An Approach To Conceptual Feedback In Multiple Viewed Software Requirements Modeling

Harry S. Delugach

Computer Science Department
Univ. of Alabama in Huntsville
Huntsville, AL 35899
phone: (205) 895-6614
fax: (205) 895-6239
Email: **delugach@cs.uah.edu**
WWW: **http://www.cs.uah.edu/~delugach**

1. INTRODUCTION

Software requirements analysis and specification is concerned with the exploratory phases of software development: namely, defining a problem and its domain, followed by the process of identifying what features and constraints must be embodied in any software that is expected to solve the problem. A multiple viewed requirements technique deals with requirements analysis based on having available diverse multiple descriptions of software requirements. Multiple views originate from the various people, perspectives and purposes involved in a system. Because these multiple descriptions are often expressed in differing notation schemes based on differing underlying paradigms and methodologies, the problem of consistency and completeness is a significant impediment to obtaining a reliable set of requirements. Although each view by itself may be demonstrated to be internally consistent and coherent, one view is not usually formally integrated with another. This paper outlines part of an approach to these multiple-viewed requirements that provides some structure for integrating and validating multiple views.

Most recent research has acknowledged the presence of multiple views, but only a few have explicitly modeled them as distinct views. The work of Nissen, et al **[Nissen96]** is an example of a practical technique that is used in commercial settings to form a framework for discussion and negotiation among participants. Its biggest drawbacks are (a) it depends upon having a skilled (human) facilitator, thus allowing for potential biases to appear and (b) there is no formal way of modeling negotiation or overlap. Viewpoints **[Nuseibeh94]** describes a partitioning of viewpoints that provides organization to the content of a viewpoint which is relevant to examining multiple viewpoints for consistency and completeness. The work of Leite [**Leite91**] addresses the question of eliciting different opinions about what a system's requirements are, and supports the negotiation process whereby the differing opinions may be reconciled. The terminology and framework of Leite's work are quite applicable to the current proposal. Their main focus is in providing an example of how viewpoints may be reconciled; in their case, by using rule-based models.

These techniques are all designed around a fairly well-defined (and pre-defined) set of interactions between views. The techniques rely on heuristics to evaluate the interactions, much as do most rule-based systems. These techniques can benefit from a uniform representation that is general enough to capture all of the relevant information in the views, with a minimum of human bias and intervention.

The approach described herein is not a methodology; as the views can be obtained from a variety of sources. Our main contribution is to incorporate conceptual graphs (a well-known semantic modeling technique; e.g., **[Sowa84] [Nagle92]**) as a general and powerful representation of the cognitive information in requirements.

2. CONTEXT OF WORK

The primary objectives of the larger project is to construct a collaborative, multiple-viewed software requirements analysis environment. An overview is shown in Figure 1. The general capabilities of the environment will be the following:

- capabilities for acquiring domain knowledge using conceptual graphs,
- translation of software requirements analysis views into conceptual graphs,
- formal analysis of the views, with respect to their overlap, consistency and completeness
- validation of resulting views through further interaction with human analysts.
- generation of software requirements specification documents.

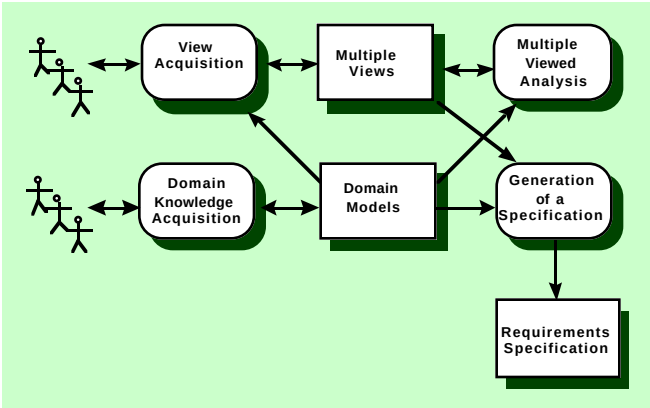


Figure 1. Multiple Viewed Requirements Environment.

