

**CompAS: A new approach to commonality and variability analysis with applications in computer assisted orthopaedic surgery**

**A Method Engineering approach**

Bart Klein 3251233 B.E.Klein@students.uu.nl

**Table of contents**

- 1. Introduction .....3
- 2. Example.....4
- 3. Process Deliverable Diagram.....6
- 4. Related literature .....14
- 5. References .....15
- 6. Appendix .....16

## 1. Introduction

Software reusability builds on the fact that it is more efficient to assemble software systems from common reusable assets (e.g. modules, objects or classes), than it is to building one from scratch (Douta, Talib, Nierstrasz, & Langlotz, 2009). A recent paradigm for software reuse is component-based programming, where the components are the building blocks for a system to be composed from. A component has several characteristics, it is a unit with contractually specified interfaces and explicit context dependencies only, can be deployed independently and is subject to composition by third parties (Szyperski, 1998). Douta et al. (2009) argue, that such a software development approach implies a move from single systems engineering to families of systems. A system family can be defined as a set of software applications sharing a large number of common properties (Parnas, 1976). Based on the efficiency of software reuse and software families CompAS is an analysis method, which focusses on the commonality and variability between systems in a particular field. This analysis can thereafter be used to support component based architectural modelling within this field.

In their paper, Douta et al. (2009) explain the two different phases ~~from which the CompAS process exists. Firstly~~, phase A: Functional evolution based commonality and variability identification, ~~secondly~~, phase B: Business-oriented variation capture. Each of these phases consists of four parts. For phase A these parts are:

1. Gather a data source, this source should contain functional system descriptions detailed enough to allow for correct functional decomposition.
2. Extract ~~from the data a set of features~~, i.e. a set of user-visible system functionalities.
3. Compute and build the family (or sub-families) evolution matrix(es). Such a matrix combines visualisation and metrics to clearly overview a large amount of data. Special software has been ~~made~~ to compute these matrices, for example Qt from Trolltech AS.
4. Identify the evolution pattern followed by each feature, which was decomposed from the functional system description, and deduce from it their commonality and variability property.

The four different parts for phase B are:

1. Gather a data source, this source should contain justification of specific system evolution.
2. List the features that the gathered descriptions propose to modify and improve.
3. For each feature define the most influential evolution factors, being the non-functional requirements that regularly motivate the implementation of new functional variations, and the resulting functional and technological variation.
4. Use the results of 2 and 3 to build the taxonomy.

The CompAS analysis method has two main deliverables, these are (an) evolution matrix(es) of the family or subfamilies, and a taxonomy of change scenarios based on evolution factors. An evolution matrix is a way to visualize the evolution of classes in object oriented software systems. Using a simple visualization approach, this matrix can greatly reduce the amount of data one has to deal with when analysing the evolution of software (Lanza, 2001). The visualization uses rectangles within a two-row matrix, this way, up to five metrics can be encoded using the rectangle's position, width, height, and colour. The taxonomy which is built in phase B of the CompAS method is based on several influential evolution factors, these evolution factors are the main non-functional reasons why software systems in a particular field were and will be modified and evolved (Douta et al., 2009). It consists of high level taxonomy categories of the specified field factors, each having several subcategories determined by the most influential subcategories.

Douta et al. (2009) don't mention specific rolls for the different parts of the CompAS method, they do however mention a domain analyst. According to Douta et al. (2009) "The identification of commonalities and variabilities mainly relies on the capabilities of the domain analyst to abstract from and refine the collected data and knowledge." This implies that it is necessary for the domain analyst to have extensive knowledge of the domain of the software systems at hand, however, CompAS also

suggests the possibility to consult domain experts. One of the known aspects of data gathering and analysis is that it is a time consuming process (Fantechi, Gnesi, John, Lami & Dörr 2003).

As previously mentioned, it is highly advised for the executors of the CompAS method to be familiar with the domain of the field in which the method is used. In their paper, Douta et al. (2009) explain the CompAS method within the field of Computer-assisted orthopaedic surgery (COAS). Three of the creators of the method, namely Gisèle Douta, Haydar Talib and Frank Langlotz are experts in this domain as they work for the MEM Research Center for Orthopaedic Surgery at the University of Bern. As the last authors background is in computer science, namely the Software Composition Group at the university of Bern, they form a group with expertise in the fields needed to develop the CompAS method. *The introduction is well written and contains, from what I can tell, all the required information. The one thing I did notice is the great focus on the method process. Maybe it would benefit from more detail on the method's purpose instead. However, as I said, it is well written.*

## 2. Example

In this chapter an overview will be given of the two phases of the CompAS method as described in Douta et al. (2009) in the form of an example. This case study in the COAS domain used by the creators of CompAS to explain their method is currently the only example in scientific literature. One reason for this might be that the method can be very well applied to the medical field, but is of less interest as domain analysis method in different types of fields.

