Software Transparency

Software transparency is a new and important concern that software developers must deal with. This paper reports on initial findings on exploring the obstacles for enabling software transparency. For providing a definition of transparency and understanding the semantics of software transparency, a SIG (Softgoal Interdependence Graph) is used, which has been refined in three versions. Based on three example situations we demonstrate the application of the transparency SIG.

DOI 10.1007/s12599-010-0102-z

The Authors

Prof. Julio Cesar Sampaio do Prado Leite Ph.D. (✉)
Departamento de Informática
Pontifícia
Universidade Católica
do Rio de Janeiro
R. Marquês de São Vicente, 225
Rio de Janeiro 22453-900
Brasil
julio@inf.puc-rio.br
url: http://www.inf.puc-rio.br/~julio

Claudia Cappelli Ph.D.
Departamento de Informática
Aplicada
Universidade Federal do Estado
do Rio de Janeiro
Rio de Janeiro,
Brasil
claudia.cappelli@uniriotec.br

Received: 2009-09-08
Accepted: 2010-02-12
Accepted after one revision by the editors of the special focus.
Published online: 2010-04-27

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Leite JCSP, Cappelli C (2010) Softwaretransparenz. WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-010-0219-1.

1 Introduction

Transparency is a concept related to information disclosure, having been used in different settings, mostly related to the empowering of citizens with regard to their rights. We argue that, in order to implement transparency, society will need to address how software deals with this concept. However, from the point of view of information system design and its supporting software, dealing with transparency poses new research questions.

We shall use a real scenario to introduce transparency and show how the latter affects software, and as such, the process of software construction. In October 2009, the city of Rio de Janeiro was chosen to host the 2016 Olympics Games. Weeks later, the city mayor announced an e-government service to allow citizens to follow government actions regarding the preparation for the games. Special attention was to be given to the financial budget with detailed information about city actions for the 2016 games. The service, named "Transparência Olímpica" (Olympic Transparency), can be accessed through the address http://www.transparenciaolimpica.com.br/.[1] It lists the first contracts, with expenditure scheduling and attached values.

Given that this e-government service is running and citizens are using it, we decided to find out how users of the service are evaluating it. Instead of using the traditional survey or a questionnaire regarding users' approaches, we looked for what they are saying about the service. Our approach was only possible precisely because of the recent trend towards individual transparency, in which citizens at large are using the Web (via blogs and micro blogs) to disseminate their opinions or their observation of real life facts. As such, we query the Web using a combination of the following keywords: "blogspot," "wordpress," "twitter," "transparência olímpica," and "problemas" (problems), targeting content providers for blogs and micro blogs (Blogspot, Wordpress and Twitter) and the topic of interest. We have found several manifestations regarding the service, but selected a few individual instances to highlight how citizens are dealing with the service. We list four observations in the form of complaints: (a) data was not being updated, (b) information had been deleted because the Mayor had indicated it to be confidential, (c) information was not accountable,[2] (d) information was not detailed enough. **Table 1** lists the Web addresses for each of these observations.

We understand that these observations reveal concerns regarding information disclosure. Citizens are complaining that the e-government service which was supposed to provide transparency is failing. It is important to stress that each of these observations was produced independently by different citizens. This scenario helps to show that the information system built to deliver the e-government service was not effective. Different issues are at stake here. In (a), data quality is at fault due to lack of updating. In (c), an auditing process should be in place. In (d) the quality of data is challenged for not being detailed enough. On the other hand, (b) shows that transparency is not always desired, as it may conflict with confidentiality.

This real situation is a demonstration that, as services are available for citizens, the latter, as users, will pose different sorts of issues related to transparency. Increasingly, information system designers and software engineers will need to address these quality issues. Organizations will be required to ensure that their computerized processes be transparent. That

[1] Visited in December 2009.
[2] That is, there were no explanations as to the origins of the information.

Table 1 Addresses for the observations on the E-Government service

Observation	Web Address	Date
(a)	http://twitter.com/fiscalizarj2016	Dec. 09
(b)	http://esportebrasilis.blogspot.com/2009/10/transparencia-olimpica.html	Dec. 09
(c)	http://intra-cranianos.blogspot.com/2009/10/olimpiadas-2016-e-outros-assuntos-para.html	Dec. 09
(d)	http://www.imil.org.br/blog/portais-de-transparencia-nao-garantem-fiscalizacao-de-gastos-com-olimpiada/	Dec. 09

is, society will demand not only that information being processed by software be disclosed, but also will seek to know about the processes which produced the information. Our research addresses the following research question: how should we build software systems supporting the demand for transparency? Our contribution is framed by a major insight: we understand that, to provide transparency, we must deal with it in the context of requirements specification. Given that providing transparency is a new requirement for software systems, we show how our approach builds on top of previous knowledge on requirements engineering, mainly the work related to Non-Functional Requirements (NFR).

Further to this Introduction, the paper has five sections. Section 2 provides an overview of the literature to present a general understanding of transparency levels and enumerate the problems for achieving transparency. Section 3 describes why requirements engineering must play an important role in software transparency. Section 4 details a major result so far, the production of an NFR catalogue for transparency, including its initial validation. Section 5 provides an example of transparency use on an Information System process (workflow) and revisits the "Transparência Olímpica" case. Section 6 concludes, stressing contributions and future research.

2 Transparency

2.1 Literature Review

Four books were influential in our understanding of transparency. Holzner and Holzner (2006) provide an in-depth study from the social and historical perspectives on what they see as a movement to open government, in which transparency is key to achieving more open and democratic societies. Henriques (2006) examines different constituents of transparency as a concept and frames them in the context of organizations, claiming that transparency will be essential for successful organizations. Lord (2006) provides arguments showing that increasing levels of transparency do not imply more democracy and peace, as such insights are located at the limits of transparency. Fung et al. (2007) use the concept of target transparency as a way for organizations to reduce specific risks or performance problems through selective disclosure and does this by providing a careful analysis of the constituents of transparency.

The issues of transparency are directly linked to information processing or information technology. Lawmakers are aware of its increasing role and several laws have been written with respect to data protection (Directive 95/46/EC of the European Parliament; European Commission 1995), data availability (Brazilian habeas data legislation; Republic of Brazil 1997), and access to information (FOIA-Freedom of Information Act; United States Department of Justice n.d.). Even more specific legislation, in special concerning the financial sector, has been written (Sarbanes-Oxley Act; U.S. Government Printing Office 2002). Weber (2008), from the point of view of law analysis, describes the ICANN (Internet Corporation for Assigned Names and Numbers) documentation's problems with regard to transparency. Weber's (2008) framework analysis sees three types of transparency: procedural, decision and subjective.

Few works have dealt with the transparency theme in the software context. Meunier (2008) states that software transparency is a condition in which all software functions are disclosed to users, and he argues that transparency is a pre-condition for proper risk management. Camp (2006) noticed that the open source movement does provide a way of disclosing software information: "The critical feature of open code is that it can be read by humans. Open code enables informed discourse about digital process application, and the assumptions underlying both," and she also observes that both law and computer programs are both called "code." Camp (2006) cites Stallman (1999) "…computer code controls and enables the actions of users, and for users to have true autonomy, they must be able to examine, alter, and redistribute the code" and stresses that this statement is key when government activities are embedded in computer code.

Notwithstanding, as Camp observes: open code does not guarantee transparency. Several situations may occur: for instance, if open code is provided as binary code, it would certainly not be easily read by humans. Source code may be written in such a way that it would be very hard to read; Camp cites a contest held at CMU IOCCC (2009) the objective of which being to produce obfuscated code, which is very hard-to-read code. Since source code may be written in different computer programming languages, then the issue of literacy in that specific programming language may also be an issue contributing to obfuscation. Of course, that code which is protected or which is not open is obfuscated. It is also true that open code does not warrant that the source code is the one the machine is using. Literature on electronic voting stress this as a key point (Bishop and Wagner 2007; Paul and Tanenbaum 2009), in particular, Paul and Tanenbaum have delved into the issue, mostly from the point of view of security, but making sure a process is in place to consider voting process transparency, by means of free software (Stallman 2009).