The primary input is either directly from human analysts or indirectly from them via CASE tools. Each diagram or specification will be translated into conceptual graphs, using algorithms along the lines of the ones already developed by us earlier **[Delugach91], [Delugach92a], [Delugach92b]**. The result is a set of specification graphs, with one conceptual graph for each originating diagram. This collection of graphs can then be analyzed according to the normal rules of conceptual graphs; e.g., they can be joined, matched, analyzed for inconsistency and incompleteness, etc.

Once the combined analysis is done, translation can be made back into one of the originating diagram views (e.g., data flow diagrams). This allows existing automated tools to perform that diagram view's own internal consistency checks, but more importantly, it allows the human analyst to see *new information*, in his original notation, that was gained by incorporating the other views. The human analyst can then examine, validate, and evaluate the new information from other views without having to explicitly see those other views in their own notations (or understand the internal representation).

From a human analyst's point of view, they create a diagram (or set of diagrams) describing a system using whatever notation they choose and tell the tool to "store" the diagrams. Other analysts have done the same, each in their own chosen notation. Then either analyst asks the system to "retrieve" the system description in their original notation. Instead of seeing the same diagrams as before, the analyst sees augmented diagrams with additional information, whose source may be traced to the various other views if desired. The analyst can then examine the new diagram(s) and provide additional information, and then "store" the new diagram, so that the process iterates. (Of course, the original diagram is still intact for tracking and editing purposes.)

The human analyst therefore has the potential to get feedback from the automated analysis so that inconsistencies and incompletenesses in their original diagrams (as well as the combined diagrams) may be flagged for being addressed by the human analyst, either then or later. The approach is thus naturally iterative.

Although it is beyond the scope of this short paper to discuss the knowledge acquisition and domain modeling portion of the environment, we consider them to be crucial to the eventual success of any such system. Descriptions of other ongoing work in this area will appear elsewhere **[Delugach96] [Wolf96]**.

3. Modeling Multiple Views with Conceptual Graphs

One central feature of this approach is the use of conceptual graphs to represent and manipulate software requirements information. Conceptual graphs are a visual form of semantics networks that is easy to understand, yet powerful enough to express a wide range of knowledge constructs. This short paper introduces conceptual graphs briefly; for a more complete introduction, see **[Polovina92]** or **[Sowa92]**.

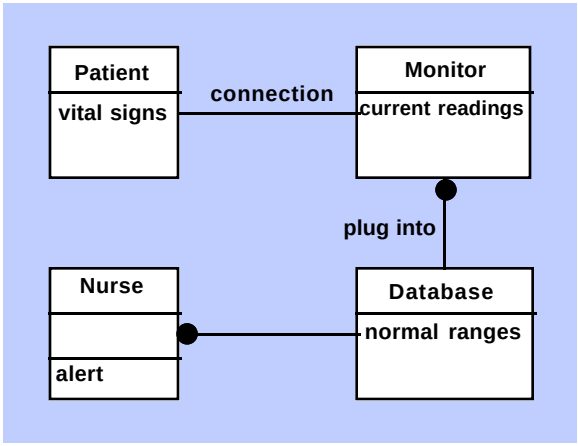
3.1 Conceptual Graphs

A conceptual graph consists of concepts and relations; concepts denoted by a box and relations denoted by a circle or oval. Relations are connected to concepts via directed links; the direction of the arrow is predetermined and usually follows a linguistic convention. A concept contains a type identifier with an optional referent which indicates a particular individual (or set of individuals) of that type. Associated with one or more conceptual graphs is a type hierarchy showing subtype/supertype relationships with multiple supertypes allowed, and a set of definitions (not shown here).

