Journal of Universal Computer Science, vol. 20, no. 6 (2014), 842-858
submitted: 1/7/13, accepted: 14/2/14, appeared: 1/6/14 © J.UCS

Using and Extending Formal Concept Analysis to Visualise Variability during Requirements Engineering

Tom Huysegoms
(Faculty of Economics and Business, KU Leuven, Leuven, Belgium
Tom.Huysegoms@kuleuven.be)

Monique Snoeck
(Faculty of Economics and Business, KU Leuven, Leuven, Belgium
Monique.Snoeck@kuleuven.be)

Guido Dedene
(Faculty of Economics and Business, KU Leuven, Leuven, Belgium
Guido.Dedene@kuleuven.be)

Antoon Goderis
(KBC Global Services, Brussels, Belgium
Antoon.Goderis@kbc.be)

Frank Stumpe
(KBC Global Services, Brussels, Belgium
Frank.Stumpe@kbc.be)

Abstract: Research on variability in software artefacts is something which is already studied extensively in research. The visualisation of variability is one aspect of this research, and results like e.g. feature diagrams are well-known and well-spread. When it concerns the origin of the variability within the phase of requirements engineering, research is much scarcer. A visualisation technique for both representing the origin and the amount of variability in requirements is not readily available in research. This paper provides a way to represent the origin of variability in requirements with the aid of a technique called formal concept analysis (FCA). Additionally the support that FCA can provide for variability related decisions during (early) requirements engineering is also depicted in this paper. Proof of the usability of FCA for the visualization, and documentation, of variability is shown with the aid of a real-life case study. FCA is also applied in the real-life case study to check the compatibility of FCA as a visualization method to support variability decision making during requirements engineering.

Keywords: Requirements Management, Variability Management, Formal Concept Analysis, Harmonization, Variabilization
Categories: I.3.8

1 Introduction

As long as all the needs for a certain software system are the same over all of the (future) stakeholders, the job of the requirements engineer is quite straightforward. The problems start to occur when the stakeholders have differences in their needs

amongst each other. Combining the different needs of all the stakeholders during requirements engineering (RE) into one set of requirements is when variability is formed based on these different needs of the stakeholders. Dealing with the variability by acknowledging and coping with it, or by ignoring it, is a form of management that needs to be done by the requirements engineer. Variability in the context of RE is here thus seen as the existence of differences in stakeholders' needs (and requirements) for a particular software system. This definition is in line with the definition of variability proposed by Kim, Her and Chang [Kim, 05]. Variability management can be defined, as stated by Schmid and John [Schmid, 04], as "encompassing the activities of explicitly representing variability in software artefacts throughout the lifecycle, managing dependences among different variabilities, and supporting the instantiation of the variabilities". One thing which is very important to take into account is that differences in stakeholders' needs only can be considered variability in requirements when these differences are explicitly taking into account into the requirements. When the differences in the stakeholders' needs are ignored, no variability will be present in the requirements. If this purposely ignoring difference in stakeholders' needs should still be considered a form of variability management or not is open for debate, given the idea that only acknowledged differences are considered variability. Variability management in this paper will always be considered in the situation where some amount of variability is present.

The importance of variability management in the context as described above has already been subject of literature since it was studied in multiple different software development paradigms. Some sort of starting point can be found in the seminal work of Kang, Cohen, Hess, Novak and Peterson [Kang, 90]. This seminal work concerning variability frameworks dates from 1990 and is called the Feature-Oriented Domain Analysis (FODA) specification. Multiple variability management approaches and variability frameworks have been developed since. Central in many of these frameworks is the concept of a feature diagram. Feature diagrams can be defined as a family of popular modelling languages used for engineering requirements according to Schobbens, Heymans and Trigaux [Schobbens, 06]. The central idea of feature diagrams is to visualize the decomposition of the root feature, which represents the main function of the to-be-created software system, into more detailed features. Feature diagrams are often used throughout many of the phases of software engineering, and quite often in RE to model common and variable aspects of the to-be-created software system.

Languages like feature diagrams offer a formal representation of the variability. The information represented by this representation is mainly on the amount of variability and how the variability is interrelated. The representation does not provide information on the necessity of the inclusion of the variability however, it does not represent the origin of the variability. The focus of this paper is on the origin of the variability before it is decided to be a necessary part of the to-be-created software system, so the focus is on the decisions to include the differences in needs as variability in the requirements. These decisions to include variability will be called the variability decisions in the remainder of this paper. More specifically the focus of the paper is a focus on the visualization of the impact of these variability decisions, the decisions whether or not to include variability in the to-be-created software system requirements, is pursued in order to somewhat facilitate the job of the

requirements engineer during variability management. The second section of this paper delves into existing research in the field of variability management and states the research question based on an identified gap in existing research. The third section defines the variability-related decisions of variability management that need to be taken during RE by the requirements engineer, and positions these decisions against the general RE process. Section four proposes an approach to visualize and document the origin of the variability. Furthermore, it visualizes variability-related decisions and their impact on the variability as they are taken during RE through the use of formal concept analysis. Section five empirically shows the added value that formal concept analysis can provide by means of application on a real-life project. Section six goes into detail on a caveat made in section four and tests whether the assumption of a flat requirements structure can be dropped without the visualization technique losing too much of its value. Section seven gives a conclusion as well as directions for future research.

