Extending the Sustainability-Quality Model for supporting the design of Persuasive Software Systems

Marcela Quispe-Cruz[1], **Nelly Condori-Fernandez**[2], [3]

[1]Universidad Nacional de San Agustín (UNSA)
Arequipa – Perú

[2]CITIC, Computer Science Department – Universidade da Coruña
Galicia, Spain.

[3]Computer Science department – Vrije Universiteit Amsterdam
The Netherlands.

mquispecr@unsa.edu.pe, n.condori-fernandez@vu.nl

Abstract. *This research aims at providing a guiding support for the selection of relevant features and quality requirements for designing persuasive software systems. To do this, a mapping between the Persuasive System Design (PSD) model and a generic Sustainability-Quality (SQ) model was carried out. As a result of this mapping, we extended the SQ model, by adding certain types of relationships with specific features of the PSD model. A Graph database tool, named Neo4j, was used for facilitating the visualization of the identified relationships. And we also used the query language Cypher in order to retrieve data from the graph Finally, we used an existing persuasive software system for illustrating the usefulness of the extended SQ model represented as graphs.*

1. Introduction

Persuasive technology can be defined as "design, research, and analysis of interactive computing products created to change people's attitudes [1] or behaviors [2]" [Fogg 2003]. As technology can be used as a promoter of sustainable behaviour, many studies have investigated the possibilities to persuade people within the context of environmental sustainability (e.g., increase consumers' awareness of energy consumption [Froehlich 2009]). However, most of these studies have shortcomings that limit their long-term effectiveness. Although behavioral models (e.g., Transtheoretical Model of behavior change [Prochaska and DiClemente 2005], the Goal-setting Theory [Locke and Latham 1991], the Fogg Behavior Model [Fogg 2009]) are very useful for conceptualizing the impact of persuasive technology, most of them cannot be applied directly to the design or assessment of persuasive systems [Harjumaa and Oinas-Kukkonen 2007, Condori-Fernández et al. 2018]. For example, through a user experience assessment of existing persuasive software applications, Condori-Fernandez et al. [Condori-Fernández et al. 2018] found that some relevant non-functional requirements had not been addressed, and consequently users experienced negatively in using such kind of systems.

[1]Attitude represents how a person thinks or feels about someone or something.
[2]Behavior represents an individual's reaction to a particular action, person or environment

As the identification and management of non-functional requirements (NFR) in software projects are challenging [Ameller et al. 2019], various assessment models have been proposed for software product quality (e.g., ISO/IEC 25010 quality model). In the last years, software sustainability has gained more attention from researchers to address sustainability requirements along the software life-cycle (e.g., [Condori-Fernandez and Lago 2018], [Calero et al. 2013], [Venters et al. 2014]). Lago et al. [Lago et al. 2015] defined software sustainability based on a four-dimensional model that adds the technical dimension to the social, environmental and economic dimensions that already appear in the Brundtland report [Brundtland et al. 1987]. Condori-Fernandez and Lago [Condori Fernandez and Lago 2018, Lago and Condori-Fernandez 2022] proposed a Sustainability-Quality (SQ) model, an instrument of the Sustainability Assessment Framework, for supporting the identification of quality requirements that contribute to the four-dimensional model of software-intensive systems [3]. The multidimensional approach of Becker et al. [Becker and et al 2016] adds the individual dimension to the four sustainability dimensions [Lago et al. 2015]. However, Calero et al. [Calero et al. 2013] define sustainability only in terms of energy consumption, resource optimization and perdurability, and they do not consider the individual, social, and economic dimensions.

In this paper, we aim to extend the SQ model [Condori-Fernandez and Lago 2018] in order to provide a guiding support for (i) the NFR identification and (ii) the selection of relevant features from the PSD Model [Oinas-Kukkonen and Harjumaa 2009] for designing persuasive systems. To do this, we first use the PSD Model as a means to extend the SQ model with the inclusion of new type of relationships. It is achieved through a mapping carried out between some elements of the PSD model and the SQ model. Second, given that designers of persuasive systems may achieve more success if adequate methodological support was available, we changed the tabular visualization of the SQL model (static) to a graph-based visualization (dynamic). Therefore, we consider that such kind of visualization can be helpful not only for identifying relevant quality requirements usually ignored in practice, but also for supporting the selection of persuasive software system features.

The following sections provide a detailed account of our paper. Section 2 describes the SQ model and PSD model on which our work is based. Section 3 presents the procedure and results of the mapping carried out between both models. In Section 4, we present the corresponding queries to support the NFR discovery and Feature selection. Section 5 illustrates the application of the most important queries in a specific example. In section 6, we conclude the paper and discuss further work.

2. Background

In this work, we consider the PSD model [Oinas-Kukkonen and Harjumaa 2009], as the theoretical framework for our research, and the SQ model [Condori-Fernandez and Lago 2018].

