DESIGNING WITH PRIORITIES AND THRESHOLDS FOR HEALTH CARE HETEROGENEITY: THE APPROACH OF CONSTRUCTING PARAMETRIC ONTOLOGY

Eivazzadeh, Shahryar; Anderberg, Peter; Berglund, Johan; Larsson, Tobias
Blekinge Institute of Technology, Sweden

Abstract

Designing for complex health care environments needs to address heterogeneous, competing, or even contradicting requirements expressed in different wordings and levels of abstraction by various actors of the health care complex environment, i.e. health care consumers, health care professionals, regulatory bodies, production lines, and marketing departments.
The method introduced in this paper, utilizes ontological structures to unify heterogeneous requirements in different levels of abstraction. A weighting mechanism, which utilizes the ontology structure, allows to prioritize the requirements, while a threshold mechanism enforces minimum required qualities in a clear and integrated way. The application of the method is not limited to designing for health care, and it might be applied in design processes for similar environments or can be used to communicate standard requirements and regulations in clear ontology structures.

Keywords: Biomedical design, Ontologies, Requirements, Priorotizing, Design validation

Contact:
Shahryar Eivazzadeh
Blekinge Institute of Technology
Health Science (HIHA)
Sweden
shahryar.eivazzadeh@bth.se

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

Health care systems are complex systems (Shiell et al., 2008), in different scales with multiple different actors (World Health Organization, 2007; Papanicolas et al., 2013), and even sometimes each actor with multiple roles (Frenk, 2010). Products or services, to be launched in a health care environment, need to address heterogeneous requirements of those multiple actors. The variety and heterogeneity of requirements make it difficult for designers to come with a clean set of clearly defined requirements. Different stakeholders express their requirements in different wordings, different scopes, and different levels of abstraction. Some requirements might contradict with each other, while others might compete in priority.

Requirements for a product or service can have two affirmative and negative facets. In one facet, the fulfillment of the requirement would increase the total value that the product or service delivers. In another facet, inability to fulfill a requirement or insufficiency in exposing a quality, invalidates the design and fails the product or service totally even if it excels in other aspects. For example, an electrical device in health care that scores high in price performance would be considered a failure if it could not comply with minimum required safety standards. At the same time, while electrical safety higher than required might contribute to the value the product delivers, but even lower —but enough— levels of safety does not invalidate the overall qualities.

Health care products and services are governed by extensive regulations and guidelines that are strictly needed to be observed (Johnson, 2012). These requirements are expressed and documented in a variety of documents, making it a challenge to extract and apply them via design in a consistent way.

2 RESEARCH CONTEXT

The method represented in this paper was the output of cycles, similar to those specified in *action research* methodology (Davison et al., 2004). The context of the research was the products' requirements specified in Future Internet Social and Technological Alignment Research (FI-STAR) project. FI-STAR is an European Commission (EC) founded project in e-health (FI-STAR, 2012). FI-STAR, as a part of phase two of Future Internet Public-Private Partnership Programme (FI-PPP) project, seeks to utilize Future Internet (FI) technologies, provided by FI-PPP phase one, in seven early trials (FI-STAR, 2012; FI-PPP, 2012). These seven trials, each are designed by different designers upon the requirements gather from different end-users. Beyond the end users, all the trials are supposed to apply or utilize technical requirements specified by the FI recommendations. The trials are connected to each other in that sense that they are needed to utilize some of the functionalities provide by FI infrastructure, called General Enablers (GEs). Also a goal of the project was to detect and suggest some of the common needed functionalities in e-health applications, which eventually are supposed to be implemented as GEs or Specific Enablers (SEs).

The aggregated requirement documents of those seven trials exposed the challenges in such situations. The requirement from each of the trials were expressed in different wording, scale, perspective, and abstraction; while the nature of many of those trials were similar to each other or some of the functionalities were expected to be the same. It was anticipated that a unifying structure can be beneficial for both design and evaluation phases.