A report to the National Research Council (Jackson et al. 2007), produced by a team of scientists chaired by Daniel Jackson identified transparency as the key issue with respect to dependability, a crucial quality for software systems. They argue that software producers should disclose their claims of dependability by making their claims, criteria and evidence available. This disclosure will provide users or customers with the grounds for informed choice. A manifesto by Weitzner et al. (2008) believes

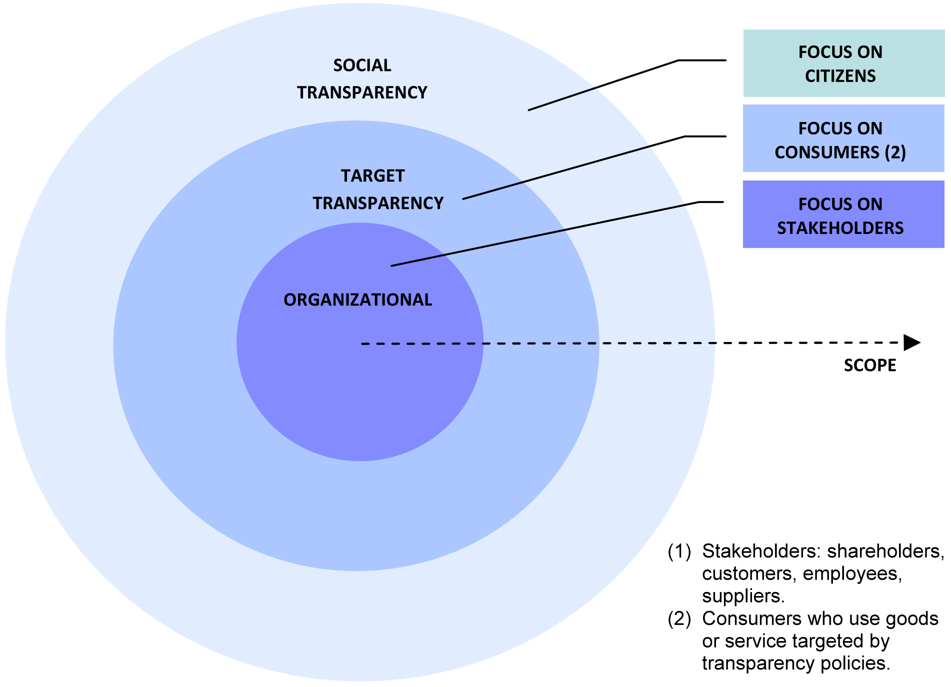

SOCIAL TRANSPARENCY
TARGET TRANSPARENCY
ORGANIZATIONAL
FOCUS ON CITIZENS
FOCUS ON CONSUMERS (2)
FOCUS ON STAKEHOLDERS
SCOPE
(1) Stakeholders: shareholders, customers, employees, suppliers.
(2) Consumers who use goods or service targeted by transparency policies.

Fig. 1 Transparency contexts (Wikipedia 2009)

that transparency is related to the right customers have to see their data and associates transparency with accountability, as it requires that information usage be transparent.

2.2 Key Concepts

The literature review helped us to unveil abstract key concepts, which summarize our understanding of transparency in general, as well as in the software context. First, we present an organizational view of transparency contexts. Second, we stress the importance of differentiating information transparency and process transparency. Third, we point out key issues that are related to software.

The literature mixes different transparency contexts. We have mapped these contexts into an "onion" chart (**Fig. 1**) to characterize the reach of transparency (Wikipedia 2009). In that Figure, we see that Social Transparency is geared towards citizens in general; Target Transparency aims at consumers of some service or goods, and Organizational Transparency focuses on an organization's stakeholders. Since automation is a key factor in modern society, software transparency will need to deal with the different focuses shown below, being orthogonal to these contexts.

Although most of the literature focuses on information transparency, at least Weber (2008) deals explicitly with process transparency. A better understanding of transparency makes it clear that the processes that produce information should themselves be transparent. This is particularly important in the case of software.

Let's use a set of possible situations to exemplify the importance of stressing the difference made between transparency of existing information and transparency of how things happen. Suppose information does exist in a company regarding safety emission levels for a certain artifact. Requiring that the information be available to customers is a form of transparency policy enabling access to the information. On the other hand, supposed one wished to know how the artifact is assembled. In this case, one will need information on the process used to assemble the artifact; one will require that this process be transparent. A citizen may be willing to buy a certain artifact with some level of radiation, as informed by company owner, but may not be willing to buy that product if people in the assembly line were exposed to higher levels of radiation. In the Introduction, one of the observations, (c), from citizens, was that the information was not verifiable, that is, the citizen was interested in knowing about the process which would guarantee the information.

With regard to software we have learned that transparency may be required for different reasons, may be related to different quality issues and is a complex matter. In Bishop and Wagner (2007) and Paul and Tanenbaum (2009) transparency is claimed to be necessary for e-vote applications, in Meunier (2008) it is seen as demanded by risk management and in Camp (2006) as necessary for software dealing with government activities. With respect to quality characteristics, we see it related to dependability (Jackson et al. 2007), to trust (Bishop and Wagner 2007; Paul and Tanenbaum 2009), to accountability (Weitzner et al. 2008) and to security (Paul and Tanenbaum 2009). Dealing with so many implications and different types and contexts is a complex endeavor, which will require specialized knowledge. The next section will detail the research challenges related to the matter.

2.3 Research Questions

Early use of the term transparency in computing was misleading.[3] The expression "transparent to the user" commonly referred to a situation where the user was using a black box, with internal details hidden away. Although the correct meaning of the word transparency is finding its way in Computer Science jargon, as seen

[3]The Wikipedia entry for transparency (computing) (http://en.wikipedia.org/wiki/Transparency(computing)) and the IBM Terminology (http://www-01.ibm.com/software/globalization/terminology/tu.jsp#t19) are examples of this usage (visited in December 2009).

in the literature cited, there is still the issue of what transparency means exactly in the context of research and practice.

What does it mean having software that is transparent? How to implement transparency? What is the implication of producing transparent software? What type of methods and tools will be needed to fulfill this demand? How does transparency relate to other quality characteristics? How does transparency relate to quality characteristics that seem to oppose transparency? What level of abstraction should we be dealing with? Is code transparency (Camp 2006) sufficient?

Our approach towards these questions is framed by our perception that dealing with software transparency should be regarded as a new quality requirement. The next section argues why we believe that requirements engineering should be the proper context in which to address software transparency.

3 Attaching Requirements to Software

It is interesting to note that Camp's discussion of information disclosure (Camp 2006) is centered at the code level. This disclosure is not just of the code as information, but also of the process that it entails. In the previous sections, we have outlined arguments showing that code level is not the correct abstract if we wish to avoid the code level of detail, avoid the peculiarity of different programming languages, and target transparency to a broad reach beyond programmers.

However, process transparency requires that the transformation steps of the process be transparent, to say that it is possible to understand its enactment. The problem is compounded when dealing with process information, since the focus is not just on understanding data, but also one involving processes and actors, as well. Similarly, in the software production context, requirements engineering is fundamental to understand what is required from the automated process, i.e., software. This line of reasoning leads us to believe that software engineers must deal with transparency during requirements definition. Moreover, Mylopoulos observes,[4] when presented to the idea of software transparency, that: "transparency is an interesting quality because it makes it necessary to attach requirements models to software."

With this insight, Mylopoulos posits that requirement models are a right vehicle for the openness necessary for transparency. In addition, it demands that requirements models need to be attached to software, which brings up the issue of trust and traceability of the code.

Using as lemma the fact that code is implementing requirements models, if the requirements models are transparent, then the code will be transparent. This assumption brings the problem of software transparency to a high level of abstraction. It is important to note that, by Mylopoulos' observation, the fact of attaching requirements models to code is necessary – but it is not stated that this is sufficient. In order for the lemma mentioned above to be true, we have to make sure that the code conforms to the requirements and that traceability back and forth is possible in order to support verification tasks.