2.1. The PSD model

The PSD model is a recent conceptualization for designing, developing and evaluating persuasive systems. It consists of the premises behind any persuasive system, the persua-

[3] Systems in which software interacts with other software, systems, devices, sensors and people.

sion context and the persuasive software system features. All persuasive software systems are based on the following premises [Oinas-Kukkonen and Harjumaa 2009]:

- **P1: Useful.** The system really serves the needs of the user.
- **P2: User-friendly.** The system should be easy to use or dealt with.
- **P3: Unobtrusiveness.** The system should avoid being disturbing while the user is performing tasks.
- **P4: Open.** Designers should make the ideas and the goals of persuasion transparent.
- **P5: Cognitive Consistency.** People like their views about the world to be organized and consistent. Inconsistency disturbs people, and they easily want to reorganize their thinking and restore consistency, perhaps even feel obliged to do so.
- **P6: Incremental.** This means that persuasion goes stepwise, and all steps contribute to the goals to be realized.
- **P7: Information Technology (IT) is never neutral.** IT always influences attitudes and behavior.
- **P8: Direct and indirect routes.** Persuasion strategies can be divided into direct and indirect routes, and persuasion will depend on the ability and motivation of people to process information.

The analysis of the *persuasion context* consists of looking into (1) the intent, (2) the event and (3) the strategy. A central feature of analyzing the intent is to consider the change type, in particular whether the persuasion aims at attitude and/or behavior change. The event comprises the use situation, user's characteristics, technological platform and environment. The strategy includes the message itself and the route to be used to achieve a goal.

The PSD model describes 28 *persuasive software system features* grouped in four categories: (i) The **primary activity support category** focuses on supporting the activities that lead to achievement of the persuasive software system goals. (ii) **Dialogue support** refers to techniques/mechanisms to motivate users to use the persuasive software system goals. (iii) The **credibility category** relates to how to design a system so that it is more credible and thereby more persuasive. iv) The **social influence category** describes how to design the system so that it motivates users by leveraging different aspects of social influence. More details of the features can be found in Appendix B.

2.2. The SQ model

It is defined in terms of four *sustainability dimensions*: (i) *Technical dimension* addresses the long-term use of software-intensive systems and their appropriate evolution in an execution environment that continuously changes. (ii) *Economic dimension* focuses on preserving capital and (economic) value. (iii) *Social dimension* focuses on supporting current and future generations to have the same or greater access to social resources by pursuing generational equity. For software-intensive systems, this dimension encompasses the direct support of social communities in any domain, as well as the support of activities or processes that indirectly create benefits for social communities. (iv) *Environmental dimension* aims at improving human welfare while protecting natural resources. For software-intensive systems, this dimension aims at addressing ecologic concerns, including energy efficiency and ecologic awareness creation.

Each dimension is characterized by a set of *Quality attributes*, which can be interdependent. Such dependency can be of two types: (i) it is *inter-dimensional* if it relates a pair of quality attributes defined simultaneously in two different dimensions (e.g. security defined in the *technical* dimension can influence security in the *social* dimension); and (ii) it is *intra-dimensional* if a dependency exists between two different quality requirements defined within the same dimension (e.g. in the *technical* dimension, security may depend on reliability).

Since our SQ model provides support to both identify design concerns, and assess the qualities of the software architecture, a set of metrics are used for measuring the quality requirements, which should be measurable. Instances of metrics can be also defined for *sustainability-related requirements*.

The list of attributes of the SQ model, and corresponding contributions to the four dimensions, can be found at [Condori Fernandez and Lago 2018]. In the following section, we explain how the SQ model has been extended, by means of a mapping with certain elements of the PSD model (i.e., premises and features).

3. Mapping the PSD model to the SQ model

3.1. Procedure

The mapping was carried out iteratively by the first two authors of the paper, which consisted of three iterations. And a first validation of the mapping was carried out in the fourth iteration.

First iteration: The PSD model was analyzed regarding the premises and features that should be considered for designing persuasive systems. Researcher 1 focused on the analysis of the categories and corresponding features of the PSD model. Researcher 2 focused on the analysis of the premises of the PSD model.

During this first iteration, both researchers used independently an Excel templatefor marking with an "X" whenever they identified some attributes of the SQ model as relevant.

Second iteration: Two online meetings were organized between both researchers for discussing regarding (i)the QAs identified in the first iteration and (ii) and the relationships that can be identified between the selected QAs of the SQ model and the elements of the PSD model (premises and features). The outcomes of the first iteration were interchanged and reviewed before the meetings. As a result of these meetings, researchers agreed on some type of relationships such as *"Helps to"* that represents a positive contribution and *"Hurts to"* that represents a negative contribution. Both type of relations were adopted from the Non-Functional Requirements framework [Chung et al. 2000]. Therefore, the SQ model is extended by means of the mapping carried out along these two first iterations. The outcome of this mapping can be found in Appendix A.

Third iteration: In this iteration, a visualization of the outcome produced in the previous iterations is implemented, by using a Graph database tool named Neo4j [4] version 4.0. To build the graph database we place the selected characteristics and QAs of the SQ model as well as the features of the PSD model in nodes and the corresponding

[4]https://neo4j.com/

relationships between them as transitions of the graph. To visualize a graph, we use Cypher that is the Neo4j's graph query language.

