Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

Classification as an Approach To Requirements Analysis

James D. Palmer, Yiqing Liang, and Lillian Wang

Center for Software Systems Engineering
School of Information Technology and Engineering
George Mason University, Fairfax, VA 22030-4444, USA

A CLASSIFICATION APPROACH TO PROBLEM SOLVING

Classification schemes have proven immensely useful in computerized systems for problem solving in the physical and biological sciences. A typical example is found in the medical domain where various taxonomies have been created for diseases, symptoms, laboratory tests, drugs, and so forth. These taxonomies may be used for characterizing specific clinical situations in expert systems that assist physicians and other health professionals in diagnosis and treatment.

Classification can be divided into several phases. The activities in the first phase are to find a category structure which can fit observations. This phase is called "cluster analysis", or "typology", "learning", "clumping", "regionalization", etc., or "classification (construction)" in our paper, depending on the field to which it is applied. There are many approaches to this cluster analysis. These include approaches such as numerical taxonomy and conceptual clustering. Once this category structure has been established, the next phase is to classify new observations, that is, recognize them as members of one category or another. There are two different situations for the activities in this phase: 1) when the category structure is completely known, this kind of activity is called "classification", "indexing", or "classifying" in this paper, and 2) if category structure is partly known or only part of the information of the observation is known, this kind of activity is called "discriminant analysis" [AND73] [GOR81].

Clancey [CLA84] has characterized classification problem solving as making a selection from a set of pre-enumerated solutions (in contrast to constructing new solutions). If the problem solver has a priori knowledge of existing solutions and is able to relate these to the problem description by data abstraction and refinement, then the problem can be solved using classification. Other artificial intelligence researchers, especially those investigating machine learning, have developed new techniques such as conceptual clustering [MIC86] (in contrast to numerical/statistical clustering), which might be used for developing classification schemes for problem solving.

CLASSIFICATION AS AN APPROACH TO REQUIREMENTS ANALYSIS

Software requirements analysis (or requirements analysis briefly) is the first important stage in the whole software development life cycle. During this stage, requirements analysts, working closely with users and customers, define a complete description of the external behavior of the software system to be built. This description is usually called software requirements, software definition, or software specification. For example, one of the requirements of a software system Howitzer Improvement Program (HIP) for upgrading the Howitzer capability is defined as:

Requirement No. 1) Default CP initialization parameters shall be stored in nonvolatile memory to maximize user friendliness.

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

Many people have proposed different requirements taxonomies [SOM85] [ROM85] [FAI85] for use during this phase. Basically requirements can be classified into functional and nonfunctional. Each of these two categories can be subdivided into subclasses and so on. Samson [SAM89] gave a rather comprehensive taxonomy of requirements. Following is a segment of the taxonomy:

- Functional
  - Software Utility
  - Software Performance
  - External Interface Requirements
  - Data Requirements
    - Logical Data Structure
    - Data Management
- Nonfunctional
  - Design Constraints
  - Quality Goals
    - Life-cycle Constraints
    - Security
    - Operational
  - Operational Constraints

Even so, requirements analysis has generally been a neglected phase for software development. The consequences of this neglect are so serious that no one involved in software engineering can afford to ignore them. Based on this, we suggest an early fix of such errors as ambiguities, inconsistencies, incompleteness and incorrect facts is required [PAL88] in the requirements analysis phase of the software engineering life-cycle.

The requirements analysis problem seems to be a good application for a knowledge-based system for detecting conflicts among requirements. A proper taxonomy would provide factual knowledge concerning this. Procedural knowledge would consist of rules for identifying potential conflicts. A taxonomy would facilitate evaluation of rules applied to large-scale systems where there are thousands of requirements, and where rules that apply to top-level conceptual nodes in the taxonomy would probably apply to descendants as well. Thus, a taxonomy lends itself to systems that feature inheritance. Samson [SAM89] first suggested a set of rules indicating how conflicts among requirements in a taxonomy may be resolved. If such a taxonomy were used for classifying (indexing) the requirements for a specific software system, these rules would use this classifying (indexing) method as access points to the requirements, and identify potential conflicts among requirements for that system. We will return to the taxonomy problem in the next section.