Ontologies have been used for a long time in health care to standardize, hence facilitate, the communication about health situation and health care interventions. While those ontologies are defined in global level application, but they can inspire using ontologies in small case-specific scales. Therefore, a pool of requirements were created from harvesting the requirement documents which was then structured as an ontology. There were several cases of ambiguity in constructing the ontology or interpreting the resulted ontology which led to priority and threshold mechanisms described late in this paper. In practice, the ontology is used only for the evaluation process in FI-STAR project, but its successful construction, ease of application in the evaluation phase, and its observed characteristics all together signal the feasibility of the method, with modifications specified in this paper, for design phase.

3 THE RESULTED METHOD

The method introduced in this paper, unifies the heterogeneous, overlapping, non-overlapping, competing, or even contradicting requirements. It also prioritizes them by their predicted contributions to the total value of the final product or service. The method can disqualify those designs that do not satisfy the designated thresholds specific to any of the requirements.
The method is elaborated with the requirements of health care systems design in mind, but it is not limited to that and can be used in other similar design contexts. A European project in health care information systems, called FI-STAR, has been the practical context of developing this method. The method tries to find a unification mechanism for the requirements communicated by the actors in a health care environment by using ontology structures. Ontology construction is a core to the method, both for the purpose of unification and for implementing prioritizing and threshold mechanisms. Ontologies are networked representation of shared knowledge about a domain, where the concepts of that domain are represented as nodes of the network, connected to each other through a limited defined set of relations (Noy and McGuinness, 2001). Usually the concepts are represented in noun form and the relations in verb form. In this sense, the heterogeneous set of requirements in a complex health care setting can be captured within an ontology, which makes is computable for ontology related algorithms. If the relations in an requirement ontology be restricted to hierarchical relations, such as parent to child, superclass to subclass, generic to specific, or superset to subset then still the ontology can capture non-functional requirements as well as those requirements that can be expressed in form of X *is of type* Y form. Hierarchical relations, excluding the equity relation, impose specific forms to networks. They work only in one way, in this sense that one concept cannot be a parent, superclass, generic form, superset to itself. An extension of the above observation is that the hierarchical relations cannot chain as loop, hence any travel between nodes would be in one direction never reaching back the same node. This form of structure, i.e. direct with no loop, is called *acyclic directed graph*. The *tree* form in graphs, is a subset of acyclic direct form. If there is a single node with no parent node then it is called *rooted tree*. As its name implies, all nodes in a rooted tree, except the one that is called *root*, have only one parent. If an acyclic graph is not in tree form, then we can convert it to a tree form by making multiple instances of those nodes that have more than one parents. In a tree, all nodes get unified, level by level, through their parent node, where ultimately they get all unified in the *root* node. Traditionally, the root nodes in tree style ontologies are labeled *thing*, as every concept is ultimately a *thing*.
Tree style ontologies of requirements expose interesting computational characteristics. Being able to traverse between any two nodes, being able to assign each node to a level of the tree, and being able to assign values to each node and connection are some of those characteristics that play role in our method.

3.1 Ontology Construction

The method is essentially a manual ontology top-down construction (Fernández-López and Gómez-Pérez, 2002), customized slightly to be able to accommodate assignment of weights and thresholds. The method, as normal in other ontology construction methods, begins by working on a pool of concepts, here the requirements, which are gathered from the actors (Alterovitz et al., 2010). This can be done by harvesting requirement documents or any document that is an output of requirement elicitation. The requirements need to be expressed in or converted to attributive form of qualities. Even many of functional requirements can be expressed in verb attributive forms such as *"can process more than X number of transactions per second"*. It is expected to encounter different wordings, different scales, different abstraction levels, overlaps, and contradictions in the requirements and their expressions.
Creating a tree style ontology is the next step and at the same time at the core of the method. All the requirements are supposed to be feed into this ontology and to expand it, to merge into another node, or to modify its structure. A repeating task is the comparison of a requirement with another requirement that is already a part of the tree. The comparison tests if the new requirement is a more generic form, a superclass, a superset, or super-type of the other requirement. In the sake of simplicity, the above cases are all addressed by using *superclass* term and vice versa the case of being more specific, subclass, subsumed, subset, or subtype is only addressed by using the term *subclass*.