### Phase A: Functional evolution based commonality and variability identification

The first step of this phase is the gathering of a set of system functional descriptions. As the field of the authors is CAOS, they acknowledge the main conference in their domain to be the annual meeting of the International Society for Computer Assisted Orthopaedic Surgery. From this meetings, the authors had a total of 1076 abstracts available, covering the years 1998-2005. Since scarce source code data were available, the authors were inspired by a study of Anton and Potts (2003) which considered literature as data source.

The second step is to extract a set of features from the data collected in step 1. The authors did this by first selecting the abstracts which contained a functional description of a system, and hereafter decomposing each of this systems into a set of features. CAOS system users and developers were consulted to evaluate the set of features.

The third step is to compute and build the evolution matrix(es). The authors identified four subfamilies, namely spine, hip and pelvis, knee and traumatology. The evolution matrices were built using a prototype support tool. An example of an evolution matrix for the knee family can be seen in figure 1. In this matrix, five metrics are encoded, the x-axis shows the computed metrics for a given year, and the y-axis shows a defined functionality. The rectangles themselves encode three data types as the width reflects the cumulative percentage of systems including the evaluated feature, the height represents the distribution over years of systems containing this feature, and finally the colour represents the percentage of system descriptions that belong to the considered family of applications for a given year.

The last step is to identify the evolution pattern followed by each feature and deducting the commonality and variability properties from these features. The authors created a table showing evolution patterns of the features by using a gray-scale. Figure 2 shows a little part of this table, the complete table of Douta et al. (2009) includes all four subfamilies and more features. In this table, the term red giant means a common feature that keeps on being very wide over time. A white dwarf is a feature that used to be wide, common, but is decreasing in width to become variable. An idle feature means that it remains relatively small, it is a rarely used variation.