The application just described, for use in requirements conflict resolution, uses a one-step inference and a single classification structure, namely, the requirements taxonomy. Requirement No. 1 can be classified as *functional / output / data storage*. Another HIP requirement is:

Requirement No. 2) Automated fire control system for the M109 self-propelled howitzer shall require real-time processing.

This is defined as a real-time system, thus it is classified as *non-functional / process constraints / real-time*. In this case, conflict is identified directly between these two requirements. They are in conflict because Requirement No. 1 calls for a static system while No. 2 calls for a real-time system. This is reflected in the following conflict identification rule:

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

IF there is a requirement that is classified as *functional / output / data storage*
AND IF there is a requirement that is classified as *non-functional / process constraint / real-time*
THEN there is a conflict between these two requirements

In addition to conflict resolution, we believe classification is a powerful tool that may be applied to the more complex task of software validation and verification, with the ultimate objective of arriving at a software test plan. This application would use multiple taxonomies and perform multi-stage inference. The remainder of this section describes the problem-solving steps for this expanded application, including identifying the different taxonomies that would be needed.

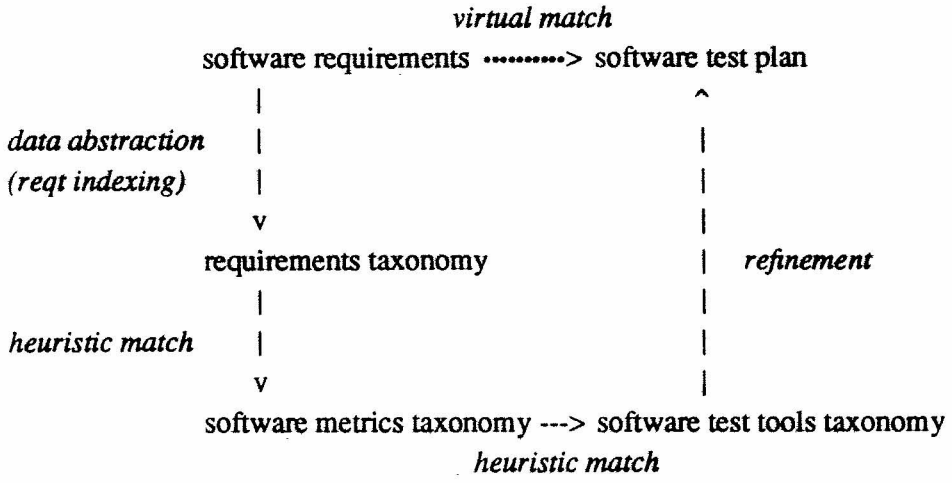
Conflict resolution as part of the activity of requirements validation is carried out in the requirements analysis stage. The implications of software validation and verification in software requirements prompt another more complicated task, that of generating software test plan from software requirements in requirements analysis stage.

Some ground work has been done for the implementation of this task. Much research has been carried out on software metrics and testing. Many software metrics have been defined and there are several hundred commercial test tools available. However, software metrics and test tools are generally applied to software projects that have been completed. Our plan is to utilize this technology, but rather than apply it to software, we would move the operations of measurement and testing to the requirements analysis stage, prior to the labor-intensive task of writing software. The basic idea is to apply these operations, in effect, to the software requirements rather than the software itself at a much later time in the software development cycle. Armed with these ideas, a test plan for the software system is generated in the requirements analysis stage, resulting in the parallel development of software testing with software development. Problems in identifying the testability of requirements volatility early in the initial stage of development will encourage refinement of those requirements before much development has been done.

At this stage, software requirements are not written in a formal language of any kind. Therefore, the problem is how to apply current technology that depends on existence of software. Our approach is to use three different taxonomies representing, respectively, software requirements, software metrics, and software test tools. The following diagram shows the flow of inference between these taxonomic components, in order to achieve software validation and verification resulting in a software test plan. Instead of one heuristic association or "great leap" [CLA84] of an abstract problem statement into features that characterize a solution, we use several "small leaps". As shown in the following diagram, one such association occurs between requirements taxonomy and software metrics taxonomy, and another, between software metrics taxonomy and software test tools taxonomy.