The algorithm begins by creating a root node labeled as *thing*. Then all the requirements in the pool would go through a process where all begin a travel beginning at the root and ending in one or more lower positions. In this sense, we call that requirement the traveling requirement as that travels from the root to some certain place or places. Each requirement begins the travel by comparing itself to the children of the root node, if there is a node that is a subclass of the requirement then it would repeat this children-comparison process with that node. Traveling deep the tree ends when it reaches a node with no child or if none of the children of a node are superclass of that requirement. If there is no child then the traveling requirement assigns itself as a new child to that node. If there is an exactly the same node, then the traveling requirement merges with that. If there exist any children that are subclass of the traveling requirement then they all change their parents to be the traveling requirement and the traveling requirement assigns itself as a child to the last node it has reached. Figure 1 is a demonstration of the final output of the method.

If a traveling requirement finds more than one superclass children node in a step, then it would replicate itself into instances of the number of those nodes and goes through each branch in parallel. In this sense, the resulting ontology might have several instances of the same concept, i.e. requirement, in different branches. This redundancy contradicts with what traditionally ontologies are, but it simplifies our method in the next step by keeping the ontology in tree format.

In the last step, the tree might need to be normalized in that sense the requirements in each level maintain the same level of generality. This can happen by manually injecting generic qualities that are superclass to nodes that are too specific for that level of ontology. This makes the ontology more readable, at the same time makes it easier to decide which level of the ontology should be considered as the unified, readable, and communicable specification of the requirements (level 3 in blue color in figure 1).

3.2 Discussing Ontology Construction Output

All heterogeneous requirements are unified ultimately in the structure of the ontology tree (see figure 1 for a sample). The tree is structured in levels, beginning with level 0 which only includes the root node (thing). The set of requirements in each level represents and unifies all the captured requirements in some degree of generality. In this sense, the root, i.e. the *thing* node, is the most general but pointless evident requirement. Going down level by level, the requirements in each level grow in number and specificness. When the number of requirements reaches the maximum of our capacity to consider in design then that level would be the *target level* (shown in blue in figure 1).

For branches of the tree that end before reaching the target level, we can continue their presence with repeating the last *leaf* node in subsequent levels. This enables us to avoid missing the requirements when deciding which level should represent all the unified requirements.

There can be ambiguity when we try to answer which of two related requirements is a subclass or superclass form for the other. For example, the topic of *safety* can be considered a superclass of *material safety* (e.g. being non-toxic) or than *operational safety*. But, some thing that is *safe* should be both *safe in material* and *safe in operation*. In this case, the so called generic form, i.e. being safe, is the set *intersection*, a common denominator, or a Boolean AND product of the two others, unlike the first case which was a *union*, a common factor, or a Boolean OR product of the two others. Here, the right wording can resolve the ambiguity, although it leads to a less intuitive hierarchy where *safe in material* is a more generic concept to being *safe* by some standard definition.

As an other example, the experience of an *efficient* solution can be fulfilled both by *fast* solutions and also *simple* ones. Here being efficient is more a *union* of both of being *fast* and being *simple* (figure 1)). At the same time, this can be reversed, where being *efficient* is a specific case of being *fast* and *simple* at the same time.

The origin of these ambiguities is that each quality, i.e. a requirement in our ontology, can be evaluated from the two affirmative and negative perspectives. In the affirmative perspective, the existence of a quality contributes to *adding value* agenda of a product or service. In the negative perspective, the nonexistence of a quality *fails* the product or service, totally or in some aspects. The concept of generality and the right wording choices work differently in these two perspectives. These two perspectives generate two different cases of generality and require different wordings usually. Switching between these two can create two different versions of an ontology of requirements.

