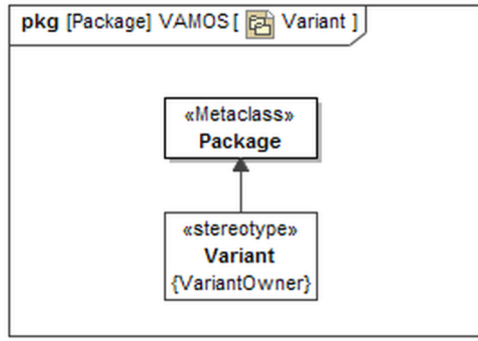

Figure 5.4: Stereotype Variant

The *Variant* stereotype is applied to a package that contains all specific elements of the variant. The package structure within the variant package is the same as for the system or a system component. In particular, the variant could also have variant configurations and variations with variants.

A variant must be owned by a *Variation* (section 5.5). That rule is assured by the constraint *VariantOwner* that is applied to the variant stereotype. The constraint is defined with OCL as follows:

```
context Variant inv VariantOwner:
self.owningPackage.oclIsTypeOf(SYSMOD::VAMOS::Variation)
```

4.2 VariationPoint

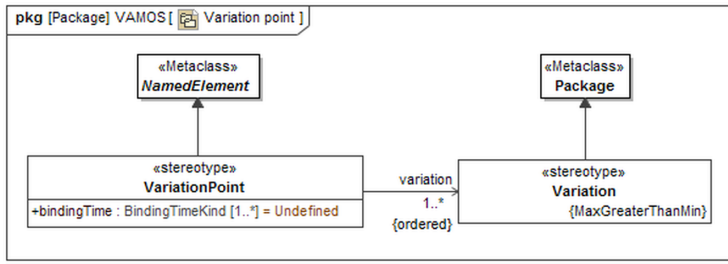

Figure 5.7: Stereotype Variation point

A *VariationPoint* marks a model element in the core that exists in different Variants. It could be any model element that has a name. The stereotype extends the abstract metaclass *NamedElement*.

The property *bindingTime* is described in section 5.1.

The property *variants* specifies a set of at least one *Variation* that contains the variant elements that could docks at the variation point. If the variation point is a classifier like a *Block* the relationship between the variant element and the variation point is typically the *Generalization* relationship.

Appendix A - Example Forest Fire Detection System

The Forest Fire Detection System (FFDS) observes a defined area for forest fires. The core use case of the system is to report a fire within a defined time frame from the burst of the fire. The example is taken from the book SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML (We15).

Section A.1 presents the core of the FFDS and gives a good overview of the system. The following sections describe the variant aspects of the FFDS.

A.1 The Core

There are many implementations possible to provide a forest fire detection service: satellites (e.g. (JPL16)), forest animals equipped with sensors (Sa07), drones, watch towers, planes, and more. To narrow the solution space we define a base architecture that presets some architecture and technical decisions. Figure A.1 depicts a sketch of the base architecture on a beer mat. Figure A.2 shows a more formal version in a SysML internal block diagram.

