Process-Oriented Metrics for Software Architecture Changeability

Nary Subramanian
Dept. of Computer Engineering
Hofstra University
Hempstead, NY 11549

Lawrence Chung
Dept. of Computer Science
University of Texas at Dallas
Richardson, TX 75081

Abstract

Changeability is an important non-functional requirement (NFR) for software systems and evaluation of software changeability will be helpful for software practitioners. Briefly stated, changeability is the ease with which software system can be changed or modified. It is widely accepted that NFRs such as changeability should be incorporated in the architectural stage of software development itself to maximize the changeability of the system. The paper proposes the framework called the Process-Oriented Metrics for Software Architecture Changeability (POMSAC) that helps generate changeability metrics for software architectures during the process of architecture development. POMSAC helps generate intuitive metrics for changeability and traces the metrics to the requirements for which the architecture exists in the first place. We present an example metrification scheme – a single-value (SV) metrification scheme- that meets the guidelines of POMSAC, and apply the SV scheme to a practical banking system. This application illustrates the advantages of the process-oriented approach.

Keywords

Non-Functional Requirements, Changeability, Software Architecture, Metrics

1. Introduction

Changeability is an important non-functional requirement (NFR) for software systems [1, 2, and 3] and evaluation of software/system changeability will be helpful for practitioners in both the industry and academia. Briefly stated, changeability is the ease with which a software system can be changed or modified. It is widely believed that NFRs such as changeability should be incorporated into a software system at the software architecture development stage itself. Therefore, assessment of architectural changeability will help software developers to confirm that the software being developed meets changeability requirements and make corrections, if needed. Several changeability measures have been proposed. A category of changes called "atomic changes" that addresses simple changes to an object-oriented system has been dealt with in [1] and a set of measures proposed to deal with these changes – these measures are NOC* (number of children in sub-tree), CBO_NA (coupling between objects but not with ancestors), CBO_IUB (coupling between objects that consists of classes using target class), and CBO_U (coupling between objects that consists of classes used by the target class). Changeability number (CN) has been proposed in [2] to measure changeability of software design, and a formula is given for measuring CN as CN = (Domain Classes Number – Simplify Classes Number)/2. In [3] a scale in terms of time spent in maintaining (assuming maintenance is related to software changes) the software is proposed, while in [4] a measure for extendability (assuming again, that changeability is related to extendability) as number of additions to the existing system is proposed. However, the following drawbacks may be pointed out in most of the metrics schemes for software changeability:

1. the definition of changeability is not universally acceptable: most schemes assume a definition of changeability which may not satisfy all constituents – different organizations or even different projects within the same organization may view changeability differently
2. the metrics are not intuitive: a formula is proposed by there is no justification as to why the formula calculates software changeability metrics (reasoning for the formula is usually there but the reasoning often does not trace the formula to the changeability requirements)
3. the metrics are not process-oriented: most changeability metrics calculate the changeability of the end product; they do not help in evaluating changeability during the development of the product – calculation of metrics during the process of software development will help the software development organization to keep track of how changeability requirements are affected by architectural changes.

In this paper we propose a new way to evaluate software changeability metrics – the process-oriented metrics that overcomes the drawbacks mentioned earlier. In order to calculate the software changeability metrics during the process of software architecture development, we want to be able to represent and reason about changeability requirements "during" the development of a changeable system, and the NFR Framework [5, 6] is a goal-oriented framework which offers the needed

concepts and techniques. This Framework is used to develop the POMSAC (Process-Oriented Metrics for Software Architecture Changeability) Framework that helps to generate process-oriented changeability metrics for software architectures. The process instituted by the POMSAC Framework is given below:

1. Develop a metrification scheme satisfying the guidelines of the POMSAC Framework – these guidelines are given in the next section.
2. Develop the Softgoal Interdependency Graph (SIG) for the NFR changeability for the domain of interest – this decomposition defines changeability for that domain and is discussed in a later section.
3. Evaluate the extent to which architectures satisfice (this is a concept of the NFR Framework which means satisfaction within limits and not absolutely).
4. Apply the metrification scheme chosen in step 1 and determine the changeability metrics.