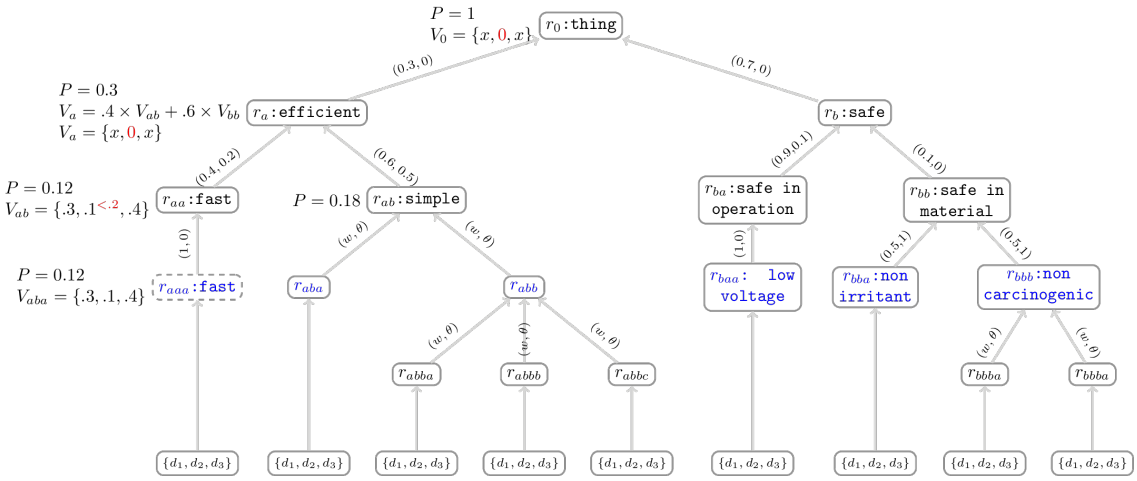

Figure 1. The method ontology structure

3.3 Implementing Priorities and Thresholds

Beyond the unification gained through the ontology, we can assign parameters to nodes and connections in order to bypass ambiguities and also to implement prioritizing and threshold mechanisms.

In our method, the value or arrays of values in the nodes, from zero to one, indicate how that quality is fulfilled by a design or collection of candidate designs. This value is originally extracted from design specification or design evaluation, but it needs to go through some calculations based on ontology connection parameters. Therefore we first focus on the explanation of the connection parameters.

We have introduced two parameters to be associated with each connection. One of these parameters is associated with a subclass to superclass relation while the other is associated with superclass to subclass relation. In this sense, each connection can work in a bilateral way and even a disputing order in hierarchy can be ignored if the right parameters are assigned. We believe that assignment of these parameters is more intuitive than choosing the very precise wording and the very precise hierarchy ordering. In the other words, those two parameters would make the ontology tolerant to some degree to the wrong or challenging hierarchical orders, while it recognizes the in parallel existence of both the affirmative and *adding value* or negative and *cause failure* facets for each quality (requirement) or its absence. The parameters for a connection are formed as an array of two numbers, each between zero to one.

The first number of the array indicates what is the average or expected contribution of a child quality to its parent quality. This value can be result of a market research or similar studies. For example in figure 1, for the children of *efficient* quality (r_a) the first value in each connection (r_{aa} and r_{ab}) indicates how much being *fast* or *simple* contributes to be considered efficient.

$$W(ra) = 0.4 \times W(raa) + 0.6 \times W(rab)$$

Values of weightings in connections (w) in the lower levels of the tree, starting from the root, are subjects of general market studies, while the deep nodes in higher nodes get their values either from the product end users or even designers themselves.

The second number indicates what is the *threshold* value for the child quality that below that value the parent quality cannot be established and its value becomes zero. For example the connection between r_{bb} and r_b (i.e. between *safe in material* and *safe*) is associated with (0.5,0.9), which means any value less than 0.9 would invalidate the state of being *safe*, regardless of other values it might received from the first parameter or from its other child nodes.

