Completeness of Conceptual Models Developed using the Integrated System Conceptual Model (ISCM) Development Process

Regina M. Gonzales
Technology, Management and Analysis Corporation
201 Church Street, Suite 200
Las Cruces, New Mexico 88001 USA
regonzal@nmsu.edu

Abstract. There exists a semantic gap between developers and users of systems. Requirements engineering is addressing this gap by introducing requirements methods, techniques, and processes that facilitate greater understanding of the customers and users needs. Many of these methods and techniques rely on the analyst or developer to model the product domain. With the complexity of systems developed today, this can be a formidable task. These systems have multiple users with diverse needs and require the integration of multiple domains in order to develop them.

Customers, users and developers are the stakeholders for a system. Increasing the communication and decreasing the semantic gap between diverse stakeholders are the aims of the Integrated System Conceptual Model (ISCM) Development Process. This process uses a combination of elicitation methods and techniques. However, the stakeholders create the initial models for the system based on their own mental models of a proposed system. This paper focuses on completeness indicators proposed in order to regulate the ISCM process.

CAPTURING REQUIREMENTS

Defining a requirements elicitation process is key to developing a complete and consistent requirements specification for systems. Most concurrent engineering efforts begin early while the product concept is still fluid and a requirements elicitation process is not defined. This scenario is detrimental to a good project process. If an engineer proceeds to design with a fluid product concept, the uncertainty practically ensures that the product developed does not meet stakeholder expectations. The goal of a requirements elicitation and analysis process should be to develop a shared vision or concept of the system to be specified before a concurrent engineering design effort begins in earnest.

This shared vision is often called a conceptual model and is an extremely useful communication tool at the onset of a project. The system conceptual model becomes the initial mapping of the intangibly abstract into something more concrete. The author refers to the system conceptual model as the Integrated System Conceptual Model (ISCM) because she wants to emphasize that it is an integration of multiple stakeholder conceptual models.

Mental models are models belonging to individuals and are used to form conceptual models, which can evolve into a group representation of a system. These conceptual models are used as a way to elicit complete requirements for a developing system. Concept of Operation (ConOp) documents use system conceptual models formally as a way to evaluate alternative development approaches and as a way to articulate the goals and objectives of a development activity.

In order for the ISCM to be useful, it must integrate viewpoints of multiple system stakeholders. As Wieringa (1997) states, "The hallmark of conceptual models is that they are conceptual structures used as a framework for communication between people." Once the ISCM solidifies, a further mapping to the requirements specification for a system is possible. Figure 1 illustrates the mapping of the stakeholder conceptual models into a single ISCM from which requirements analysis and specification can begin. Once the ISCM is created it must be iterated with the stakeholders until it represents a consensus system conceptual model and all the issues and questions are addressed. This mapping represents a stepwise approach to capturing requirements.

INFORMATION TRANSFORMATIONS IN THE ISCM DEVELOPMENT PROCESS

Developing an ISCM requires three transformations as illustrated in Figure 2. The first is from the stakeholders mental models into a stakeholder conceptual model on a piece of paper. There is significant research on mental models as they apply to existing systems including natural systems, e.g. physics. To the author's knowledge, there is no research on using mental models to generate new systems based on experience with multiple existing systems.

The second transformation is from each informal stakeholder conceptual model to a Unified Modeling