The application of the POMSAC process will be described in subsequent sections. In order to illustrate the use of POMSAC, we have applied it to a bank loan system that meets the requirements of the Barclay Bank Code of Business [7, 8]. Further discussion on the application of the NFR Framework to the design of the bank loan system can be seen in [9]. In this paper we consider the two architectures of the bank loan system to measure their changeability using POMSAC. POMSAC is similar to POMSAA (process-oriented metrics for software architecture adaptability) [10] and POMSAE (process-oriented metrics for software architecture evolvability) [11] (the reason for separate frameworks for dealing with adaptability, evolvability and changeability is clarified in Section 2). This application of POMSAC will also highlight its advantages compared to other metrics for changeability.

2. The POMSAC Framework

The POMSAC Framework consists of six major components: a set of *softgoals* for representing NFRs, design components and claims, a set of *contribution types* for relating softgoals to other softgoals, a set of *methods* for refining softgoals into other softgoals, a set of *correlation rules* for inferring potential interactions among softgoals, a *labeling* procedure which determines the degree to which a design component satisfices a softgoal, and a set of *metrification schemes* to map labels to numbers. The partial ontology of the POMSAC Framework is given in Figure 1.

1. Softgoals can be of several types – the NFR softgoals (depicted by a cloud), the design softgoal (depicted by a dark cloud), and the claim softgoal (depicted by a dotted cloud). The design softgoal represents a design component, while a claim softgoal represents a claim (for any item of the Framework).
2. Contribution types connect various softgoals – the links may connect several softgoals to one softgoal in an AND-decomposition (depicted by single arc) or in an OR-decomposition (depicted by double arc).
3. Methods are ways to refine or decompose one softgoal into offspring softgoals for purposes of clarity and achievement of better designs. The softgoals are changeability related.
4. Correlation rules help determine the interactions between different changeability-related NFRs for a design component.
5. Labels indicate the degree to which their associated softgoal (or links) are satisficed – the various satisificing degrees are given in Figure 1.
6. Metrification schemes map qualitative labels into quantitative scores for a given architectural design. Labels of NFR softgoals, design softgoals, claim softgoals and links, in some combination (either only one of these, any two of these, any three of these or all of these), may be converted to numbers. There are several different metrification schemes, including: 6a) *Max and Min Values*: In this scheme the max and min values are computed for the labels; 6b) *Single Values*: Here one value is computed for the labels; 6c) *Probabilistic:* Here probabilities are computed for the labels. The metrification scheme guidelines are given in the next section.

Elements 3 and 4 above necessitate a separate framework for dealing with changeability: POMSAC has the knowledge of methods and correlation rules relating to the NFR changeability, while POMSAA [10] and POMSAE [11] have a knowledge base of methods and correlation rules that relate to adaptability and evolvability, respectively.

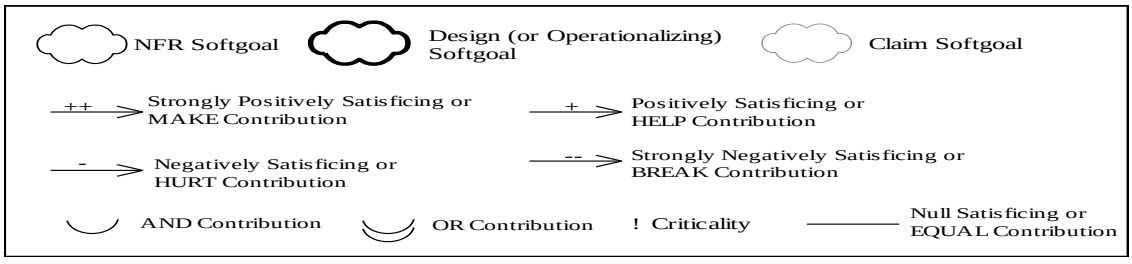

Figure 1. The Ontology (partial) of the POMSAC Framework

2.1 Guidelines for Metrification Schemes

Any metrification scheme M converts labels of Step 5 above into metrics. This conversion is accomplished using the guidelines of Figure 2.