2 Existing Requirements Variability Visualization Research

The importance of variability management specifically within the field of RE is acknowledged as an important contemporary research topic. Because it is so important, the research on variability management in RE can be positioned in the dimensional representation of RE by Pohl [Pohl, 93]. Variability management concerns the "agreement" dimension. The objective related to this dimension is increasing what Pohl calls the common system specification. This common system specification is that part of the requirements where the different stakeholder's views have been unified into one agreed upon specification. However, at the same time Pohl acknowledges that some requirements will remain on which none or only partial agreement can be reached between stakeholders. How to manage this absence of agreement, and thus variability, is exactly what will be studied in this paper.

In the field of research on variability management during RE there are several frameworks available to manage and visualize the variability. Probably the most popular one is feature modelling, with visualization through feature diagrams, as mentioned before [Schobbens, 06]. Another research effort on managing variability is known as viewpoint-oriented RE and leans more closely to the ideas of Pohl. Viewpoint-oriented RE is based on the fact that stakeholders can have different views. In order to handle requirements which are identical between stakeholders, also called crosscutting requirements, the concept of aspects is used [Pu, 09]. Resolving crosscutting requirements is called viewpoint resolution [Sampaio do Prado Leite, 91], this resolution of variability is focused however on implementation time and not on design time. While both feature modelling and viewpoint-oriented RE offer a visualization technique, neither of them offers a representation of the origin of variability, as already mentioned before. The same issue is present in the research on decision modelling, which is a framework for capturing the kind of decisions that need to be made in order to arrive at a specific product in the product line [Schmid, 11].

There is another strand of research which can be traced back to the concept of a requirements modelling language as introduced in 1986 [Greenspan, 86]. Based on this seminal work several requirements modelling frameworks were developed. The

most renown requirements modelling frameworks which use some kind of visual representation of the requirements in this area of research are the KAOS method [Darimont, 96] and the i* framework [Yu, 97]. Both modelling frameworks focus on goal modelling, albeit both with a completely different focus. The advantage of goal modelling is that it focuses itself on early requirements engineering, so it is not possible to have an implementation bias or a focus on specifications rather than on requirements, as it is the case in feature modelling.

The central premise of the KOAS method is the decomposition of goals into sub-goals, in order to arrive at a level of goals which are readily implementable. The decomposition of the goals can be visualized through a graph representing this decomposition. In this KAOS method there is also room for alternatives through construct of OR-decomposition. This provides the possibility to introduce variability in the goal decomposition, and to represent variability visually to a certain extent. The problem is however that the origin of the variability is not visualized in the decomposition graphs of KAOS, and as a result the method provides insufficient information to support the variability decision making process, just like with feature diagrams although the focus of KAOS is more towards early requirements engineering, there where the variability decisions are taken.

In contrast to the KAOS method and other research mentioned above, the i* framework does provide the origin of the goals, or as they call it the 'why' component of the goals. The i* framework is basically a set of two models, i.e. the strategic dependency model and the strategic rationale model. It is the strategic rationale model that provides the link between the goals and why these goals are needed. As a result, the need for requirements can be visually modelled. The drawback of the strategic rationale model, and thus of the i* framework is that the visual representation of variability is almost completely lost due to the focus on the origin of the requirements. Therefore, one can state that although the i* framework solves the problem that KAOS has with representing the origin of the variability at the same time it loses some of the power of KAOS to represent variability in goals. In the context of this research some kind of combination of both representation of the variability and the origin of this variability is needed in order to support the communication during the variability decision making process.

A limitation of all the research mentioned above, besides the i* framework, is that although these frameworks focus on providing an overview of the variability in the requirements, they always start from the assumption that the decisions concerning what variability to keep and what variability to put out of scope are already taken. These frameworks do not provide a form of systematic documentation of these variability decisions taken. The central question in the research in this paper therefore revolves around the visualization of the origin of the variability in order to be able to support the variability decisions between the different stakeholders during RE. Without the visualization of the origin of the variability, supporting the variability decisions is impossible. More specifically the research question addressed in this paper is: "How can variability be visualized to make the impact of variability decisions more clear to all stakeholders?" If the impact of variability decisions can be visualized, this visualization also serves as a form of documentation which can be used to support the choices which were made ex post.

3 Harmonization and Variabilization

Looking at the RE process, many possible ways of identifying phases in this process can be found. According to Nuseibeh and Easterbrook requirements engineering phase consists of following core activities: requirements elicitation, requirements modelling and analysis, requirements communication, requirements agreement, and requirements evolution [Nuseibeh, 00]. Van Lamsweerde sees the requirements engineering phase as a cyclical process starting with domain understanding and requirements elicitation, continued with requirements evaluation and negotiation, followed by requirements specification and documentation, and ending with requirements consolidation before going back into the domain understanding and requirements elicitation once again to start the circle over [van Lamsweerde, 09]. While Nuseibeh and Easterbrook define five sub phases of requirements engineering van Lamsweerde only defines four. Upon closer inspection it can be seen however that the fifth sub phase of Nuseibeh and Easterbrook is more or less equivalent to the start of a reiteration of the requirements engineering phase by van Lamsweerde. This iterative cycle view is by far the most common view on RE. Another possible division of RE, one more useful in this context, is to split the process up into two big parts or concerns. This division is based on the concept of the RE problem as defined by Zave and Jackson [Zave, 97], and later refined by Jureta, Mylopoulos and Faulkner [Jureta, 08]. The RE problem is described as the need to obtain specifications that fulfil the requirements of a to-be-created software system within a particular domain context.