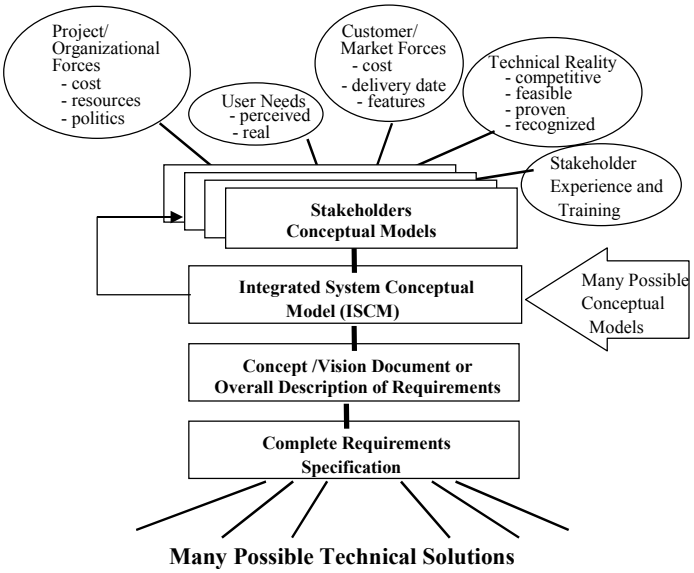

Figure 1 Role of Conceptual Models in Requirements

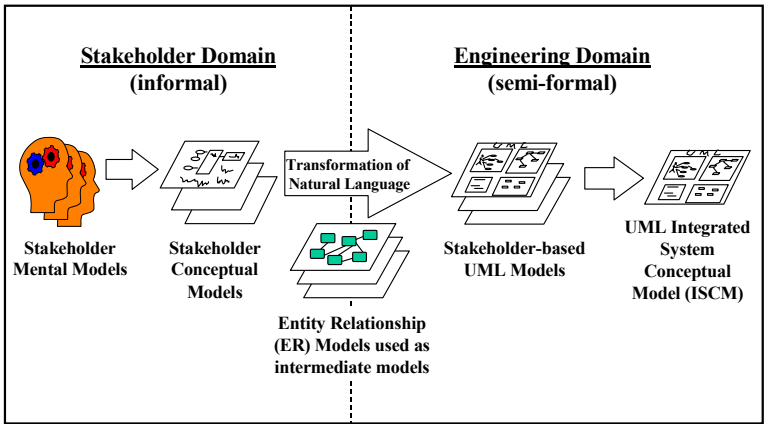

Figure 2 Transformations in the ISCM Development Process

Language (UML) stakeholder model. UML is considered a semi-formal representation of information (Darke and Shanks, 1996). This transformation requires knowledge of UML. UML is a language with syntax and semantics expressed using formal techniques (Object Management Group, 1999). The third transformation is from multiple UML stakeholder models to an ISCM. Completeness indicators are proposed in this paper for a UML model. These indicators can be applied to individual stakeholder UML models and to the ISCM.

The ISCM development process emphasizes that the engineer-analyst is playing more of a facilitator role in developing the ISCM by integrating stakeholder viewpoints. They provide structure and process but the content comes directly from the stakeholders. The ISCM development process does not rely on hours spent attending group meetings but rather on the models developed by the stakeholders and integrated by the analyst. In fact, an older approach called the Delphi Process (Delbecq, A. L., 1975) is often advocated especially for high-technology systems. The Delphi Process is one in which the model can be reviewed individually and contributions given to the facilitator. When conflict or questions occur, the facilitator can decide if it is best addressed individually or as a group.

Darke and Shanks (1996) present a framework for understanding and comparing viewpoint approaches. They outline six components regarding viewpoint development, viewpoint agent, viewpoint representation, process model, management of conflicts, viewpoint integration, and the role of viewpoint development. Viewpoint agents can be any stakeholder or non-human agent, such as an existing database, that maintains and accepts responsibility for a viewpoint. A single stakeholder could play multiple organizational roles and thus have multiple viewpoints.

Viewpoints can be represented using informal, formal or semi-formal representation techniques. The suitability of a particular technique depends in part on the role of the viewpoint development, i.e. requirements acquisition versus requirements modeling. Different techniques also reflect different perspectives, i.e. data, process or behavior. Informal representations allow requirements freedom and allow for incompleteness, inconsistency, complexities and ambiguities, but do not provide support for reasoning about requirements specifications. Informal approaches do not facilitate analysis and comparison of viewpoints. Darke and Shanks (1996) indicate that there may be a spectrum from informal to more formal techniques applied as a greater understanding of the system and its organizational context develop. They assert that semi-formal and formal representation schemes tend to focus on the syntax of representations at the expense of their meaning and socio-organizational interpretation. This balance is addressed in the ISCM development process by allowing the stakeholders to introduce the model in an informal way. The facilitator then brings the stakeholders to a common representation by championing the semi-formal representation. There is no expectation that the stakeholders ever master the semi-formal representation, but simply that they grow to understand the syntax.