M1: label(leaf softgoal)	→	metric(leaf softgoal)
M2: label(contribution)	→	metric(contribution)
M3A: criticality(softgoal)	→	metric(criticality(softgoal))
M3B: criticality(contribution)	→	metric(criticality(contribution))
M4: {metric(leaf softgoal), metric(criticality(leaf softgoal)), metric (contribution), metric(criticality (contribution))}	→	metric(individual contribution of leaf softgoal)
M5: {metric_i(individual contribution of child softgoal_i)}	→	metric(parent softgoal)
M6: {metric_i(individual contribution of child softgoal_i)}	→	metric(parent contribution)

Figure 2. Guidelines for Metrification Schemes

In Figure 1, guidelines M1, M2, M3 state the rules for metrification of an element of the framework: thus M1 says that the label of leaf softgoals gets converted into a metric, M2 says that the label of a contribution converts into the metric for the contribution; and M3 says that the criticality gets converted into the metric for criticality – however, since criticalities can be assigned to two elements of the framework, the softgoal and the contribution, M3 is broken into two parts: M3A applies to the criticality of the softgoal while M3B applies to the criticality of the contribution. M4 states that for any leaf softgoal, the metric of its label, the metric of its criticality, the metric of its contribution to its parent, and the metric of its contribution's criticality together form the metric for the individual contribution of that leaf softgoal. M5 says that the metric of all individual child softgoal contributions result in the metric for the parent softgoal. M6 applies to contributions that have children (for example, in the form of claim softgoals) and states that the metric of the parent contribution is computed from the metric of the contributions of all child softgoals of that contribution. In applying M2 the following ordering among the contributions should be maintained:

MAKE > HELP > HURT > BREAK

where ">" means "stronger positive satisficing".

2.2 Decomposition of the NFR Changeability – the SIG

The second step in the POMSAC process (given in the Introduction) is the development of the Softgoal Interdependency Graph (or SIG). The SIG first decomposes the NFR changeability for the domain of interest – here bank loan system – this decomposition defines changeability for that domain. Then the extent to which the architectures satisfice the various NFRs are evaluated – this process captures traceability between architectures and their changeability requirements. Figure 3 shows the decomposition of changeability for the bank loan system. Each cloud in Figure 3 is a softgoal in the NFR Framework and each softgoal is named using the convention *Type[Topic1, Topic2,…]*, where *Type* is a non-functional aspect (e.g., changeability) and *Topic* is the system to which *Type* applies (e.g., bank loan system), and the decomposition can take place along Type or Topic. In Figure 3 the bank loan policies [7, 8] have been used to guide the decomposition; thus the NFR softgoal of interest, viz., Changeability[Architecture, Bank Loan System] is AND-decomposed (indicated by the single arc) into three child softgoals: Changeability[Architecture, NFR] (meaning changeability of non-functional requirements of architecture), Changeability[Architecture, FR] (meaning changeability of functional requirements of architecture), and Changeability[Architecture, System Workload] (meaning changeability of architecture to accommodate varying system workloads). AND-decomposition means that all children must be satisficed in order for the parent to the satisficed. The NFR softgoal Changeability[Architecture, NFR] is AND-decomposed into three softgoals – Accuracy[Update, Statements] (meaning the statements that are updated should be accurate), Informativness[Update, Statements] (meaning the statements that are updated should be informative), and Time[Update, Base Rate] (meaning the base rate for the loans should be updated in a timely fashion upon any changes). The NFR softgoal Time[Update, Base Rate] is considered critical and is indicated as such by the '!' mark next to it. The NFR softgoal Changeability[FR] is AND-decomposed into Changeability[Base Rate] (meaning accommodation of changing base rates) and Changeability[Statement] (meaning accommodation of changing statements including format and content changes), and both of these softgoals are marked critical (by the '!' symbol). The NFR softgoal Changeability[Architecture, System Workload] is OR-decomposed (indicated by the double arc) into softgoals Simultaneity[Processing Customer Statements] (meaning customer statements are processed simultaneously) and Sequentiality[Processing Customer Statements] (meaning customer statements are processed sequentially); OR-decomposition means satisficing of either child satisfices the parent.