This argumentation brings the problem of software transparency to the realm of requirements, posing new challenges in an area that has evolved rapidly since its characterization in 1993, with the First IEEE International Symposium on Requirements Engineering. As such, we could understand that some of the questions raised in Sect. 2 should be answered mainly from the perspective of requirements engineering, making sure that requirements models are transparent. Preliminary work on the suitability of modeling languages to transparency requirements (Cappelli et al. 2007) concluded that an intentional model[5] (Leite and Cappelli 2008; Yu 1994) is better for the task, since who (actors) and why (goals) are explicitly represented. As such, dealing with transparency was framed as dealing with a quality requirement, i.e., a non-functional requirement (Chung et al. 2000), or a softgoal, using the terminology of intentional modeling (Mylopoulos et al. 1992). Understanding transparency as a non-functional requirement and framing the problem of achieving transparency in the context of intentional modeling brings forth different possibilities for exploring the issues raised before. The central one, providing a definition of transparency, is rewritten as finding a SIG, a Softgoal Interdependence Graph (Chung et al. 2000). In Sect. 4, we detail how this SIG was built and validated over a series of versions.

Another consequence of explicitly representing non-functional requirements as softgoals is the capability of reasoning (Giorgini et al. 2002) about contribution links, which is basically the problem of requirements interaction (Robinson et al. 2003), and dealing with antagonistic requirements (Cappelli 2009). We will deal with these issues in Sect. 5. The issue of how to attach requirements to software is dealt with in Sect. 6.

4 Transparency as an NFR

Why should we treat transparency as a non-functional requirement? Different reasons make us believe this is a proper position. First, it is a quality issue; that is, it is orthogonal to the software functionality. Having transparency or not having transparency will not impact what the software does. Second, the characteristic is general, and as such, spreads to different parts of a given software system. Although quality issues can be modularized, as for instance by means of aspect-oriented ideas, it is usually required by different functional parts of a given artifact. Third, it is not amenable to the typical measurement treatment as the one applied to functional characteristics; that is, we cannot say that something is or is not transparent. We will need to use a less objective judgment, like almost transparent, or transparent enough, and so on. In that sense, transparency is a kind of characteristic that defies the notion of satisfaction in the traditional sense. The NFR uses Simon's ideas to understand the degree of how a softgoal is fulfilled. Simon (1969) coined the term "satisfice," which is central to his theory of decision behaviors. Fourth, our group has been working with the NFR framework for some time now (Cysneiros et al. 2003; Chung and Leite 2009).

Why should we use The Non-Functional Requirements Framework (Chung et al. 2000) to represent transparency? The NFR Framework (Chung et al. 2000) was devised to promote the non-functional requirements as first citizens in requirements models. It has been

[4]Personal communication.

[5]Intentional modeling refers to requirements modeling languages that explicitly deal with goals. Kaos (van Lamsweerde 2009) and i* (Yu 1994) are the most prominent ones.

widely used and evolved in requirements engineering research, mainly due to its uniqueness in dealing with the interactions of requirements, being unique in this approach. Instead of modeling requirements just as functions to be performed, the framework addresses quality attributes such as: security, usability, performance and precision, for example. As it promoted these characteristics, the NFR framework acknowledged their substantial difference with respect to functional characteristics and used Simon's ideas of "satisfice." This difference led to the characterization of non-functional requirements as softgoals, that is, goals that could be achieved but would need a notion of "satisfice" instead of the traditional satisfaction to measure its degree of achievement. As such, the NFR framework and its main model, the SIG, Softgoal Interdependence Graph, is an instance of intentional modeling, and well suited to deal with a soft concept, as is the case of transparency.

A SIG is composed of nodes and links. The nodes are either a softgoal or an operationalization of a softgoal's nodes that are named with the *type* (quality) of the softgoal and the *topic* (the context in which the softgoal is being applied). *Type* is the jargon used by the NFR framework to distinguish between the title of the softgoal and the *topic* to which it is applied. Links are either contributions links or correlations links. Links are labeled to describe their strengths (make, help, hurt, break), or whether they are decomposition (AND) links or specialization links (OR). Contribution links are solid arrows and correlation links are dashed arrows. Contribution links are used to describe a hierarchy of softgoals, and correlations are used to describe relations among different hierarchies.

In this section, we describe the several interactions we performed and their according versions for finding a transparency SIG. Each version was built and validated by a different strategy. The denotation of each *type* was, mainly, extracted from Wordnet (http://wordnet.princeton.edu/) with some adaptations and is shown in **Table** 2.

4.1 Version 1

We have reported our first attempt to encode transparency as a network of non-functional requirements in Cappelli et al. (2007).

The SIG is the representation used in the NFR framework for showing the relationships among different softgoals. In **Fig.** 2 SIG we have used the "some+" edge, meaning that there is a positive contribution of unknown strength from the nodes towards their ancestors. This graph has 34 softgoals, including transparency itself, and four softgoals were factored: Usability, Auditability, Accessibility and Informativeness. This graph depicts that these quality factors contribute to the notion of transparency, and, as such, their fulfillment, through the notion of "satisfice," will provide a degree of how transparency would be "satisficed."

In order to produce this graph, we have followed a process composed of three main steps.

(1) The first step used an elicitation strategy based on the systematic review procedure (Biolchini et al. 2005). A systematic review is a process to guide literature review using well established criteria. It has been used in the research area of software experimentation.

The following characteristics drove the systematic process for literature review:

- *Keywords*: transparency, organizational transparency, software transparency process transparency.
- *Paper and book Sources*: Internet (Google Portal) and our university central library
- *Intervention*: In the search the similarity between concepts will be observed.
- *Effect*: At the end of this systematic review, a collection of characteristics to better define transparency concepts in organizational context should be available.
- *Application*: The organizations will know what is expected from them when someone or some organism expects transparency.
- *Experimental Design*: The literature in different knowledge areas will be studied to extract the meaning of transparency in this reference context. Then characteristics cited about transparency will be analyzed and organized, thereby identifying their commonality.
- *Source Selection*: To be wide-ranging the search began on the Internet using the Google portal to discover which areas use the term "transparency." To complement this first step, another search was made at the central library. During this work, we discovered the use of this term in some areas, such as: Computer Science, Communication, Sociology, Physical, Cinema and Politic Science.
- *Source Identification*: The information obtained in the books at the library was manually collected through reading. We have used Google with the following keywords: transparency, organizational transparency, software transparency and process transparency.
- *Inclusion and exclusion criteria*: The papers and books must be available on the Internet or at the library, provide a transparency definition and explain transparency characteristics aiming to answer the first two questions.
- *Preliminary study selection process*: Each publication obtained had its abstract or summary analyzed and, based on the inclusion and exclusion criteria, some were selected.

The review identified 10 sites, 20 books[6] and 15 scientific papers and, following analysis, 3 sites, 5 books and 1 scientific paper were selected as information sources. By using these sources, we have produced a list of quality terms associated with transparency.

(2) The second step compared the list produced by step one with the Chung et al. (2000) non-functional requirements list. The intersection of these two lists produced a list of *types* related to transparency.

(3) The third step was the construction of the graph shown in **Fig.** 2. The following heuristics were used to produce the graph: (a) First, we studied relations among the types and separated them into groups, (b) and for each group we identified the dependencies between the types. (c) If one type depends on others to be achieved, then this one will be placed on a higher level. The resulting SIG was comprised of 34 nodes arranged in 3 levels.

The second level of decomposition was formed by the following softgoals: Usability, Auditability, Accessibility and Informativeness. All of the softgoals were mapped as "some+", a relationship denoting a positive contribution in order to achieve the higher softgoal.

[6] The books (Holzner and Holzner 2006; Henriques 2006, Fung et al. 2007) were not included in this survey. We have found them very recently.

