Model-Based System Architecting : 'Cook Book' for Dilemma Exploration

Richard Doornbos, Wouter Tabingh Suermondt, Tjerk Bijlsma

TNO-ESI

Contents

Date	December 2018
Version	1.0
Project name	Octo+/SA
Project number	060.31772/01.01
Document number	2018-10263
Contact	richard.doornbos@tno.nl

Abstract

An architectural dilemma is an important and difficult architectural choice that has simultaneously advantages and disadvantages. Exploring the effects on a variety of system aspects and KPIs is an important task for system architects. The clarity this exploration process provides helps in the end to decide what is the best choice. This exploration can be performed systematically and in a stepwise approach, based on the Model-Based System Architecting methodology. This paper provides the guidelines to execute and consolidate the dilemma exploration, in order to make a balanced and underpinned decision.

Keywords: dilemma exploration; architectural decision making; model-based system architecting; multidisciplinary design; quantitative analysis; architecting support; system complexity; impact analysis; technology change

1. Introduction

Making choices is essential for product innovation. One of the key tasks of a system architect working in an industrial environment is to understand the impact of an architectural choice. The expected impact of the change should be made as concrete as possible and, when feasible, in quantitative terms. When multiple options exist for a single choice, i.e. when various realization options exist, their effects on crucial system aspects such as cost, performance, and quality must be compared. Choosing the best option can be difficult as each choice has different advantages for different aspects: a dilemma.

To grasp the impact of choosing an option, one should be at least knowledgeable about the relations and dependencies between critical parameters of the system architecture under design. E.g. how is a particular choice affecting the cost, performance, quality? One can pose questions about the needed investment for manufacturing in the factory, the cost of replacement for the service department, etc. These consequences can go far beyond the properties or performance impact on the product, usually considered by a product architect!

In industrial practice these dependencies are not crystal clear and reside mainly in architects' heads, because these relations are seldom made explicit, let alone be captured, modeled or documented, and conscientiously maintained in a tool. Due to this fact, rarely agreement exists on these important matters [12], [13]. Many architectural concepts, mechanisms and parameters are often misunderstood, and consequently parameter *values* are regularly debated, as quantitative models are lacking.

Depending on the company and its approach to knowledge and expertise management, there is sometimes discussion on who is knowledgeable or responsible for a specific area. In large companies the knowledge and skills are distributed among a large group of employees. As each knowledge area has a vague boundary, overlap of these areas cause many times disagreement. The factor of multi-disciplinarity adds to the problem as languages differ per discipline.

It is clear that for technically complex systems the impact of a choice is hard to establish in practice, especially when dealing with large teams of architects, in multi-site development, and in the context of a large number and variety of stakeholders. The system architect should be supported in this daunting task as much as possible.

In this white paper we describe guidelines for dilemma exploration using the model-based system architecting (MBSA) approach [7], [5], [14]. The exploration will lead to an well-understood and communicable overview, and the gained insights will be extremely useful in decision making.

1.1. Intended audience

This document is intended for everyone involved in the activity to resolve the architectural dilemmas. We can generally

identify several classes of audience:

- The system architects and the domain architects.
- The direct managers of the architects.
- The decision makers related to project milestones.
- The engineering specialists and other experts that contribute in the exploration process.
- Other stakeholders of the system.

1.2. Outline

Chapter 2 describes in detail the application of MBSA in dilemma exploration, in the form of a cookbook. Guidelines are given step-by-step to enable a structured way of working, hence the term 'cookbook'. The instructions of what and how to use the tool, DAARIUS, is described explicitly in the boxes in each section.

2. Exploring an architectural dilemma

A system architect performs a large variety of activities as mentioned in the Introduction. Variability and context-dependency of architecting activities play a major role in MBSA. This chapter will show a typical flow of a systematic model-based way of working.

An architectural dilemma is a difficult architectural choice due to tension in the architecture caused by conflicting interests or interacting effects. For example, system performance and system cost are connected in an intricate way: deciding for more expensive resources will have a positive effect on performance, but a negative effect on cost (lower is better). What to choose?

Very often the most important work for a system architect is creating clarity in dilemmas, or making certain tensions in architecture or system design explicit. This will help in deciding what to do as it provides an underpinning or substantiation by quantification. To create clarity, the dilemma should be analyzed: find out which parts of the systems are involved, what is being affected and how big the impact is (e.g. on system KPIs). This question cannot be answered immediately due to the intricate dependencies and complexity of the system and its context.

The MBSA approach will help in systematically uncovering and exploring the relations. The list of steps below can be followed to explore an architectural dilemma. In the end a balanced decision in the dilemma can be made based on the provided architectural information and the built-up knowledge and reasoning lines.

> We have for the sake of clarity dissected dilemma exploration into a sequence of discrete steps. Please keep in mind that in practice these steps can be taken in a different order, partially executed, or even skipped.

We have experienced that sometimes the dilemma 'evaporates' when doing the exploration. The dilemma was in that case not a real dilemma, but a poorly understood problem or based on wrong assumptions (that were corrected during the exploration). In those cases no further steps are needed, and the exploration ends.

