Knowledge Architect: A Tool Suite for Managing Software Architecture Knowledge

Peng Liang, Anton Jansen, Paris Avgeriou

February, 2009
RUG-SEARCH-09-L01

Software Engineering and Architecture (SEARCH) Group
Department of Mathematics and Computing Science
University of Groningen
PO Box 407, 9747 AG Groningen
The Netherlands

Abstract

Management of software architecture knowledge (AK) is vital for improving an organization's architectural capabilities. To support the architecting process within our industrial partner: Astron, the Dutch radio astronomy institute, we implemented the Knowledge Architect (KA): a tool suite for creating, using, translating, sharing and managing AK. The KA tool suite entails specialized support for integrating the various process activities and supporting collaboration between the stakeholders. This report discusses the tools included and features of KA. We also discuss different usages of the KA for capturing and using AK to support the architecture process.

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge. We would like to thank the master students Jens Rasmussen, Natasja Sterenborg, Hubert ten Hove, Alex Haan, Tjaard de Vries, Joris Best, Marco van der Kooi and Reinder Kamphorst for their work on the development of the Knowledge Architect tool suite.

Content

1 Introduction ... 3
2 Motivation ... 3
3 KA Tool Suite ... 3
3.1 Knowledge Repository ... 4
3.2 Document Knowledge Client ... 4
3.3 Excel Plug-in ... 7
3.4 Python Plug-in ... 8
3.5 Knowledge Explorer ... 9
3.6 Knowledge Translator ... 10
4 Conclusion and Future Work ... 11
5 Reference ... 12

1 Introduction

Management of software architecture knowledge (AK) is vital for improving an organization's architectural capabilities [1]. In the context of the Griffin [2] research project, in order to support the architecting process within our industrial partner: Astron, the Dutch radio astronomy institute, we implemented the Knowledge Architect (KA): a tool suite[1] for creating, using, translating, sharing and managing AK. The KA tool suite entails support for integrating the various architecting process activities and supporting collaboration between the stakeholders.

2 Motivation

The KA tool suite is developed iteratively according to requirements and feedbacks of the case studies from Astron. The initial set of use cases for using architectural decision (an important type of AK) was specified in [3]. Generally the KA tool suite addresses the following features to serve the architecting process:

- A central *knowledge hub*. In a large project multiple stakeholders are involved in the different process activities and typically manage and maintain their part of the relevant AK. The knowledge hub is critical for gathering all the AK in one resource, and providing an interface to all involved stakeholders to manage, and evolve it;
- *Traceability management*. In a collaborative architecting process, AK entities are produced by various stakeholders. Traceability needs to be established between these collaboratively produced artifacts (e.g. a requirement leads to a design decision, and when one changes the other needs to be updated). This is of paramount importance during the architecture iterations, but also for the architecture evolution;
- *Knowledge translation*. Typically stakeholders come from different backgrounds and have their own perspectives on software architecture, usually limited to individual AK entities. Effective knowledge translation enables various stakeholders to understand each other and speak through a "common language". Furthermore knowledge translation provides the ability to present the "big picture", and especially the complex relationships between different parts of the knowledge;
- *Automated checking*. Different stakeholders working at varied activities and at different time, may touch upon the same or related AK entities. Automated checking may help to identify the conflicts, inconsistencies and incompleteness in the collaboratively produced AK entities. Especially when the amount of knowledge increases, this type of automated support is the only way to effectively manage it.

3 KA Tool Suite

Currently, the KA tool suite consists of 6 tools: Knowledge Repository, Document Knowledge Client, Excel Plug-in, Python Plug-in, Knowledge Explorer and Knowledge Translator. A brief description of each tool is provided, while a detailed presentation follows in the next subsections:

- **Knowledge Repository** is at the heart of the tool suite: a central location, which provides various interfaces for other tools to store and retrieve AK.
- **Document Knowledge Client** is a Word plug-in that supports capturing (annotating) and using (storing and retrieving from the Knowledge Repository) AK within software architecture and requirement documents inside Microsoft Word.
- **Analysis Model Knowledge Clients**: are tools that support capturing (annotating) and using (storing and retrieving from the Knowledge Repository) AK of quantitative analysis models. Specifically, two knowledge clients are developed (Excel and Python plug-in):
    - **Excel Plug-in** supports capturing and using AK of quantitative analysis models inside Microsoft Excel [4].

[1] Part of the tool suite can be downloaded from http://search.cs.rug.nl/griffin

    - **Python Plug-in** supports capturing AK from quantitative analysis models described in Python.
- **Knowledge Explorer** is a tool for analyzing the relationships AK entities have. It provides various visualizations to inspect AK entities and their relationships.
- **Knowledge Translator** is a (semi-)automatic tool to translate the formal AK based on one AK domain model into the AK based on another, so that various stakeholders can understand each other when they use different AK domain models to document AK. The Knowledge Translator can be potentially used across the different architecting activities when translations are needed due to the heterogeneous AK domain model being used by AK producer and consumer, e.g. a requirements engineer and an architect use different AK domain models to produce and consume requirements (part of AK).

3.1 Knowledge Repository

The Knowledge Repository as depicted in Figure 1 is a central location, which provides an interface to store and retrieve AK across a wide range of architecting activities. The tool makes heavily use of technologies developed for the semantic web. The Knowledge Repository is built around Sesame, an open source RDF store [5], that offers functionality to store and query information of ontologies [6]. AK Domain models are modeled as ontologies, which are expressed in OWL [7]. The Protégé tool is used to create the OWL definition of the domain model [8], which in turn is uploaded to Sesame. To provide some intelligence in the Knowledge Repository, Sesame is extended with the inferencer OWLim [9], which offers OWL Lite [7] reasoning facilities. This inferencer is mostly used to automatically infer the inverse (for the traceability) and mapping (for the knowledge transformation) relationships that exist between AK entities. In this way, a user does not have to manually define them. The Repository API provides the interfaces to communicate with all the Knowledge Clients (Document Knowledge Client, Excel and Python Plug-in) to store the annotated AK into the repository. The Query Engine is used to query the AK entities and their relationships in the repository, and visualize them in the Knowledge Explorer. The Knowledge Translator also employs the OWLim engine to perform the automatic translation. All the surrounding tools are described in details below.