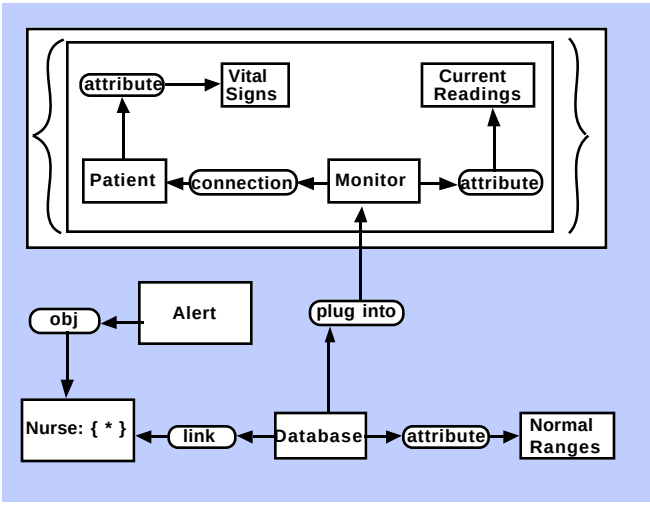
Starting along the lines of our earlier work [**Delugach91**], [**Delugach92a]**, various notations in software requirements are provided with translations to and from conceptual graphs. Due to space limitations, we do not present the algorithms here. An early version of the data flow translation appears in [**Delugach92a]**.This paper will show two examples: a Rumbaugh OMT object diagram **[Rumbaugh91]** and a data flow diagram.

3.2 OMT object diagrams

Figure 2(a) shows a typical view of a system using OMT. Boxes represent object classes, with attributes and operations shown. Figure 2(b) shows the initial conceptual graph that would result from a first translation step.



(a)



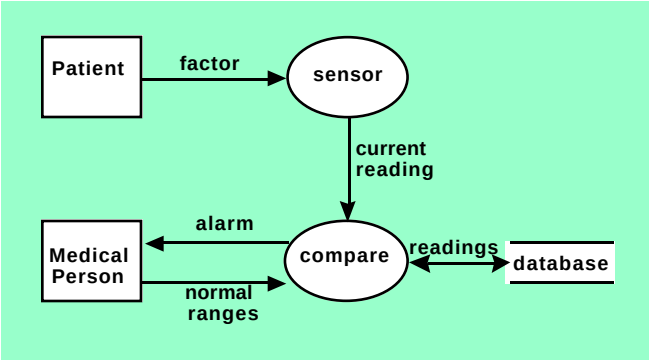
NURSE < ENTITY.
DATABASE < ENTITY.
PATIENT < ENTITY.
MONITOR < ENTITY.
ALERT < PROCESS.

(b)

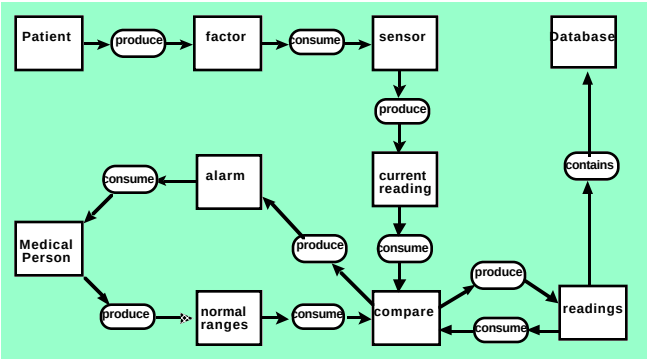
Figure 2. Object diagram and its conceptual graph.

3.3 Data flow diagrams

Figure 3(a) shows another typical view of a system using a data flow diagram (DFD). Obviously a small example, it nonetheless captures some of the features of a view that this work will consider. There are labels which originate from real-world entities, whose meaning is implied (e.g., "patient") but not further described in the view. There are relationships called for by the particular notation, in this case, data flows between processes, actors or data stores.