SYSTEM REQUIREMENTS CHARACTERISTICS

Each industry specifies characteristics that a good requirements specification should have. These characteristics are important to understand because a

requirements specification is the goal of the requirements elicitation process. In the *IEEE and ANSI 830 Standard for Software Requirements Specification (SRS)* (IEEE, 1994), they list the following as characteristics of a good SRS: Correct, Unambiguous, Complete, Consistent, Ranked for importance and/or stability, Verifiable, Modifiable, and Traceable.

Similar criteria are called properties in the *IEEE 1233 Guide for Developing System Requirements Specification* (IEEE, 1996) they include the following.

- Unique Set: Each requirement should be stated only once
- Normalized: Requirements should not overlap
- Linked Set: Explicit relationships should be defined among individual requirements
- Complete: Should include all the requirements identified by the customer, as well as those needed for the definition of the system
- Consistent: Should be consistent and non contradictory in the level of detail, style of requirement statements, and in the presentation material
- Bounded: The boundaries, scope, and context for the set of requirements should be identified
- Modifiable: Should be modifiable
- Configurable: Versions should be maintained
- Granular: This should be the level of abstraction for the system being defined

Kar and Bailey (1996) discuss the characteristics of individual requirements versus characteristics of the aggregate requirements set. The characteristics they assign to the individual requirements include: Necessary, Concise, Implementation free, Attainable, Complete, Consistent, Unambiguous, Standard Constructs and Verifiable. The characteristics they assign to the aggregate requirements include, Complete and Consistent.

There are many other books (Davis, 1993; Pfleeger, 1998) on requirements that enumerate similar characteristics. It is widely accepted within the requirements engineering community that if all of the key stakeholders are not considered in the elicitation effort, the requirements are likely incomplete. If the requirements specification is developed from an ISCM based on stakeholder input, it aids in achieving the other characteristics, such as Understandable, Consistent and Modifiable. Developing the requirements specification based on a model, in essence a conceptual model, is the recommended approach of the Software Engineering community as discussed in the IEEE and ANSI 830 Standard for Software Requirements Specification (SRS) (IEEE, 1994). How to go about developing the model is not discussed in that standard. The primary characteristics of complete and consistent are the measure for the quality of the ISCM. The other characteristics, primarily understandability, are byproducts of the ISCM Development Process.

UNDERSTANDING COMPLETENESS

Each standard for requirements specifications (IEEE, 1993; IEEE, 1996; Kar and Bailey, 1997) maintain that a set of requirements should be complete. Some, including the IEEE Std 1233, recommend ways to achieve completeness that amount to requirements elicitation techniques. However, none give a way to evaluate completeness or even assess relative completeness. There is research under way to formalize many terms used in software and systems engineering. Briand et al. (1996) formally define cohesion and coupling using graph theory. However, there is little information on quantifying completeness of models, especially at the early conceptual phase.

Completeness refers to wholeness or entirety of an object. In this paper, completeness refers to the entirety of a conceptual model. A conceptual model represents concepts in the real-world problem domain (Larman, 1998). When referring to software requirements, specification completeness is also described as external consistency. External consistency of a software specification means that it is consistent and meets the requirements written in the system specification. However, for completeness of a system specification or model, external consistency is based on the needs of the people who will be using the system and the constraint envelope introduced by the environments in which the system will operate or be exposed to.

Stakeholders are inherently inconsistent. There are also problems with identifying all the right stakeholders and of getting adequate information from each of them. Therefore, in the strictest sense, completeness of a conceptual model is practically unattainable. However, proposed in this paper are operational indicators of completeness of an ISCM in terms of the incremental completeness of each stakeholder conceptual model. Equation 1 and 2 represent a way to think about completeness in terms of error associated with getting complete information from the stakeholders.

$$X = \left\lfloor \sum_{1}^{n} (\chi \pm \varepsilon) \right\rfloor + E \qquad \text{Eq} - 1$$

$$E = K + M \qquad \text{Eq} - 2$$

X •represents the desired ISCM and χ represents the contributions of the individual stakeholder conceptual models. The error (ε) is the incompleteness of the individual models and is a function of the modeling accuracy and the ability of the stakeholder to represent the knowledge they possess. The error or incompleteness of the ISCM (E) is a function of the compromise factor (K) and the information missing (M) because an entire set of stakeholders is not included in the summation. This equation does not take into account that the problem or requirements may change over time.