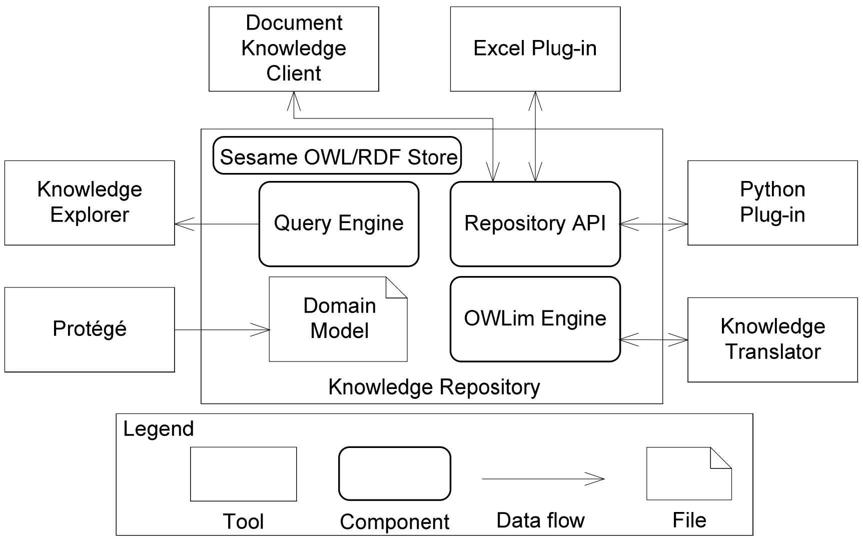


Figure 1. The Knowledge Repository with other tools in the KA tool suite

3.2 Document Knowledge Client

The Document Knowledge Client is a plug-in to capture and use explicit AK inside Microsoft Word 2003. The plug-in adds a custom button bar (see Figure 2) and provides additional options in some of the context aware popup menu's of Word (e.g. Figure 3 and Figure 4). The tool automatically adapts at start-up time to the AK domain model used in the Knowledge Repository, which is specified using OWL [7]. Various AK domain models can be deployed in the Knowledge Repository for different users (stakeholders), who annotate the AK using the AK domain models they can understand. Hence,

the tool can be reused with other AK domain models. Figure 2 presents the buttons that give access to the bulk of the functionality of the tool. In short, they give access to the following functionality (the numbers below correspond to the button numbers in Figure 2):
(1) Add current selected text and/or figure(s) as a new AK entity.
(2) Add current selected text and/or figure(s) to an existing AK entity.
(3) Create an AK entity table at the end of the document.
(4) Color the text of the AK entities based on their type.
(5) Color the text of each AK entities based on its completeness.
(6) Show a list of AK entities in the current document.
(7) Export AK entities of the document to a XML file.
(8) Import AK entities from the document into the AK repository.
(9) Connect to an AK repository.
(10) Enable the plug-in for the current active document (i.e. read and show the annotations)
(11) Open the settings menu.
(12) Display the plug-in version & authors' information.

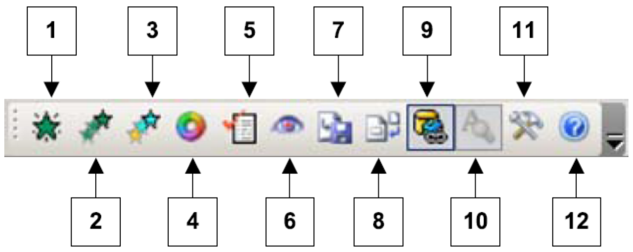
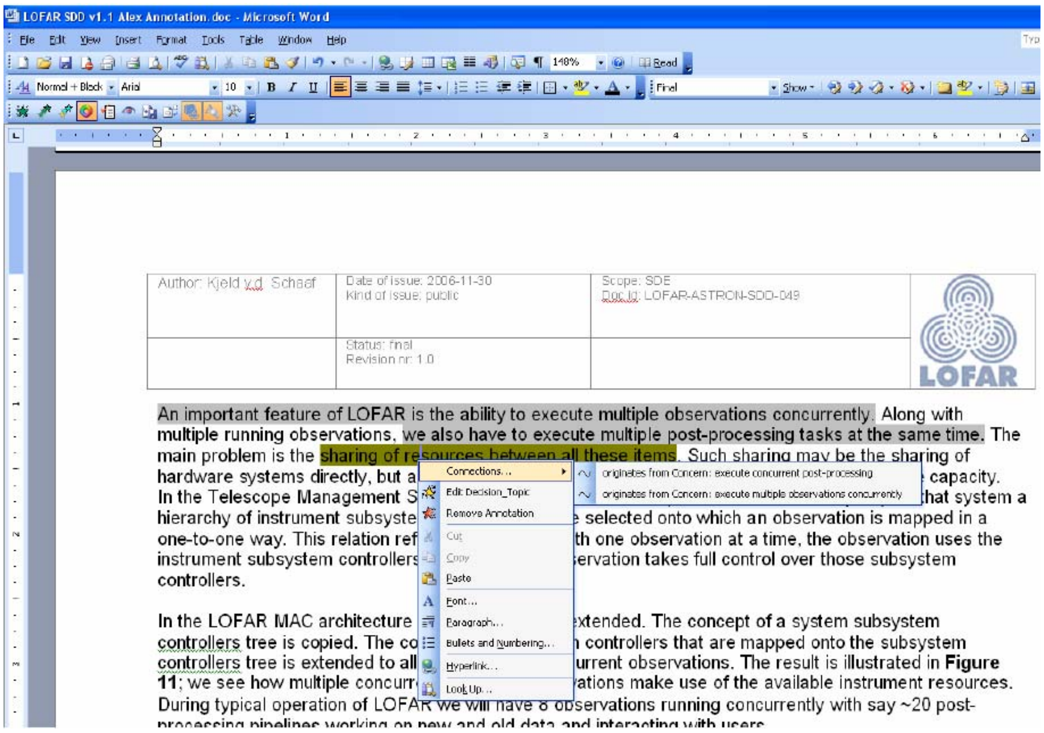