The first part or concern, (forming the RE problem) is the central focus during requirements analysis when all the statements from the involved stakeholders are gathered and brought together. Activities like requirements gathering and requirements elicitation can also typically be linked to this first concern. Once all statements are collected, the requirements engineer has a complete overview of how the specific project's RE problem looks like for the to-be-created software system.

The second part or concern then comes into play. This concern (solving the RE problem) is central during requirements specification when the requirements engineer develops a set of specifications to which the to-be-created software system should adhere to. The objective at this point in time is to take into account as much as possible the statements issued by the stakeholders. Requirements validation can also typically be linked to this second RE concern. The remark can be made that there are RE activities which cannot be clearly be linked to one of the two concerns described here. Requirements management for instance typically encompasses both RE concerns. Nevertheless, a focus on these key concerns is important in variability management, since RE becomes complex when variability is introduced. Variability management boils down to taking two decisions. These decisions are called harmonization and variabilization and are explained in more detail in [Huysegoms, 13].

Harmonization starts after gathering the requirements of the to-be-created software system from the demand side stakeholders, the stakeholders who utter requirements, with the identification of common requirements in order to obtain a view on the amount and areas of variability in the requirements. It should be more than just a passive process of comparing requirements by the requirements engineer alone. When the demand side stakeholders and the requirements engineer engage

actively in discussions on the requirements, these requirements (and the underlying variability) are put to the test. It is more than just an observation in the sense that harmonization also involves negotiation with the demand side stakeholders in order to obtain more common requirements. Harmonization impacts Pohl's "Agreement" dimension [Pohl, 93]. This is beneficial to the demand side stakeholders in a sense that requirements shared by more stakeholders typically have a higher chance to be retained and implemented in the to-be-created software system. The systematic discussions will also have impact the supply side stakeholders, the stakeholders who will be providing the to-be-created soft-ware system. The benefits of a systematic approach for the supply side are known to be: opportunities for rapid new development due to reuse, decreased development costs and decreased effort that needed to implement instances of the to-be-created software system [Coplien, 98].

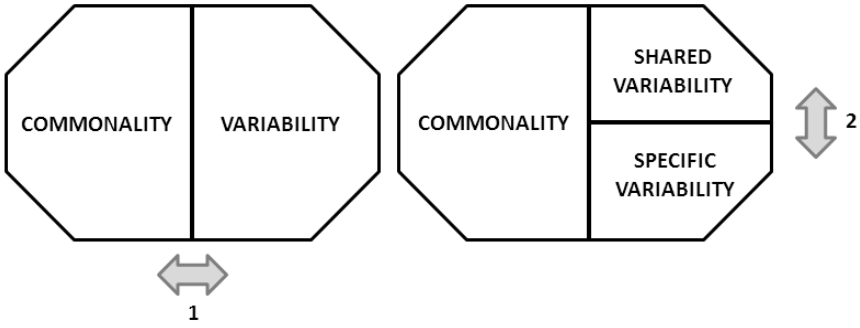

Figure 1: Harmonization (1) and Variabilization (2)

Variabilization is the second decision concerning the variability, which can be taken after the harmonization is done. Variabilization is a negotiation on which requirements not common for all demand side stake-holders should be retained within the RE problem of the to-be-created software system during further development. Indeed, not all collected requirements will also be effectively incorporated into the to-be-created software system. Some requirements are not entirely common to all stakeholders, but still shared among some of them. Such shared variability can be retained, while other, more specific, requirements will be considered out of scope for the to-be-created software system. Once again, like the harmonization, the variabilization decision can better be taken consciously by the requirements engineer with all possible input of the stakeholders, since they are not only paying for the to-be-created software, but they will also have to cope themselves with the variability they deem important but is not retained to be supported by the to-be-created software. Figure 1 represents both harmonization (1) and variabilization (2) decisions.

Now that both variability decisions are known, one can try to link them within the RE concerns defined earlier. The variability decisions do not link to the concern of 'forming the RE problem' since the overview of the totality of the requirements is only obtained after the RE problem is formed. Without an overview of the requirements, and the variability herein, it would be impossible to decide anything in terms of harmonization or variabilization. The concern of 'solving the RE problem' on the other hand may seem a good candidate to link the variability decisions to at

first sight. However for this concern the set of requirements that needs to be satisfied is considered fixed, albeit that some can be mandatory and some can be optional. The problem is that during harmonization requirements can still shift due to the discussions between the requirements engineer and the stakeholders. These shifts make it very difficult to solve the RE problem optimally a priori. It can thus be said that the variability decisions don't fit well to either of the RE concerns. This could explain why contemporary research rooted in the research of Zave and Jackson [Zave, 97] has difficulties providing explicit visualization support for the variability decisions. The solution is to add some kind of third step located in between defining and solving the requirements problem.

The requirements engineer should be the one responsible for the 'harmonization and variabilization' concern. He or she will need to communicate with all the other stakeholders on the variability in the requirements so that conscious decisions on the variability can be made. Prior research [Huysegoms, 11] has shown that the absence of conscious decisions and communication about variability, may lead to failed software projects. Visualization is often an effective way to support communication as it provides support for increasing the focus on the appropriate issues. At the same time the visualization can be used later on as a form of documentation on the variability decisions. The use of the visualization technique to solve the complete traceability problem [Gotel, 94] in RE is however out of scope in this paper.