The completeness indicators proposed in this paper become a way to gauge if the error ε for an individual model has leveled off. At this point, the completeness indicators are not changing. These indicators can also be used in the same way for the ISCM error E to know when it is time to move on in the requirements elicitation process with an ISCM that is as representative as possible given the set of stakeholders.

THE ISCM DEVELOPMENT PROCESS SYSTEMS MODEL

Figure 3 is a control systems diagram illustrating the ISCM development process. There is a problem, p(n), that is introduced to various stakeholders, S_1 and S_2, and they produce a model of the problem, $m_1(n)$ and $m_2(n)$, with help from the analyst (A) and an analysis method embodied in a tool (T). The model is iterated via a feedback loop with the stakeholder until the completeness indicators introduced in this research remain relatively unchanged. There is an optional multiplier, β_1 and β_2, for weighting the contribution of each stakeholder. Discrete time is used because it indicates that the stakeholders are in essence sampled at intervals instead of continuously, i.e. a brainstorming session with all stakeholders present.

Preconditions and Distortion. The goal of the ISCM Development Process described in this paper is to faithfully model (*m*) the bounds that the stakeholder (S) places on the problem (*p*). No judgment is made about the correctness of these bounds at an early conceptual stage. Completeness at this conceptual level represents a measure of the structure, clarity and consistency of information as related by the stakeholder. There are certain preconditions that are assumed true at the onset of the process as shown in Table 1.

The stakeholders have a relevant contribution to solving the problem. However not necessarily the entire solution.
The stakeholders have a unique perspective and bind the problem from his or her perspective.
The analyst has reasonable training in requirements and in using modeling methods and tools.
The tool and the embodied method are adequate to represent the problem domain.

Table 1 Preconditions to System Modeling

Each of these preconditions represents a body of research that is not addressed in this paper. Given these preconditions, Table 2 illustrates further analysis of the elements for the error (ε) for individual stakeholder models. Both address the noise and distortion components introduced by the ISCM development process.

x(n) is the transmission error and represents the stakeholders' inability to express his or her conceptual model (problem bounds).
r(n) is the reception error and represents the analysts' inability to receive and model what the stakeholder is expressing.
f(n) is the filtering error because the tool/method is based on a paradigm. Tools/methods are like off-the-shelf filters with some ability to customize. As with all filters, distortion is introduced.
u(n) is the understanding error and represents the stakeholders' ability to relate to the model as portrayed using the modeling tool/method when the model is fed back to the stakeholder.

Table 2 Noise and Distortion Components

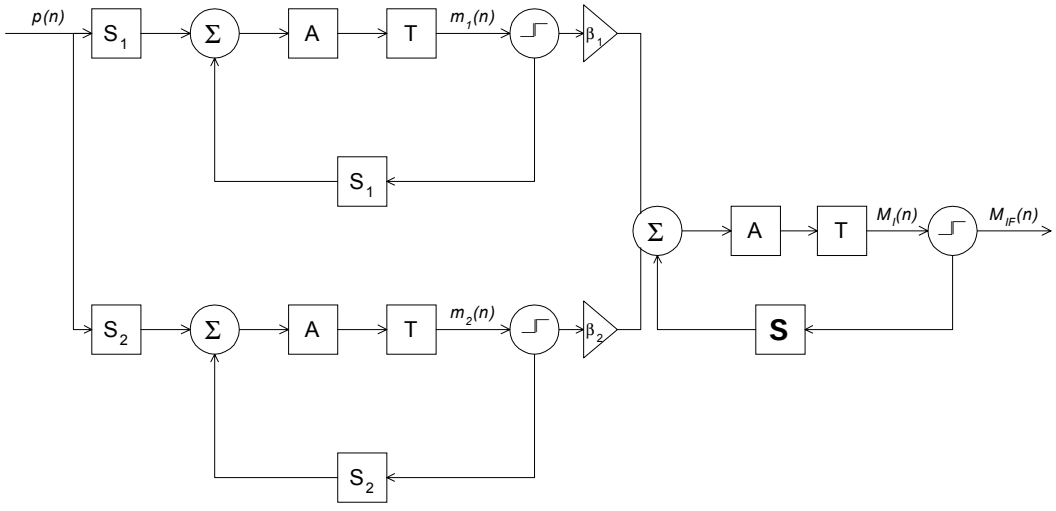

Figure 3 Systems Model of ISCM Development Process

Externalizing a model that embodies the knowledge that a stakeholder possesses requires iteration. In the ISCM Development Process, the assertion that when prompted, stakeholders are able to formulate conceptual models of yet to be systems, even if they are incomplete, is the basis for the iteration process. As illustrated in Figure 3, once the initial models