Figure 2. The Document Knowledge Client button bar

AK capturing: Capturing AK in a Word document involves buttons 1 and 2, but can also be performed by selecting a piece of text and right clicking and choosing the appropriate option from the pop-up menu. When adding a new AK entity, a menu appears, which allows the user to provide the following additional information about an AK entity:

- **Name** that identifies the AK entity.
- **Type** of the AK entity, which is one of the concepts of the AK domain model being used.
- **Status** of the AK entity, which describes the level of validity of the AK entity (potential validity states: *To be reviewed*, *Reviewed*, *To be discussed*, *To be checked*, *Validated*, *Obsoleted*).
- **Connections**. The user can add and remove relationships to other AK entities as shown in Figure 3. Based on the earlier defined AK entity type and the domain model, the tool determines the type of relationships that might be available for new relationships to other AK entities.


LOFAR SDD v1.1 Alex Annotation.doc - Microsoft Word

Author: Kjeld v.d. Schaaf	Date of issue: 2006-11-30 Kind of issue: public	Scope: SDE Doc.Id: LOFAR-ASTRON-SDD-049	LOFAR
	Status: final Revision nr: 1.0		

An important feature of LOFAR is the ability to execute multiple observations concurrently. Along with multiple running observations, we also have to execute multiple post-processing tasks at the same time. The main problem is the sharing of resources between all these items. Such sharing may be the sharing of hardware systems directly, but a … capacity.
In the Telescope Management S… that system a hierarchy of instrument subsyste… selected onto which an observation is mapped in a one-to-one way. This relation ref… th one observation at a time, the observation uses the instrument subsystem controllers… ervation takes full control over those subsystem controllers.

In the LOFAR MAC architecture … xtended. The concept of a system subsystem controllers tree is copied. The co… controllers that are mapped onto the subsystem controllers tree is extended to all … urrent observations. The result is illustrated in **Figure 11**; we see how multiple concurr… ations make use of the available instrument resources. During typical operation of LOFAR we will have 8 observations running concurrently with say ~20 post-processing pipelines working on new and old data and interacting with users

originates from Concern: execute concurrent post-processing
originates from Concern: execute multiple observations concurrently

Figure 3. A software architecture document with colored AK entities and pop-up menu for tracing the relationships of an AK entity

AK traceability: An AK entity can be edited or removed by choosing the appropriate option from the pop-up menu when right-clicking on the text of an AK entity. In the same menu, the relationships among AK entities can be followed. Thus useful traceability among AK entities is provided. Figure 3 exemplifies this, under the "Connections..." the pop-up menu lists the relationships an AK entity has, clicking on them moves the cursor to the appropriate piece of text. This allows for a hyper link style of navigation inside an architecture document. Navigating back to the originating AK entity is easy due to the automatically created inverse relationships. To enhance the readability and making spotting existing AK entities easier, the tool can color texts based on their type (button 4 in Figure 2). Figure 3 gives an example of the effect of this coloring. Thus spotting AK entities becomes easy. Simply browsing through an annotated document gives the reader a sense of where most of the relevant AK resides in the document. The colors used for each type can be configured by the user per AK repository.

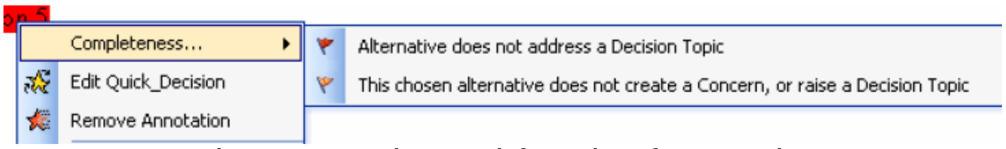

Alternative does not address a Decision Topic
This chosen alternative does not create a Concern, or raise a Decision Topic

Figure 4. Incompleteness information of an AK entity

Design maturity assessment: The Document Knowledge Client supports the architect in assessing the completeness of the architecture description. Based on the AK domain model, the tool performs model checks, which are based on conformity rules, to identify incomplete parts. For each AK entity inside the document, a completeness level is determined. The completeness levels are named after the colors that are used to color the text of the AK entity. To find out the reason why the tool deems a certain AK entity to be incomplete, the user can use the "Completeness...." option of the context popup menu to see which rules are not adhered to. Figure 4 presents an example of this. The tool distinguishes the following four different completeness levels (ordered from high to low severity):

- **Red** One or more primary rules are violated.
- **Orange** The primary rules are adhered to, but one or more secondary rules are violated.
- **Yellow** Both primary and secondary rules are passed. However, the AK entity has not achieved the status of validated yet.
- **Green** Both primary and secondary rules are passed. In addition, the AK entity has been validated by a reviewer.