4 Formal Concept Analysis for Requirements Engineering

Formal concept analysis (FCA) was created as mathematical theory with the purpose to identify concepts and to create a mathematical order among these concepts. A good way to state the aim of FCA can be found in the work of Wille [Wille, 05]: "The aim and meaning of FCA is to support the rational communication of humans by mathematically developing appropriate conceptual structures which can be logically activated." This clearly states that FCA can be used to support the rational communication by humans, which is the objective of this research on variability. While originally mostly used in mathematical theory, the visualization aspects of FCA became more and more popular over the years due to the fact that they are easily readable. More on the ease of readability and communication capabilities of FCA can be found in the work of Eklund, Ducrou and Brawn [Eklund, 04]. Nowadays the visualization capacities of FCA are used not only to purely visualize data, but also to mine knowledge out of data. An overview on all the uses of FCA can be found in the work of Poelmans, Elzinga, Viaene and Dedene [Poelmans, 10]. Within this work it can be seen that FCA has not much been used in the field of RE, which strengthens the choice for FCA as a novel approach to visualize the variability decisions in RE. In order to comprehensibly explain the technique of FCA a small example is given below.

The table on the left of Figure 2 lists certain types of fruits with their colour and flavour. The types of fruit are the objects O = {apple, banana, cherry, lemon}. The colour and the flavour of the fruit are called the properties A = {red, green, yellow, sweet, sour}. Objects and their properties are related as indicated by the X's in the table. The whole of objects (O), properties (A) and their one-to-one pair wise relations (I) is called the formal context (O, A, I). An example of such a relation is the pair

{apple, red}. This formal context (O, A, I) will be mathematically transformed into a structure called a concept lattice. The transformation of the formal context into a concept lattice is one where no information is lost. The concept lattice obtained after transformation can be represented visually to facilitate communication on the formal context. The graph representing the concept lattice belonging to the formal context on the fruit can be seen on the right side of Figure 2.

	Red	Green	Yellow	Sweet	Sour
Apple	X	X		X	X
Banana			X	X	
Cherry	X			X	
Lemon			X		X

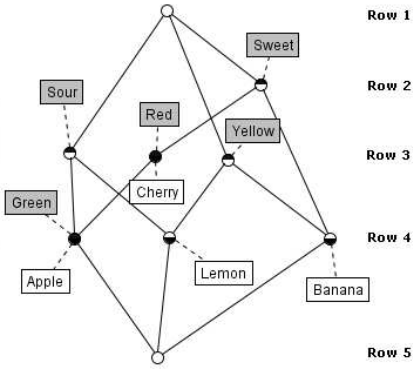

Figure 2: The Formal Context of Fruit Types

A set of objects X and a set of properties Y together form a formal concept (X, Y) of the formal context (O, A, I) if and only if each object of X has all the properties of Y, and if each object which does not belong to X has at least one property which is not in Y. In the case the above is true, X will be called the extent of the formal concept, and Y will be called the intent of the formal concept. An example of a formal concept (X, Y) for the formal context on fruit is the one where X = {apple, cherry} is the extent and Y = {red, sweet} is the intent. Once all formal concepts are identified, they can be partially ordered. If there are e.g. two formal concepts (X, Y) and (V, W) one can order them as (X, Y) ≤ (V, W) when the set objects X is a subset of the set of objects V. Alternatively, they can also be ordered as (X, Y) ≤ (V, W) when the set of properties W is a subset of the set of properties Y. In the formal context on the fruit, it can be seen that e.g. the formal concept (X, Y) with extent X = {apple, cherry} and intent Y = {red, sweet}; and the formal concept (V, W) with extent V = {apple, banana, cherry} and intent W = {sweet} can be partially ordered in the way that (X, Y) ≤ (V, W) since {apple, cherry} is a subset of {apple, banana, cherry} and {sweet} is a subset of {red, sweet}.

This partial ordering of the formal concepts is exactly what is visualized in the graph of the concept lattice. In Fig. 2 the formal concept (X, Y) = ({apple, cherry}, {red, sweet}) is the represented by the second node on the third row, while the formal concept (V, W) = ({apple, banana, cherry}, {sweet}) is represented by the node on the second row. The formal concept (V, W) is located higher than the formal concept (X, Y) which is the logical representation of the partial ordering (X, Y) ≤ (V, W). The advantage of creating a concept lattice is this visualization of that partial ordering and the resulting visualisation of the level to which properties are shared by objects. The latter is exactly what is usable in the context of variability management to

communicate more clearly about requirements and the variability in requirements. In order to be able to represent the requirements in a lattice the stakeholders which utter the requirements will be the concepts, and the requirements themselves will be the properties.

5 QBS, a Real Life Formal Context

In order to test the visualization capabilities of FCA for variability a case study was conducted. This case study focused on QBS, a customer relationship management system within the context of a major national insurance company which works with local insurance agencies. All the insurance agencies work semi-independently with their own business vision and strategy and can vary both in size and types of customers they wish to address. In total there are over 300 local agencies which, although they work semi-independently, still are dependent on the central headquarters' IT department for their software systems. Since the current version of QBS became outdated a newer version was to be made.

	# of requirements (# requirements represented from total amount requirements)
Total amount of requirements	**233**
Total amount of unified requirements	**66 (233 of 233)**
Unified requirements shared ≥ 50%	**16 (82 of 233)**
Unified requirements shared <50%	**50 (151 of 233)**
Unified requirements shared 100%	**1 (12 of 233)**

Table 1: Overview of the QBS Requirements

This provided an opportunity to execute the RE phase with an appropriate focus on the variability decisions. The requirements gathering was done through individual interviews with a representative set of insurance agents used by the company for all IT projects. A list of 233 uttered requirements originating from 8 different insurance agents was extracted. The issue was that some requirements within the requirements list were actually duplicates of each other since some of the requirements were stated by multiple agents in different wordings, like mentioned by Pohl [Pohl, 93]. Therefore a normalization step was performed during which duplicate requirements were identified and a 'normalized' formulation of these requirements was defined.