provided by the stakeholders are iterated to a point where the completeness of the model is at an acceptable threshold level, the individual models are integrated and the ISCM is iterated with the entire set of stakeholders. Since it is difficult to precisely determine completeness, this paper proposes completeness indicators that can be used to determine when the iteration process has reached diminishing returns.

DETERMINING COMPLETENESS

The models are derived in the context of the UML translation mentioned previously, and the completeness indicators are evaluated in this context as well. Figure 4 is an activity diagram that details the translation method. Classes, packages and attributes are rigorously identified based on input from the stakeholder. The analyst introduces structure, and to some degree, distortion as part of the method of deriving the UML class diagrams. Abstract classes are created to group logically connected entities. The operations are not defined until the behavior of the system is realized with sequence diagrams. At this time, the operations are identified based on the stakeholder scenario information in their conceptual model and added to the classes.

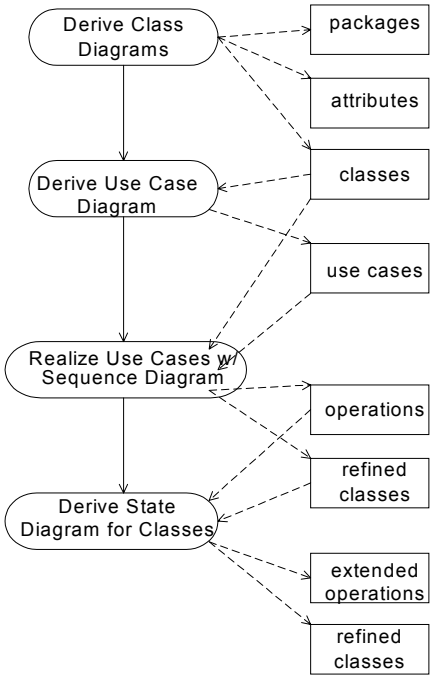

Figure 4 Activity Diagram of UML Stakeholder Model Development Method

Based on these operations and the stakeholder conceptual model data the state diagrams are derived for the behavior of the classes. At this stage, there are certain, minimal operations (events) that are added as demanded by the state diagram for closure and sequencing of states. These operations do not correspond to a sequence diagram. In summary, there are two places where information is added.

- Abstract classes and packages are added to organize and structure the information given by the stakeholder
- Event operations are added at the time the state diagrams are derived for closure and sequencing.

Developing Completeness Indicators. Four areas have been researched in order to derive a basis for evaluating the completeness of a model.

- The things that the analyst would normally ask the stakeholder when evaluating a stakeholder conceptual model and performing the iteration on the models.
- The goodness criteria or heuristics for models expressed in the literature that is related to completeness and can be quantified.
- Other completeness or consistency indicators discussed in the literature pertaining to requirements specifications or models.
- The model checking done by the tool used (iLogix Rhapsody).

Analysts Evaluation of the UML Stakeholder Models. Analysts heuristically evaluate the completeness of models based on their experience and knowledge base. Some of the questions that an analyst would go back and ask of the stakeholder in order to derive a better model are a basis for formulating completeness indicators for a stakeholder conceptual model. For this research using OO analysis as expressed using UML, some of these concerns include:

- abstract classes where only one or no children exist; more children may be identified since abstract class behavior is implemented by children classes
- abstract classes that are represented at the highest level of abstraction (i.e. not contained in a package) and therefore seem to need further structural elaboration because the model seems unbalanced
- isolated parent classes for which all the connections are made to the child class; it may be possible to generalize the relationship to the parent
- abstract operations that are represented by vague words like "involve" or "interact" require further elaboration
- cases where a verb is made into a noun and represents a class, such as "checks performed"