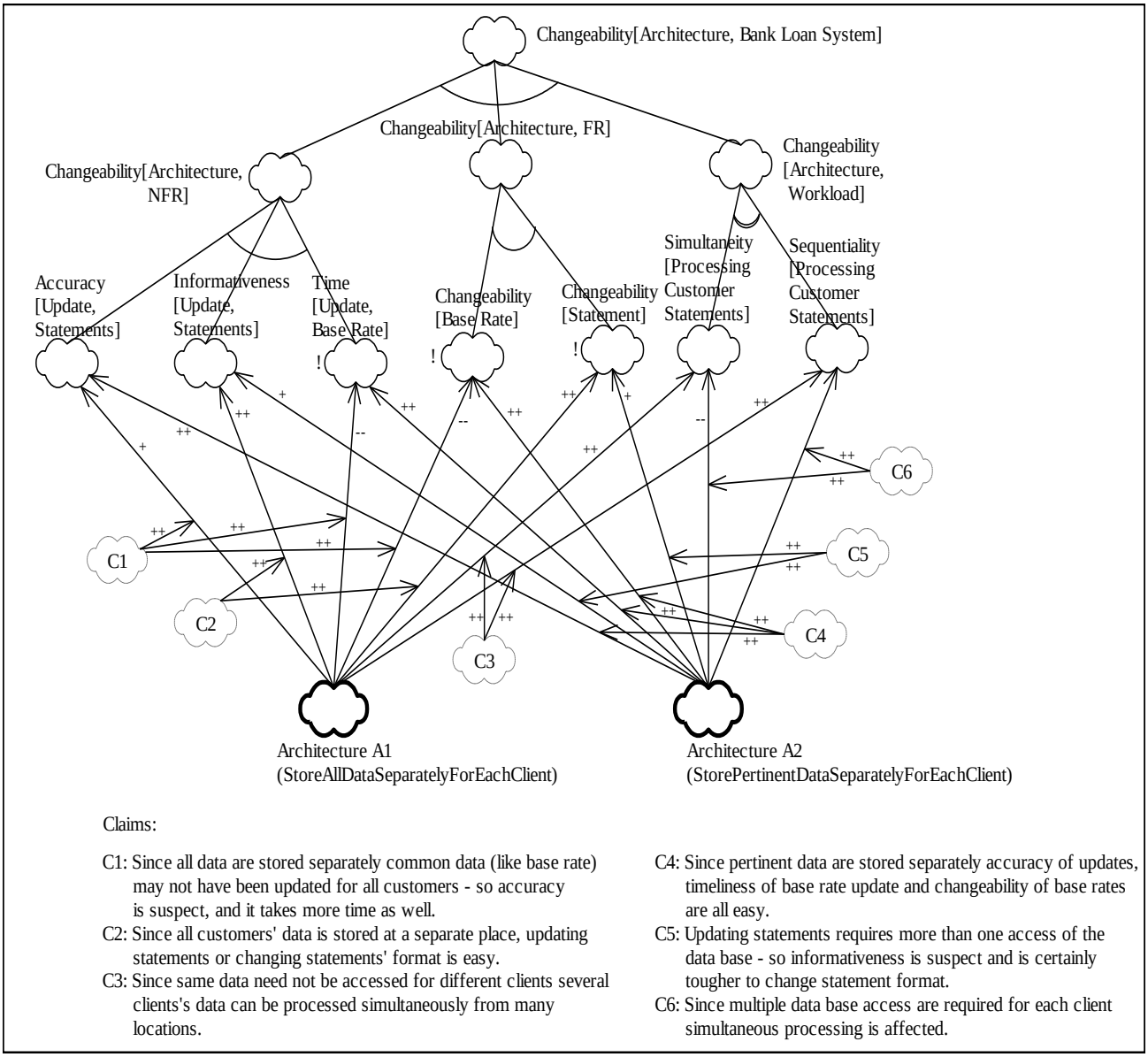

Figure 3. The Softgoal Interdependency Graph for POMSAC

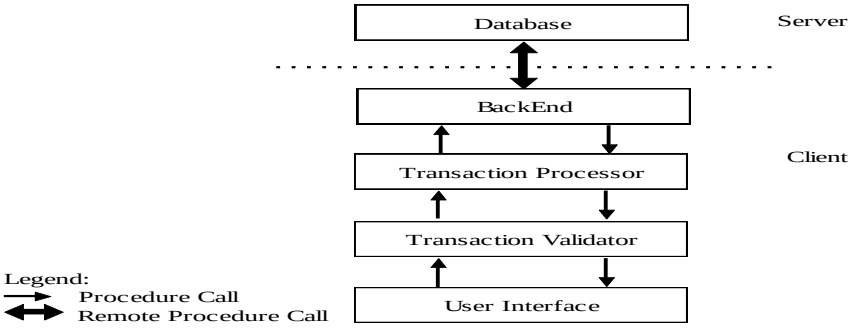

Figure 4. Architecture for the Bank Loan System

3. Application of the POMSAC Framework

In this section we will apply the POMSAC Framework to the bank loan system. However, we first need the architectures for the bank loan system and the architectures are discussed next.

3.1 Architectures for the Bank-Loan System

Software architectures have several constituents: components, connections, patterns, constraints, styles, and rationale [12, 13]. We consider two architectures A1 and A2 for the bank loan system – both have client-server style, the constraints on both are that the response time should be fast, rationale for choosing them is changeability, the connections are procedure calls and remote procedure calls, while the pattern (the repeating motif) is database access for almost all transactions. The main difference between A1 and A2 is that A1 stores data for each customer separately on the server (we can call A1 as StoreAllDataSeparatelyForEachClient), while A2 stores data common to all customers (for example, base rate of loans) at one place on the server and only data unique to a customer separately on the server (we can call A2 as StorePertinentDataSeparatelyForEachClient). The component (depicted by boxes) and connection (depicted by arrows) diagram for the two architectures are given in Figure 4. The database resides on the server; on the client side there is the user interface which is accessed by the bank loan officers, the transaction authenticator that authenticates transactions, transaction processor that processes authenticated transactions and the back end that includes the communication with the server. As can be seen both the architectures are the same at a high level of abstraction – only their functionality is different which may be localized in any one or more of the components on the client or the server. Thus by simply looking at the architecture it is almost impossible to decide which is more changeable.

3.2 Sample Single-Value Metrification Scheme

In order to apply the POMSAC Framework we need a metrification scheme satisfying the guidelines of Figure 2. While several different schemes are possible, we provide a sample single-value metrification scheme in Figure 5. This scheme assumes that the metrics are between +1 and -1. Thus the softgoal metrics rules M1.1 and M1.2 allocate a metric of +1 for satisficed softgoals and a metric of -1 for denied softgoals. The rule M2 allocate metrics for contribution between +1 and -1 depending on the contribution type (here it is assumed that the contributions are themselves satisficed – if not the contributions are given a metric of 0). The M3 rules give the metrics for the criticalities of a softgoal and a contribution and they can range between 0 and 1. M4 gives the formula for computing the metric propagated by a child softgoal to its parent. M5 gives the rules for combining propagated values from multiple children for AND, OR, and for leaf NFR softgoals. M6 gives the method to compute metrics for a contribution that has children.