The dilemma exploration is supported by DAARIUS, see the Appendix on Tool support. This requires some preparations; we assume the following items are in order before starting the exploration:

- a *Project* for the system already exists in the DAARIUS tool.
- access to DAARIUS is arranged for all users.

In the following sections the use of DAARIUS is indicated for each step:

> In these text boxes explicit DAARIUS tool instructions are given. Note that 'MBSA terms' such as *Project*, *View*, *Block*, and *Text* are in italic. Further information about these concepts, their usage and the functionality of the tool can be found in [3] and [4].

2.1. Starting point: shared high-level system information

Step 0: share system knowledge

It is possible to use a CAFCR analysis results [2], or to use available overviews (e.g. A3 architectural overviews) as starting points to achieve a shared knowledge level. Typically, these results usually contain uncertainties or are still somewhat vague. This naturally leads to a lot of misunderstandings and misconceptions within the team, which eventually may cause repetition of work and significant delays.

When available, the following small set of views on the system can be helpful to use as a starting point. Usually these views are in the form of sketches or diagrams:

- Functional View showing the decomposition of system functions. This may include system dynamics in the form of process descriptions.
- Physical View showing the decomposition of tangible system parts and relevant environment.
- Stakeholder View showing the stakeholders and their concerns and needs.

When these views are not available, just start with step 1.

2.2. Create information overview

Step 1: identify knowledge domains

Give a name to the dilemma and try to explain the current thoughts. Be prepared for a lot of discussion, questions, and requests for explanation. Identify the important knowledge domains related to the current dilemma. This is needed to be sure all relevant arguments and consideration are taken into account.

- It is important to give a concise name to the dilemma, to be able to refer to it in various communications.
- The dilemma 'owner', the person responsible for investigating the dilemma, or the one to resolve it, should be named. Of course, in practice usually a team is involved in the exploration and resolution of a dilemma, but it is important to assign a single responsible person.
- Use the 4-layer MBSA overview structure to guide your identification process: stakeholders, system, aspects, realization elements / building blocks, see Figure 1.
- Use for each stakeholder, aspect, building block a separate knowledge domain, see [5].
- Give names to the domains, use terms well-known in the organization.

> In DAARIUS:

- create a *View* for the 4-layer MBSA overview structure. Its title is summarizing the dilemma: "<Project name> / <Dilemma name>". Note that explanations about how to operate DAARIUS can be found in [3] and [4].
- create for each knowledge domain a named *Block*, and add a background *Image* for establishing a dedicated visual area on the screen for the information, and add a *Text* with the knowledge domain name to be used as a label for the visual area.
- organize the background images in the 4-layer structure, see Figure 1.

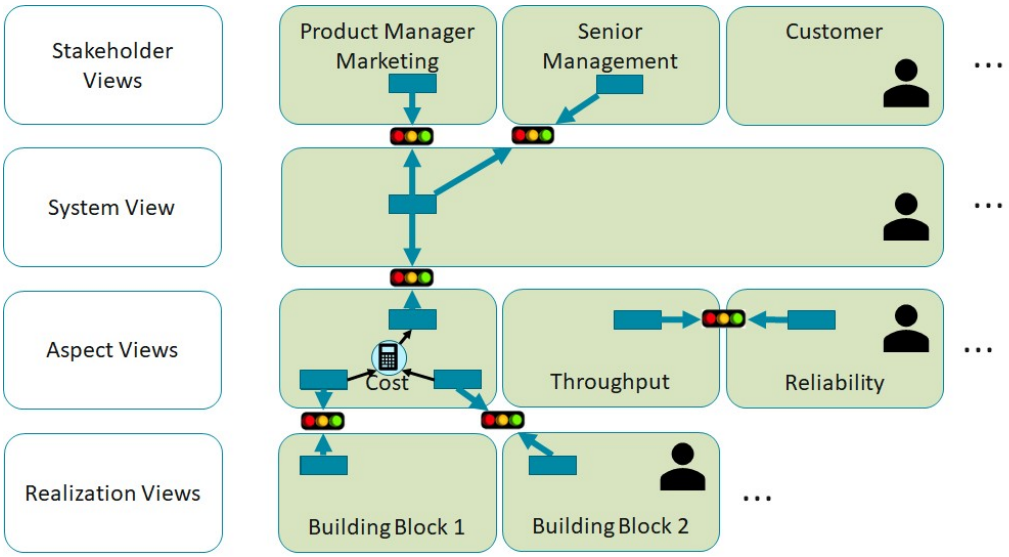

Figure 1. The MBSA structure containing the information of and the information flow between the most important system elements and stakeholders [7].

Step 2: determine owners

Determine the owner of each knowledge domain determined in the previous step. There is only one person who is responsible and can make decisions [1]. Attaching a name to knowledge domains helps tremendously in the communication process, as it clearly defines the role with respect to the system and within the architecting process.

In DAARIUS:

- add to each knowledge domain's main *Block* a new *Block* with the owner's name [2].
- when possible add an *Image* with the picture of the person. Respect privacy regulations and individual preferences.