Fourth iteration: A third participant was involved, a professional in software engineering, for the review of the mapping (only relationships between QAs and Features of the PSD model). This review was in two stages. The first stage was in offline mode, where the reviewer received the corresponding instructions and an excel file that contains (i) definitions of QAs of the SQ model as well as definitions of features of the PSD model; and (ii) the relations to the PSD features with their corresponding explanations. For each identified relationship, the following question was formulated: *Do you agree with the identified relationship? If not, please explain your own rationale.*

In the second stage, an online meeting was held with the participation of our external reviewer and the two researchers that defined the first version of mapping. As a result of this review, two type of changes were implemented regarding:

- Type of relationship: With the purpose of highlighting a greater contribution from some quality attributes to some features of the PSD model, or vice versa, a new relationship *"Strong helps to"* was added. For example, an *"Strong helps to"* was identified for the relationship between the context completeness attribute and two Primary Task Support features: tailoring, and personalization.
- Features: Due to the relevance of some few features (e.g., simulation, reduction) some few relationships were also added to certain QAs. For example, researchers had only considered that unobtrusiveness HELPS to the tunneling feature, when this QA could also helps to the reduction feature.

Figure 1 shows the resulting scheme in the graph database.

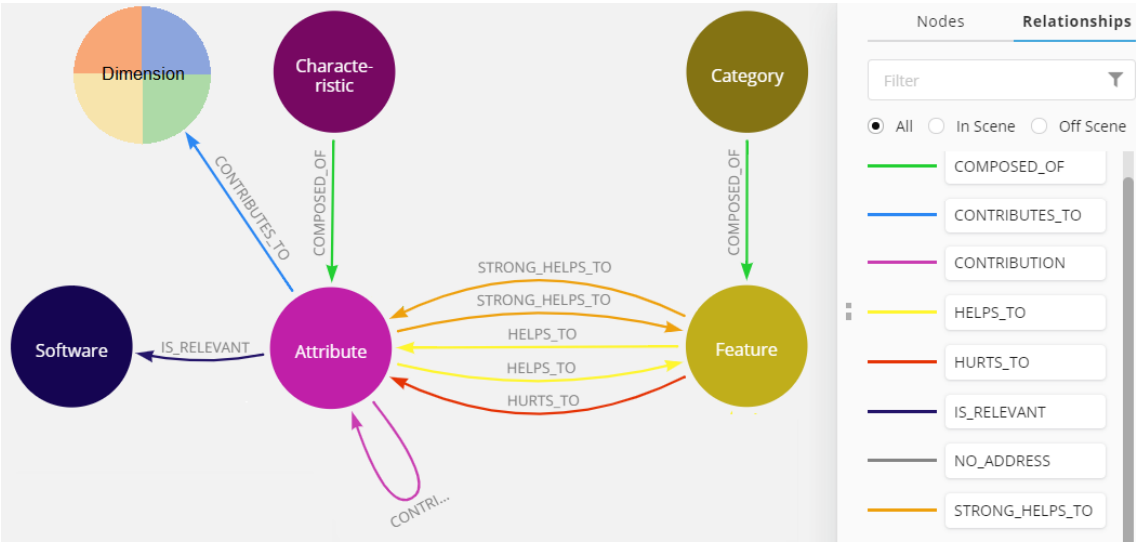

Figure 1. Scheme with types of nodes and relationships in the graph database.

The graph database consists of:

- Four kinds of coloured edges: The edges HELPS_TO (orange), STRONG_HELPS_TO (brown) and HURTS_TO (red) represent that the source node (QA or Feature) helps, strongly helps or hurts the target node (QA or Feature), respectively.

COMPOSED_OF (green) represents a composition relationship.

- Four kinds of coloured nodes: *Characteristic*-node (purple) represents a quality characteristic of the SQ model, *Attribute*-node (fucsia) represents a QA of the SQ model, *Category*-node (olive) represents a features category of the PSD model, *Feature*-node (lime) represents a feature of the PSD model.

3.2. Results

Next, we discuss the type of relations with the PSD model at two levels: Premises and Features.

3.2.1. Premises

Thanks to the mapping with the premises of the PSD model, we were able to identify two new QAs that were not present in the SQ model: Unobtrusiveness (P3) and Transparency (P4). Other premises such as P1 (useful) could be directly related to usefulness of the SQ model; whereas user-friendly (P2) was related to usability (operability). The premise Incremental (P6) was related to maintainability since the system should be easy to modify for adding new suggestions or recommendations.

There were also some premises like Cognitive consistency (P5) and Direct and indirect routes (P8) that could not be related to any specific QA. We consider that both premises could be addressed by more than one QA. For example, attributes regarding context coverage, satisfaction, and effectiveness could enable that software systems can implement their persuasion strategies into direct and indirect routes (P8). It is also important to note that a persuasive system that causes any user disturbance would affect to efficiency (more time to perform tasks) and timeliness (being not opportune). For this reason, Unobtrusiveness (P3) was also related to both QAs.

3.2.2. Features

The mapping with the features of the PSD model allowed us to extend the SQ model, by using some the identified type of relations between QAs and features of the four categories: primary task support, dialog support, system credibility support and social support. Figure 2 summarizes the corresponding features (Fx) and premises (Px) mapped to the quality characteristics that contribute to each sustainability dimension.