Table 2 Definitions for the types used in the Transparency SIG

NFR Framework characteristics	Definitions
Accessibility	**The quality of being easy to meet deal with**
Portability	The quality of being light enough to be carried
Availability	The quality of being at hand when needed
Publicity	The quality of being open to public view
Usability	**The quality of being able to provide good service**
Uniformity	The quality of lacking diversity or variation
Simplicity	The quality of being free from difficulty or hardship or effort
Operability	The quality of being treated by surgical operation
Intuitiveness	The quality of being spontaneously derived from or prompted by a natural tendency
Perform ability	The ability of giving a good performance
Adaptability	The ability to change (or be changed) to fit changed circumstances
User-friendliness	The ability to use easily
Informativeness	The quality of providing or conveying information
Clarity	The ability to be free from obscurity and easy to understand
Completeness	The quality of being complete and entire; having everything that is needed
Correctness	The quality of being conform to fact or truth
Current	The quality of occurring in or belonging to the present time
Comparable	The ability to be compared
Consistency	The ability to express logical coherence and accordance with the facts
Integrity	The quality of being undivided or unbroken completeness, or totality with nothing wanting
Accuracy	The quality of being near to the true value
Understandability	**The quality of comprehensible language or thought**
Conciseness	The ability to express a great deal in just a few words
Composability	The ability to put together out of existing material
Decomposability	The ability of separating into constituent elements or parts
Extensibility	The quality of being protruded or stretched or opened out
Dependability	The quality of being dependable or reliable
Auditability	**The ability to examine carefully for accuracy with the intent of verification**
Validity	The quality of being valid and rigorous
Controllability	The ability of being certain of something
Verifiability	The quality of being tested (verified or falsified) by experiment or observation
Traceability	The quality of following, discover, or ascertain the course of development of something
Accountability	The quality of being explained; made something plain or intelligible

4.2 Version 2

Later on, we re-organized the transparency SIG using a collaboration process with six software engineering researchers. The goal of the process was to discuss the SIG previously produced. The process used clustering techniques together with a consensus meeting. The participants were: two Ph.Ds, two Ph.D. candidates and two Masters' degree candidates, all of them working in the software engineering field. Each subject was asked to review the existing SIG and to propose a new clustering of the softgoals. At this time, the group understood that, instead of using the "some+" contribution, we should follow the Chung et al. (2000) decomposition strategy for SIG catalogues, and, as such, we have used the AND relationship which is a stronger linkage than "some+". Each subject brought his or her proposal to a meeting and presented to the group. A moderator used the whiteboard to consolidate the consensus. After drawing different alternatives, the group reached an agreement over a new SIG (**Fig.** 3). The resulting SIG was composed of 33 nodes arranged in 3 levels. The second level of decomposition was formed by the following softgoals: Accessibility, Usability, Informativeness, Understandability and Auditability. As can be seen in the second level, one more node was created. The third level nodes were redefined and rearranged.

Fig. 3 was submitted to 16 international modeling experts,[7] who used a questionnaire to evaluate the proposed SIG (**Fig.** 3) and to suggest some changes. It is interesting to note that one of the

[7]A number of participants of the February 2008 IFIP 2.9 meeting and participants of the 3rd International i* Workshop.

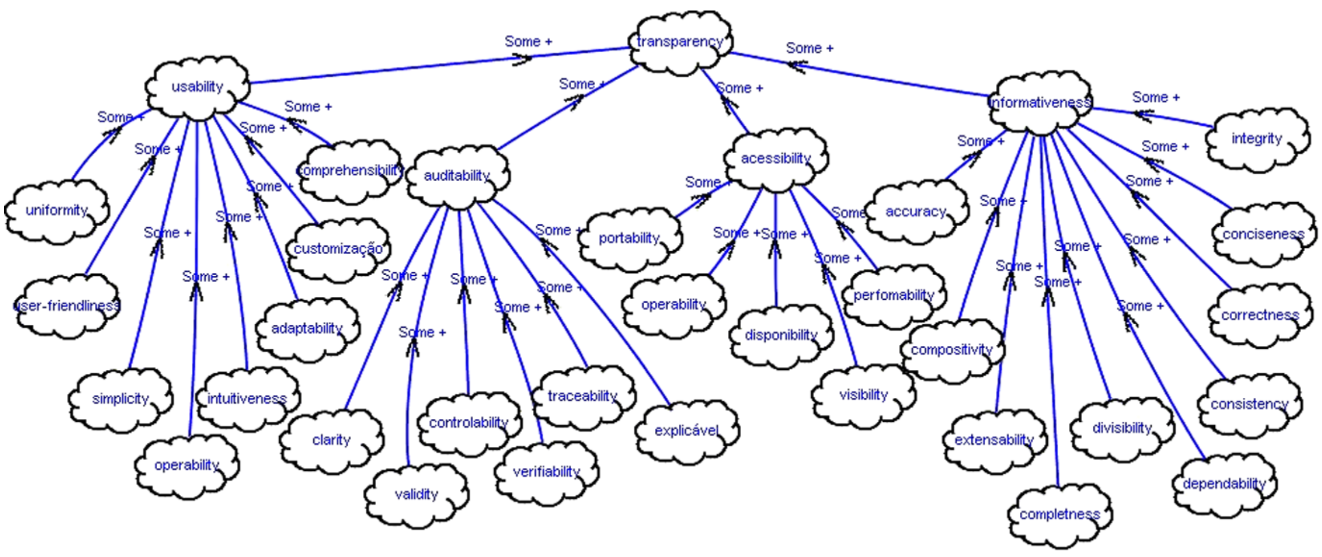


Fig. 2 Softgoal interdependency graph (SIG) – Version 1

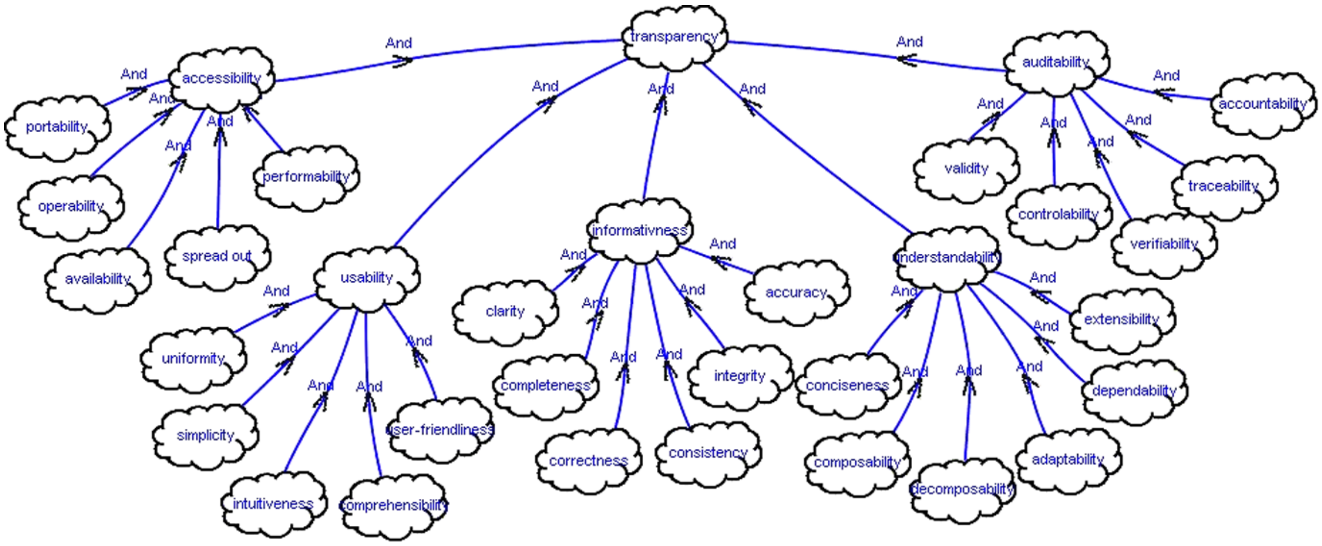


Fig. 3 Transparency network (Leite and Cappelli 2008) – Version 2

suggestions was the replacement of the AND by the "Help" contribution.

4.3 The Final Version

Cappelli (2009) continued refining the SIG, using questionnaire responses submitted to 16 international modeling experts and including the correlations among leaf softgoals in different sub-trees (see **Fig. 4**). These correlations were found based on the application of an on-line questionnaire (http://pes.inf.puc-rio.br/questionario/), which was answered, anonymously, by approximately 20 people.

The Transparency SIG (Cappelli 2009) is the first systematization of transparency of which we are aware; this has been done using the NFR Framework semantics of decompositions, collaborations and operationalizations (Chung et al. 2000). Operationalization, in the NFR Framework (Chung et al. 2000), is the linkage of a non-functional requirement to possible "implementations" in functional terms. The next Section, in which we explore the use of the Transparency SIG, shows an example of an operationalization.

5 Applying the Transparency SIG

This Section is organized by the presentation of three distinct examples. The first one is based on Cappelli's thesis of bringing transparency to business processes; the second one is based on the analysis of different types of elections systems; and the third is based on the e-government service case, "Transparência Olímpica."