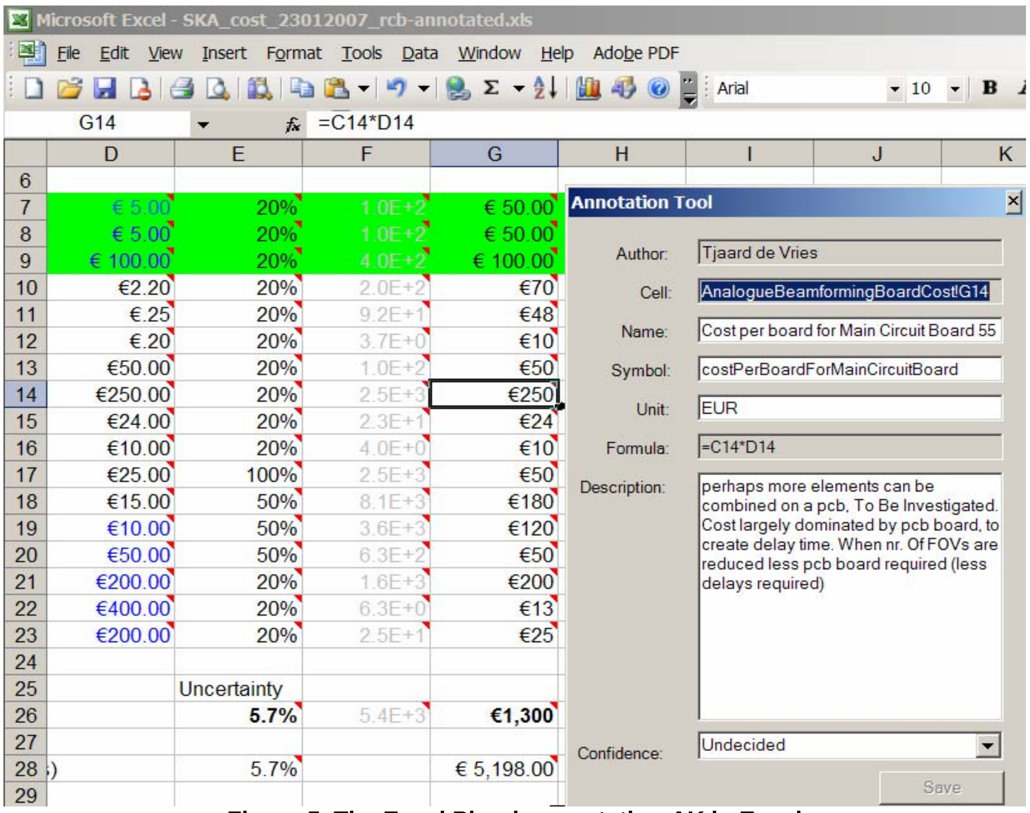
The distinction between primary and secondary rules is a pragmatic one. Primary rules are those rules that check whether the document is complete enough to provide a minimum level of traceability. This minimum level should ensure the existence of at least one reasoning path a reader could follow. Secondary rules focus more on the completeness of the architecture design. Both the primary and secondary rules are specific to the domain model, as they use the concepts and relationships of the domain model to attach semantics to missing information.

3.3 Excel Plug-in

The Excel Plug-in implements a domain model for analysis models in Microsoft Excel. The tool supports analysts in making the AK produced during architecture analysis explicit. The aim is to facilitate the sharing of AK to other analysts and the analysis results in a transparent manner to other stakeholders. The major part of the knowledge of an analysis model in Excel is found in the cells found on different worksheets. The domain for Excel reflects these concepts. The visible representation of a cell is normally a *Number*, which we see as the instance of a particular *System Parameter*. If a single cell is selected, the equation bar presents the *Analysis Function* that calculates the *Number* of a particular *System Parameter*. More information on these concepts can be found in [4].

Often labels surrounding the cell denote the semantic meaning of a cell. However, the texts of these labels are not formally related to any *System Parameters*. The tool allows analysts to make special annotations to make these relationships explicit. Figure 5 presents how a cell is annotated. For a *System Parameter*, a name, symbol, unit, and description can be defined. In the same window, an analyst can define the confidence of the *Analysis Function* as well. For reviewing purposes, the tool tracks the review state of each cell and allows for comments.

The author of these AK entities is automatically determined using the build in office author tracking system. Furthermore, since annotating cells one by one is time-consuming, a feature is available to annotate whole tables of cells without much effort by codifying the two dimensions used in a worksheet area.


Microsoft Excel - SKA_cost_23012007_rcb-annotated.xls
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10 B
Annotation Tool

G14 =C14*D14

	D	E	F	G	H	I	J	K
6								
7	€ 5.00	20%	1.0E+2	€ 50.00				
8	€ 5.00	20%	1.0E+2	€ 50.00				
9	€ 100.00	20%	4.0E+2	€ 100.00				
10	€2.20	20%	2.0E+2	€70				
11	€.25	20%	9.2E+1	€48				
12	€.20	20%	3.7E+0	€10				
13	€50.00	20%	1.0E+2	€50				
14	€250.00	20%	2.5E+3	€250				
15	€24.00	20%	2.3E+1	€24				
16	€10.00	20%	4.0E+0	€10				
17	€25.00	100%	2.5E+3	€50				
18	€15.00	50%	8.1E+3	€180				
19	€10.00	50%	3.6E+3	€120				
20	€50.00	50%	6.3E+2	€50				
21	€200.00	20%	1.6E+3	€200				
22	€400.00	20%	6.3E+0	€13				
23	€200.00	20%	2.5E+1	€25				
24								
25		Uncertainty						
26		**5.7%**	5.4E+3	**€1,300**				
27								
28	)	5.7%		€ 5,198.00				
29								