*Tough domain but well explained example*

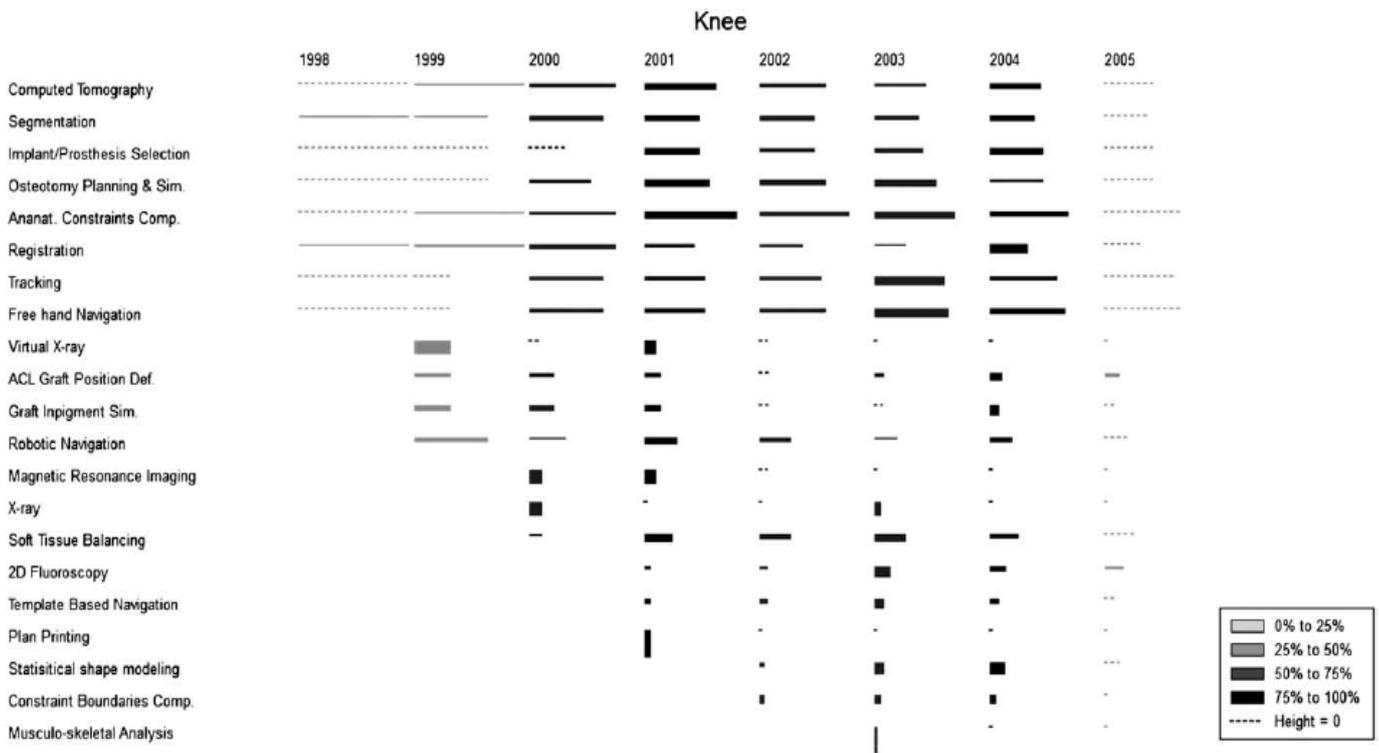


Figure 1: Evolution matrix of the knee family (Douta et al., 2009)

Knee	
Computed tomography	Red Giant
Tracking	Red Giant
Free hand navigation	Red Giant
2D fluoroscopy	Idle
Segmentation	Red Giant
Implant/prosthesis selection	Red Giant
Registration	White Dwarf
Virtual X-ray	Idle
Robotic navigation	White Dwarf
Trajectory definition	
Template based navigation	Idle

Figure 2: Parts of identified common and variable features in the knee family (Douta et al., 2009)

### Phase B: Business-oriented variation capture

The first step of this phase is to gather a set of functional evolution descriptions. The authors collected abstracts which focussed on proposing improvements to a given aspect of already existing systems.

In the second step  authors identified from the abstract nine factors which they see as the main factors of CAOS evolution. These factors are: Visualization, accuracy and safety, planning and outcome optimization, system handling, invasiveness, radiation, time, robustness and cost.

The authors partitioned the features in seven categories, figure 3 shows an example of one of these categories with a computation of the influence of the previously identified evolution factors. The estimated most influential factors for the category are highlighted in red. In this example the most influential factors of the 3D-modeling category are visualization and accuracy.

The final step of this phase is to use the results of the two previous steps to build the taxonomy. Douta et al. (2009) depict subcategories of the seven taxonomy categories by explaining how the most influential evolution factors contribute to the functional and technological variations in the related CAOS functionality. In the example of 3D-modeling, this method depicts three subcategories, namely output, input and method. Douta et al. (2009) argue that different methods of 3D modeling were

proposed, each having different types of information as input, and a different nature of information as output.

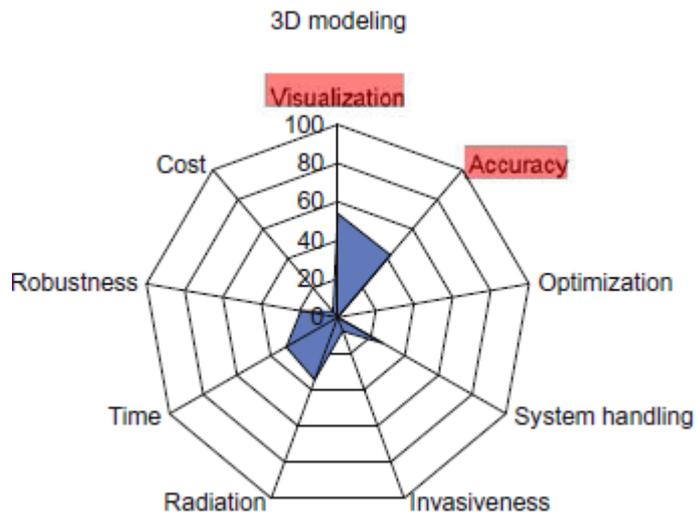


Figure 3: 3D modeling most influential evolution factors (Douta et al., 2009)