Step 3: determine shared aspects and concepts

Determine the important system aspects and concepts of each knowledge domain at the overview level that are related to the dilemma, e.g. reliability, cost. Identify what the relation is of a certain knowledge domain to the dilemma. Why is this domain mentioned? What concept, term, notion is used to express this? If no relation can be found, leave it out. To check if the concept is at the correct level, the overview level, the concept (term, or notion) should be relevant to other knowledge domains.

In DAARIUS:

- create in the MBSA overview *Blocks* that represent the identified concepts. Name them accordingly.
- if possible add *Parameters* to the *Block* it belongs to or refines. If the value is not known leave it open. Do not discuss values now.

Step 4: determine critical parameters

Determine the important and critical parameters. Parameters are attributes of concepts: they refine or characterize the concept and have a value and a unit, see [7]. For example: a concept 'power supply' has a parameter 'output power'. What are the parameters that have an effect at the system level, on the system structure or system behavior, or on the business *related to the dilemma*? These parameters are generally called 'critical'.

- In case the critical parameters are unknown, determine why? Is there a gap in knowledge? This is a serious issue and should be addressed with priority.
- The 'owner' of the critical parameter is by nature determined by its knowledge domain. It is important to assign an owner to have a single responsible person; this is a good way to control and manage the discussions about the critical parameter.

In DAARIUS:

- add the found *Parameters* to the *Block* it belongs to.
- add the value and unit. If the value is not known just make an estimate.

Step 5: identify the main system process

Identify the system process(es) related to the dilemma [3]. Each system has a core functionality that can be expressed as a process: the system does something with input and produces output. E.g. a printer delivers ink into paper using a very specific and well-defined process. Explore the process steps, its inputs, its outputs and the sub-systems that realize the function. This will provide the right context for the dilemma exploration.

In DAARIUS:

- create a *View* for the process. Its title is summarizing the main output deliverable or the process: "<Project name> / Aspects / <Process name>". In this way it is positioned in de aspects layer, see Figure 1.
- add a background *Image* for establishing a dedicated area, in order to be able to show the interaction with other systems, material streams, etc.
- create for each process step a named *Block*
- add *Texts* or *Images* to explain the process.

Step 6: identify variation points

Determine the variations that are important at system level. Where are items in the system or its context that can change: product configurations, scenarios, user modes, input materials, etc.? This is important as the dilemma may work out differently for each variation, and it is crucial to understand the impact across the variations.

In DAARIUS:

- create a *Block* for each variation point and add it to the related concept. E.g. a 'product configuration' is added to the 'product' *Block*.
- add a *Parameter* to the *Block* to set the selected variant. The parameter name should explain the kind of variation, e.g. 'configuration', 'scenario', 'user mode', 'type'.
- assign a 'variation'-*Representation type* [4] to the *Parameter* to make it clearly visible. See [3] for details.
- add a *Transformation* to enable the selection out of a list of variants [5].
- the *Transformation* will adapt the value of the output parameters, depending on the selected variant.

Step 7: determine realization options

Determine the realization options, the possible ways to realize a particular part of the system. These realization options have associated properties that differ per variant, which can be expressed in a so-called option sheet. A simple example is shown in Table 1. Note that the values are not essential in this phase of the exploration.

- The system parts, often called 'building blocks' must be positioned in the realization layer (see Figure 1).
- Limit yourself to the main choices that in essence expose or create the dilemma at system level.

Table 1. An example of an option sheet

Power supply type	cost	weight	output power
Regulated	100 $	1 kg	200 W
Unregulated	50 $	1 kg	200 W
Programmable	1000 $	2 kg	150 W

In DAARIUS:

- add a *Parameter* to the *Block* that represents the building block. Use a name that explains the type of realization option.
- assign an 'option'-*Representation type* [4] to the *Parameter* indicating that realization variants exist.
- add a *Transformation* to enable the selection out of the list of realization options [5]. Option sheets can be input for this step. It is clear from these sheets which parameters are dependent (in our example: cost, weight, output power); obviously these should be the outputs of the *Transformation*.
- (optional) the *Transformation* will automatically set the value of output parameters, depending on the selected option.

Step 8: identify influences on other knowledge domains

Identify the (consequences and) influences on other domains due to a choice for a certain realization option. In particular, analyze whether the dependent parameters in the previous step are used in other domains. There should already be relations to the dilemma as identified in step 3 and 4. It is important to do this for all realization options in order to fully cover all impact. This tracing of influences crosses many knowledge domains: e.g. one can reason from realization element via aspects and system domain to the stakeholders domains [6]. Also dependencies between the domains in one layer (so-called 'horizontal' dependencies) may exist [7].

In DAARIUS:

- recognize the existing relations between the domains, and check if there is an influence when choosing a certain realization choice (see Step 7: determine realization options).
- add new *Relations* representing these influences, when not already present in the tool.
- refine these relations by connecting a *Validation* between the dependent parameters [8]. This *Validation* can be used as a 'traffic light' that indicates the value consistency.
- it is useful to indicate visually the influences. This can be done by choosing a clearly visible *Representation type*.