5.1 Bringing Transparency to Business Processes

Cappelli (2009) refined the third level of SIG, by defining possible operationalizations of each of the leaf softgoals in the context of her doctoral thesis on applying the transparency concept towards business processes. **Fig. 5** shows a possible operationalization of the softgoal Accountability within the domain of busi-

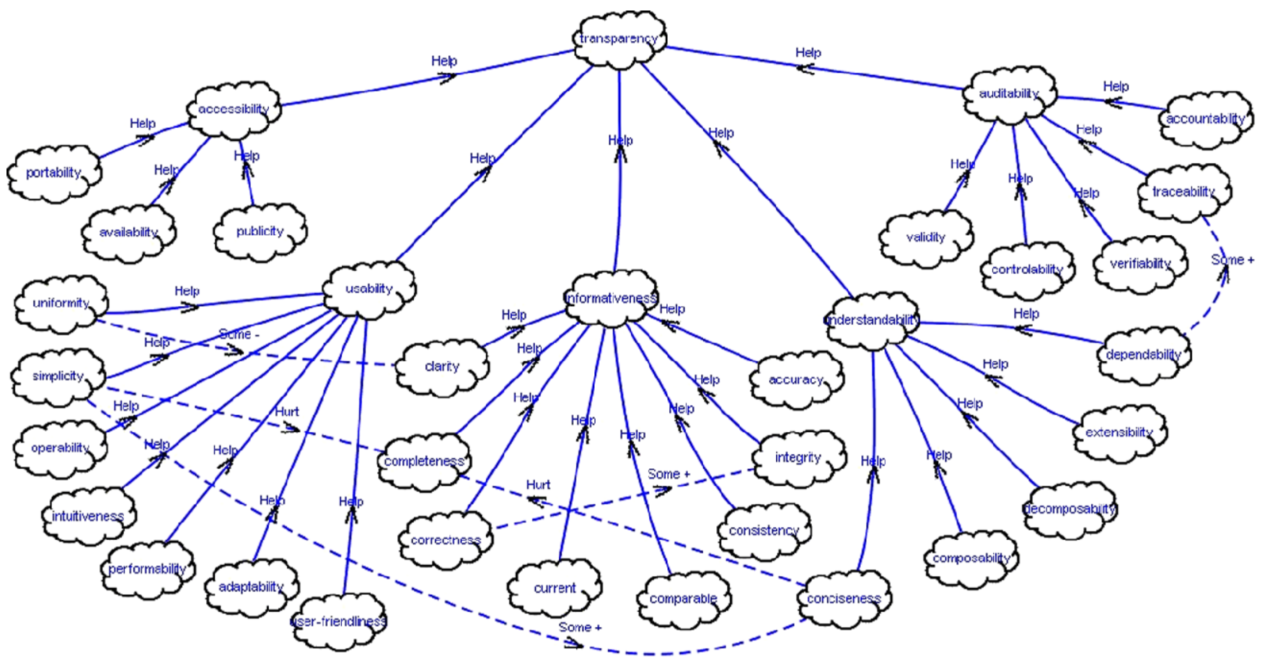


Fig. 4 Transparency SIG (Cappelli 2009) – final version

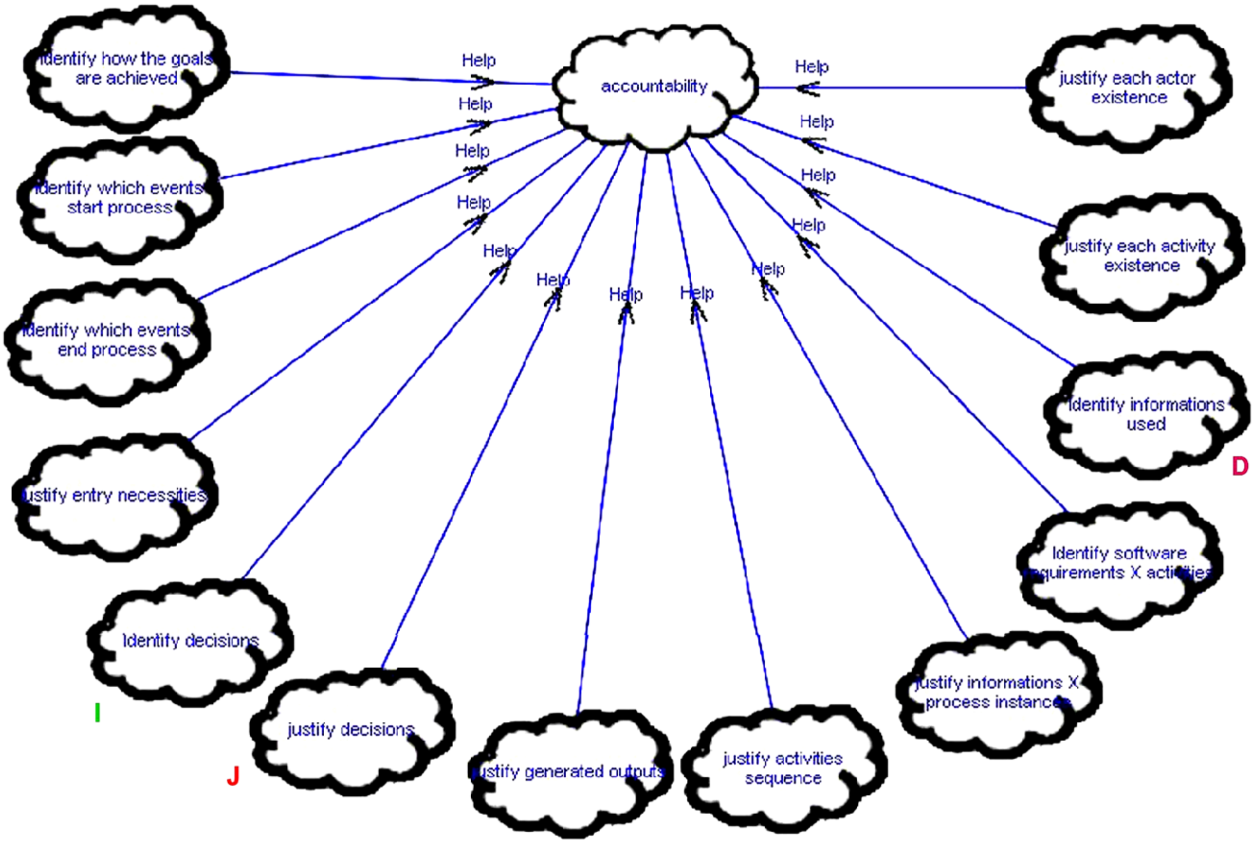

accountability
Identify how the goals are achieved
Identify which events start process
Identify which events end process
Justify entry necessities
Identify decisions
justify decisions
Justify generated outputs
justify activities sequence
Justify informations X process instances
Identify software requirements X activities
Identify informations used
justify each activity existence
justify each actor existence
Help
D
I
J

Fig. 5 The operationalization of accountability (Cappelli 2009)

ness processes, and, as such, the *topic* "Business Process" is part of the node name. It is important to notice that, in this case, Accountability is seen as helping Auditability, which "Helps" Transparency.