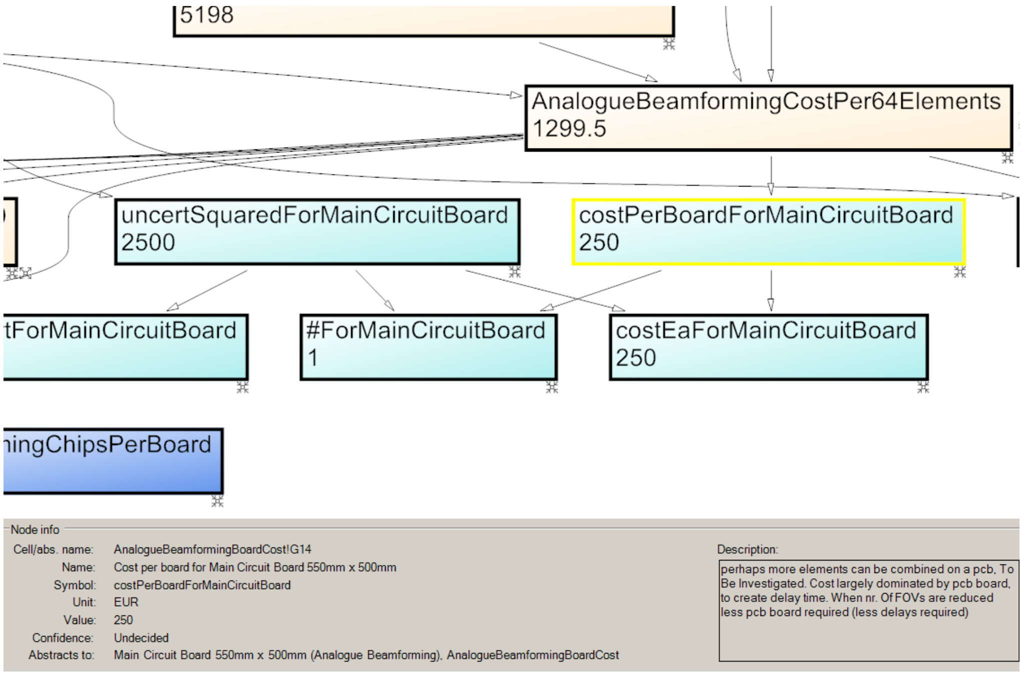
Author:	Tjaard de Vries
Cell:	AnalogueBeamformingBoardCost!G14
Name:	Cost per board for Main Circuit Board 55
Symbol:	costPerBoardForMainCircuitBoard
Unit:	EUR
Formula:	=C14*D14
Description:	perhaps more elements can be combined on a pcb, To Be Investigated. Cost largely dominated by pcb board, to create delay time. When nr. Of FOVs are reduced less pcb board required (less delays required)
Confidence:	Undecided

Save

Figure 5. The Excel Plug-in annotating AK in Excel

An important feature of the tool is that it is capable of automatically inferring the dependencies among the *System Parameters* (i.e. cells) by analyzing the *Analysis Functions* used. Hence, the dependency relationships are automatically captured. As a consequence, it is rather easy to define a scenario, just selecting a cell part of a particular scenario and hitting a task bar button will pop up a menu to define the properties of a scenario. The associated System Parameters and Analysis Functions are automatically inferred.

To facilitate verification, the tool offers a visualization of the dependency graph. An example of this graph is presented in Figure 6. A node in the graph represents a *System Parameter* or a *Component*, i.e. a set of *System Parameters*. An arrow between two nodes indicates that the *Analysis Function(s)* of the *System Parameter(s)* of one node uses the *System Parameters* in its calculations, thereby creating a dependency between the *System Parameters*. The name of a node comes either from a user annotation or a default cell name. The bottom part of the figure visualizes the details of a selected node.


5198
AnalogueBeamformingCostPer64Elements
1299.5
uncertSquaredForMainCircuitBoard
2500
costPerBoardForMainCircuitBoard
250
tForMainCircuitBoard
#ForMainCircuitBoard
1
costEaForMainCircuitBoard
250
ingChipsPerBoard

Node info

Cell/abs. name:	AnalogueBeamformingBoardCost!G14
Name:	Cost per board for Main Circuit Board 550mm x 500mm
Symbol:	costPerBoardForMainCircuitBoard
Unit:	EUR
Value:	250
Confidence:	Undecided
Abstracts to:	Main Circuit Board 550mm x 500mm (Analogue Beamforming), AnalogueBeamformingBoardCost

Description:
perhaps more elements can be combined on a pcb, To Be Investigated. Cost largely dominated by pcb board, to create delay time. When nr. Of FOVs are reduced less pcb board required (less delays required)