An example of graph that shows different type of relationships like COMPOSED_OF (i.e., Satisfaction composed of Trust) HELPS_TO, and HURTS_TO (i.e., relationships between Trust and Features of the *System Credibility Support* category) is shown in Figure 3. In the graph, we considered some features of the *Primary task support* and *System Credibility Support* categories that might *help to* the system in providing information that is truthful. Thus, a system, which incorporates the features like expertise, real-world feeling, leverages roles of authority, and third-party endorsements, makes easy verify the accuracy of site content, and consequently achieves that the stakeholders become confident.

Looking at the same graph (Figure 3), we also see the relationship HELPS_TO from the node labelled “Self-monitoring” (feature) to the node “Trust” (QA). This is be-

QUALITY CHARACTERISTICS	TECH	ENV	ECON	SOC	PREMISES (Px) AND FEATURES (Fx)
Compatibility					F01 to F07
Context coverage					F02 to F05, F07 to F11, F14
Effectiveness					F01, F06, F08 to F21
Efficiency					P3, F01
Freedom from risk					F15 to F28
Functional suitability					F02, F04 to F11, F14, F16, F18 to F20, F22 to F28
Maintainability					P6
Performance efficiency					
Portability					
Reliability					F05, F08 to F14, F21
Satisfaction					P1, F05, F15, F16, F18 to F21
Security					F15
Usability					P2, F02 to F05, F13
Accessibility					F03, F04
Robustness					
Survivability					
Data Privacy					F05
Timeliness					P3, F02, F07
Reguation compliance					
Scalability					
Tailorability					F03
Unobtrusiveness					P3, F2
Transparency					P4, F08-F11

Figure 2. SQ model characteristics related to Features (F) or Premises (P) of the PSD model (Tabular view)

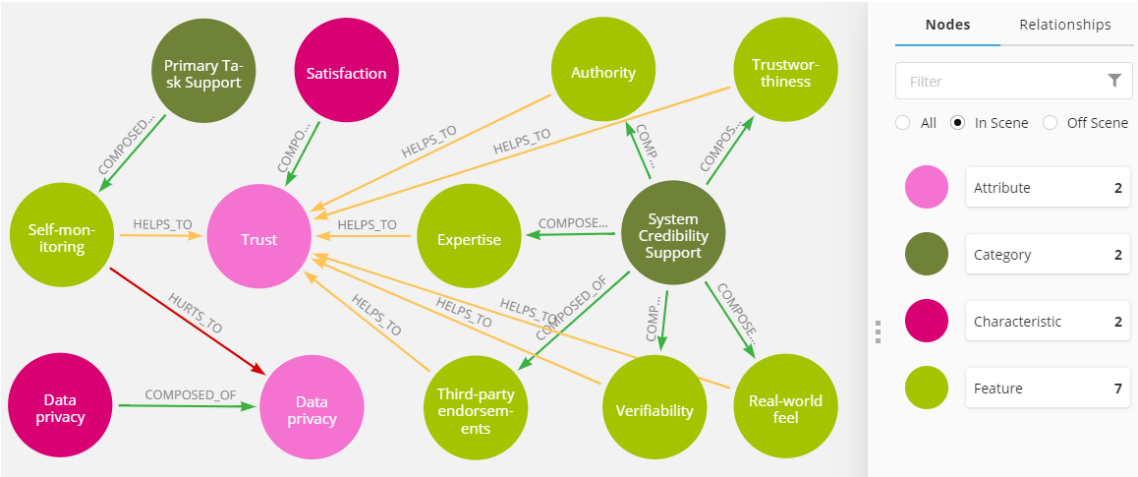

Figure 3. An illustration of relationships: HELPS_TO, HURTS_TO and COMPOSED_OF between the PSD model and the SQ model (Satisfaction in terms of Trust, and Data privacy)

cause having such feature implemented (user can track his/her own performance through the system), it might help to address the quality attribute "Trust" due to stakeholders are confident in that product. However, we can also observe that the "Self-monitoring" feature could also affect negatively to data privacy (HURTS_TO). This is because a system tracking the performance or status might cause some privacy concerns due to the collection of personally identifiable information.