Using a partial description of a software acquisition process, we have applied three operationalizations from the Transparency SIG (**Fig. 5**) to illustrate a new version of the process which will be "more" transparent, for the Account-

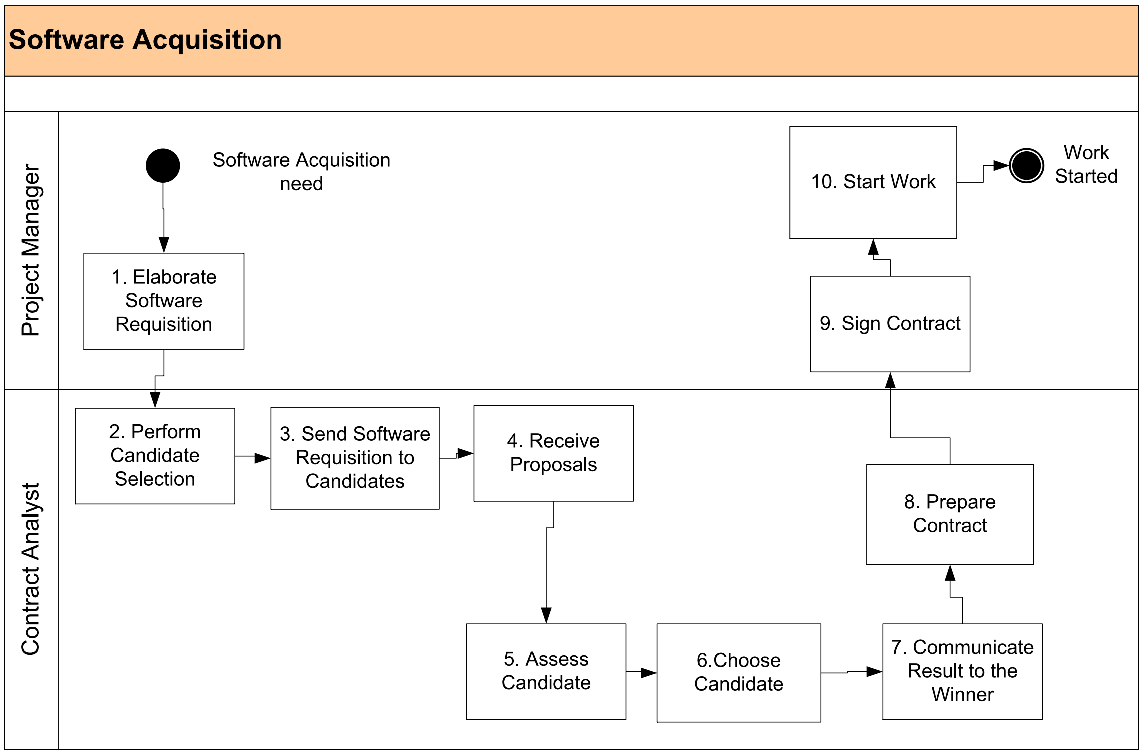


Fig. 6 A software acquisition process

ability operationalizations included in the process will "Help" the Auditability, which will "Help" Transparency (see **Fig.** 4).

Fig. 6 provides, using a BPMN (Business Processing Modeling Notation), a partial description for a software acquisition process in an organization. In this version, the process does not consider the issue of Auditability and, as such, bears a problem with Transparency.

Fig. 7 shows a new description for the software acquisition using two Accountability SIG activities (**Fig.** 5): Identify decisions in the process (marked with I) and Justify decisions in the process (marked with J). As such, two new activities are added to the process (**Fig.** 7, activities A and B). Another Accountability operationalization function, Identify information used in the process (marked with D), requires the addition of three documents to the process (**Fig.** 7, documents C, D, E).

If we do compare the two versions of the process, it is reasonable to claim that the second version, **Fig.** 7, makes the contract analyst perform two activities that will be of fundamental importance for any auditing operation in the future. It is also reasonable to claim that, by making the documents in the process explicit, the organization is making sure that accountability is being enforced.

This is only a small example of the role that Transparency SIG may have in the efforts towards making processes more transparent. Note that if process descriptions do consider transparency, these requirements will need to be cast on the supporting software as well.

5.2 Analyzing Correlations in the Context of Electoral Systems

An electoral system is an information system that supports an election process. In Cappelli et al. (2010) we explored how transparency would interact with security as desired quality characteristics of three types of election processes. We compared the Transparency SIG (**Fig.** 4) with a Security SIG taken from (Chung et al. 2000) in the analysis involving an indirect elicitation strategy entailing seven stakeholders.

The purpose of the elicitation was to find correlations among softgoals. As we can see from **Fig.** 8, we elicited several correlations in the Transparency and Security SIG. Most of them, surprisingly, are positive correlations, with only two negatives, which are those involving Confidentiality and Traceability and Auditability and Confidentiality. This example shows a way of dealing with the relationship of transparency with other quality characteristics, even if they seem to oppose transparency. Of course the results found in Cappelli et al. (2010) are bound by the processes examined as well as by the viewpoint of the stakeholders consulted.

The point in this example is that analysis of transparency relationships can be done early on. The NFR analysis stressed the negative impact of confidentiality over traceability and of auditability over confidentiality, so that designers and customers of an electoral system will more clearly see that opting for a given quality will impact another. In particular, it will provide a starting point for the discussion of possible implementations. Of course, the mapping of these interactions brings more complexity towards requirements analysis, but also may uncover problems that will be hard to deal later during software design. It also helps to uncover the rationale for design decisions.

5.3 Transparency in the Case of "Transparência Olimpica" Site

In the Introduction, we have reported on the site "Transparência Olímpica" as a government service provided by the mayor of Rio de Janeiro to inform about

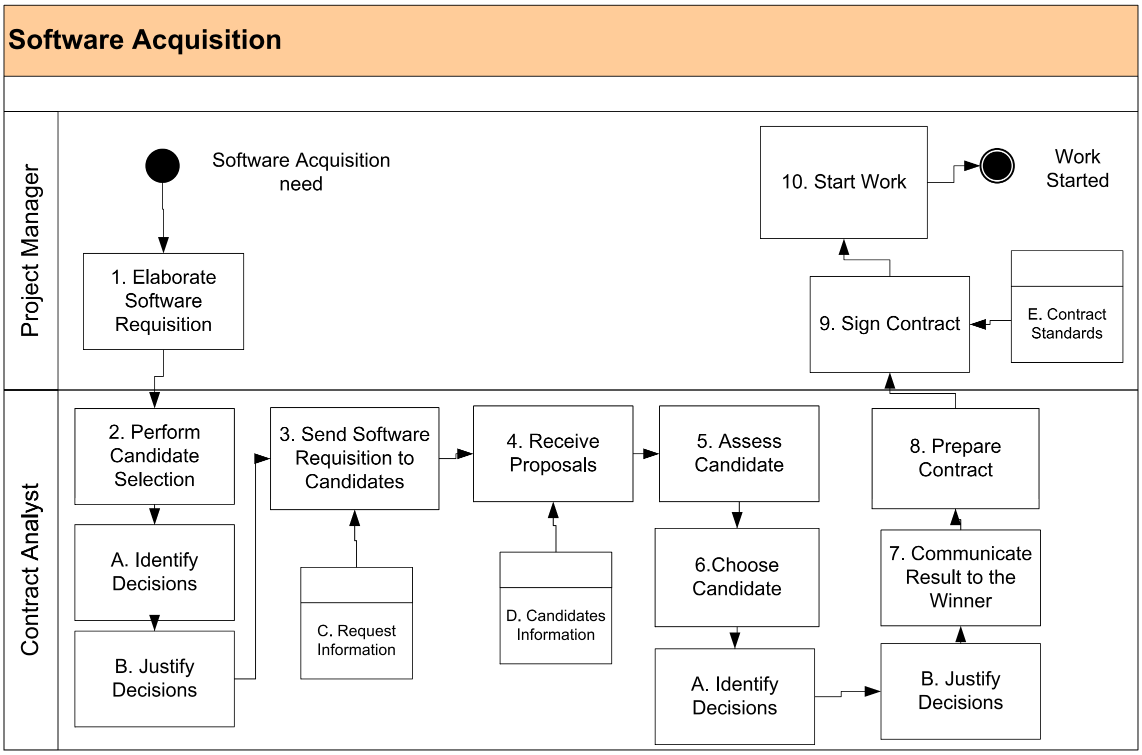
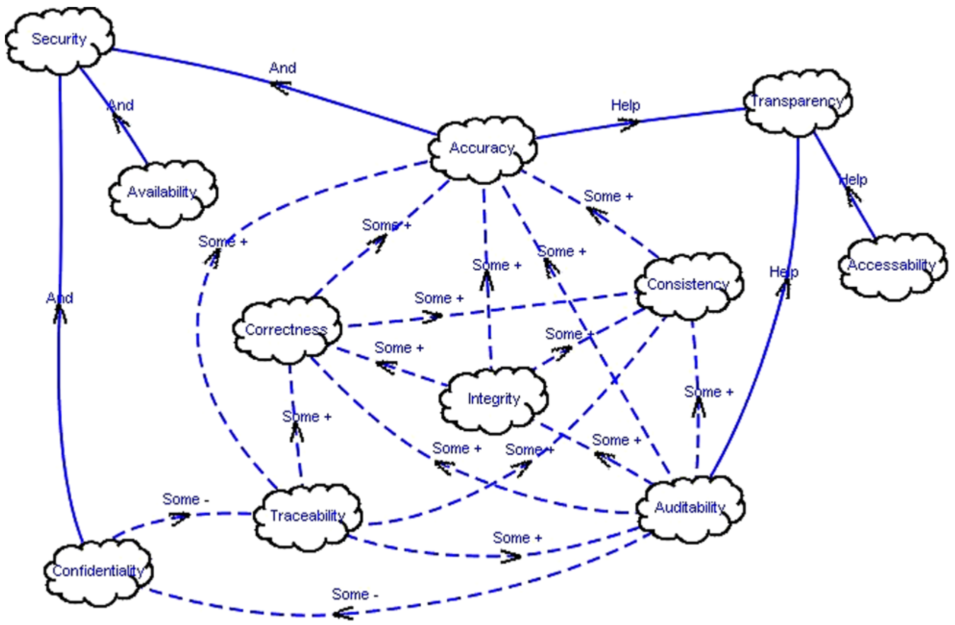