Figure 6. An excerpt of the system parameter dependency graph

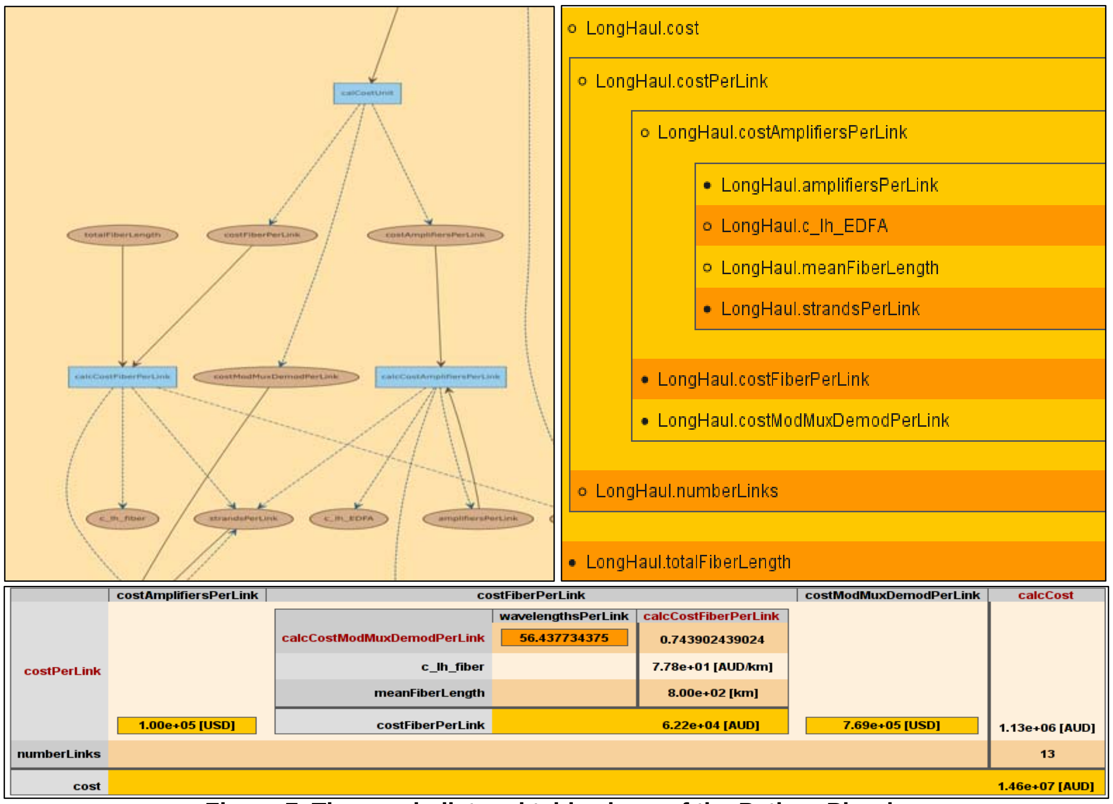
There is a correspondence between the nodes in the dependency graph and the cells of the Excel worksheets. Making a selection of cells in a worksheet, also selects the related nodes in the dependency graph and vice versa. An analyst can use this selection mechanism to easily create new collections of System Parameters. The color of a node in Figure 6 indicates the collection a node belongs to. A user can expand or collapse nodes, thereby providing a way to view either more abstract *System Parameters* or more detailed ones. Orthogonal to this, is the ability to filter *System Parameters* based on the scenarios they are involved in.

3.4 Python Plug-in

Similar to the Excel Plug-in, the Python Plug-in provides functionality to codify the AK of analysis models. In this case, the analysis models are expressed using the Python programming language. Interesting enough, both plug-ins assume quite similar domain models. Hence, most of the concepts discussed in the Excel Plug-in section also apply to this plug-in. The only major difference, concept-wise, lies in the use of an *Analysis Function*. In Excel, such a function is always associated with a unique set of *Numbers*, which directly relate one-by-one to *System Parameters*. Since *Analysis Functions* in Python can be reused, a relationship is needed to make a distinction between multiple sets of such *Numbers*.

Both plug-ins, however, do differ a lot in the way in which the information is presented and captured. The AK is captured using a two stage approach. The first step is manual and involves annotating those methods in the Python code that represent analysis functions. This in contrast to the Excel Plug-in, where the function in every cell is considered to be an Analysis functions. The Python Plug-in uses these annotations in the second step. For each annotated function, it automatically captures the involved *System Parameters* and *Numbers* together with their relationships using a run-time dependency analysis. Once this knowledge is captured, the knowledge is uploaded to the AK repository.

The Python Plug-in offers a similar graph-like view as the Excel Plug-in, as illustrated on the top left of Figure 7. In addition, the plug-in comes with specialized table and list views. The table view, visualized in the bottom of Figure 7, presents the *Numbers* and *Analysis Functions* in a structured manner. This view uses both the horizontal and vertical dimensions to lay out the instances of both concepts. The list view, found on the left side of Figure 7, is a collapsible/expendable list of *System Parameters* presenting their underlying dependency tree and their associated set of *Numbers* used in different *Scenarios*.



- ○ LongHaul.cost
  - ○ LongHaul.costPerLink
    - ○ LongHaul.costAmplifiersPerLink
      - ● LongHaul.amplifiersPerLink
      - ○ LongHaul.c_lh_EDFA
      - ○ LongHaul.meanFiberLength
      - ● LongHaul.strandsPerLink
    - ● LongHaul.costFiberPerLink
    - ● LongHaul.costModMuxDemodPerLink
  - ○ LongHaul.numberLinks
- ● LongHaul.totalFiberLength

	costAmplifiersPerLink	costFiberPerLink			costModMuxDemodPerLink	calcCost
costPerLink			wavelengthsPerLink	calcCostFiberPerLink		
		calcCostModMuxDemodPerLink	56.437734375	0.743902439024		
		c_lh_fiber		7.78e+01 [AUD/km]		
		meanFiberLength		8.00e+02 [km]		
	1.00e+05 [USD]	costFiberPerLink		6.22e+04 [AUD]	7.69e+05 [USD]	1.13e+06 [AUD]
numberLinks						13
cost						1.46e+07 [AUD]