In a more precise wording, for a connection weighting (w, θ):

$$\text{parent} = \sum_{i=1}^{n} w_i \times \text{child}_i \qquad (1)$$

$$\exists child_i : (child_i < \theta_i) \Rightarrow parent \coloneqq 0 \quad (2)$$

3.4 Prioritized Requirement Selection

Every node in the output technology can be assigned a value which indicates what is the share of that quality, i.e. fulfillment of that requirement, in the overall value that the product or service deliver. This value is the chain production of the values of all nodes and w value in each connection, starting from the root node and ending at the last connection to the node itself. If we maintain to keep distribute connection values as percentages of number 1, then the value in each node would be less than 1, while the sum of all nodes in each layer would be exactly 1.

3.5 Design Disqualification Based on Thresholds

Any candidate design can be compared to a leaf node of the ontology to see how it fulfills that requirement. This comparison determines the value or value array for that leaf node. Other nodes in the tree get their values based on these initial values in leaf nodes and the parameters of connections between leaf nodes and them.

In figure 1, the candidate three designs are shown by {d1,d2,d3} at the bottom of the ontology. Technically in ontology literature, they can be considered as *instances* connecting to *classes* (Hitzler et al., 2012), hence they become the leaf nodes themselves, but in our method any reference to a leaf node only means a quality class that is leaf in the ontology and not the designs themselves. The comparison of various designs with a requirement creates an ordered set of fulfillment values, ranging from zero to one. The values would be multiplied by the *weighting* value in their parent connection and summed by their parent, but if the values are less than the specified θ threshold they would turn of the parents to zero.

For example in figure 1, V_{aba}, with values 0.3, 0.1, 0.4, means that design number 1 (d_1) is supposed to fulfill 0.3 out of 1.0 in being fast (i.e. requirement r_{aba}), design number 2 can fulfill 0.1 out of 1.0, design number 3 can fulfill 0.3 out of 1.0. The values in r_{aba} would be copied to its parent (r_{ab}) because (1,0) indicates no threshold (0_θ <) and to be fully (1) —not partially— responsible for its parent value. But in the next step, i.e. from r_{ab} to r_a, because the value of the second design is less than threshold (0.1 < 0.2), i.e. it is not designed to be *fast* enough, then the value of r_a, i.e. *efficiency* would be zero for design 2 (d_2), regardless of other values it gathers from the other child node (i.e. r_{ab}). Here the connection between r_{ab} and r_a defused r_a and any upper node in the chain, as the design did not fulfill an essential requirement.

3.6 Communicating Regulations and Standard Guidelines

Health care products and services are governed by intensive regulations and standards that are supposed to ensure safety and reliability of any product or service which can have direct sever impacts on the health of health care consumers (Johnson, 2012). Any design process for health care needs to find its product related regulations and guidelines and comply with them. Communicating those regulations and guidelines can be eased and clarified through suggesting partial ontology trees, the same as in our method, that cover the required qualities.

These partial ontology trees can be integrated to the product requirement ontology in its early stages of construction. It can also be imagined that many of compatible regulations can be aggregated into a unique grand ontology. This grand ontology can be stored and maintained in a centralized way by a regulatory body. This grand ontology can be partially defused to become suitable for design of specific cases.

3.7 Some Limitations and Challenges

The ontology construction method relies on micro decisions in each stage of the algorithm, trying to find out which node is a superclass to the other. These micro decisions can be subjective (Abels et al., 2005), creating different versions of the ontology when applied to the same context but in different times. Those micro decisions can also create different ontologies, when being done by a loosely couple group who do content curation in the sake of extracting the requirements and