Step 9: focus on the important stuff

Reduce the amount of domains/connections/relations by hiding them from view. This clean-up based on the thus far developed insights, creates more focus on the dilemma and makes it easier to explain.

- Obviously, this is a generic guideline that can be done often.

In DAARIUS:

- hide the less relevant *Blocks*, *Images*, *Relations*, and *Parameters*.
- check if the visual indications of the previous step now really stand out.

Step 10: re-assess the influences

Estimate the largest influences and most important relations between the domains (that remained after step 9) involved in the dilemma. This may be an influence via different paths. This step intends to limit the exploration effort: in this way we focus on the important influences only.

In DAARIUS:

- check if the influences (represented in a *Relation* or visible as a traffic light) have to be changed due to new insights about their importance.
- adapt the associated parameters when needed.

Step 11: determine the workflow

Create a view of the workflow. How is the system playing its role in its context [9]? Which actors are involved, how are materials entered into and moved out of the system, is there interaction between the actors, how much time is involved in

all steps, etc.? This will lead to clear indications where the dilemma has an impact on the workflow.

In DAARIUS:

- create a separate *View* for the workflow. Its title is capturing the workflow process: "<Project name> / Aspects / Workflow"
- add a background *Image* for establishing a dedicated area, in order to be able to show the interaction with other knowledge domains.
- create for each workflow step a named *Block*
- add *Texts* or *Images* to explain the step.
- create for each interaction a named *Relation*
- (optional) add *Texts* or *Images* to indicate the (expected) impact of the choices / dilemma.

Step 12: check if the dilemma is visible

Check the visibility and existence of the initial identified dilemma at the aspect level (layer 3). At this level the influence on the various system aspects (see Step 3: determine shared aspects and concepts) must be clearly visible: e.g. one choice should improve one aspect while deteriorating another aspect, and another choice should have the reverse effect.

In DAARIUS:

- check if the parameters in the aspect layer (connected via a *Relation* or visible as a traffic light) do change according to expectations, when choices in the realization layer are taken.
- if not according to expectations, try to find out why. Adapt the relations, models or parameters when needed.

Step 13: check the dilemma owner

Typically, a dilemma is 'owned' by (or the responsibility of) one architect, see Step 1: identify knowledge domains. The dilemma owner has the responsibility to explore and resolve the dilemma. Check if the 'dilemma owner' is still correct, given the gained insights.

In DAARIUS:

- check if the name of the dilemma owner is still correct. If not the case, adapt the name.
- for completeness, the team can also be named, and added to the dilemma *View* (see Step 1: identify knowledge domains).

Step 14: check the knowledge domain owners

Revisit the owners of all knowledge domains. Sometimes the earlier indicated owners (see Step 2: determine owners) are considered incorrect when time passes and the exploration progresses.

In DAARIUS:

- check if the names of the knowledge domain owners are still correct. If needed, adapt the names.
- for consistency, the involved team members (when present in the tool) should also be adapted.

2.3. Zoom-in on knowledge domains

Step 15: identify the important concepts and relations in the knowledge domains

Step into the knowledge domains and work out the most important and so-far 'unknown' (or unclear) concepts, parameters, and relations [10].

- This is work for the knowledge domain owners as they (and the supporting teams) have domain knowledge and are responsible for the information about the domain.

In DAARIUS:

- create a *View* for each knowledge domain. Its title is the name of the knowledge domain. Use the hierarchy of names for the *Views* following the MBSA overview structure: "<Project name> / <Layer name> / <Knowledge domain name>".
- unhide the background *Image*, in order to be able to show the interaction with elements (*Blocks* and *Parameters*) of other knowledge domains.
- unhide the existing *Blocks* of this knowledge domain (added in Step 3: determine shared aspects and concepts, Step 4: determine critical parameters, Step 7: determine realization options and Step 8: identify influences on other knowledge domains). Position them inside the background image, indicating that their are part of the domain.
- create for new concepts, parameters, and relations a named *Block, Parameter,* and *Relation*. Connect them via *Transformations* or *Relations*.
- add *Texts* or *Images* to explain the added elements, to express assumptions, etc.

Step 16: quantify

Quantification of the concepts, parameters, relations. Here the intention is to refine qualitative relations. A qualitative relation between concepts can be refined by translating it into (a set of) quantitative relations (functions) between parameters of the two concepts, see also Step 4: determine critical parameters.

- Be aware that this is an expensive step: it takes a lot of effort to create quantitative relations. This can take easily ten times the effort of creating qualitative relations!
- Decompose concepts into smaller concepts, when relevant for expressing values.
- Characterize each concept using parameters. E.g. a 'Power supply' (concept) has the following parameters: 'power output', 'power input', 'output voltage', etc. When relevant also 'weight', 'width', 'height' and 'length', etc.
- Determine values and units for parameters; when this is difficult, use indications (+/0/-), or by reference (comparisons to the values of existing systems or components).