(a)



Medical_Person < ANIMATE_ENTITY.
Patient < ANIMATE_ENTITY.
DATABASE < ENTITY.
sensor < PROCESS.
compare < PROCESS.

(b)

Figure 3. Data flow diagram and its conceptual graph.

4. VALIDATING MULTIPLE VIEWS

After views have been acquired, they can be combined via conceptual graph operations, such as joining around coincident concepts. Figure 5 shows the combined graphs from Figure 2(b) and Figure 3(b).

There are several validation and verification techniques using the combined graphs. Two important ones are (a) feedback into one's original diagramming technique, and (b) paraphrasing graphs into natural language.

4.1 Originating Notation

Using the translation from conceptual graphs to the various notations, the analyzed requirements can be presented to each original requirements specifier in their original notation, with additional features obtained during analysis with the other views. As an example, we show in Figure 4 an augmented data flow diagram that would be presented to the DFD specifier. Some additional features have appeared:

1. The "Medical Person" has been changed to "Medical Person / Nurse" reflecting the conceptual graph join.
2. An "alert" process has appeared, since it was introduced by the OMT specifier as an operation.
3. An unknown "flow" appears between "alert" and "Medical Person / Nurse" since there is an (obj) relation between those two in the conceptual graph.
4. A "Monitor" actor has appeared due to the appearance of the entity in the OMT's conceptual graph.

Note that feature 1 is subject to change based on the natural language feedback. Feature 3 represents an incompleteness; the DFD specifier must decide the direction and label of the flow (or to eliminate it entirely). Feature 4 also is an incompleteness; the DFD specifier is invited to provide flows for the monitor.

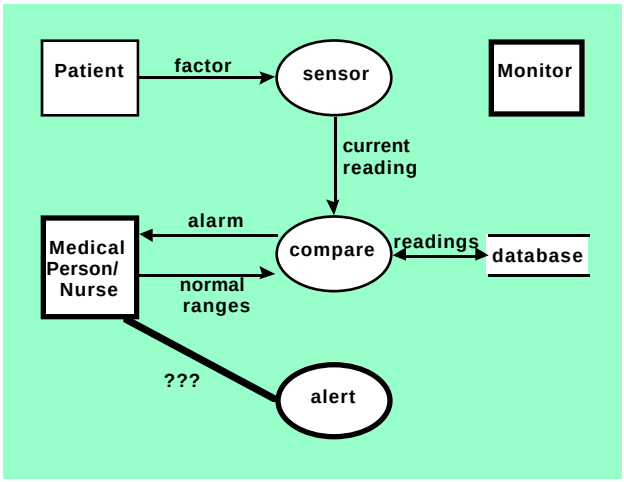


Figure 4. Augmented Data Flow Diagram.