3.3 Calculation of Metrics Using the Sample SV Scheme

The complete SIG for architectures A1 and A2 is shown in Figure 3 – the justifications for the claims are depicted by the claim softgoals (labeled C1 through C6) and the claims are also given in that figure. Figure 6 shows the step-wise application of the SV scheme to architecture A1. First metrics M1 and M3 are assigned to the design softgoal (here the design softgoal A1 represents the entire architecture A1 – this need not be the case; individual constituents of architecture

1. M1 (softgoal metrics)
   M1.1: A satisficed softgoal gets a metric of 1
   M1.2: A denied softgoal gets a metric -1
   M1.3: A softgoal with any uncertainty gets a metric between 1 and -1
2. M2: (contribution metrics)
   A contribution's metric (CM) is computed as follows:
   MAKE = +1, HELP = +0.5, HURT = -0.5, BREAK = -1, EQUAL = 1
3. M3: (criticality metrics)
   M3A: A softgoal's criticality is assigned a metric between 0 and 1
   M3B: A contribution's criticality is assigned a metric between 0 and 1
4. M4: The metric propagated (C) by a softgoal to its parent is given by:
   (metric of softgoal + criticality of softgoal) * (metric of contribution + criticality of contribution)
5. M5: For each parent softgoal in the SIG:
   M5.1: if the children are all connected by AND, the metric of the parent is
   the minimum of the metrics propagated by all its children.
   M5.2: if the children are all connected by OR, the metric of the parent is
   the maximum of the metrics propagated by all its children.
   M5.3: the metric of a parent NFR softgoal which is also a leaf (i.e., has only operationalizing
   softgoals connected to it) is given below where Ci is the metric propagated by each child:
   parent softgoal metric = $\Sigma\{Ci\}$ if $1 \geq \Sigma\{Ci\} \geq -1$
   parent softgoal metric = 1 if $\Sigma\{Ci\} > 1$
   parent softgoal metric = -1 if $\Sigma\{Ci\} < -1$
6. M6: Metric of a parent contribution is CM+the minimum of its children's propagated values.

Figure 5. Sample Single-Value Metrification Scheme

A1 may be treated as separate design softgoals, but we have given priority to simplicity) and to the claim softgoal (Figure 6a); then M6 and M4 are applied to the contribution of the claim softgoal (Figure 6b). Then M4 is used to compute the metric propagated by the design softgoal to each of the parent NFR softgoals considering criticality (Figure 6c). Then M5 is used to progressively propagate the metrics up the SIG till the metrics for the main NFR softgoal, viz., Changeability[Architecture, Bank Loan System] are calculated (Figure 6d). Thus based on POMSAC, architecture A1 gets a changeability metric of **0.5**, while architecture A2 gets (by a similar process) a changeability metric of **1.0**.

3.4 Discussion of POMSAC Metrics

The process-oriented metrics easily overcome the three main drawbacks of the currently used metrics mentioned in the Introduction. Firstly, almost any definition of changeability is accommodated in POMSAC since the definition is captured in the NFR softgoal decomposition for the domain. Different decompositions capture different definitions. Secondly, the metrics are justified by the requirements – thus when architecture A1 obtained a metric of 0.5, we know exactly why – it is because of the changeability requirements, the contributions that A1 makes to the various NFR softgoals, and the metrification scheme used. Any changes in one of these factors could affect the metric calculated but again we know why. Finally, the process-oriented metrics are process-oriented – they help (re-)calculate metrics during the process of architecture development; thus metrics for partial architectures may be evaluated and can be used to guide the development process – any modifications to the architectures that affect changeability metrics may be scrutinized for omissions/commissions. This also helps analyze reasons for strengths/weaknesses of architectures. Thus POMSAC provides a useful alternative approach to calculating software architecture changeability metrics.