In DAARIUS:

- for each concept, add parameters using named *Parameters*.
- add a value and the unit. If the value is not known just make an estimate.
- add in the comment-field of the added parameters an explanation.
- use separate *Text* elements to express any assumptions.
- for each dependent parameter create a *Transformation* using the parameters as input and output. A *Transformation* uses a *Model,* see [4].
  - E.g. a 'Power supply', can be refined using the parameter 'output current'. This parameter can be calculated by dividing the 'power output' by the 'output voltage' (this calculation is the model).
- for each qualitative *Relation* create a (set of) *Transformation(s)* using the parameters as input and output.
  - E.g. a qualitative relation 'Connector' - 'part of' - 'Power supply', can be refined using the parameters 'cost' of the power supply and 'cost' of the connector. The 'part of' is refined into the cost calculation: the cost of the connector is added to that of other elements to calculate the total cost of the power supply.

Step 17: model the usage scenarios

Work out conceptually usage scenarios [9]. What is the end user doing with the system? Which steps are taken by which actor, in which order and at what time? Note that this more detailed exploration will provide requirements, assumptions and constraints for each step in the scenarios, which should be captured carefully to establish the link to the dilemma.

- The scenarios can be expressed in state machines, Gantt charts, swim lanes, or time lines. Choose a usable granularity level for discussion.
- Determine for each step requirements, assumptions, constraints, timing information, and specific properties related to the dilemma.
- In case of a multitude of scenarios, think of weighting the scenarios.
  - Weighting scenarios can be done e.g. by estimating occurrence at the customers, or measured in field data. This should help to focus on the most important, worst case, or best case scenarios.

In DAARIUS:

- use the workflow *View* to add the newly discovered information.
- add the discovered requirements, assumptions and constraints in *Parameters*, *Images* or in *Text* elements.
- add values and units. If the value is not known just make an estimate.
- assumptions should be checked using a *Validation* [8].

Step 18: connect to existing dilemmas

Try to connect the reasoning about the dilemma to already existing information about other dilemmas, good/bad-weather scenarios, and MBSA analyses. This will deepen the understanding and allows to relate system considerations and reasoning threads.

In DAARIUS:

- add the discovered relations between dilemmas by adding *Relation* and in *Text* elements.
- when needed create a new dedicated *View* to elaborate on the relation between the dilemmas.

Step 19: quantify parameters that cross the domain boundary

Quantification of parameters that 'cross' the boundary of the knowledge domains. When different knowledge domains use the same parameter, one should be very careful: it is by no means clear if the parameters have the same meaning! Typically, different domains have different vocabularies and for good reasons. Sometimes a translation is required, however it is clear that a strong dependency may exist.

- Use simple statements to explain the parameter in both knowledge domains. Making things quantitative and using simple formulas quickly leads to better understanding of the key differences.
- Indicate also the assumptions made. The validity range for the values can be expressed in each domain, or for each calculation.
- Check on inconsistencies in values.

In DAARIUS:

- check the values and units of the existing *Parameters* that exist in both the knowledge domains.
- add *Texts* or *Images* to explain the elements and their differences.
- for each parameter a *Validation* [8] should be created to check on value equality in both domains. Here it is used as a traffic light.

Step 20: check assumptions in the domain

Re-assess or validate the assumptions, models and statements in each knowledge domain. This is needed to increase the validity and precision of the reasoning over knowledge domains.

- Express the assumptions and statements using parameters and value ranges.
- If possible create a traffic light to indicate the status of an assumption: a green traffic light shows that the assumption is met, red indicates a violation. An orange color may indicate that the assumption is nearly met.

In DAARIUS:

- check the resulting colors of the existing *Validations*.
- are the values and units of the existing *Parameters* (especially those that cross the knowledge domain boundary) still OK?
- update all parameters, or validation models when needed.

2.4. Refine iteratively

Refine the information in the overview and all knowledge domains to arrive at a sensible level to be able to easily reason about the dilemma. This may take many iterations. Parts of steps 1 to 20 can be re-done, deepened, and checked with newly acquired knowledge and skills [11].

Step 21: add models

Assess whether more detailed models are needed to understand and calculate certain aspects and their associated parameters in the context of the dilemma exploration. The more detailed models typically have a higher level of accuracy. Note that accuracy alone should not be the main reason to add the models. Sometimes an accuracy of 20% as perfectly acceptable, in other situations a higher accuracy is required to make a decision [12] in the (early) architecting phase of system development. Detailed models are usually already developed within a certain knowledge domain, and the knowledge domain owner will know about them.

- Identify the modelers among the experts and specialists in each knowledge domain.
- Identify their models and the overall information about these models (purpose, input / output, assumptions, context and other contributing factors). Note that in industrial practice also many valuable models exist that are not shared.
- Check the expert models for calculation time, and if they are fit for the purpose of analyzing the dilemma. Discuss whether more abstract or simplified models are appropriate.

In DAARIUS:

- identify the *Transformations* under consideration.
- replace the *Models* used in the *Transformations* by the expert models, see [4].
- validate the output of the new models by comparison with the old ones, and with other references. Did the related traffic lights change colors?