Figure 7. The graph, list and table views of the Python Plug-in

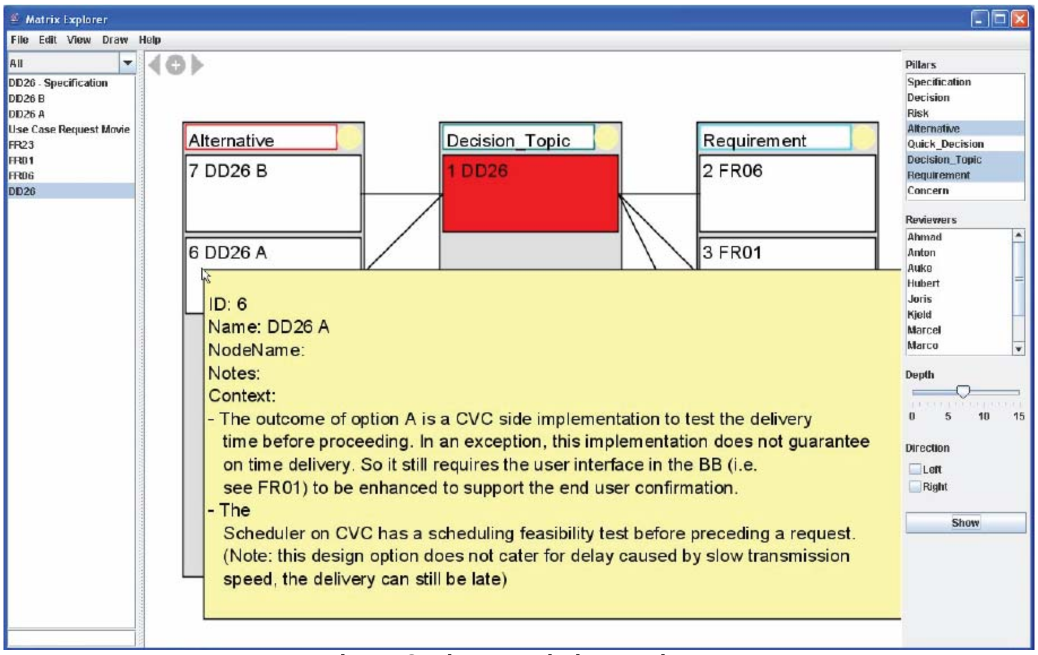
3.5 Knowledge Explorer

Typically, the size of an AK repository will be considerable containing thousands of AK entities. Finding the right AK entity or even worse a number of related AK entities, from such a big collection is not trivial. Hence, there is a need for a tool to assist in exploring an AK repository. The Knowledge Explorer is such a tool that supports users in visualizing AK instances and their relationships, and dealing with the size and complexity of an AK repository.

Figure 8 presents a screenshot of the tool. On the left hand side the search functionality offered by the tool is found. Users can type in queries in the see-as-you-type search box on the bottom left to look for specific AK entities. The resulting AK entities of this search action are shown in the list on the left hand side. The results can be filtered using the drop down box on the left, thereby reducing the size of the found results. The filtering is based on the type of the AK. The available options are taken from the used domain model. Double clicking on one of the search results in illustrating a number of related AK entities in columns.

The selected AK entity (e.g. DD26) is placed in the center column and is indicated with a red background color. All its related AK entities are shown with forward and backward traceability links in the columns left and right of the selected one. Double clicking on these related AK entities changes the focus of the visualization accordingly, by brining them in the center column. The columns represent the concepts of the domain model that are selected from a list on the top right and visualized as grey columns in the middle. In the case of Figure 8, these columns are the Alternative, Decision Topic, and Requirements concepts. The column concept allows for easy inspection of whether an AK entity is (in)directly related to other AK entities of a specific type. For example, this allows for checking whether a requirement eventually leads to a specification (i.e. the detailed design of a decision). This is simply achieved by only enabling the requirement and specification column. To get additional information about an AK entity, the mouse can be hovered over an AK entity and a pop-up window will present this information. Another way to deal with the size of the AK repository is by using the list found in the middle right. This list presents all the AK entities authors and provides the opportunity to either include or exclude AK entities from specific authors for the visualization in the middle.

The last mechanism that helps dealing with the AK repository size is the slider on the middle right. This slider controls the distance from the selected AK entity at which related entities are displayed. This distance is defined as the maximum number of relationships that may be followed to find a related AK entity. By moving the slider to the right, more distant related AK entities are visualized, whereas moving the slider to the left reduces this number.



ID: 6
Name: DD26 A
NodeName:
Notes:
Context:
- The outcome of option A is a CVC side implementation to test the delivery
  time before proceeding. In an exception, this implementation does not guarantee
  on time delivery. So it still requires the user interface in the BB (i.e.
  see FR01) to be enhanced to support the end user confirmation.
- The
  Scheduler on CVC has a scheduling feasibility test before preceding a request.
  (Note: this design option does not cater for delay caused by slow transmission
  speed, the delivery can still be late)

Figure 8. The Knowledge Explorer

3.6 Knowledge Translator

The purpose of the knowledge Translator is to translate the AK in various AK domain models from one to the other and vice versa. This allows various users to understand the AK codified in different AK domain models. This is critical for stakeholders from different backgrounds to understand each other in a collaborative architecting process. As shown in Figure 9, the Knowledge Translator consists of two major functions: the AK domain model mapping and AK entities transformation. In the central AK repository, AK codified using different AK domain models co-exist, since stakeholders use their specific AK domain models to codify the AK.

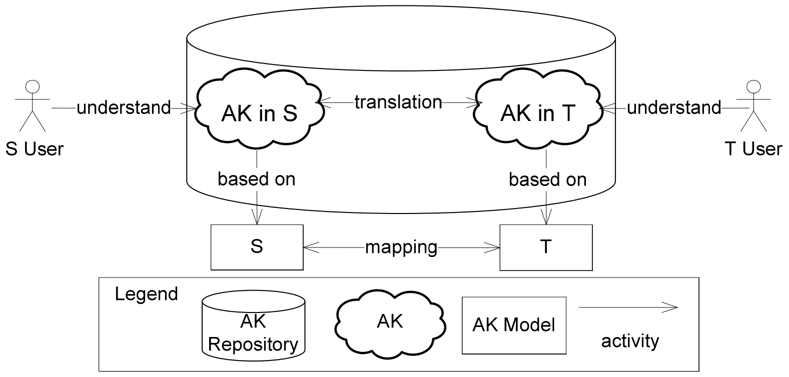


Figure 9. AK translation based on AK model mapping

The tool can assist AK domain experts to perform AK domain model mapping by creating the concept mapping relationships (e.g. *superClassOf*, *subClassOf*, *equivalentClass*, *noMatchingPair*) manually between two AK models as shown in Figure 10. Subsequently the tool can translate the AK entities in a specific AK concept (e.g. *Requirement*) in one AK model into the corresponding AK entities of an AK concept (e.g. *Concern*) of another AK model.