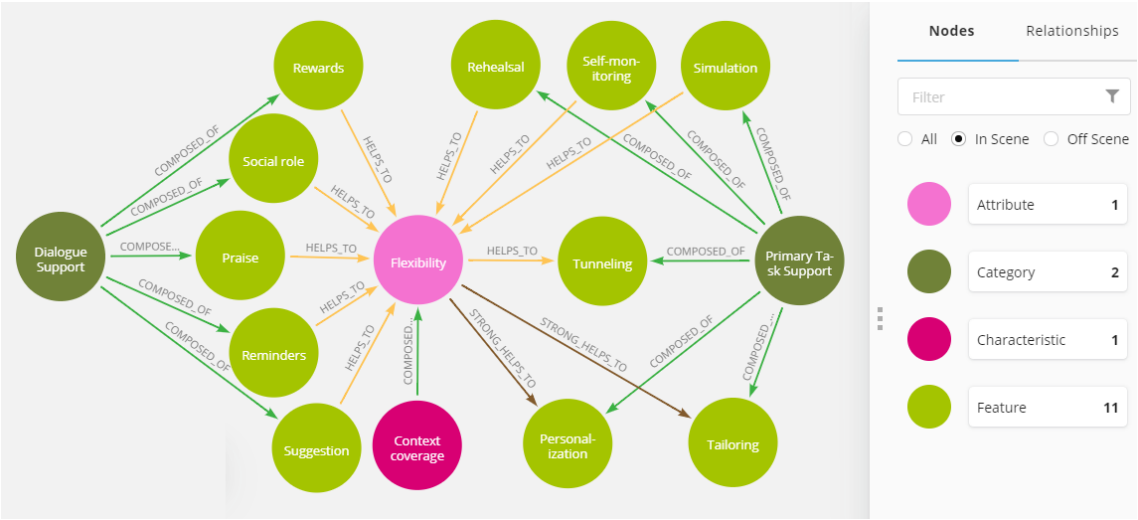

Figure 4. An illustration of relationships: HELPS_TO and COMPOSED_OF between the PSD model and the SQ model (Context coverage in terms of Flexibility)

In Figure 4, we illustrate the relationships for the quality attribute: Flexibility. In this case, we want to remark the arrows direction. For example, five features of the category *Dialogue Support* ("Suggestions", "Praise", "Rewards", "Social role" and "Reminders") contribute (helps to) to address the quality attribute: Flexibility. The implementation of the Praise, Rewards, Reminders, and Suggestions features can help users keep moving towards their goal or target behavior in contexts beyond those initially specified in the requirements. The social role feature might help in communication between users and specialists for example, consequently help the system to be used in contexts beyond those initially specified in the requirements as well.

We can also have contributions in the other direction, (QA→FE), such as Flexibility might help to the Tunneling, Personalization and Tailoring features. This means that if a system can be used in contexts beyond those initially specified in the requirements then the system should be able of: (i) offering personalized content or services; (ii) providing information that should be tailored to the potential needs, interests, personality, usage context, or other factors relevant to a user group; and (iii) guiding the user through a process or experience providing opportunities to persuade it.

4. Retrieval of Features and Quality Attributes using Cypher

The importance of the extended SQ model in a graph database lies in the fact that relationships between features, quality attributes and dimensions of sustainability can be consulted. We take advantage of neo4j's Cypher language to facilitate the designer's query. To better illustrate the retrieval from the graph database using Cypher, we first present the syntax of the language [Neo4j 2022].

Cypher's syntax The Cypher query language depicts patterns of nodes and relationships and filters those patterns based on labels and properties. For instance, nodes are represented with parentheses around the attributes and information regarding the entity. Relationships are depicted with an arrow (either directed or undirected) with the relationship type in brackets.

```
//node
(variable:Label {propertyKey: 'propertyValue'})
//relationship
-[variable:RELATIONSHIP_TYPE]->
//Cypher pattern
(node1:LabelA)-[rel1:RELATIONSHIP_TYPE]->(node2:LabelB)
```

Keywords: Cypher contains a variety of keywords for specifying patterns, filtering patterns, and returning results. Among those most common are: (i) MATCH is used before describing the search pattern for finding nodes, relationships, or combinations of nodes and relationships together. (ii) WHERE in Cypher is used to add additional constraints to patterns and filter out any unwanted patterns. (iii) RETURN formats and organizes how the results should be outputted. Just as with other query languages, you can return the results with specific properties, lists, ordering, and more.

Table 1 illustrates some queries in the graph database (Q1, Q2 and Q3) with a brief description of them. According to the syntax shown above, the query Q1 will search for the pattern of the node (*Characteristic* label) connected by the relationship (*COMPOSED_OF* type and outgoing direction away from the first node) to another node (*Attribute* label and property called name with value of input variable *$attrib*) and this is connected by the relationship (*CONTRIBUTES_TO* type and outgoing direction away from the second node) to the node *Dimension* label). In Q2 and Q3 we have queries of two types of relationships between nodes (of types *HELPS_TO* and *STRONG_HELPS_TO*).

Table 1. Queries

ID	Query in graph database	Description	Input	Output
Q1	MATCH p=(:Characteristic)-[:COMPOSED_OF]->(:Attribute{name:$attrib})-[:CONTRIBUTES_TO]->(:Dimension) RETURN p;	Knowing a quality attribute, the query helps in retrieving the corresponding sustainability dimensions.	Quality attribute	Sustainability dimensions and quality characteristic
Q2	MATCH p=(:Attribute {name:$attrib})- [:HELPS_TO | :STRONG_HELPS_TO]-(:Feature)<- [:COMPOSED_OF]-(:Category) RETURN p;	Knowing a quality attribute, the query helps in retrieving the corresponding feature and category.	Quality attribute	Features, categories
Q3	MATCH p=(:Feature{name:$fea})-[:HELPS_TO | :STRONG_HELPS_TO]-(:Attribute)-[:CONTRIBUTES_TO]->(:Dimension) RETURN p;	Knowing a feature, the query supports retrieving quality attributes helped by it and sustainability dimensions related.	Feature	Quality attributes and sustainability dimensions

5. Use Case: Illustration of Queries in the Graph model

In this section, we firstly describe the case from the well-being domain. Then, the use of the queries, defined in the previous section, is illustrated in two steps, which allow us to: (i) discover the relevant NFR and (ii) identify the features that were not present in the existing software application.