After this normalization only 66 unified requirements remained, which were shared by one or more agents, like mentioned in Table 1. These 66 unified requirements represent the 233 'original' requirements, as mentioned between brackets. Out of these 66 unified requirements 16 were shared by 50% or more of the agents. These 16 unified requirements represent 82 out of the 233 original requirements. The remaining 50 unified requirements were shared by less than 50% of the agents and represented 151 of the original requirements.

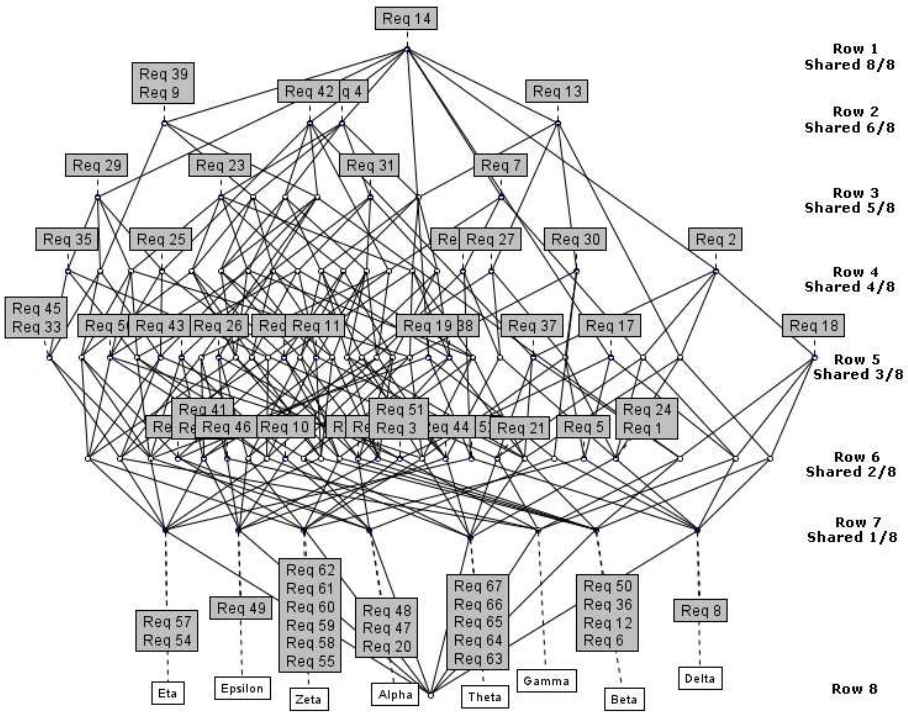

Figure 3: The QBS Unified Requirements Concept Lattice

As it is not possible to disclose the exact content of all the requirements themselves due to legal reasons, the requirements received a number as a unique identifier by which they can be identified and they will be also referenced through this number in the remainder of the paper. The interviewed agents will be denoted by letters of the Greek alphabet. The resulting concept lattice graph shown in Figure 3 provides a clear overview of the requirements and the amount and structure of the variability present in the requirements.

While the concept lattice graph may seem quite complex at first, it is fairly easy to extract some interesting observations straight away. Some of the more interesting figures mentioned in Table 1 can be deducted from the concept lattice graph. There is one requirement (req 14) shared by all agents as can be seen on row 1 in Figure 3. There are 22 requirements not shared at all; these are uttered by only a single agent as can be seen on row 7. The 16 requirements linked to nodes on rows 1-4 are shared by at least 50% of the stakeholders.

Besides the information present in Table 1, some additional information on the variability within the requirements can easily be read from the concept lattice graph. Agent Gamma did not state any requirements which were unique for him, which makes him a 'good' stakeholder in a sense that he generates less variability compared

to agent Zeta, who has no less than 6 unique requirements. When harmonization is started it would thus be more beneficial to try to persuade agent Zeta to drop some of his own specific requirements, than it would be to persuade agent Delta or Epsilon, since agent Zeta is more likely to be a bigger source of specific variability in the end. The concept lattice graph hence offers interesting clues to the requirements engineering on what to focus on during the harmonization efforts.

It is also quite easy to perform additional analysis by drawing additional concept lattice graphs on subsets of concepts and/or attributes which you want to study more in depth. A good way to devise subsets of requirements is to delineate categories with in the set of requirements. In the QBS case study several sub-lattices were drawn, based on delineating categories of requirements. The formal contexts for these categories consisted of all the concepts (agents) from the original lattice, but only a subset of the properties (requirements). Here concept lattice graphs are given for both the 'ease of use' type of requirements and the 'data feeds' type of requirements. These two specific categories of requirements are chosen as illustrative examples, since the 'ease of use' requirements contains much variability (the requirements are almost not shared at all) while the 'data feeds' requirements contains much less variability (the requirements are shared by almost everyone). This amount of variability is reflected in the shape of the concept lattice graphs.