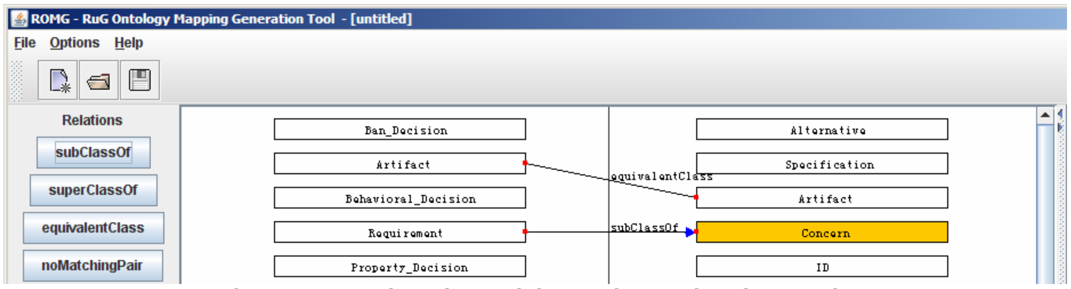


Figure 10. AK domain model mapping tool and example

The AK translation can be done manually or automatically. Manual translation is a two step process that is integrated in the Document Knowledge Client. The first step consists of a user selecting an AK entity that has been annotated with an AK concept of an AK domain model. In the second step, the user specifies the relationship (using the *sameAs*[2] relationship) with the corresponding AK entity of an AK concept of another AK domain model. The manual translation is comparable with natural language translation, in which the translation precision rate depends on the understanding of the human translator. The advantage of manual translation is that the precision rate of the translation is very high with an experienced AK expert, and the disadvantage is that translation cost is quite high, since the AK expert has to translate the AK entities one by one.

Automatic translation uses an automatic classifier to classify the AK entity in an AK concept of an AK domain model into the corresponding AK entity in an AK concept of another AK domain model. The classifier uses as a basis for this the concept mapping relationships. The advantage of automatic translation is that the translation cost is dramatically decreased since all the translation work is done by the automatic classifier without the intervention of human experts. However, the disadvantage is that the precision rate of the translation will be lower than manual translation. For example, *a* is an AK entity in concept *A*, and *A* has *superClassOf* mapping relationship with concept *B* and *C*. In the automatic translation, the automatic classifier will decide whether it will translate AK entity *a* into *B* or *C* depending on how intelligent the automatic classifier is [10]. Since the perfect automatic classifier (one hundred percent correct translation) does not exist, the precision rate is always lower than the manual translation.

Manual and automatic translation have their respective advantages and disadvantages on translation cost and quality, and stakeholders can select an appropriate manner by trading off quality and cost in their own context. The initial cost-benefit ratio analysis about the AK translation cost and quality has been investigated in [10].

4 Conclusion and Future Work

This research aims at improving the architecting process through a knowledge management support mechanism. A central knowledge repository based tool suite KA is implemented for capturing and using AK in architecting process. This technical report presents a panorama view of various features provided by the KA tool suite for AK management and introduces these features by examples. In a practical aspect, the KA tool suite has been used and validated in two industrial case studies of Astron for quantitative analysis of architecture design [4] and enrichment of software architecture documentation [11].

We have planned further case studies of KA tool suite in the architecting activities in Astron. Based on the current trial, following enhancements are scheduled to the KA tool suite: (1) develop additional AK annotation plug-ins for email client and web pages (e.g. wiki) which is the source of most of text-based architecture documentation and communication; (2) enhance the AK annotation functionality of various plug-ins by considering the context knowledge of the AK to be annotated; (3) provide more accurate quality prediction of AK translation for users to select a better knowledge translation method.

[2] *sameAs* is used to specify the “same as” relationship between AK entities, and *equivalentClass* is used to specify the “same as” relationship between AK concepts.

5 Reference

[1] H. van Vliet. Software Architecture Knowledge Management. In Proceedings of the 19th Australian Software Engineering Conference (ASWEC), pp. 24–31, 2008.

[2] GRIFFIN: a GRId For inFormatIoN about architectural knowledge, http://griffin.cs.vu.nl/

[3] J. van der Ven, A. Jansen, P. Avgeriou, D. Hammer. Using Architectural Decisions. In Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA), pp. 1–10, 2006.

[4] A. Jansen, T. de Vries, P. Avgeriou, M. van Veelen. Sharing the Architectural Knowledge of Quantitative Analysis. In Proceedings of the 4th International Conference on the Quality of Software Architectures (QoSA), pp. 220-234, 2008.

[5] J. Broekstra, A. Kampman, F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Proceedings of the 1st International Semantic Web Conference (ISWC), pp. 54–68, 2002.

[6] G. Antoniou, F. van Harmelen. A Semantic Web Primer. 2004, MIT Press, USA.

[7] W3C, OWL web ontology language - W3C recommendation. 2004. http://www.w3.org/TR/owl-features/

[8] H. Knublauch, R.W. Fergerson, N.F. Noy, M.A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In Proceedings of the 3rd International Semantic Web Conference (ISWC). pp. 229-243, 2004.

[9] A. Kiryakov, D. Ognyanov, D. Manov. Owlim - a Pragmatic Semantic Repository for OWL. In Proceedings of the 6th Web Information Systems Engineering (WISE). pp. 182-192, 2005.

[10] P. Liang, A. Jansen, P. Avgeriou. Sharing Architecture Knowledge through Models: Quality and Cost. The Knowledge Engineering Review 24(1), 2009.

[11] A. Jansen, P. Avgeriou, J.S. van der Ven. Enriching Software Architecture Documentation. Journal of Systems and Software, 2009. (accepted)