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472



The dotted arrow labeled *virtual match* indicates the final match we see as the result of a series of heuristic matches and other steps and does not exist in itself.

An example using Requirement No. 1 can be traced through the whole process. Initially this requirement is classified, as described earlier, as *functional / output / data storage*. A heuristic match from this classification against the software metrics taxonomy would result in selection of a metrics of "completeness". This, in turn, is mapped by a heuristics match to a test plan consisting of test tools such as inspection, snapshot, test case generator. Validation of the software would occur when test attributes are satisfied.

Thus, completion of the steps of data abstraction, multi-stage heuristic match between different taxonomies, and refinement would provide a satisfactory solution in the form of a test plan for the software requirements.

BRIDGING THE GAP BETWEEN REQUIREMENTS AND TAXONOMY

Classification problem solving for requirements analysis is based on the supposition that individual requirements for a software system are indexed according to a taxonomy. Obviously, classifying (indexing) requirements is the bottle-neck to the whole approach. It is especially prohibitive for large scale systems with thousands of requirements as the object. Neither a classic numerical taxonomy nor conceptual clustering may apply, as observing the attributes and the variable values of each requirement is impractical. Manual processing is certainly neither manageable nor dependable. Natural language processing to understand requirements so as to classify is impossible at this stage.

Our approach to classifying (indexing) a large set of software requirements specifications is to use semi-automatic natural language processing, based on syntactic analysis, coupled with human input using a human-computer interface. From comprehensive reading of several existing requirements specifications, we find that individual non-functional requirements are often characterized by nouns that indicate their quality goals. Thus, identification of these nouns represents a good first approach to classifying non-functional requirements. A requirement such as: "The system should have the maximum availability" clearly indicates it could be classified as a non-functional quality goal with the noun "availability" as indicator. Functional requirements

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

should, in most cases, imply an action or performance attribute required for the system. Thus, verbs in functional requirements statements should reflect the requirements classification. Reading of requirements specifications proves this assumption: verbs in the statements or the nouns or gerunds following certain verb patterns can be used as a means to classify the requirement.

We have investigated using verbs which have been identified in previous research. Studies indicate that verbs can be classified according to their functions [CHO81]. But we are interested more in software function related verbs and their taxonomy. For example, the verb CLUSTER is most often related to software requirements that can be classified as: functional / data requirement / data management / processing. *PSL/PSA (Problem Statement Language / Problem Statement Analyzer)*, an automated requirements specification tool developed at the University of Michigan [TEI77], recognizes 58 relationship types. PSL relationships may be likened to "verb" which, together with PSL objects, serves to generate "sentence". For the Requirement No. 1 example, identifying "memory" and "parameters" as PSL objects, we can relate them to one another with the PSL relationship type "CONTAINED" to form the PSL sentence "parameters CONTAINED-BY memory". However, this relationship does not indicate the functions for the entire requirement, as we do by classifying it as *functional / output / data strorage.* Wood and Sommerville [WOO88] and Maarek and Kaiser [MAA87] have identified some basic functions for software components, such as control, communicate, search, etc., but do not systematically relate them to as many verb classifications as possible.

We believe a Unix* system to be a more comprehensive system that can be used to describe various kinds of software functions. Verb selections used in describing Unix commands provide a list of more than 100 verbs related to and covering most software functions. Efforts were made to generate a verb taxonomy according to their semantic functions. As discovered by [CHO81], this taxonomy is rather bushy, not deep. For example, the verb CLUSTER has such subconcepts as categorize, classify, collect, gather, group, etc. These may be called synonyms to the primary verb.

From this we manually map this verb taxonomy onto a software functional requirements taxonomy and discover a set of rules. This set of rules is empirical and based on experience. It corresponds most closely to the "rule of thumb" approach from AI. Experience to date shows that based on a set of 81 software requirements for HIP, this set of heuristic rules is able to correctly classify more than 50% of the requirements. Eleven of them had apparent problems that could be recognized; these were identified by the system. The system could resolve the problems with these problematic requirements and classify seven of them correctly.

The approach we adopt is as follows: the set of heuristic rules is first used to automatically classify (index) as many requirements as possible. For requirements that the system is not sure how to classify, the system prompts the user with several options. And it identifies to users those requirements it cannot classify at all and requests a rephrasing of the requirement from the user according design templates.