5.1. Case description:

There exists some software applications which aim to prevent and reduce RSIs in office workplaces, where the user has to typically be seated for a long time in front of a screen, typing on the keyboard and using a mouse as main peripheral devices. For illustrating the usefulness of our operationalized model, we have selected the Workrave application.

Workrave is probably one of the most complete applications of its class. It considers micro-pauses, rest breaks, and guidance for exercise routines. This software is based on timers and keyboard/mouse activity, which determine when the actions must be displayed on screen. The user interface offers to configure a good number of parameters and provides a monitor on micro-breaks, rest breaks and working hour limit. The remarkable feature is the Training support, using an animated virtual human to demonstrate the exercises in addition to a textual description (see Figure 5), which could have some positive effects on attaining coaching goals.

5.2. Features selection and NFR discovery

To illustrate the usage of the queries, we will consider only Flexibility and Timelines as relevant NFRs that were not addressed in Workrave:

- **Flexibility.** Workrave can be used in two specified contexts (writing and reading mode). However flexibility is desirable since users might want to use it in another different contexts to the specified. For example during a videoconference (listening).
- **Timeliness.** As Workrave is not aware on the context of usage, the main functionality (i.e., Take-a-break notification) might occur at a non-favourable time.

By considering both NFRs and using the extended SQ model, we were able to identify some new potential features that should be considered in further Workrave versions. In this paper, we illustrate some of the identified features as potential improvements for two functionalities of Workrave.

As shown in Figure 5, one of these functionalities is to provide statistics about user activity. For example, this functionality in the current version is limited to display measures of the keyboard/mouse usage (e.g., how long the mouse has been in use, the total and net distances the mouse has been moved). This means that the statistics of the actual user activity could not become accurate whether a user would be doing anything other than typing. Using query Q2 shown in the Table 1, which receives as input the quality attribute Flexibility, we get as output features (like personalization, tailoring and tunneling) with which they help each other (HELP_TO and STRONG_HELP in both directions). In our case, if workrave was flexible regarding its context of usage, the implementation of the *tailoring feature* would be favoured and consequently the reporting of such type of statistics would be improved as well. Another relation observed in the query Q2, it is that

self-monitoring feature helps to flexibility. As designers, we think that this feature could effectively help to address the flexibility requirement since Workrave would be able to provide other means to track user status through self-monitoring. This information might enable the system to be used in contexts beyond those initially specified in the requirements. As shown Figure 5, the second functionality of Workrave is regarding "Guided exercises". The fact that this functionality could be done at an opportune moment (timeliness), it would favor the use of Workrave to guide users more appropriately (*tunneling)*, which could provide better opportunities to persuade them along the way. As categories are also output of query Q2 (See Table 1), we can also retrieve the corresponding categories to which the features belong (i.e., Primary task support).

Figure 5. Mapping between NFRs and features for Workrave's functionalities: User activity statistics (left) and Guided exercises (right)

Using Flexibility and Timeliness as inputs, we can also use the query Q1 (see Table 1) to determine which sustainability dimensions can be addressed (output). This is also illustrated in Figure 5, where we obtain the Economic dimension, since there is a relation from flexibility (CONTRIBUTES_TO-type relationship). The quality characteristic Context coverage, is another output from query Q1, since there is a relation (COMPOSED_OF-type relationship) with flexibility. In similar way, we can retrieve the social dimension, by having Timeliness as input for the query Q1.

6. Conclusions and Future Work

The present research aims to operationalize a generic Sustainability-Quality (SQ) model for both assessing and designing persuasive systems through a mapping with the PSD

model. As a result of this mapping, the operationalization consisted of identifying: (i) the QAs that might be addressed by the corresponding features or premises of the PSD model; and also those QAs that might help to the implementation of the PSD features, (ii) explicit relationships between QAs and features of the PSD model like *Strong Helps to*, *Helps to* and *Hurts to*. Moreover, the presentation of our operationalized SQ model has been visually represented by using graphs. We consider that this type of visualization might be more efficient because we can take advantage of the graph query language (Cypher).

This mapping (with the PSD model) has also helped us to uncover two new QAs (unobtrusiveness and transparency) that were identified as relevant for designing persuasive systems, which were included in the sustainability-quality model.

This mapping is the first attempt that tries to bridge the gap between both the software engineering and the persuasive technology communities. Our extended SQ model can be a helpful instrument not only for software engineers, who could discover NFR and select specific features for developing persuasive systems in a easier way, but also for researchers from the Persuasive Technology area, who could be interested in the design and evaluation of persuasive techniques/strategies. Besides that, the extended SQ model can be also used as a tool to identify qualities and features that might contribute to one or more dimensions of software sustainability. We also used an existing RSI application (Workrave) for illustrating how the extended SQ model helps in discovering missing quality requirements and potential features that were not present in the software app. Although in this research, we focus on the specific characteristics of persuasive systems, the SQ model can be used for any other type of software-intensive systems.

As future work, we are going to evaluate the implementation of the SQ model(with Neo4jgraph technology), by involving developers or software designers. Their feedback will be very useful not only for reviewing our list of predefined queries, which were formulated for retrieving quality requirements and features, but also for evaluating how usable is our graph-based approach.