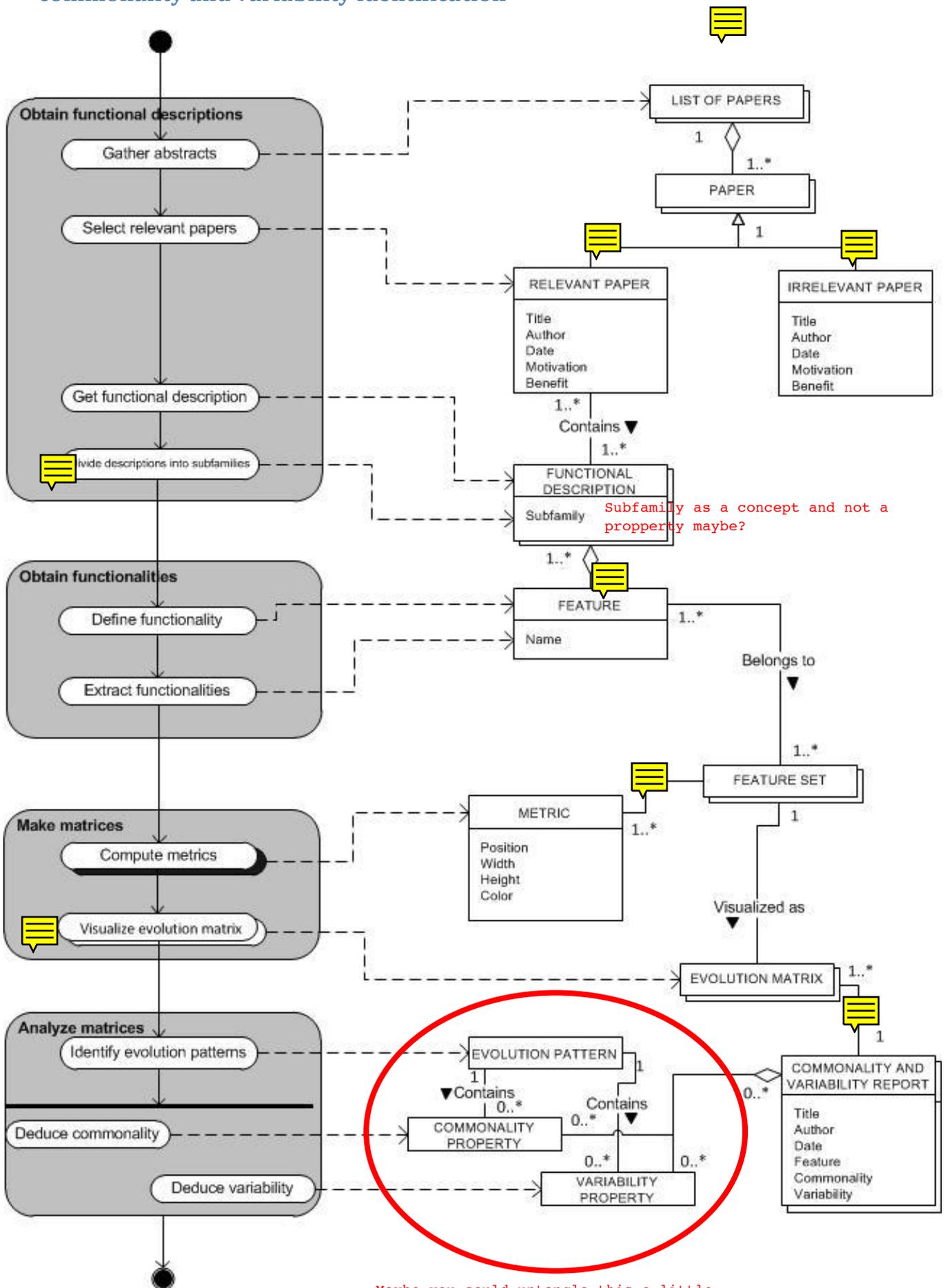
### 3. Process Deliverable Diagram

This report contains the process-deliverable diagrams (PDD) for the CompAS method. A PDD is a modelling technique which links processes within a method to the deliverables they produce. The model consists of two integrated diagrams, on the left side, the processes are modelled, based on the UML activity diagram, and on the right side, the deliverables are modelled based on the UML class diagram. The two models are integrated by using a dotted arrow, which links the activity to the produced deliverable (Weerd & Brinkkemper, 2008).

For the CompAS method, two PDD's are depicted (figures 4 and 5). The method has two PDDs because it consists of two different phases, according to Douta et al. (2009), these phases are independent from each other, and the domain analyst is free in his choice to apply both phases, or just one of them. As Douta et al. (2009) don't mention specific roles for different parts of the method, the role is fitted in the diagram. One can assume that all activities in the method are performed by one person, by Douta et al. (2009) called the domain analyst. This person should however have extensive knowledge of the domain, but can, if necessary, get help from a domain expert.

The various activities and deliverables in the PDD are described in tables, an explanation of the various activities of the two phases of the CompAS method is depicted in tables 1 and 2 (activity tables for phase A and B). As previously mentioned, phase A and B can be performed on their own, therefore the activities are described in two separate activity tables. The mentioned concepts in the diagrams are depicted in table 3 (concept table).