4. Conclusion

In this paper we have presented the process-oriented metrics for software architecture changeability (POMSAC). POMSAC differs from other techniques for measuring architectural changeability in the following ways:

1. it is process-oriented: architectural changeability metrics can be computed *during* the process of software development incrementally as additions and changes are being made
2. provides a means for representing software artifacts
3. provides a method to capture design rationale – this is very important since these rationales help in making change decisions (and any change decision can be recorded as well)
4. traces the metrics to the changeability requirements
5. maintains an historical record – this will help in any change decisions.

The POMSAC Framework is based on the NFR Framework [5, 6] and provides a set of guidelines that any metrification scheme for process-orientation should satisfy. In order to illustrate POMSAC we present a sample single-value metrification scheme that satisfies the guidelines. The single-value metrification scheme is applied to two architectures for a bank loan system and the changeability metrics for the two architectures are computed. The process illustrates the advantages of the POMSAC Framework over other changeability metrics in the literature.

There is still more work to be done – POMSAC has to be tested using a deeper NFR-decomposition scheme, has to be tested on complex architectures, needs to be tested on different architectural styles, and for different architectural views such as logical, physical, deployment, etc[1]. Other possible avenues for further research include automated tool support for applying POMSAC, studying scalability of POMSAC possibly with the aid of automated tool, and applying of POMSAC to other software systems for further feedback on the strengths and weaknesses of the framework. However, it is our opinion that our preliminary studies show that the POMSAC Framework will be useful to software organizations in practice.

5. References

[1] M. A. Chaumun, H. Kabaili, R. K. Keller, F. Lustman and G. Saint-Denis, "Design Properties and Object-Oriented Software Changeability", *Proceedings of the Conference on Software Maintenance and Reengineering*, Feb-March, 2000, Zurich, IEEE Computer Society, pp. 45-54.

[2] J. Garzas, M. Piattini, "Analyzability and Changeability in Design Patterns", *The Second Latin American Conference on Pattern Languages and Programming*, Special Session on Software Pattern Applications, August 2002, Brazil.

[3] N. E. Fenton, "*Software Metrics – A Rigorous Approach*", Chapman & Hall, London, 1991.

[4] T. Gilb, "*Principles of Software Engineering Management*", Addison Wesley, England, 1988.

[5] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, Boston, 2000.

[6] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", *IEEE Transactions on Software Engineering*, Vol. 18, No. 6, June 1992, pp. 483-497.

[7] Barclays Bank PLC, *The Barclays Code of Business Banking*, London, England, effective 31st Jan. 1992.

[8] Barclays Bank PLC, The Barclays Code of Business Banking, London, England, May 1993.

[1] We thank the reviewers of the paper for pointing out possible improvements for POMSAC's application.