Acknowledgment

This work has received support by the projects PDC2021-121239-C31, and KUSISQA 014-2019-FONDECYTBM-INC.INV. The research of Dr. Condori-Fernandez has been carried out as part of CITIC, Research Center accredited by Galician University System, funded by "Consellería de Cultura, Educación e Universidade" from Xunta de Galicia.

References

Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araujo, J., Biffl, S., Cabot, J., Cortellessa, V., Méndez, D., Moreira, A., Muccini, H., Vallecillo, A., Wimmer, M., Amaral, V., Bühm, W., Bruneliere, H., Burgueño, L., Goulão, M., Teufl, S., and Berardinelli, L. (2019). Dealing with non-functional requirements in model-driven development: A survey. *IEEE Transactions on Software Engineering*, pages 1–1.

Becker, C. and et al (2016). Requirements: The key to sustainability. *IEEE Software*, 33(1):56–65.

Brundtland, G., Khalid, M., Agnelli, S., Al-Athel, S., Chidzero, B., Fadika, L., Hauff, V., Lang, I., Shijun, M., Morino de Botero, M., Singh, M., and Okita, S. (1987). Our com-

mon future (brundtland report). Technical report, World Commission on Environment and Development.

Calero, C., Moraga, M. Á., and Bertoa, M. F. (2013). Towards a software product sustainability model. *CoRR*, abs/1309.1640.

Chung, L., Nixon, B., Yu, E., and Mylopoulos, J. (2000). *Non-functional requirements in software engineering*. SPRINGER, LLC, New York, NY, USA.

Condori-Fernández, N., Bolos, A. C., and Lago, P. (2018). Discovering requirements of behaviour change software systems from negative user experience. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 222–223.

Condori-Fernandez, N. and Lago, P. (2018). Characterizing the contribution of quality requirements to software sustainability. *Journal of systems and software*, 137:289–305.

Condori Fernandez, N. and Lago, P. (2018). *A Sustainability-quality Model: (version 1.0)*. VU Technical Report.

Fogg, B. (2003). *Persuasive Technology: Using Computers to Change What We Think and Do*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Fogg, B. (2009). A behavior model for persuasive design. *Proceedings of the 4th International Conference on Persuasive Technology - Persuasive '09*, page 1.

Froehlich, J. (2009). Promoting energy efficient behaviors in the home through feedback: The role of humancomputer interaction. In *HCIC 2009 Winter Workshop*, volume 9, pages 1–11.

Harjumaa, M. and Oinas-Kukkonen, H. (2007). An analysis of the persuasiveness of smoking cessation web sites. In *The Second International Symposium on Medical Information and Communication Technology*.

Lago, P. and Condori-Fernandez, N. (2022). *The Sustainability Assessment Framework (SAF) Toolkit: Instruments to help Sustainability-driven Software Architecture Design Decision Making*. S2 Group, Vrije Universiteit Amsterdam.

Lago, P., Koçak, S. A., Crnkovic, I., and Penzenstadler, B. (2015). Framing sustainability as a property of software quality. *Communications of the ACM*, 58(10):70–78.

Locke, E. and Latham, G. (1991). A theory of goal setting & task performance. *The Academy of Management Review*, 16:480–483.

Neo4j (2022). Cypher introduction.

Oinas-Kukkonen, H. (2013). A foundation for the study of behavior change support systems. *Personal Ubiquitous Comput.*, 17(6):1223–1235.

Oinas-Kukkonen, H. and Harjumaa, M. (2009). Persuasive systems design: Key issues, process model, and system features. *Communications of the Association for Information Systems*, 24:485–500.

Prochaska, J. O. and DiClemente, C. C. (2005). The transtheoretical approach. *Administration and Policy in Mental Health and Mental Health Services Research*, pages 147—171.

Venters, C., Jay, C., Lau, L., Griffiths, M. K., Holmes, V., Ward, R., Austin, J., Dibsdale, C. E., and Xu, J. (2014). Software sustainability: The modern tower of babel. In *RE4SuSy: Third International Workshop on Requirements Engineering for Sustainable Systems*.