**Figure 4: Process-deliverable diagram phase A: Functional evolution based commonality and variability identification**



**Table 1: Activity table phase A**

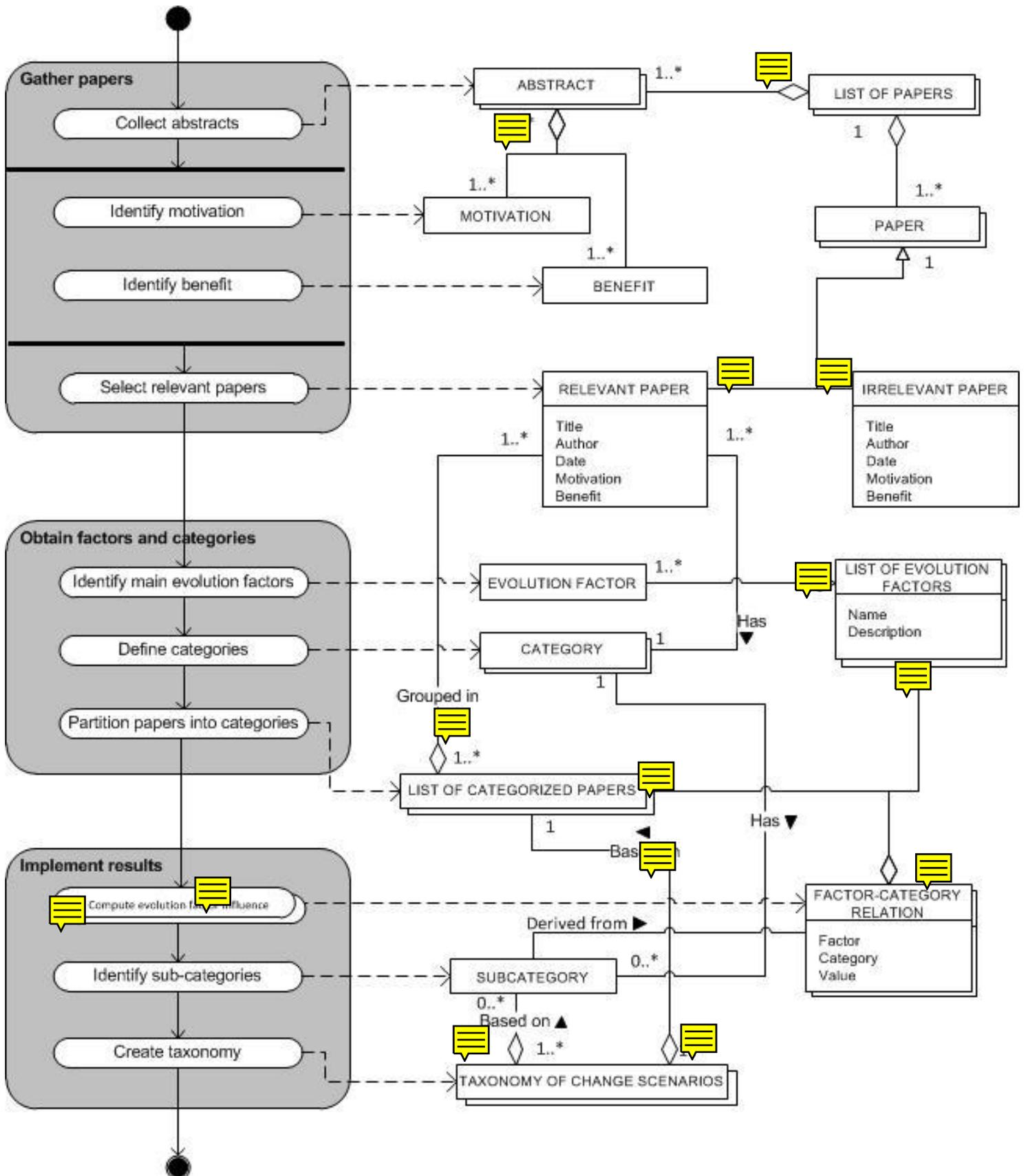
Table is well written. I can't find anything wrong with it.