Fig. 7 A software acquisition process instantiated for transparency

Fig. 8 Security versus Transparency (Adapted from Cappelli et al. 2010). Some "Help" contributions of the Transparency SIG and some "And" contributions from the Security SIG were left out of the graph so as not to clutter it



the ongoing works for the 2016 games. We have also mentioned that, by querying the Web, we have found several manifestations regarding the quality of the service.

In that particular case, it is interesting to note that all the comments about the service could be mapped to our Transparency SIG. As such it is reasonable to posit that, if the designers of the service used the proposed SIG, all of these comments could have been addressed at the service definition time.

The following argumentation serves the goal of showing a possible answer to the question of what it means to have transparent software and how to implement it. Let us look at each observation and argue about how the SIG (see **Fig. 6** and **Table 2**) could have helped. In (a), the criteria, quality of data being updated, is exactly the softgoal Current that "Help" Informativeness to "Help" Transparency. In (b), an antagonistic analysis, similar to the one performed in Sect. 5.2, could have shown that Availability to "Help" Accessability to "Help" Transparency would "Hurt" Confidentiality; thus requiring operationalizations that would point to policies of what is considered confidential and, as such, would not be available. In c), there is the case of Accountability to "Help" Auditability to "Help" Transparency; here also requiring operationalizations that, certainly, would be similar to those enumerated in Sect. 5.1 (see **Fig. 5**).

On the other hand, we have also observed that the service does not fulfill the softgoal of Publicity to "Help" Accessibility that "Help" Transparency. We checked the "Transparência Olímpica" site using Brazilian software (daSilva), which works for Portuguese; the software checks for compliance with the WCAG 1.0 recommendation issued by the W3C. We found three occurrences of the lack of textual description for images used in the site. This fact will be an obstacle for voice browsers, a tool used by persons with visual impairment. A proper operationalization for Publicity should refer to the W3C guidelines.

6 Conclusion

We have argued that transparency is a concern that information system designers must address as society demands more openness. We described an existing e-government service and showed how citizens demand transparency from this service. Our literature review is evidence Based on available knowledge, we have stressed the key concepts: of transparency contexts, of differentiating information transparency and process transparency, and of software transparency.

Since transparency is a quality characteristic we have argued that in the context of information system design, it is proper to be dealt with during the requirements definition, and as such posing the challenges in the context of requirements engineering. In that context, the usage of the NFR framework is well justified as the basis for treating transparency. The bulk of our work has been trying to pin down the semantics of software transparency using the NFR framework. Being able to deal with the constituents of transparency, it will be possible to better evaluate the degree of transparency of a given software. Although our taxonomy, expressed by Transparency SIG, may be seen as incomplete, we have shown its utility by examples in the analysis of different situations. It is also important to stress that our taxonomy allows for variability by the combination of topic and operationalizations. So, for different topics, we may have different operationalizations, as seen in Sect. 5. We have also shown, in Sect. 5, the handling of antagonistic requirements by using the correlation links associated with strength labels.

Regarding future work, **Fig. 9** depicts an environment we have been investigating as to support software transparency. It inherits the baseline idea from a previous work (Leite et al. 1997), and uses a mix of intentional modeling (i*; Yu 1994) with a scenario and light ontology (Language Extended Lexicon; Breitman and Leite 2003). Other investigators also have pursued the combination of intentional models with a scenario-oriented model in requirements engineering (Rolland and Salinesi 2009; Castro et al. 2009). We understand that in our context, the idea of traceability automation is key, that is, the traces are automatically generated by the environment. Some first results already are available in this area (Egyed and Grünbacher 2002), and we have implemented a similar scheme in our lexicon and scenario editor (Fernandes et al. 2005). So, by taming the tracing aspect we are dealing with important support for helping the Auditability subtree (**Fig. 5**), which, however, comprises just 6/33 of the problem.

Of importance as well is exploring the issue of reusability, since the Transparency SIG with proper operationalizations by topic is a first step towards building a transparency catalogue (Cysneiros et al. 2003; Chung et al. 2000) and, as such, rendering the enactment of quality-based reusability possible (Leite et al. 2005).

Bringing the focus of transparency towards a requirements perspective is important to allow the inheritance of several results of an area, which since its inception has been looking outside the realm of software engineering. In a recent report on the challenges of requirements in the context of high complex systems, Jarke et al. (2009) have noticed the evolution from "user" to "citizen" and the consequences of this observation. As we have pointed out in **Fig. 9**, the upper area where software has to communicate with citizens and abide with the softgoals of Understandability, Usability and Informativeness is, undoubtedly, a complex issue in terms of software design.

Based on the fact that software transparency must be dealt with and building on our first efforts to deepen the understanding of what transparency is, we believe that the most pressing issues are the following ones:

- The issue of trust is a major roadblock to software transparency. Science (Kramer 2007), as well as poets,[8] have taught the importance of abstraction. We cannot deal with transparency if we will have to look at every single line of code. The volume and complexity would be enormous. We will need abstractions faithful to the real code. As such, the issue of trust is essential. How to guarantee that the requirements, which

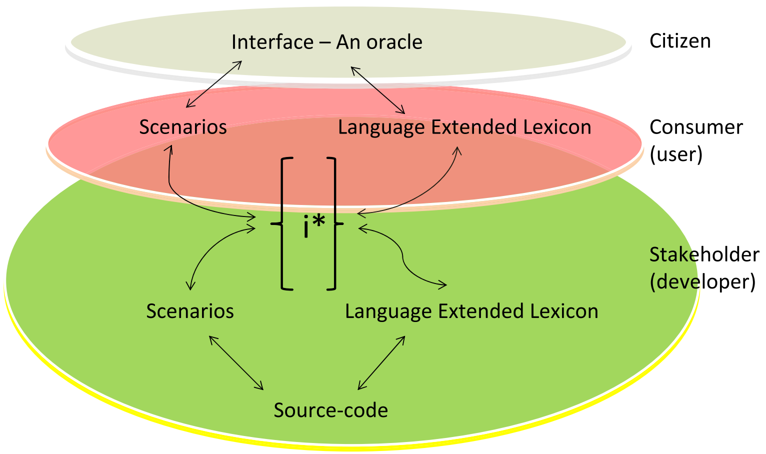


Fig. 9 The Transparency environment

[8]See "On Exactitude in Science" by Jorge Luis Borges (http://en.wikipedia.org/wiki/On_Exactitude_in_Science).

Abstract

Julio Cesar Sampaio do Prado Leite,
Claudia Cappelli

Software Transparency

Software transparency is a new and important concern that software developers must deal with. As society moves towards increased automation, if citizens wish to exercise their right to know, the transparency of public services and processes acquires fundamental importance. Informed discourse is only possible if processes affecting the public are open to evaluation. Achieving software transparency to this level of openness faces several roadblocks. The paper reports on initial findings on exploring the obstacles for enabling software transparency.

Keywords: Software, Transparency, Information transparency, Open society, Requirements engineering

will bring transparency to the processes, are being executed as planned?