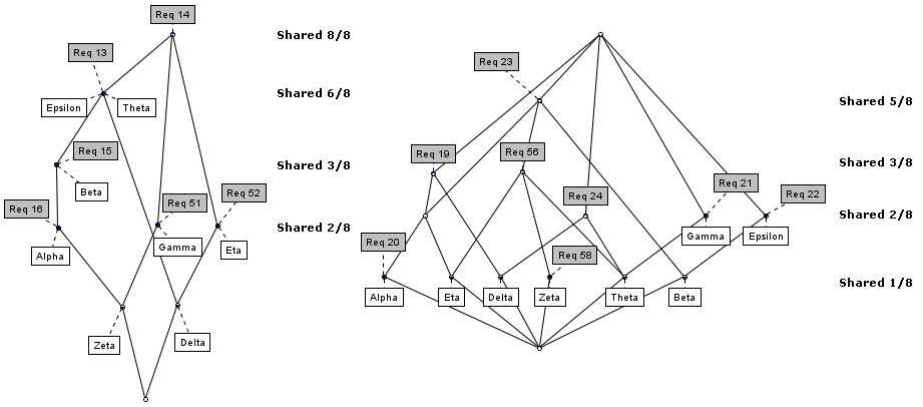

Figure 4: The 'Data Feeds' Lattice (left) and the 'Ease of Use' Lattice (right)

The 'data feeds' concept lattice graph on the left in Figure 4 has a slimmer shape (less formal concepts at the same level in the concept lattice graph) compared to the 'ease of use' concept lattice graph on the right in Figure 4. The 'data feeds' lattice is three concepts wide at most (the level shared by 2/8) while the 'ease of use' lattice is at most six concepts wide (the level shared by 1/8).

The advantage of analysing subsets of requirements according to their category, besides the higher clarity of the graphs themselves, makes it possible for the requirements engineer to quickly focus on the requirement types which contain more variability or are considered of higher priority, and thus demand more attention during the variability decisions sub-phase. Once the lattices representing subsets of requirements are drawn, the same information can be gathered as described for Figure

3 earlier on. For instance, from Figure 4 it can easily be deduced that the ease of use requirements of agent Gamma are a subset of those of agent Theta and that those of agent Epsilon are a subset of those of agent Beta. Agent Gamma and Epsilon are thus more 'harmonized' agents than agents Theta and Beta respectively. Another example in Figure 4 is that agent Epsilon and Theta are completely 'harmonized' with each other concerning the data feeds since they are linked to the same node in the data feeds concept lattice graph. It must be noted that by only analysing subsets of requirements some of the information on the variability is lost, since the interactions between the subsets are not be represented in the graphs of the subsets. Therefore the general graph cannot be left aside in favour of the sub-set graphs.

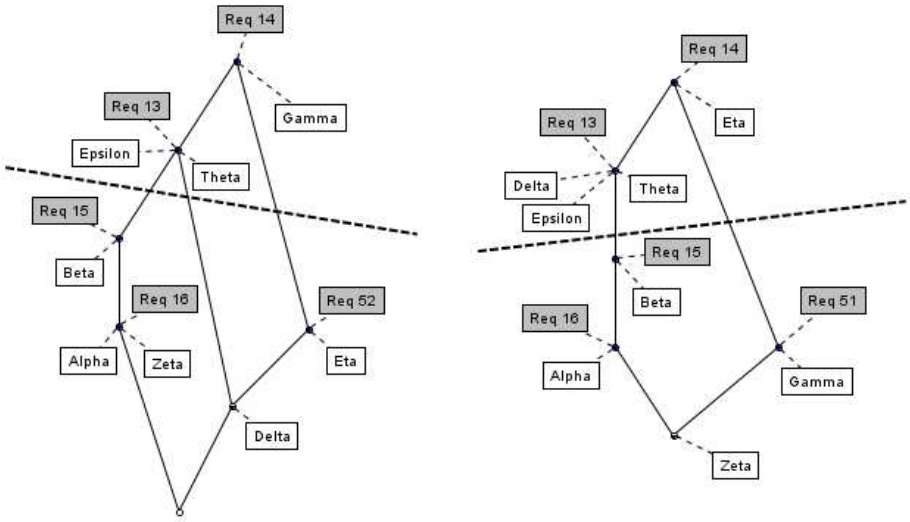

Figure 5: The 'Data Feeds' Lattice without req 51 (left) or without req 52 (right)

Besides the pure visual representation of variability, the effect of harmonization and variabilization decisions can also be easily visualized, and thus documented, through FCA lattice graphs. Harmonization can be seen as trying to bring the requirements higher in the lattice graph through negotiation with the stakeholders, or to make stakeholders dropping requirements if they are too low on the graph. An example is given based on the 'data feeds' concept lattice graph depicted Figure 4. After analysing this graph, requirements 51 and 52 were identified as good candidates to be dropped, by Gamma and Zeta; and by Delta and Eta respectively. Visually this can be easily identified as requirements 51 and 52 are located to the 'side' of the lattice graph compared to the main body of requirements. This resulted in the lattices shown in Figure 5. As can be seen in both of these concept lattice graphs, the removal of one requirement immediately leads to an even more harmonized set of requirements, as the lattice becomes even more slender with a maximum width of only two concepts.

Variabilization can be visualized on the lattices as well. It is essentially positioning some imaginary boundary line within the lattice at a certain level to divide

the requirements which will be supported and the requirements which will be left aside. Typically the 'variabilization boundary line' divides the graph into an upper, supported part (shared variability) and a lower unsupported part (specific variability). Assuming that shared variability is variability shared by at least half of the QBS agents, this variabilization boundary would run were the dotted lines are marked on Figure 5.