---

* Unix is a Trade Mark of AT&T

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

FUTURE WORK

This research is supported by the Virginia Center for Innovative Technology and the US Army. George Mason University is working on the part of automatically classifying requirements and metrics taxonomy, while Sonex Enterprises Inc. is working on the tools taxonomy part. A prototype has been constructed and has been successfully demonstrated. A system to implement the system is now under development.

The three taxonomies of software requirements, software metrics, and software test tools, especially the latter two, need much more refinement. The verb taxonomy needs enhancement as well. The same is true with the rules that implement the mappings between these taxonomies.

A more schematic approach should be sought for automatically classifying requirements. Automatic indexing techniques in information retrieval, with a knowledge-based system to assist, may provide a good guide for future direction.

Multi-inheritance instances should be taken into consideration. Ranking techniques should be included for the user's decision process in selecting from matched results. This is particularly true when there is more than one hit resulting from a match. In our situation, where there are several steps that do matching, with probably more than one hit from each match, this is a necessity.

Classification schemes other than the enumerative ones could be adopted for comparison. The facet scheme [PRI87] should be one of those considered.

REFERENCES

[AND73] Anderberg, Michael R., *Clustering Analysis for Applications*, Academic Press, NY, 1973.

[CHO81] Chodorow, Martin S., "Growing taxonomic word trees from dictionaries," workshop presentation, IBM Research Center, Yorktown Heights, NY, 1981.

[CLA84] Clancey, W. J., "Classification Problem Solving," *Proceedings of National Conference on Artificial Intelligence, August 6-10, 1984, University of Texas at Austin*, AAAI, Los Altos, CA, pp. 49-55.

[FAI85] Fairley, Richard E., *Software Engineering Concepts*, McGraw-Hill, 1985.

[GOR81] A. D. Gordon, *Classification*, Chapman and Hall, London, 1981.

[MAA87] Maarek, Yoelle and Gail E. Kaiser, "Using conceptual clustering for classifying reusable Ada code," *ACM SIGAda International Conference, Boston, MA, 1987*, pp.208-215.

[MIC86] Michalski, Ryszard S. and Robert E. Stepp, "Learning from Observatin: Conceptual Clustering," in *Machine Learning — an Artificial Intelligence Approach*, Ed. by Ryzard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell, Morgan Kaufmann, Los Altos, CA, 1986, pp.331-363.

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

[PAL88] Palmer, James, "Impact of Requirements Uncertainty on Software Productivity", in *Productivity: Progress, Prospects, and Payoff, 27th Annual Technical Symposium Sponsored by Washington, D. C. Chapter of the ACM and NBS, Gaithersburg, MD, June 1988*, pp.85-90.

[PRI87] Prieto-Diaz, R. and P. Freeman, "A Software Classification Scheme for Reusability", *IEEE Software*, Vol. 4, No. 1, January 1987, pp.6-16.

[ROM85] Roman, Gruia-Catalin, "A Taxonomy of Current Issues in Requirements Engineering," *IEEE Computer*, Vol. 18, No. 4, April 1985, pp. 14-22.

[SAM89] Samson, Donaldine E., *Automated Assistance for Software Requirements Definition*, Ph.D. Dissertation, George Mason University, Fairfax, VA, 1989.

[SOM85] Sommerville, I., *Software Engineering*, Addison-Wesley, 1982, 1985.

[TEI77] Teichroew, D. and E. Hershey, "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing System,", *IEEE Transactions on Software Engineering*, Vol. 3, No. 1, 1977, pp. 41-48.

[WOO88] Wood, M. and I. Sommerville, "An Information Retrieval System for Software Components", *Software Engineering Journal*, Vol. 3, no. 5, September 1988, IEE, London, pp.198-207.

ISSN: 2324-9773

Palmer, J. D., Liang, Y., & Wang, L. (1990). Classification as an approach to requirements analysis. Proceedings of the 1st ASIS SIG/CR Classification Research Workshop, 131-138. doi: 10.7152/acro.v1i1.12472

ISSN: 2324-9773