Step 22: reason across domains

With the growing knowledge and confidence we can now reason across multiple domains about the dilemma. This should be done and exercised with an audience (in the team of architects, or other stakeholders) to receive immediately feedback when things are missing, or wrong turns are taken. Admit and express that this activity is continuously in progress and the most difficult part of the dilemma exploration. This obviously requires a large degree of trust and a cooperative mindset within the team.

- Start the reasoning about the dilemma at the choices available in the realizations. From there follow the relations and traffic lights to other knowledge domains. Explain the impact in each domain. By following the relations the reasoning thread can traverse across many knowledge domains, including those of the stakeholders.
- It is also possible to start the reasoning thread at the stakeholders' concerns. In fact one can start anywhere.
- Note that the reasoning is done in the perspective of making a decision in the dilemma.

In DAARIUS:

- viewing and navigating the overview and knowledge domain views are the major activities.
- add or change information when there is a need to.

2.5. Finalize

Step 23: create a clear message

Now it should be possible using all the gathered information that explicitly describes the dilemma, to create a clear message. This can be done in the form of a story or presentation, in order to provide clarity or stimulate discussion about the dilemma in the team of architects or for senior management (and other decision makers). Having a clear story is crucial to obtain management commitment and buy-in based on insights.

- The impact at the stakeholder level is obviously of great interest for the stakeholders. Start therefore the reasoning about the dilemma at this level and work towards the choices available in the realizations.
- Show the trade-offs that have to be made by indicating the impact of each realization choice.

In DAARIUS:

- (optional) create a special *View* for the explanation of the dilemma for a given audience. Its title can be chosen: "<Project name> / <Dilemma name> / <Audience>"
- typically the 4-layer overview is used and adapted to tell the story or reasoning line.
- for this purpose specific colors (defined in the *Representation types*, see [3]) can be used to visually support the reasoning.

Step 24: decide on the dilemma

In the end a balanced decision in the dilemma can be made based on the provided architectural information and the built-up knowledge and reasoning lines.

- The evaluation of the choices must be presented in a concise way, e.g. in Pugh matrices showing benefits vs. costs comparisons.
- Decisions are usually taken by the architect, the team, and senior management (and other stakeholders such as representatives of business units, and marketing and customer support departments) together in a decision meeting setup. Therefore the decision meeting must be thoroughly prepared to be able to make an substantiated decision.

In DAARIUS:

- similar to the previous step, create a special *View* for the explanation of the dilemma for a given audience.
- the impact of the decision can be shown using DAARIUS in real-time by selecting a certain choice in the realization layer. Parameter values and traffic lights should show the effects.
- Keep this *View* as simple as possible, because the audience is not used to the way of representation and the visualization, and may have difficulty in understanding or gets distracted (by irrelevant details). In any case, other forms of presentation can be created based on this *View*.

Acknowledgements

We thank Océ's architects, engineers, and managers for their collaboration and contributions to this work.

References

- [1] Gerrit Muller, *Systems Architecting: A Business Perspective*, http://www.gaudisite.nl/SystemArchitecting.html.
- [2] Gerrit Muller, *Architecting for Business Value*, http://www.gaudisite.nl/ABV.html.
- [3] *Design Framework - user manual*, ESI, 2016.
- [4] Wouter Tabingh Suermondt, Richard Doornbos, Tjerk Bijlsma, *Design Framework Tutorial*, Version V0.1, 2018-08-20, ESI.
- [5] Tjerk Bijlsma, Wouter Tabingh Suermondt, Richard Doornbos, *A knowledge domain structure to enable system wide reasoning and decision making*, submitted, 2018.
- [6] Bas Huijbrechts, *Octo+ 2018 - Project plan* (Océ and ESI collaboration), TNO doc. no. 2017-10193, ESI, 2017.
- [7] Richard Doornbos, Wouter Tabingh Suermondt, Tjerk Bijlsma, *Model-Based System Architecting in a nutshell*, ESI, 2018.
- [8] Ronald Fabel, Lou Somers, *Virtual product development @ Océ*, ESI Symposium presentation 2012.
- [9] https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two.
- [10] Richard Koch, *The 80/20 Principle: The Secret to Achieving More with Less*, ISBN 978-0-385-49174-7, 1999.
- [11] ISO42010, https://en.wikipedia.org/wiki/ISO/IEC_42010.
- [12] Ivano Malavolta, Patricia Lago, Henry Muccini, Patrizio Pelliccione, Antony Tang, *What Industry needs from Architectural Languages: A Survey*, IEEE Transactions on Software Engineering, 39(6), pp. 869-891, 2013.
- [13] TOGAF, *Tools for Architecture Development*, http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap38.html#tag_39_02, 2018.
- [14] Bas Huijbrechts, Richard Doornbos, Ronald Fabel, Bas Stottelaar, *Model-Based System Architecting - Quantitative Early Architecting*, white paper v1.0, 2016.