Activity	Sub-activity	Description
Obtain functional descriptions	Gather abstracts	ABSTRACTS within the domain are gathered and are put in a LIST OF PAPERS. Doua et al. (2009) do this at a conference of the specific domain.
	Select relevant papers	ABSTRACTS are read, and the PAPER is defined as either a RELEVANT PAPER or an IRRELEVANT paper. IRRELEVANT PAPERS are omitted for the remainder of the method.
	Get functional description	From the selected RELEVANT PAPERS, FUNCTIONAL DESCRIPTIONS of technologies in the domain are identified.
	Divide descriptions into subfamilies	The FUNCTIONAL DESCRIPTIONS belong to a category of technology in the domain. The FUNCTIONAL DESCRIPTIONS are divided into these categories.
Obtain functionalities	Define functionality	Functionalities which are provided to assist the user in performing a step in the domain are defined, Doua et al. (2009) define this as a FEATURE
	Extract functionalities	The functionalities which form a FEATURE are conceptualized and FEATURES which belong to the same sub-family are bundled in a FEATURE SET.
Make matrices	Compute metrics	METRICS are calculated from the FEATURE SET, examples include percentage of systems including a feature and distribution over years of systems containing the feature (Doua et al., 2009).
	Visualize evolution matrix	From the METRICS, which contain information about the width (percentage of systems including the feature), height (distribution over years of systems containing this feature), position (year on the y-axis and functionality on the x-axis) and color (percentage of system descriptions that belong to a family in a given year) of a FEATURE SET, software is used to visualize an evolution matrix (Doua et al. (2009) use the Qt software of Trolltech).
Analyze matrices	Identify evolution patterns	From the results visualized in the EVOLUTION MATRIX, EVOLUTION PATTERNS are identified. For this, a table is used explaining the correlation between evolution patterns and commonality and variability
	Deduce commonality	From the EVOLUTION MATRIX, COMMONALITY PROPERTIES, being similarities in occurrences, between the FEATURES in a specific FEATURE SET are deduced. The results are reported in the COMMONALITY AND VARIABILITY REPORT, which is the main deliverable of phase A.

	Deduce variability	From the EVOLUTION MATRIX, VARIABILITY PROPERTIES, being differences in occurrences, between the FEATURES in a specific FEATURE SET are deduced. The results are reported in the COMMONALITY AND VARIABILITY REPORT, which is the main deliverable of phase A
--	--------------------	---

Your activity side is rather simple to follow and mostly correct, however you should revise your deliverables side, I still found many notational errors.

**Figure 5: Process-deliverable diagram Phase B: Business-oriented variation capture**



**Table 2: Activity table phase B**

looks clean

Activity	Sub-activity	Description
Gather papers	Collect abstracts	ABSTRACTS within the domain are gathered and are put in a LIST OF PAPERS. Douta et al. (2009) do this at a conference of the specific domain.
	Identify motivation	From reading ABSTRACTS motivations for the design and/or use of the described contribution of the paper is identified as MOTIVATION.
	Identify benefit	From reading ABSTRACTS discussions of the benefits of the paper are identified as BENEFITS.
	Select relevant papers	ABSTRACTS are read, and the PAPER is defined as either a RELEVANT PAPER or an IRRELEVANT paper. IRRELEVANT PAPERS are omitted for the remainder of the method.
Obtain factors and categories	Identify main evolution factors	Non-functional reasons why systems in the domain were and will be modified and evolved are identified as EVOLUTION FACTORS and are bundled in a LIST OF EVOLUTION FACTORS
	Define categories	CATEGORIES within the domain are identified. Examples from Douta et al. (2009) include 3D modeling and navigation within the computer assisted orthopaedic surgery domain.
	Partition papers into categories	The RELEVANT PAPERS are partitioned into the identified CATEGORIES of the domain at hand.
Implement results	Compute evolution factor influence	For each CATEGORY, the level of influence from each EVOLUTION FACTOR is estimated (1), resulting in a FACTOR-CATEGORY RELATION, visualized by Douta et al. (2009) as a radar graph.
	Identify sub-categories	In the TAXONOMY, SUBCATEGORIES are described which will evolve because of specific evolution factors. These SUBCATEGORIES are derived from the FACTOR-CATEGORY RELATION.
	Create taxonomy	SUBCATEGORIES that match the EVOLUTION FACTORS of each CATEGORY are depicted in a TAXONOMY OF CHANGE SCENARIO.

**Remarks**

1. Douta et al. (2009) do not elaborate on how this estimation is done. I assume they  count how many times a specific evolution factor is named in papers within a certain category.