constructing the ontology. Here the point is that like a design, for the same context, there can be multiple versions of good ontologies and multiple versions of bad ontologies. The method proposed in this paper limits decisions to those micro decision and automates the rest, this would probably lead to similarity in implementations but does not guarantee that.
The major limitation in the threshold mechanism is probably the simplistic way of recognizing when it should trigger a rejection. In real life, a complex situation about product requirements might determine if a design should be rejected or not. In our method, triggering a rejection only depends on deeper child nodes to where the threshold value is defined; in real life, it can depend on other nodes in other branches in a more sophisticated way. For example, a design for a health technology which is rejected because it could not satisfy minimum required *safety in operation* value can be still a valid option, saving lives of many, if it shows high scores in *safety by informing* and *inexpensive* requirements; while these two requirements are in two different branches not necessarily below *safety in operation*.
The limitation in sophistication of triggering rejection by threshold mechanism can sometimes be amended by restructuring and modifying the ontology. In the above example, the *safety by operation* can lower its threshold requirement, instead increase its share, in competition with *safety by informing*, in its parent *safety* node; at the same time the *safety* could demand higher threshold. This way would let the *safety by informing* to compensate proportionally low values in *safety by operation* to some extent and transfer the trigger more on to the *safety* node. A higher artificial node called *safe or inexpensive* can be asserted above the *safe* and *inexpensive* to collect these two with right thresholds and factors of contribution.

4 CONCLUSION

Designing product or services for health care complex environments can encounter challenges in the stage when designers need to clarify and solidify the requirements. The challenge with the requirements in such complex environments can be that they originate from heterogeneous sources, are specified by different wordings, are expressed in various levels of abstraction, and expose multiple facets. The method introduced in this paper tries to address these challenges by utilizing the unification nature of tree-style ontologies; therefore it constructs a tree ontology out of the requirements and provides dimension of computability to the ontology and the requirements within that. The method of constructing the ontology is a simple top-down, with possibility to be extended in automation or matching aspects.
The method enhances the constructed ontology by assigning weighting and threshold values to the connections and quality fulfillment degree to the nodes. The enhanced ontology facilitates prioritizing the requirements in a design and detecting, hence rejecting, those designs that do not fulfill a minimum of a critical requirement. The method can also be used to communicate a set of regulations and guidelines in an integrated and clear way, by communicating a partial pre-built ontology. Any constructed ontology from a design case can be an input for other related knowledge elicitation and inference algorithms.

ACKNOWLEDGMENT

The authors acknowledge the contribution of FI-STAR project partners, providing the context for this paper. FI-STAR received funding from the European Commission under the Seventh Framework Programme (FP7).

REFERENCES

Abels, S., Haak, L., and Hahn, A. (2005). Identification of Common Methods Used for Ontology Integration Tasks. In Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems, IHIS ’05, pages 75–78, New York, NY, USA. ACM.

Alterovitz, G., Xiang, M., Hill, D. P., Lomax, J., Liu, J., Cherkassky, M., Dreyfuss, J., Mungall, C., Harris, M. A., Dolan, M. E., Blake, J. A., and Ramoni, M. F. (2010). Ontology engineering. Nature Biotechnology, 28(2):128–130.

Davison, R., Martinsons, M. G., and Kock, N. (2004). Principles of canonical action research. Information Systems Journal, 14(1):65–86.

Fernández-López, M. and Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. Knowledge Engineering Review, 17(2):129–156.

FI-PPP (2012). About | FI-PPP.
FI-STAR (2012). About FI-STAR: FI-STAR.
Frenk, J. (2010). The Global Health System: Strengthening National Health Systems as the Next Step for Global Progress. PLoS Medicine, 7(1).
Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (2012). OWL 2 web ontology language primer. W3C recommendation, 27(1):123.
Johnson, J. A. (2012). FDA regulation of medical devices. Congressional research service, June, 25.
Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880.
Papanicolas, I., Kringos, D., Klazinga, N. S., and Smith, P. C. (2013). Health system performance comparison: New directions in research and policy. Health Policy, 112(1–2):1–3.
Shiell, A., Hawe, P., and Gold, L. (2008). Complex interventions or complex systems? Implications for health economic evaluation. BMJ : British Medical Journal, 336(7656):1281–1283.
World Health Organization (2007). Everybody's business–strengthening health systems to improve health outcomes: WHO's framework for action.