6 Dropping the Flat Requirements Structure Assumption

Up until now everything concerning the visualization looks good. The FCA technique, as a visualization technique used here, however also has some significant shortcomings the way it is used in the previous sections. The concept lattice graphs as they are used here have no possibility to deal with different levels of requirements. While in the case of QBS this 'flat' structure of all requirements being on the same level can be more or less assumed due to the fact that all requirements were considered early requirements on the business level, this is certainly not always the case. In reality requirements are more complex and they are often related to each other as being refinements of each other. The KAOS method mentioned before even has such refinement as a central premise. Extending the concept lattice graphs with a construct similar to the one that KAOS uses for refinement in order to be able to show refinement relations amongst requirements is a good direction to further improve the visual representation of variability. Through this requirements refinement an extra dimension to the concept lattices would be added. The problem is that the concept lattices are limited to two dimensions, i.e. the 'object' dimension and the 'property' dimension. A third extra dimension should be introduced now, the 'refinement' dimension. In order to do this a way has to be found to incorporate more information in the lattice.

The extra dimension of requirements refinement can be visualized by somehow using the horizontal dimension in the lattice graph. Since the vertical dimension is already used fully for the partial ordering, the horizontal dimension is the most straightforward option. An example for this can be found in Figure 6. The left hand side of Figure 6 is a detail of Figure 5 with a representation of the refinement of requirement 15 into requirements 15a and 15b with horizontally aligned edges. The problem with this visualization is however that the refined requirements are not part of the original lattice structure (part of the formal context now). When requirements 15a and 15b would be included in the formal context Figure 5 would turn into the concept lattice graph on the right hand side of Figure 6.

The problem with the concept lattice on the right hand side of Figure 6 is that in order to make all refinement lines visible, concept point A should be split up, as now requirement 15 and 15a are on the same concept point, and no refinement edge between them can be shown. This split would also make the fact visible that there are two edges between concept points A and B, one normal concept lattice edge and one refinement edge.

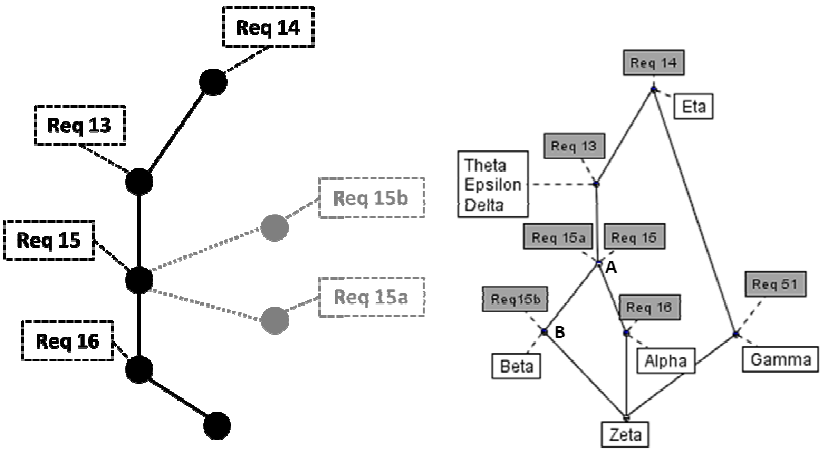

Figure 6: The 'Refinement' Dimension in Concept Lattices

These adjustments would however inherently make the concept lattice graph more complex as more nodes and edges would appear. One must know that these refinement adjustments are only done for one single requirement in the example here, in reality they should be done for all requirements when requirement refinement would be included. As a result, the power of the visualization to readily support communication with non-technical stakeholders would probably drop significantly, and the time investment new stakeholders must make to get to understand the visualization would rise.

Another very important issue that arises from requirements refinement is the levelling of requirements. In the example of Figure 6 it was easy to make o formal context of all the requirements at one level since they were all considered at the same level. In reality however all requirements arrive without any indication on which level they should be situated, and the same stakeholders can even switch levels himself when uttering multiple needs. A solution would be to include all the requirements on all levels into the same formal context and thus dropping the refinement dimension by 'ignoring' it, but this would make the concept lattice very cluttered. At the same time there would be requirements uttered by one stakeholder on one level which may fulfil other requirements (partially) of another stakeholder on a different level. But at this point it might perfectly well be that the second stakeholder is also happy when the requirements of the first stakeholder are fulfilled while his own requirements are not completely fulfilled, since they sufficiently encompass their own needs.

The only way to really solve all the issues mentioned in a way that it remains easily comprehensible is to return to a single level of requirements like it was assumed before. The requirements engineer will then first have to equalize all the requirements at a chosen level before visualizing the variability with FCA. This equalizing can also be seen as part of the iterative aspect of the RE cycle, where the levels of the requirements are being communicated back to the stakeholders which uttered them, and at the same time their opinion on whether requirements of other

stakeholders can suffice their own needs can be questioned. RE thus remains a process where human interaction is needed, but where FCA can surely provide some visual aid in the communication.

7 Conclusion and Future Research

A previous case study [Huysegoms, 13] already demonstrated the importance of good and clear communication during RE. Variability management, which is a substantial part in RE from the moment several stakeholders are involved, is one aspect of requirements management for which the communication is even more key if the variability decisions (as a part of the RE problem) are to be taken in a traceable and objective way. In order to achieve this, a way of visualizing the impact of variability decisions is studied. The technique of formal concept analysis is identified as a good, yet unexplored, potential candidate visualization method, since no mention is made of its use in this particular context in existing surveys on formal concept analysis [Poelmans, 10].