**Table 3 – Concept table**

good section, practically no remarks

I like the phase-labeling

Concept	Description	Phase
LIST OF PAPERS	A collections of the gathered PAPERS.	A, B
PAPER	A paper form a scientific source, describing system functionalities from the specific domain.	A, B
RELEVANT PAPER	A PAPER which is determined to be relevant by the domain analyst. It is relevant when it contains descriptions on functionalities in the domain. A PAPER contains the attributes title, author, date, textual motivation and benefit.	A, B
IRRELEVANT PAPER	A PAPER which is determined to be irrelevant by the domain analyst because it does not contain functionality descriptions on the domain at hand.	A, B
FEATURE	A FEATURE defines the functionalities needed to perform a step in the domain at hand.	A
FUNCTIONAL DESCRIPTION	A detailed description of a FEATURE, explaining all functionalities within this FEATURE leading to the successful completion of a step within the domain. It contains the attribute name.	A
FEATURE SET	A FEATURE SET consists of one or more FEATURES which belong to the same sub-family within the domain at hand.	A
METRIC	METRICS are calculated to determine where to put a FEATURE in the visualization of the EVOLUTION MATRIX. A Metric contains the attributes position, width, height and color.	A
EVOLUTION MATRIX	A visualization of the evolution of specific FEATURE within the specific domain. METRICS used in this visualization are position, width, height and color.	A
EVOLUTION PATTERN	A pattern which can be derived from the EVOLUTION MATRIX, it describes the evolution of a certain FEATURE over time.	A
COMMONALITY PROPERTY	The value of how far a certain FEATURE is common between categories.	A
VARIABILITY PROPERTY	The value of how far a certain FEATURE is variable between categories.	A
COMMONALITY AND VARIABILITY REPORT	A report which explains the found COMMONALITY PROPERTIES and VARIABILITY PROPERTIES.	A
ABSTRACT	Part of the definition given by Graetz (1985) explains the best how this concept should be viewed in this context ‘the abstract is a time saving device that can be used to find particular parts of the article without reading it’. Douta et al. (2009) use the ABSTRACTS to determine whether or not a paper is relevant.	B
MOTIVATION	The MOTIVATION for the design and/or use of the described contribution of a PAPER given by the author (Douta et al., 2009).	B
BENEFIT	The claimed BENEFIT of the described contribution of a PAPER by the author.	B

EVOLUTION FACTOR	A non-functional reason why systems within the domain were and will be modified and evolved (Douta et al., 2009).	B
LIST OF EVOLUTION FACTORS	A LIST OF EVOLUTION FACTORS contains one or more EVOLUTION FACTORS and is used as a basis for the FACTOR-CATEGORY RELATION. It contains the attributes name and description.	B
CATEGORY	A group with a name which can bind RELEVANT PAPERS sharing features.	B
LIST OF CATEGORIZED PAPERS	RELEVANT PAPERS grouped together in a CATEGORY based on shared features.	B
FACTOR-CATEGORY RELATION	In Douta et al. (2009) shown as a radar graph, a FACTOR-CATEGORY RELATION is an overview showing which EVOLUTION FACTOR is of importance to which CATEGORY.	B
SUBCATEGORY	A SUBCATEGORY is a part of a category which is used in the TAXONOMY OF CHANGE SCENARIOS because it is derived from main evolution factors.	B
TAXONOMY OF CHANGE SCENARIOS	A TAXONOMY OF CHANGE SCENARIOS highlights how each of the estimated most influential evolution factors contributes to the functional and technical variations introduced in the related functionality of the domain (Douta et al., 2009).	B

## 4. Related literature

The origins of the CompAS analysis method are in the field of component based software development. This becomes clear when one understands that the method tries to identify common and variable components of software systems in a particular field in order to improve efficiency of software engineering via software reuse. Capretz, Capretz and Li (2001) define component based software development as a set of prebuild, standardized software components available to fit a specific architectural style for some application domain. As the CompAS method is used to identify commonalities between software systems, it can be used to determine reusable software components and thus, save time and money when developing component based software systems.

The deliverable of the first phase of the CompAS method is the evolution matrix. The implementation of this method is explained in a paper by Lanza (2001). Two useful approaches to reduce complexity in software evolution are software visualization and software metrics (Lanza, 2001). In his paper, Lanza (2001) combines these two approaches into the evolution matrix, which allows for a quick understanding of the evolution of an object-oriented system at system and class level.

The second phase of the CompAS method focusses on the gathering of non-functional requirements. A rather simple definition of non-functional requirement is given by Gilman (2007): A non-functional requirement is an attribute of or a constraint on a system. However, more elaborate definitions exist, for example given by Wiegers (2003): A non-functional requirement is a description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behaviour. Dousta et al. (2009) argue that the domain analyst must identify non-functional requirements that motivate the implementation of new functional variations.

As software reuse and component based software development are approaches to lower development time and costs, several methods have been developed of which CompAS is one. One popular domain analysis method is the one described by Kang, Cohen, Hess, Novak and Peterson (1990) called Feature Oriented Domain Analysis (FODA). This method supports reuse at the functional and architectural levels. Via domain analysis, the common functionality and architecture of applications in a domain, called domain products, are produced. The main goal of this method is the development of generic, widely applicable domain products. Abstracting away factors that make one application different from another can result in the genericness. In a later paper of Kang, Lee and Donohoe (2002) this approach was extended to the Feature oriented Reuse Method (FORM), which consists of two major processes. The first process, asset development, consists of analysing a product line and developing architectures and reusable components based on analysis result. The second process, product development, includes analysing requirements, selecting features, selecting and adopting an architecture and adapting components and generating code for the product. (Kang et al., 2002). Comparing these methods to CompAS we can say that FODA and FORM are more general methods whereas CompAS lends itself more to the medical domain. However, the second phase of CompAS, developing a taxonomy, is a method not used in FODA and FORM, and therefore a strong point of CompAS.