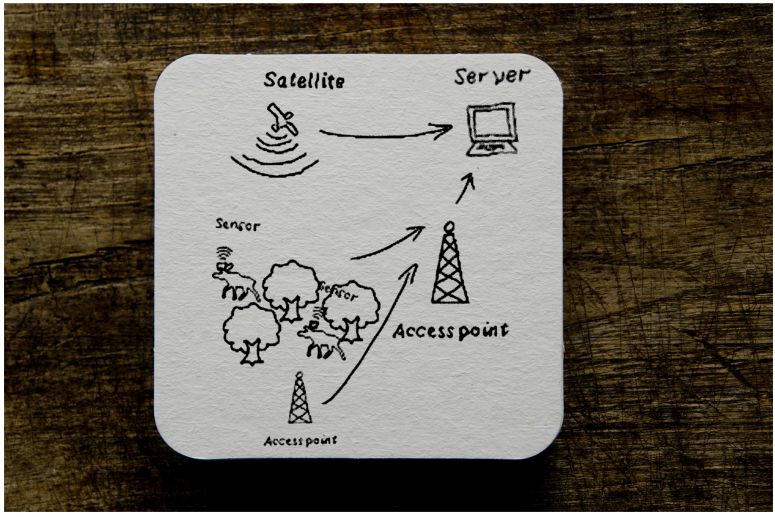

Figure A.1: Sketch of the Base Architecture of the FFDS on a beer mat

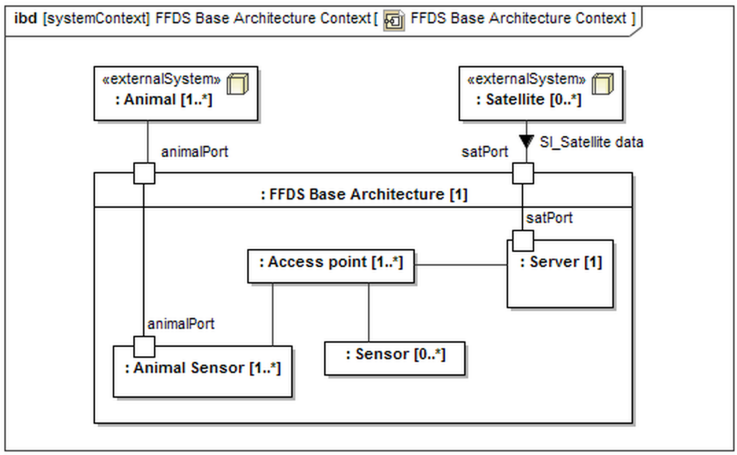

Figure A.2: Base Architecture of the FFDS

The base architecture presets the option for a satellite observation. The satellites are external to the system. The animal sensors are the special feature of the system and a mandatory part. The sensors are

connected with access points to communicate with a central server.

Appendix B - Example Virtual Museum Tour System

The example of the Virtual Museum Tour (VMT) is taken from my book *Model Based System Architecture* (WeLa15) that I have written together with Jesko G. Lamm, Stephan Roth and Markus Walker. The VMT system provides virtual visits of a museum especially outside the opening hours. The customer watches the museum through the eyes of a robot. The system provides fix tours where the customer just watches the video stream from the robot whiles it moves automatically through the museum and listens to the explanations from a tour guide. The system also provides custom virtual museum tours where the customer can control the robot and steer it through the museum.

Section B.1 gives a brief overview of the core including a description of the purpose of the VMT. More details of the example can be found in WeLa15. The following sections describe the variant aspects of the VMT.

Note that the example presented here slightly differs from the example in WeLa15. The books have both their own lifecycle and herewith their own example model.

Bibliography

(Al16) allrecipes.com, accessed January 2016.

(CVL16) http://www.omgwiki.org/variability/doku.php. accessed March 2016.

(JPL16) http://wildfire.jpl.nasa.gov/. accessed January 2016.

(Ky90) Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. *Feature-oriented domain analysis (foda) feasibility study.* Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, 1990.

(Ky02) Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe. *Feature-oriented product line engineering.* IEEE Software, 19:58-65, 2002.

(MOF15) Object Management Group. Meta Object Facility (MOF) Version 2.5. 2015.

(OCL15) Object Management Group. Object Constraint Language (OCL) Version 2.4. 2014.

(Poh05) Klaus Pohl, Günter Böckle, and Frank J. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer, 1 edition, 2005.

(Sa07) Sahin YG. Animals as Mobile Biological Sensors for Forest Fire Detection. Sensors (Basel, Switzerland). 2007;7(12):3084-3099.

(Sy15) Object Management Group. *OMG Systems Modeling Language (OMG SysML) - Version 1.4.* formal/2015-06-03.

(We08) Tim Weilkiens. *Systems Engineering with SysML/UML.* Morgan Kaufmann, 2008.

(We14) Tim Weilkiens. *Systems Engineering mit SysML/UML.* 3. Auflage, dpunkt-Verlag, 2014.

(WeLa15) Tim Weilkiens, Jesko G. Lamm, Stephan Roth, Markus Walker. *Model-Based System Architecture*. Wiley, 2015.

(We15) Tim Weilkiens. *SYSMOD - The Systems Modeling Toolbox*. Leanpub, 2015.

(We16) Tim Weilkiens. *The Myth of the Association*. Blogpost www.model-based-systems-engineering.com, 2015.

Index

R

S

T

U

V

X