- The issue of cost is a major roadblock. We, software engineering researchers, have to find ways of providing transparency without increasing the cost of producing software. It seems that the collaborative movement behind an open source is a promising starting point.
- The issue of performance is also a major concern. How to assure trust without interfering with system performance? Remember that in an ideal world people would like to be informed about how software executes its processes.
- Another challenge is how to deal with citizens as our "customers," as noted before. Software engineers have not been too keen on human-computer interaction, which was delegated to another research area (HCI), in which a great deal of progress has certainly been made. However, we foresee that dealing with transparency "customers" requires different sorts of models and strategies than needed for just dealing with users, even in a general sense, such as when dealing with interfaces for the Web. On the other hand, in the Web services context (Hendler et al. 2008), we will also have to consider software agents as demanding transparency for proper collaboration.

Of course, achieving transparency for processes – and, in particular for software – is a challenging endeavor. But we envision that this important and pressing issue must be taken under consideration by both researchers and industry. By outlining the results and the challenges for Software Transparency, we are aligned with a number of observations made by Jarke et al. (2009) with respect to requirements for facing engineering challenges.

Acknowledgements

The authors are grateful to the editors and reviewers. Their positive criticism and their assistance in helping us express our ideas were fundamental in achieving a much better text. Financial support for this work was provided by Faperj and CNPq.

References

Biolchini J, Mian PG, Natali AC, Travassos GH (2005) Systematic review in software engineering: relevance and utility. Technical report ES67905, PESC – COPPE/UFRJ, 2005. http://cronos.cos.ufrj.br/publicacoes/reltec/es67905.pdf

Bishop M, Wagner D (2007) Risks of e-voting. Communications of the ACM 50(11):120–120. doi:10.1145/1297797.1297827

Lamsweerde A van (2009) Requirements engineering: from system goals to UML models to software specifications. Wiley, New York

Breitman K, Leite JCSP (2003) Ontology as a requirements engineering product. In: 11th IEEE international conference on requirements engineering. IEEE Comp Soc Press, Los Alamitos, pp 309–319

Camp LJ (2006) Varieties of software and their implications for effective democratic government. In: Proceedings of the British academy, vol 135, pp 183–185

Cappelli C (2009) An approach for business processes transparency using aspects. Dissertation, Departamento de Informática, PUC-Rio, Ago (in Portuguese)

Cappelli C, Oliveira AP, Leite JCSP (2007) Exploring business process transparency concepts. IEEE Comp Soc Press, Los Alamitos, pp 389–390

Cappelli C, Cunha H, Gonzalez-Baixauli B, Leite JCSP (2010) 25th transparency versus security: early analysis of antagonistic requirements. In: ACM symposium on applied computing, requirements engineering track. Sierre (accepted)

Castro J, Kolp M, Liu L, Perini A (2009) Dealing with complexity using conceptual models based on tropos. In: Conceptual modeling: foundations and applications. Springer, Berlin, pp 335–362

Chung L, Leite JCSP (2009) On non-functional requirements in software engineering. In: Borgida A, Chaudhri V, Giorgini P, Yu E (eds) Conceptual modeling: foundations and applications. Springer, Heidelberg, pp 363–379

Chung L, Nixon B, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. Kluwer, Norwell

Cysneiros LM, Yu E, Leite JCSP (2003) Cataloguing non-functional requirements as softgoal networks. In: Proc. of requirements engineering for adaptable architectures. 11th international requirements engineering conference, pp 13–20

Egyed A, Grünbacher P (2002) Automating requirements traceability: beyond the record & replay paradigm. In: Proc. 17th IEEE international conference on automated software engineering. IEEE Comp Soc, Washington, p 163

European Commission (1995) Directive 95/46/EC of the European parliament – data protection. http://ec.europa.eu/justice_home/fsj/privacy/index_en.htm

Fernandes L, Leite JCSP, Breitman K (2005) C&L uma ferramenta de apoio à engenharia de requisitos. Revista de Informática Teórica e Aplicada 12(1):23–45

Fung A, Graham M, Weil D (2007) Full disclosure, the perils and promise of transparency. Cambridge University Press, Cambridge

Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastián R (2002) Reasoning with goal models. In: Proc. ER 2002. 21st international conference on conceptual modeling. Springer, Berlin, pp 167–181

Hendler J, Shadbolt N, Hall W, Berners-Lee T, Weitzner D (2008) Web science: an interdisciplinary approach to understanding the web. Communication of the ACM 51(7):60–69 doi:10.1145/1364782.1364798

Henriques A (2006) Corporate truth the limits to transparency. Earthscan, London

Holzner B, Holzner L (2006) Transparency in global change: the vanguard of the open society. University of Pittsburgh Press, Pittsburgh

IOCCC (2009) International Obfuscated C Code Contest. http://ioccc.org/main.html

Jackson D, Thomas M, Millett L (2007) Software for dependable systems: sufficient evidence? The National Academies Press, http://www.nap.edu/catalog.php?record_id=11923#toc

Jarke M, Loucopoulos P, Lyytinen K, Mylopoulos J, Robinson W (2009) Manifesto – high-impact requirements for software-intensive systems In: Dagstuhl seminar proceedings – 08412, 1862–4405. http://drops.dagstuhl.de/opus/volltexte/2009/2028/pdf/08412.SWM.Paper.2028.pdf

Kramer J (2007) Is abstraction the key to computing? Communications of the ACM 50(4):36–42

Leite JCSP, Yu Y, Liu L, Yu E, Mylopoulos J (2005) Quality-based software reuse. In: CAiSE, pp 535–550

Leite JCSP, Rossi G, Balaguer F, Maiorana V, Kaplan G, Hadad G, Oliveros A (1997) Enhancing a requirements baseline with scenarios. Requirements Engineering 2(4):184–198

Leite JCSP, Cappelli C (2008) Exploring i* characteristics that support software transparency In: Proc 3rd international i* workshop, CEUR workshop proceedings, vol 322, pp 51–54. http://CEUR-WS.org/Vol-322/

Lord KM (2006) The perils and promise of global transparency. State University of New York Press, New York

Meunier P (2008) Software transparency and purity. Communications of the ACM 51(2):104. doi:10.1145/1314215.1314232

Mylopoulos J, Chung L, Nixon B (1992) Representing and using nonfunctional requirements: a process-oriented approach. IEEE Transactions on Software Engineering 18(6):483–497

Paul N, Tanenbaum AS (2009) Trustworthy voting: from machine to system. Computer 42(5):23–29. http://dx.doi.org/10.1109/MC.2009.169

Republic of Brazil (1997) Lei N° 9.507, de 12 de Novembro de 1997. http://www.planalto.gov.br/ccivil/Leis/L9507.htm

Robinson WN, Pawlowski SD, Volkov V (2003) Requirements interaction management. ACM Computing Surveys 35(2):132–190. doi:10.1145/857076.857079

Rolland C, Salinesi C (2009) Supporting requirements elicitation through goal/scenario coupling. In: Conceptual modeling: foundations and applications. Springer, Berlin, pp 398–416

Simon HA (1969) The sciences of the artificial. MIT Press, Cambridge

Stallman R (1999) The GNU Manifesto. Originally written in 1984; later version available. In: DeBona C, Ockman S, Stone M (eds) Open codes: voices from the open code revolution. O'Reilly, Cambridge, pp 53–70. http://www.gnu.org/gnu/manifesto.htmlv

Stallman R (2009) Why 'Open Source' misses the point of free software. Communications of the ACM 52(6):31–33. doi:10.1145/1516046.1516058

United States Department of Justice (nd) Freedom of Information Act (FOIA). http://www.usdoj.gov/oip/index.html

US Government Printing Office (2002) Public law 107–204 – Sarbanes-Oxley Act of 2002. http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/content-detail.html

Weber RH (2008) Transparency and the governance of the internet. Computer Law & Security Report 24(4):342–348

Weitzner DJ, Abelson H, Berners-Lee T, Feigenbaum J, Hendler J, Sussman GJ (2008) Information accountability. Communications of the ACM 51(6):82–87. doi:10.1145/1349026.1349043

Wikipedia (2009) Transparency (social). http://en.wikipedia.org/wiki/Transparency_(social)

Yu E (1994) Modeling strategic relationships for process reengineering. Dissertation, University of Toronto, Graduate Department of Computer Science, pp 124 ff