The CompAS method is developed by Dousta et al. (2009), and therefore this paper is the first mentioning of this method. As previously mentioned, this is also the only reference to this method at this moment. Dousta et al. (2009) perform a case study of the CompAS method within the COAS field, with the aim of finding common system components through domain analysis, in order to give system developers the possibility of software reuse, which saves time and money. Even though there are currently no references to this method in another domain than CAOS as described by Dousta et al. (2009), one can imagine this method being useful in other software system domains. The evolution matrix used in this method gives an overview of the evolution and commonality of certain software features. As this method gives possibilities for software reuse, system developers in the field of e.g. operating system software and gaming could possibly take advantage of this method.

Douta et al. (2009) stress the fact that the currently presented work can only be used as a tool to support and strengthen one part of domain analysis. The method is lacking the dimension that constitutes dependency handling. Future work on this method can therefore include evolving the presented taxonomy into an ontology, in order to provide the possibility of modelling dependencies. Chandrasekaran, Josephson and Benjamins (1999) define ontologies as “content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge”. In the CAOS domain of Douta et al. (2009), these objects could for example be change scenarios and surgical procedures.

## 5. References

- Anton, A.I., Potts, C. (2003). Functional paleontology: the evolution of user-visible system services. *IEEE Transactions on Software Engineering*, 29(2), 151 - 166.
- Capretz, L.; Capretz, M.; Li, D. (2001) Component-Based Software Development. *The 27th Annual Conference of the IEEE*, vol. 3, 1834-1837.
- Chandrasekaran, B., Josephson, J.R., Benjamins, V.R. (1999) What are ontologies, and why do we need them?. *Intelligent Systems and their Applications, IEEE*, 14(1), 20-26.
- Douta, G., Talib, H., Nierstrasz, O., & Langlotz, L. (2009). CompAS: A new approach to commonality and variability analysis with applications in computer assisted orthopaedic surgery. *Information and Software Technology*, 51(2), 448-459.
- Fantechi, A., Gnesi, S., John, I., Lami, G., Dörr, J. (2003) Elicitation of use cases for product Lines. *Proceedings of the 5th International Workshop on Product Family Engineering (EPF)*, Siena, 152-167.
- Glinz, M. (2007) On non-functional requirements. *Proceedings 15th IEEE International Requirements Engineering Conference*, Delhi, 21–26.
- Graetz, N, 1985, Teaching EFL students to extract structural information from abstracts. *Reading for Professional Purposes: Methods and Materials in Teaching Languages* Leuven: Acco, pp. 123 – 135
- Kang, K.C., Cohen, S., Hess, J.A., Novak, W.E., Peterson, S.A. (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. *Technical Report*, Software Engineering Institute, Carnegie–Mellon University.
- Kang, K.C., Lee, J., Donohoe, P. (2002) Feature-Oriented Product Line Engineering. *IEEE Software*, 19(4) 58-65.
- Lanza, M. (2001). The evolution matrix: recovering software evolution using software visualization techniques. *IWPSE '01 Proceedings of The 4th International Workshop on Principles of Software Evolution*. New York, New York, USA pp. 37-42.
- Parnas D.L. (1976). On the design and development of program families. *IEEE Transactions on Software Engineering*, 2(1), 1–9.

Szyperski, C. (1998). *Component Software – Beyond Object-Oriented Programming*. Reading, MA: Addison-Wesley.

Weerd, I. van de, & Brinkkemper, S. (2008). Meta-modeling for Situational Analysis and Design Methods. In M. Syed, & S. Syed (Eds.), *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications* (pp. 33-58). Hershey: Idea Group Publishing.

Wiegers, K.E. (2003) *Software Requirements*, 2nd ed., Redmond, Washington: Microsoft Press.

## 6. Appendix

### Template for a change scenario taxonomy

Depicted in Figure 6 is a template for the creation of a change scenario taxonomy as created by Doua et al. (2009) in phase B of the CompAS method: Business-oriented variation capture.

A. Category	
	A.1 Subcategory resulting from evolution factor
	A.2
	A.3
B. Category	
	B.1 Subcategory resulting from evolution factor
	B.2
	B.3
C. Category	
	C.1 Subcategory resulting from evolution factor
	C.2

**Table 4: Taxonomy of change scenario template**

In the right column of this taxonomy template, the categories are placed, these are the categories in which the relevant papers were categorized. In the left column, the subcategories are placed which are the result of the main evolution factors of the category.