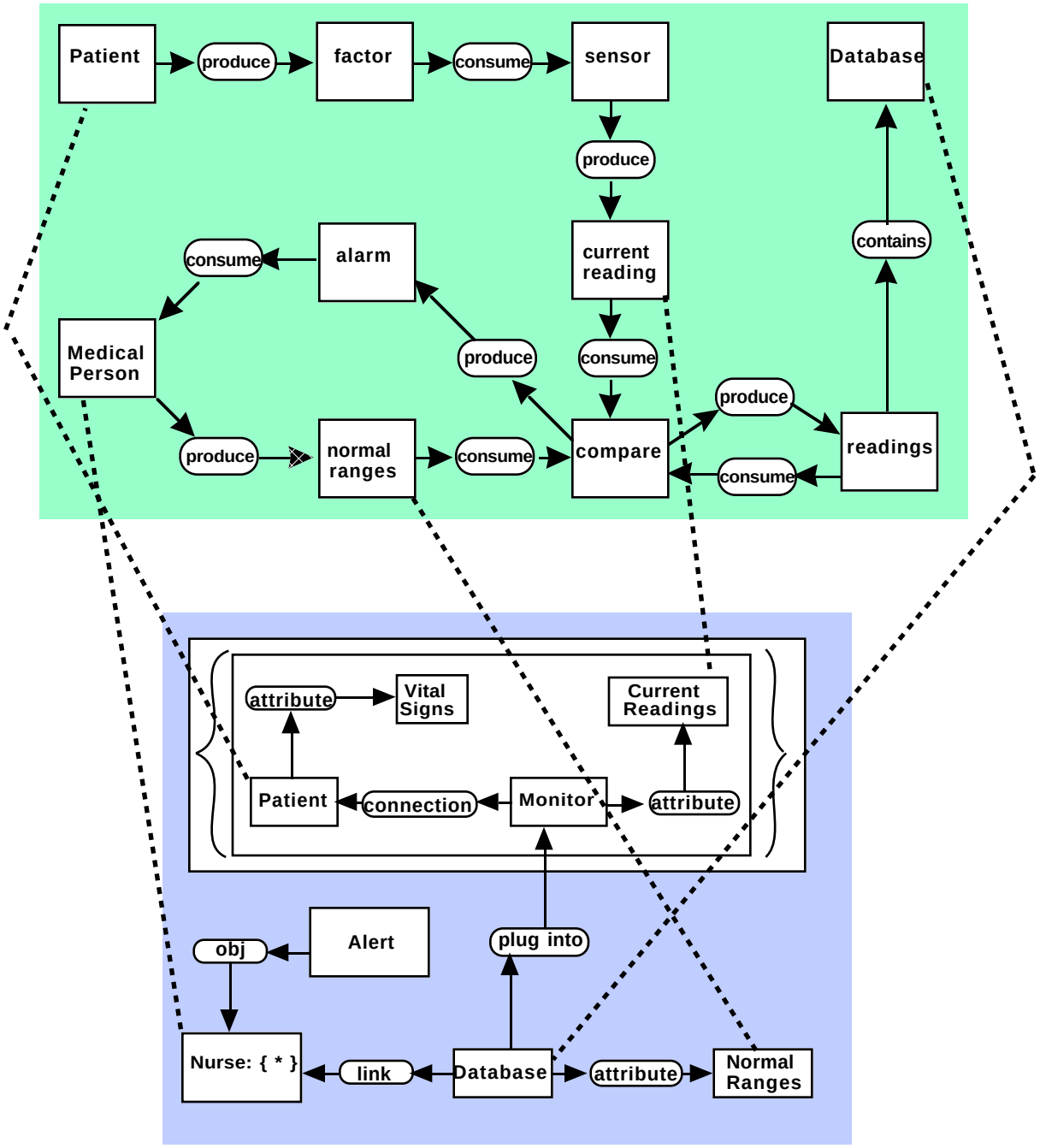


Figure 5. Combined specification graphs.

4.2 Natural Language

Paraphrasing consists of presenting each graph, or the combined graphs, or the analyzed combined graphs in English (or other natural language) for each requirements specifier to evaluate and respond to. For example, using Figure 5, an automated system produces the following results to which the specifier can respond.

1. *A patient produces factors, is connected to a monitor, has vital signs, and there are zero or more patients*. The OMT specifier and DFD specifier both agree with this.
2. *A medical person/nurse is linked to a database, undergoes an alert, consumes an alarm, and produces normal ranges*. The DFD specifier notes that a nurse is not supposed to produce the normal ranges; they are to come from a doctor. Therefore the

DFD needs changing and the term "medical person" is not the same as "nurse".

3. *A database contains readings, is linked to nurse, has normal readings*. OMT specifier notes that the link to nurse can be further specified as "accesses".
4. *Normal ranges is produced by medical person / nurse, and is consumed by compare process*. This validates the DFD specifier's earlier conclusion that "medical person" is not the same as "nurse".
5. *Current reading is produced by sensor process, is attribute of monitor, and is consumed by compare process, There are zero or more current readings*. OMT specifier notes that the current readings are really related to the patient's vital signs rather than just a monitor feature. Both specifiers note that sensor process and monitor entity are related, suggesting that the sensor process perhaps ought to be located in the monitor.
6. *A compare process has one normal ranges, many current readings*. Both specifiers agree with this.