Appendices

Appendix A: Tool support

The tool named 'DAARIUS' (Digital Assistant for ARchitects using Information for Understanding Systems) is intended to operationalize and support the MBSA approach, see [3] and [4]. Although the MBSA approach can be applied using pen and paper, the amount of information quickly becomes intractable. The tool supports the MBSA language [5] and can consolidate the large amount of architectural information, e.g. system concepts, parameters, and models. This tool also enables easy sharing within the team (and other stakeholders), automatic manipulation (execution, checking, coupling, navigation), and professional management (reuse, governance, storage). The main features of DAARIUS:

- Strong system architecture language, with only a few basic concepts
- Calculation propagation, consistency checks (traffic lights), generation of graphs
- Linking of Views
- Interoperability with external modeling tools (Matlab, Excel, Python, Frink)
- Central storage (single source)
- Multi-user, web-based
- Multi-project
- Industry scalable
- Versioning support

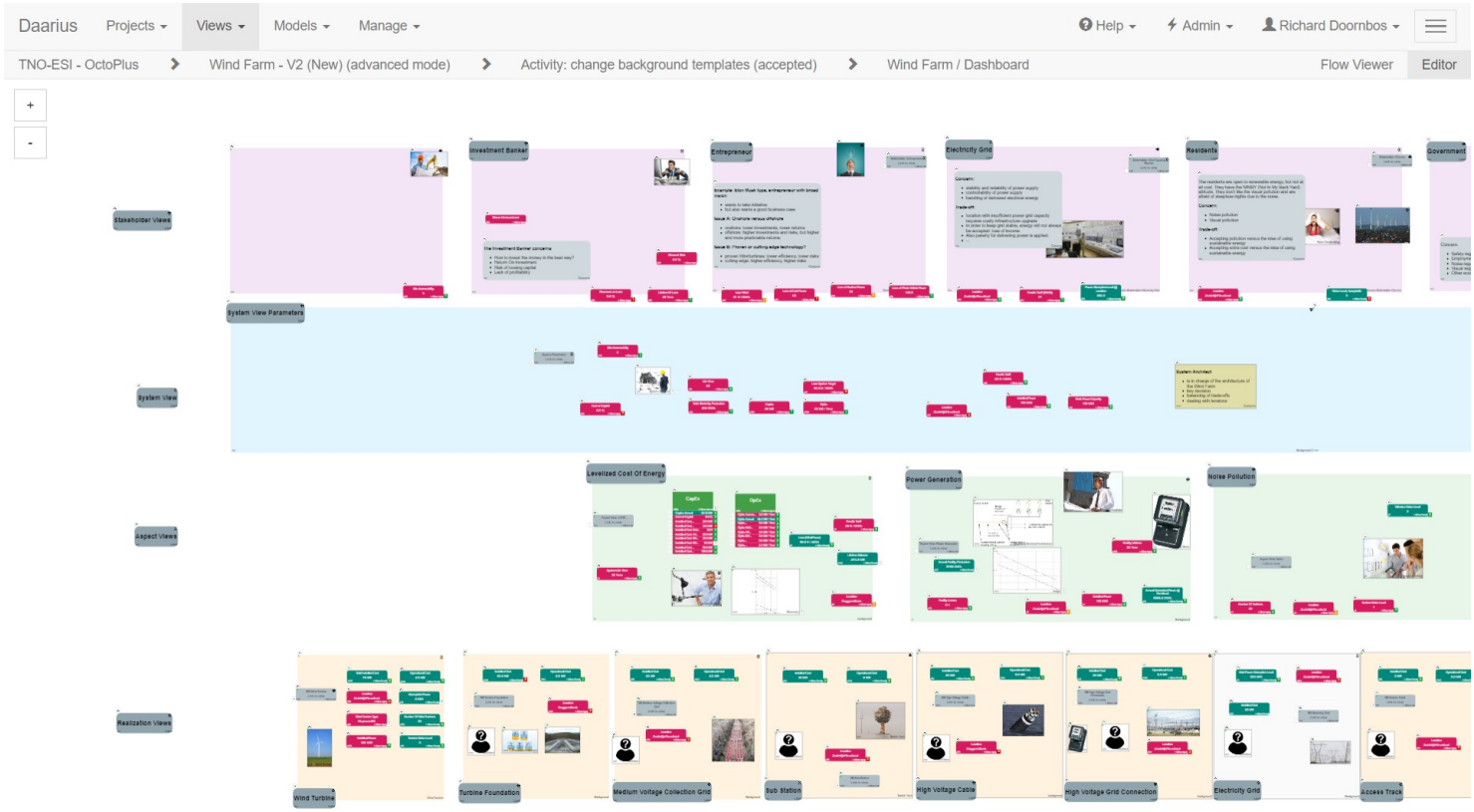

Figure 2. A screen shot of DAARIUS.