Goodness Indicators or Heuristics in the Literature. In the user guide for UML (Booch et al. 1999), the chapters end with a section that provides characteristics of a well-structured model. There are other heuristics set forth in the literature as well about the goodness of a requirements specification (Kotonya & Sommerville, 1998). There are many "how to" books for modeling, but few specify criteria for determining goodness. In most cases, the

criteria that is set forth are difficult to measure, some are taken care of with good modeling practices, and a few pertain to completeness indicators. An excerpt from Booch et al. (1999) is given below:

"A well-structured classifier

- Have both structural and behavioral aspects
- Is tightly cohesive and loosely coupled

A well-structured package

- Is cohesive, providing a crisp boundary around a set of related elements
- Is loosely coupled, exporting only those elements other packages really need to see, and importing only those elements necessary and sufficient for the elements in the package to do their job.
- Is not deeply nested, because there are limits to the human understanding of deeply nested structures
- Owns a balanced set of contents; relative to one another in a system, packages should not be too large or too small."

From these guidelines, the measure of nesting and of balance would contribute to indicators of completeness. While nesting and balance are a function of both the system being modeled and the modeling techniques, they are useful as indicators of where more elaboration may be needed.

Other Completeness Indicators in the Literature. Gunnar Overgaard (1999) correlates collaborations (sequence diagrams) to object diagrams and use cases. He defines consistency by saying that all action sequences contained within a collaboration that realizes a use case should also be contained in the operations for a class. In this research, the author uses this assertion to address the issue of operations that are introduced by state diagrams.

Heimdahl and Leveson (1998) formally address completeness with respect to a set of criteria related to robustness. A response is specified for all possible input and input sequences. They also address internal consistency, which for their purposes is defined as a specification free from conflicting requirements and undesired non-determinism. Their work focuses on modeling the requirements using Statecharts (Harel, 1990), more specifically on the safety behavior of a system. Statecharts are also used in UML to model the behavior of classifiers. However, in order to perform the level of completeness and consistency checking that is performed in Heimdahl and Leveson the specification must be at a more mature stage than the requirements discovery stage that is addressed by this research. The analysis they describe is better performed once the models are integrated and iterated with the stakeholder community.

Mayr and Kop (1998) describe a technique called KCPM that they use for deriving models based on natural language requirements by formulating the content into structured tables. They discuss incompleteness of a model as where the structured table does not have an entry. This work does not deal with the abstract nature of the 'things' introduced or the need for more elaboration based on an unbalanced model. These drawbacks may be due to the tabular formulation of the data. The areas of incompleteness addressed in their work are multiplicity, quantification of nouns, and finding synonyms. While very structured and systematic in their approach, the information sought does not address conceptual completeness, i.e. is the entire domain body represented to an adequate degree of elaboration.

Model Checking Done by the iLogix Tool. Much of the feedback offered by the iLogix Rhapsody tool, as well as others, aides the modeler in constructing the model. An example of this feedback is a warning about states with no exit transitions. Some of the warnings refer to things that are nice to have, but not necessary, like description annotations for the various modeling elements. At some point, these descriptions are added as the model evolves and this provides a glossary of the system. One of the warnings that relates to the conceptual completeness and consistency addressed in this research, ''Element with no relation,'' is a strong indicator of a class that is not in a collaboration. However, iLogix Rhapsody, as well as most other tools, does not check sequence diagrams.

ILogix Rhapsody has a powerful automated code generation capability. The errors and warnings are given in the context of animating and providing code generation; therefore, it is very rigorous about checking the model. This feature also expects a certain level of system resolution that may not be possible at the requirements discovery stage, i.e. types for all attributes and arguments. The analyst must use the information to define the model as much as possible, bearing in mind that a conceptual model is what is being built.

COMPLETENESS INDICATORS

The completeness that is being addressed by this research is a measure of resolution of information. This research does not evaluate the correctness of the information given by the stakeholders per se. There is also the aspect of internal consistency between the static and behavioral views in the model. This internal consistency reflects primarily on the completeness of the behavioral view because the conceptual model reflects principally the static view, which is more complete initially. The scenarios provided by the stakeholders tend to be very sketchy initially when using the ISCM Development Process. With iterations, the scenarios become more concrete because the stakeholder conceptual model is better specified.