5. CONCLUSION AND FUTURE WORK

This paper described one part of a multiple viewed development environment that deals with integrating and validating each view through feedback from the people who have created each view. As part of a larger body of research, it is being developed along with the knowledge acquisition and (most importantly) domain modeling efforts to begin exploring how these ideas all interact with each other.

The examples suggest that conceptual feedback is a valuable feature for any requirements development environment. Overlap and inconsistency are readily apparent through two kinds of feedback: translation to one's originating notation and natural language paraphrasing. Obviously the techniques need to be applied to substantial development efforts in order to determine their effectiveness in the large scale.

Two strengths of a conceptual graph based approach are (a) it easily supports natural language interaction with the environment and (b) it does not enforce much structure or a certain methodology, thus providing generality. It is not yet clear whether this environment will (or should) ultimately lead to a coherent methodology. At the outset, we are avoid prescribing a methodology so that several different ones may be tried. An interesting future development might be to treat a methodology itself as a view and hence to model multiple methodologies using this approach. Such a daunting task would nonetheless be useful, both as a research tool in exploring the relationship between various methodologies and as a practical development tool, since large projects usually comprise subsystems that may be developed by different vendors using different methodologies.

6. REFERENCES

[Delugach91] Delugach, Harry S. "A Multiple Viewed Approach to Software Requirements", Ph.D. dissertation, Computer Science Department, University of Virginia, Charlottesville, VA, 1991.

[Delugach92a] Delugach, Harry S., "Specifying Multiple-Viewed Software Requirements With Conceptual Graphs," *Jour. Systems and Software*, vol. 19, pp. 207-224, 1992.

[Delugach92b] Delugach, H.S. "Analyzing Multiple Views Of Software Requirements" [In] *Conceptual Structures: Current Research and Practice*, Eklund, P., Nagle, T., Nagle, J. & Gerholz, L. [Eds.], 1992, 391-410, Ellis Horwood.

[Finkelstein92] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. & Goedicke, M. "Viewpoints: a framework for integrating multiple perspectives in system development", *International Journal of Software Engineering and Knowledge Engineering*, 2 (1), 1992, 31-57.

[Leite91] Leite, J.C.S.P. and Freeman, P.A. "Requirements Validation through Viewpoint Resolution", *IEEE Trans. on Software Eng*., 1991, 17 (12), 1253-69.

[Nagle92] Tim Nagle and Jan Nagle and Laurie Gerholz and Peter Eklund, eds , *Conceptual Structures: Current Research and Practice*, Ellis Horwood, 1992.

[Nissen96] Hans Nissen, Manfred Jeusfeld, Matthias Jarke, Georg Zemanek and Harald Huber, "Managing Multiple Requirements Perspectives with Metamodels," *IEEE Software*, 12(6), pp. 37-48, 1996.

[Nuseibeh94] Nuseibeh, B., Kramer, J. and Finkelstein, A. "A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications", *IEEE Trans. on Software Eng*., 20 (10), 760-73, 1994.

[Polovina92] Polovina, Simon and Heaton, John, "An Introduction to Conceptual Graphs", AI Expert, pp. 36-43, 1992.

[Rumbaugh91] Rumbaugh, James, Blaha, Michael, Premerlani, William; Eddy, Frederick; and Lorensen, William, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Sowa84] Sowa, John F., *Information Processing in Mind and Machine*, Addison-Wesley Publ., Reading, MA, 1984.

[Sowa92] Sowa, J. F. "Conceptual Graphs Summary," in *Conceptual Structures: Current Research and Practice*, Tim Nagle and Jan Nagle and Laurie Gerholz and Peter Eklund, eds., Ellis Horwood, 1992 pp. 3--52.