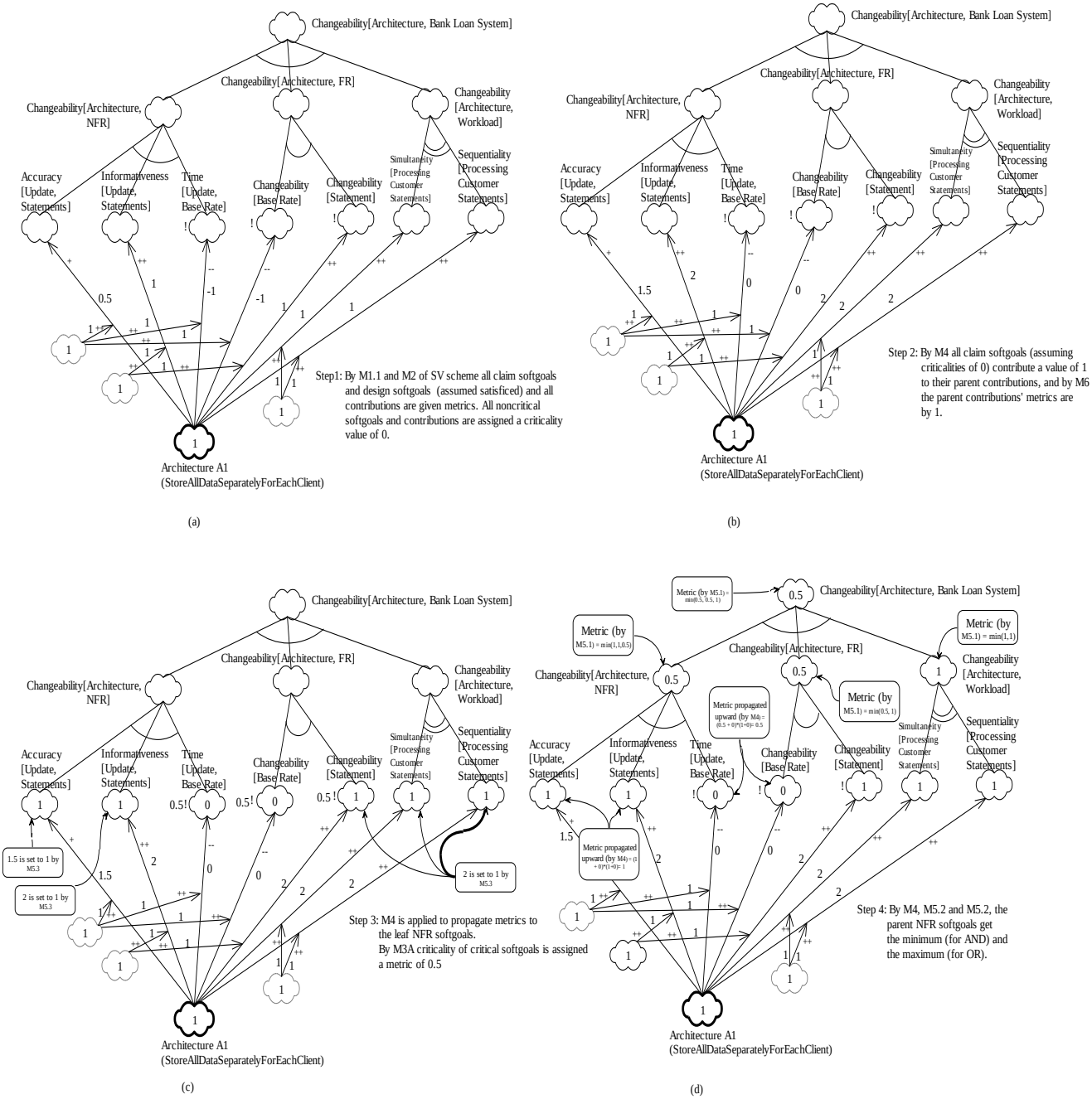

Figure 6. Step-by-step Propagation of POMSAC Metrics up the SIG

[9] L. Chung, B. A. Nixon, and E. Yu, "Dealing with Change: An Approach Using Non-Functional Requirements", *Requirements Engineering Journal*, Volume 1, Number 4, January 1997.

[10] L. Chung and N. Subramanian, "Process-Oriented Metrics for Software Architecture Adaptability", *Proceedings of the International Symposium on Requirements Engineering*, Toronto, Canada, Aug-Sep., 2001, pp. 310-311.

[11] N. Subramanian and L. Chung, "Process-Oriented Metrics for Software Architecture Evolvability", *Proceedings of the International Workshop on Principles of Software Evolution*, IEEE Computer Society, Helsinki, Finland, September, 2003, pp. 65 -70.

[12] M. Shaw, and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[13] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, SEI Series in Software Engineering, Addison-Wesley, 1998.