Abstract Classes. Abstract classes represent a generalization and have children sub-classes. If abstract classes are identified and no children or only

one child is specified then this condition is an indication of incompleteness. The first proposed completeness indicator is the ratio of abstract classes, where the number of children is greater than or equal to two, to the total number of classes contained in the class diagrams. Guidelines for identifying possible abstract classes include, 1) classes that are introduced in order to structure the class diagram view, 2) classes that are mentioned as generalizations by the stakeholder, i.e. ‘‘documents like registration document, grade report, etc.’’ and 3) classes that contain only abstract operations.

Abstract Operations. The second indicator of completeness is the ratio of all non-abstract operations specified in class diagrams to the total number of operations specified in the class diagrams. There are often operations specified that are clearly abstract, like ‘‘interact.’’ The operations for which the analyst cannot outline in a method (OO term) for realizing an operation are considered abstract. Identifying abstract operations is a function of the analyst as well as the information provided. A guideline to use is that if the analyst has reservations on how an operation might be realized by a method, in the context of the problem domain, then the operation should be considered abstract. Given the diverse composition of a group of stakeholders and the preconditions set in Table 1 regarding the analyst, if the analyst is uncertain then at least one of the stakeholders will have questions. Often the details of implementation are not clear, but are routine in nature and can be elaborated later. For example, operations like printing or generating a document are resolved enough at the conceptual level and are not considered abstract.

Package Nesting. The third indicator of completeness is a function of the average nesting value where nesting is a function of package containment. A class has a nesting value of one if it is within the <<toplevel>> (Object Management Group, 1999) package, each subsequent package containment increments that nesting value by one. The average nesting is taken over all the classes in a UML model. The average is calculated by summing the classes at the various nesting levels, multiplying each sum by the nesting level, then sum each of these values and divide by the total number of classes to get the average.

The third indicator of completeness must be defined given the average nesting value and the standard deviation of that average. It is necessary to first select a nesting value goal that the analyst wishes to achieve at this stage of system definition. Since an initial ISCM definition is at an early stage and involves diverse stakeholders, an ideal nesting value of two (2) is chosen. This ideal nesting value reflects the experience of the author and other analysts. It is usually the case that in resolving level *n* in a model, that level *n-1* is truly resolved and not likely to change significantly. This nesting goal normalizes the average nesting value obtained and allows establishing a completeness indicator for nesting.

Balance. The forth indicator of completeness is the balance of a model and is based on the standard deviation of the average nesting value.

Class Consistency. All classes discussed by the stakeholder should be involved in some scenario that elaborates the behavior of the class and therefore its purpose. The fifth completeness indicator is the ratio of all the classes that are contained in a sequence diagram to the number of classes contained in the class diagrams. Abstract classes are included even though there are only instances of non-abstract classes specified in a collaboration when the system is implemented. During conceptual modeling, the stakeholders tend to specify behavior for abstract classes using abstract operations. During the course of resolving the conceptual model further, both the abstract classes and the abstract operations are better specified such that behavior can be attributed only to non-abstract classes.

Operation Consistency. All operations contained in classes should be in a collaboration that realizes a use case (Overgaard, 1999). The sixth indicator of completeness is the ratio of the total messages specified in the collaboration diagrams to the total number of operations in the class diagrams.

CONCLUSIONS

A UML stakeholder model can be evaluated for conceptual completeness using completeness Indicators. In this paper six indicators of completeness are proposed. Based on the research there are aspects of a conceptual model that can be further defined formally in UML as quantifiable indicators of completeness. The completeness indicators are a necessary part of regulating the ISCM Development Process or any process that develops conceptual models. Further validation of these indicators is required in order to understand and possibly modify them such that they are providing the desired regulating effect.

FUTURE WORK

The completeness indicators developed in this research need to be further analyzed. The first step would be to perform a Monte Carlo analysis of the indicators as portions of a model are removed. This would provide further insight into the nature of the indicators, their variability, their correlation to expected results based on expert experience, etc. This would also provide the opportunity to write software to help extract the data required to calculate these indicators and automate their calculation to the extent possible.