With the aid of a real-life case study within the insurance sector called QBS the value of formal concept analysis is tested. This pilot study resulted in the indication that visualization through formal concept analysis can create value as it is a way to visually identify the points of attention that need to be focused upon when making the variability decisions. The QBS case study however revealed that using formal concept analysis only on the total set of requirements in real life is insufficient, since the lattice can become quite complex. In order to have a better visualization additional lattices can be created based on subsets of requirements divided by their type, which is context depended. It is not good however to only focus on the subset lattices as some information on the interaction between the types of requirements can only be observed in the lattice on all the requirements.

Whereas in the pilot-study the researchers were taking the role of requirements engineers, future research will focus on providing a validation cycle with the actual requirements engineers of the QBS project. Based on the case study the added value of visualizing variability through concept lattices can be shown to practi-tioners in order to generate feedback from them. Next, the utility of the FCA visualization for the stakeholders will also be tested, by investigating how the FCA-graph might influence their willingness to negotiate. Besides validation future research will also look into adding even more information in the visualization. Right now there is no way to visualize conflicting requirements or requirement dependencies. Adding this would improve the value of the variability concept lattices. At the same time the issues mentioned in the sixth section would also benefit from further research.

Acknowledgements

We gratefully acknowledge KBC Global Services NV (member of the KBC Group) for funding this re-search. We also gratefully acknowledge all participants of the case study for their invested time and effort.

References

[Coplien, 98] Coplien, J.,Hoffman, D., Weiss, D., 1998. Commonality and variability in software engineering. IEEE Software 15, p.37-45.

[Darimont, 96] Darimont R., van Lamsweerde A., Formal refinement patterns for goal-driven requirements elaboration, In: ACM SIGSOFT Symposium on Foundations on Software Engineering, p.179-190, 1996

[Eklund, 04] Eklund, P., Ducrou, J., Brawn, P., 2004. Concept Lattices for Information Visualization: Can Novice Read Line Diagrams? Lecture Notes in Computer Science 2961, p. 57-73.

[Gotel, 94] Gotel, O.C.Z. , Finkelstein, A.C.W., 1994. "An analysis of the requirements traceability problem," 1st IEEE International Requirements Engineering Conference. Colorado Springs, USA.

[Greenspan, 86] Greenspan S.J., Borgida A., Mylopoulos J., A requirements modeling language and its logic, Information Systems, 11 (1), p.9-23, 1986

[Huysegoms, 11] Huysegoms, T., Snoeck, M., Dedene, G., Goderis, A., 2011. Requirements for Successful Software Development with Variability: A Case Study. Communications in Computer and Information Science 219, p. 238-247.

[Huysegoms, 13] Huysegoms, T., Snoeck, M., Dedene, G., Goderis, A., Stumpe, F., 2013. A case study on variability management in software product lines: identifying why real-life projects fail. International Journal of Information Systems and Project Management 1, p. 31-42.

[Jureta, 08] Jureta, I., Mylopoulos, J., Faulkner, S., 2008. "Revisiting the Core Ontology and Problem in Requirements Engineering," 16th IEEE International Requirements Engineering Conference. Barcelona, Spain.

[Kang, 90] Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S., Feature-Oriented Domain Analysis (FODA) feasibility study, Software Engineering Institute, Carnegie Mellon University, 1990.

[Kim, 05] Kim, S.D., Her, J.S., Chang, S.H., 2005. A theoretical foundation of variability in component-based development, Information and Software Technology 47, p. 663-673.

[Nuseibeh, 00] Nuseibeh, B., Easterbrook, S., 2000. "Requirements engineering: a roadmap," Conference on The Future of Software Engineering. New York, USA.

[Poelmans, 10] Poelmans, J., Elzinga, P., Viaene, S., Dedene, G., 2010. Formal concept analysis in knowledge discovery: A survey. Lecture Notes in Computer Science 6208, p. 139-153.

[Pohl, 93] Pohl, K., 1993. The three dimensions of requirements engineering, Lecture Notes in Computer Science 685, p. 275-292.

[Pu, 09] Pu, Y.N., Liu, Q., 2009. "A Viewpoint-Oriented Requirements Elicitation Integrated with Aspects," WRI World Congress on Computer Science and Information Engineering, Beijing, China.

[Sampaio do Prado Leite, 91] Sampaio do Prado Leite, J.C., Freeman, P.A., 1991. Requirements validation through viewpoint resolution. IEEE Transactions on Software Engineering 17, p. 1253-1269.

[Schmid, 04] Schmid, K., John, I., 2004. A customizable approach to full lifecycle variability management, Science of Computer Programming 53, p. 259-284.

[Schmid, 11] Schmid, K., Rabiser, R., Grünbacher, P., 2011. "A comparison of decision modeling approaches in product lines," 5th Workshop on Variability Modeling of Software-Intensive Systems. New York, USA.

[Schobbens, 06] Schobbens, P., Heymans, P., Trigaux, J.-C., 2006. "Feature Diagrams: A Survey and a Formal Semantics," 14th IEEE International Requirements Engineering Conference. Minneapolis, USA.

[van Lamsweerde, 09] van Lamsweerde A., Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley, 2009

[Wille, 05] Wille, R., 2005. Formal concept analysis as mathematical theory of concepts and concept hierarchies. Lecture Notes in Computer Science 3626, p. 1-33.

[Yu, 97] Yu E.S.K., Towards modelling and reasoning support for early-phase requirements engineering, In: IEEE International Symposium on Requirements Engineering, p.226-235, 1997

[Zave, 97] Zave P., Jackson, M., 1997. Four dark corners of requirements engineering. ACM Transactions on Software Engineering Methodology 6, p. 1-30