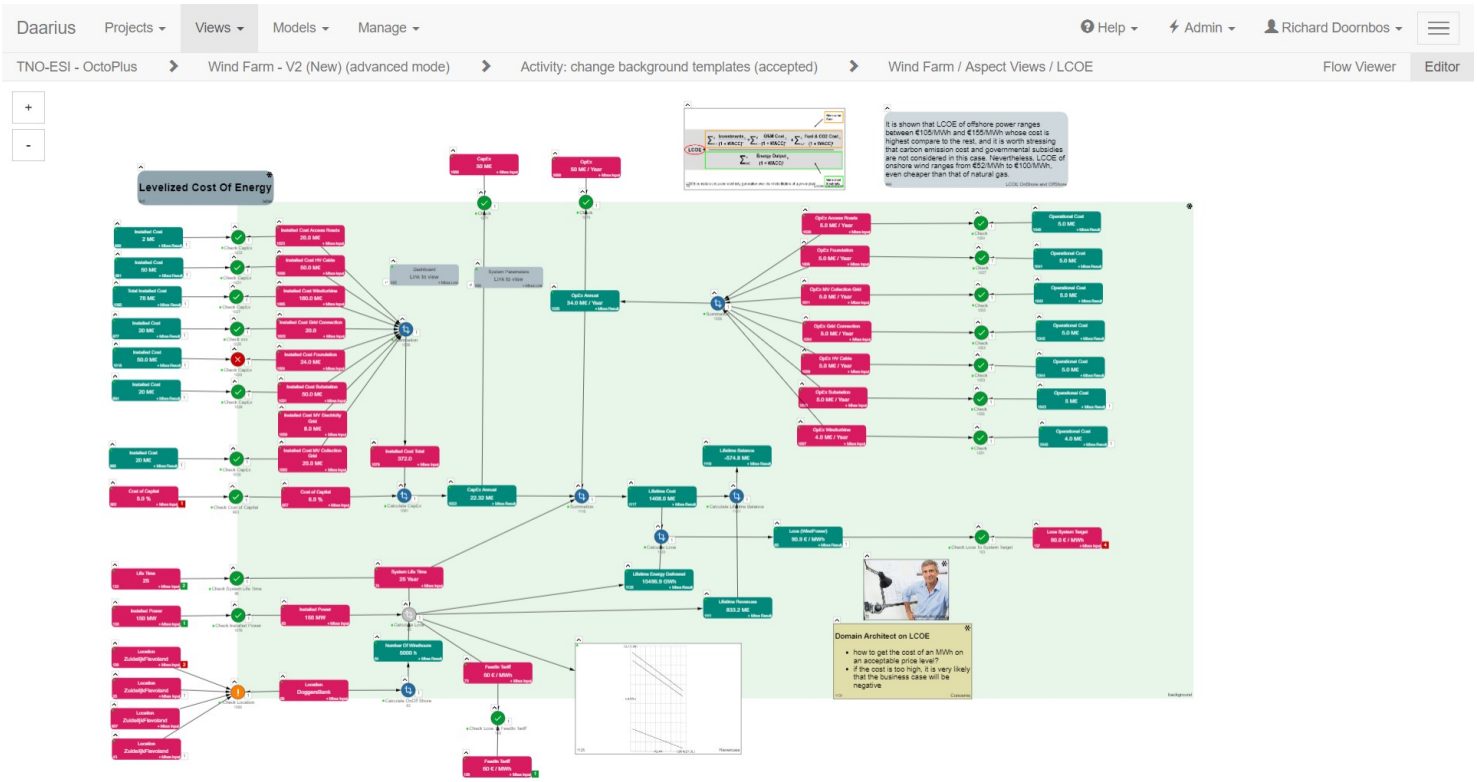

Figure 3. A DAARIUS screen shot of a dedicated View.

Appendix B: End notes

1. Of course more people may work in a domain and can contribute valuable information
2. There is some freedom here: it is also possible to add a *Parameter* named 'owner' and use as value the name of the person. Furthermore, the ownership can when needed be made explicit by adding an 'owns'-*Relation* between the person and domain *Block*.
3. It is possible that this step has already been addressed to some degree in Step 0: share system knowledge
4. In DAARIUS all elements can be visualized in a specific way. The assigning of a *Representation type* to each element determines the color and shape of the element.
5. This is in essence selection of values from a lookup table (a specific type of *Model*).
6. This reasoning (vertically) over layers and domains is called in the CAFCR framework of Gerrit Muller [1] 'threads of reasoning'. See also [7] for discussion about reasoning across the MBSA structure.
7. Horizontal dependencies require in general special attention as they tend to complicate matters significantly. There is a significant semantical difference between horizontal and vertical relations in the MBSA structure; for more discussion see [7]
8. A traffic light is a particular use of a *Validation* that checks if both inputs are equal (green), very close (orange) or not equal (red). *Models* with inequalities or range checking are also possible. See [3] and [4] for more details.
9. This is a perspective at the super system level, thus is a view of the context of the system. We consider this view as a system aspect, and therefore resides at the aspect level in the overview.
10. This can be similarly along the steps Step 4: determine critical parameters to Step 6: identify variation points, and Step 9: focus on the important stuff, only now in the knowledge domain.
11. This acquiring of knowledge, modeling and reasoning skills is also a goal of this dilemma exploration activity. The growth of the knowledge and insights should drive the architect (and his team) to iterate over the overviews and knowledge domains, and 'continuously' add new information.
12. This is obviously just an indication. Use a reasonable accuracy while considering the effort to calculate it.