REFERENCES

Booch, G., Rumbaugh, J., and Jacobson I., The Unified Modeling Language User Guide. Addison Wesley, 1999

Briand, L. C., Morasca, S., Basilli, V. R., "Property-based Software Engineering Measurement," IEEE Transactions on Software Engineering, Vol 22, No. 1, January 1996.

Darke, P. and Shanks, G., "Stakeholder Viewpoints in Requirements Definition: A Framework for Understanding Viewpoint Development Approaches." Requirements Engineering (1996) 1, 88-105, 1996 Springer-Verlag London Limited.

Davis, A. M., Software Requirements: Objects, Functions and States. Prentice Hall, New Jersey, 1993.

Delbecq, A. L., Van de Van, A. H. and Gustafson, D. H., Group Techniques for Program Planning: A Guide to Nominal Group and Delphi Processes, Scott, Foresman and Company, Glenview, Illinois, 1975.

Gonzales, R.M. and Lovelace, N., "Using Stakeholder Mental Models to Create an Integrated System Conceptual Model for Systems," Proceedings of the eighth Annual International Symposium of the International Council on Systems Engineering, Volume 1, July 26-30, 1998.

Gonzales, R.M. and Wolf, A.L., "A Facilitator Method for Upstream Design Activities with Diverse Stakeholders," Proceedings of the Second International Conference on Requirements Engineering, April 1996, pp. 190-197, IEEE Computer Society Press.

Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., and Trakhtenbrot, M., "Statemate: A Working Environment for the Development of Complex Reactive Systems," IEEE Transactions on Software Engineering, Vol. 16, No. 4, April 1990.

Heimdahl, M. P. E., Leveson, N. G., "Completeness and Consistency in Hierarchical State-Based Requirements," IEEE Transactions on Software Engineering, Vol. 22, No. 6, June 1996.

IEEE Standards Board, "IEEE Std 1233-1996, IEEE Guide for Developing System Requirements Specifications," IEEE Standards Collection, IEEE Inc., New Jersey, 1996.

IEEE Standards Board, "IEEE Std 830-1993, Recommended Practice for Software Requirements Specifications," IEEE Software Engineering Standards Collection, IEEE Inc., New Jersey, 1994.

IEEE Standards Board, "IEEE Std 1362-1998 (Incorporates IEEE Std 1362a-1998) IEEE Guide for Information Technology -- System Definition -- Concept of Operations (ConOps) Document," IEEE Software Engineering Standards Collection, IEEE Inc., New Jersey, 1998

Kar, P. and Bailey, M., "Characteristics of Good Requirements," Proceedings of the Sixth Annual International Council on Systems Engineering, Vol 2, July 1996, pp. 284-291.

Kotonya, G., Sommerville, I., Requirements Engineering Processes and Techniques, 1. Edition, John Wiley & Sons, 1998.

Kop, C., Mayr, H. C., "Conceptual Predesign Bridging the Gap between Requirements and Conceptual Design," Proceedings of the Third International Conference on Requirements Engineering, 1998.

Larman, C., Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall, New Jersey, 1998.

Object Management Group, Inc., OMG Unified Modeling Language Specification, Version 1.3, www.rational.com/uml, June 1999.

Overgaard, G., "A Formal Approach to Collaborations in the Unified Modeling Language", UML'99 - The Unified Modeling Language. Beyond the Standard. Proceedings, Second International Conference, Fort Collins, CO, USA, pg. 99-115, October 1999.

Pfleeger, S. L., Software Engineering: Theory and Practice, Prentice Hall, March 1998.

Wieringa, R.J., Requirements Engineering: Frameworks for Understanding, John Wiley & Sons, New York, 1996.

BIOGRAPHY

Dr. Regina M. Gonzales is a Project/Site Manager at TMA. She is also a College Assistant Professor at New Mexico State University. Regina has worked and consulted in industry and government in the areas of requirements capture/modeling, computer/software design, process, project management and training for over 16 years. She is the current chair of the Requirements Working Group within INCOSE. She has a Ph.D. in Computer Engineering with a specialty in Requirements Engineering from NMSU, an MS in Computer Science from University of Colorado, an MS in Electrical and Computer Engineering from University of Arizona, B.S. in Electrical and Computer Engineering from NMSU.

Previous Menu/Document