7. Appendix A: Mapping between PSD model and the SQ model

	System credibility			Explanation
Quality attributes	***Strong Helps***	***Helps***	***Hurts***	
Effectiveness		X		A system that incorporates features such as trustworthy, incorporing expertise, surface credibility, incorporing real-world feeling, leverages roles of authority, third-party endorsements and facilitates verifiability of the accuracy of site content achieves (HELPS) that the system be effective.
Economic risk mitigation			X	Social support (all features) might have a negative effect (HURTS) on the user's financial status in medium or long term due to the excessive usage of the (persuasive) software system.
Environmental risk mitigation			X	Social support (all features) might have a negative effect on the environment in medium or long term due to the excessive usage of the (persuasive) software system. For instance, when groups of motivated users became addicts of the system, energy consumption would increase, which can cause a negative effect on the environment.
Health and safety risk mitigation		X		As all features contribute to design more credible systems and thus more persuasive, we think that the likelihood of happening a health and safety risk would be low
Functional appropriateness		X		The functions facilitate (HELP) the accomplishment of specified tasks and objectives related to features that the system should have: Expertise, Real-world feel, Authority, Third-party endorsements.
Functional correctness		X		The correct functions facilitate (HELP) the accomplishment of specified tasks and objectives such as features that the system should have: Expertise, Real-world feel, Authority, Third-party endorsements.
Functional completeness		X		Functional completeness facilitates (HELPS) the accomplishment of specified tasks and objectives such as features that the system should have: Expertise, Real-world feel, Authority, Third-party endorsements.
Maturity		X		If the system facilitates verifiability of the accuracy of site content then the system becomes more reliable
Trust		X		A system that incorporates features such as trustworthy, incorporing expertise, real-world feeling, leverages roles of authority, third-party endorsements and facilitates verifiability of the accuracy of site content achieves (HELPS) that the stakeholders are confident that a product or system will behave as intended, that is, it is seen as trustworthy.
Accountability		X		If actions of an entity can be traced uniquely to the entity, this contributes (HELPS) for the system be viewed as trustworthiness
Authenticity		X		If the identity of a subject or resource can be proved to be the one claimed, this contributes (HELPS) for the system be viewed as trustworthiness
Confidentiality		X		If system ensures that data are accessible only to those authorized to have access, then this contributes (HELPS) for the system be viewed as trustworthiness
Integrity		X		If the system prevents unauthorized access, or modification of, computer programs or data, this contributes (HELPS) to the system be viewed as trustworthiness.
Transparency		X		The transparency of the purpose of the system HELPS make the system more credible (System Credibility Support - all features) and thereby more persuasive.

Figure 6. Relationships between quality attributes (SQ model) and features of system credibility support(category of the PSD model)

8. Appendix B: Features of the PSD model

Categories	Description	Features	Description
Primary Task support	The primary activity support category focuses on supporting the activities that lead to achievement of the BCSS goals.	F01: Reduction	A system that reduces complex behavior into simple tasks helps users perform the target behavior, and it may increase the benefit/cost ratio of a behavior.
		F02: Tunneling	Using the system to guide users through a process or experience provides opportunities to persuade along the way.
		F03: Tailoring	Information provided by the system will be more persuasive if it is tailored to the potential needs, interests, personality, usage context, or other factors relevant to a user group.
		F04: Personalization	A system that offers personalized content or services has a greater capability for persuasion.
		F05: Self-monitoring	A system that keeps track of one's own performance or status supports the user in achieving goals.
		F06: Simulation	Systems that provide simulations can persuade by enabling users to observe immediately the link between cause and effect.
		F07: Rehearsal	A system providing means with which to rehearse a behavior can enable people to change their attitudes or behavior in the real world.
Dialogue Support	Dialogue support refers to techniques/mechanisms to motivate users to use BCSS.	F08: Praise	By offering praise, a system can make users more open to persuasion.
		F09: Rewards	Systems that reward target behaviors may have great persuasive powers.
		F10: Reminders	If a system reminds users of their target behavior, the users will more likely achieve their goals.
		F11: Suggestion	Systems offering fitting suggestions will have greater persuasive powers.
		F12: Similarity	People are more readily persuaded through systems that remind them of themselves in some meaningful way.
		F13: Liking	A system that is visually attractive for its users is likely to be more persuasive.
		F14: Social role	If a system adopts a social role, users will more likely use it for persuasive purposes.
System Credibility Support	The credibility category relates to how to design a system so that it is more credible and thereby more persuasive.	F15: Trustworthiness	A system that is viewed as trustworthy will have increased powers of persuasion.
		F16: Expertise	A system that is viewed as incorporating expertise will have increased powers of persuasion.
		F17: Surface credibility	People make initial assessments of the system credibility based on a firsthand inspection.
		F18: Real-world feel	A system that highlights people or organization behind its content or services will have more credibility.
		F19: Authority	A system that leverages roles of authority will have enhanced powers of persuasion.
		F20: Third-party endorsements	Third-party endorsements, especially from well-known and respected sources, boost perceptions on system credibility.
		F21: Verifiability	Credibility perceptions will be enhanced if a system makes it easy to verify the accuracy of site content via outside sources.
Social support	The social influence category describes how to design the system so that it motivates users by leveraging different aspects of social influence	F22: Social learning	A person will be more motivated to perform a target behavior if (s) he can use a system to observe others performing the behavior.
		F23: Social comparison	System users will have a greater motivation to perform the target behavior if they can compare their performance with the performance of others.
		F24: Normative influence	A system can leverage normative influence or peer pressure to increase the likelihood that a person will adopt a target behavior.
		F25: Social facilitation	System users are more likely to perform target behavior if they discern via the system that others are performing the behavior along with them.
		F26: Cooperation	A system can motivate users to adopt a target attitude or behavior by leveraging human beings' natural drive to co-operate.
		F27: Competition	A system can motivate users to adopt a target attitude or behavior by leveraging human beings' natural drive to compete.
		F28: Recognition	By offering public recognition for an individual or group, a system can increase the likelihood that a person/group will adopt a target behavior.

Figure 7. Features of the PSD model [Oinas-Kukkonen and Harjumaa 2009] [Oinas-Kukkonen 2013]