DEASEL : An Expert System for Software Engineering

by Jon D. Valett and Andrew Raskin

ABSTRACT

For the past ten years, the Software Engineering Laboratory [1] (SEL) has been collecting data on software projects carried out in the Systems Development Branch of the Flight Dynamics Division at NASA's Goddard Space Flight Center. Through a series of studies using this data, much knowledge has been gained on how software is developed within this environment. Two years ago work began on a software tool which would make this knowledge readily available to software managers. Ideally, the Dynamic Management Information Tool (DynaMITe) will aid managers in comparison across projects, prediction of a project's future, and assessment of a project's current state. This paper describes an effort to create the assessment portion of DynaMITe.

1.0 Backround

Assessing the state of a software project during development is a difficult problem, but its solution contributes to the success of the project. By determining a project's weaknesses early in its life cycle, problems can be dealt with quickly and effectively. For the software manager to perform this assessment he needs easy access to detailed, accurate information (knowledge) regarding past projects within the development environment. He then incorporates this information with his own knowledge of software engineering to make some assessment of a project's strengthes and weaknesses. The DynaMITe Expert Advisor for the SEL (DEASEL) is the first version of an expert system that attempts to simulate this process.

2.0 Developing and Using Rules

Basically, DEASEL assesses an ongoing project by attempting to answer a simple question such as "How is my project doing?" To answer this question DEASEL utilizes a knowledge base of rules for evaluating software projects. This knowledge base consists of rules derived from two sources: the SEL database and experienced software managers. DEASEL uses these rules along with data on the project of interest, to give the manager a relative rating of the quality of that project.

2.1 Corporate Memory

Of course, a major effort in the development of the DEASEL system was the actual collection of knowledge. To derive rules from the corporate memory, former studies [2,3,4,5,6,7,8] performed by the SEL were reviewed to find relationships that affect the quality of a software project. That is, many studies of data concerning the SEL environment have been done within the last ten years. These studies give some idea of the cause and effect of technologies and methodologies on a software project. Thus, relationships like "increasing tool use will increase productivity" are found. Because of the interdependencies amoung the items the strength of each relationship is then determined. For example, many different factors may influence productivity, therefore the determination of which of these have the most and which the least influence must be made. This has been a long and difficult process because of the amount of data and the problems with determining what data is relevant to the assessment process.

2.2 Knowledge from Software Managers

The other source of knowledge is the experienced software managers, who have certain "rules of thumb" they use to evaluate a software project. They are questioned to obtain this subjective information which is then used along with the more objective material to produce the knowledge base. Again the determination of the strengthes of the relationships must be performed. The entire process of collecting knowledge is long and difficult and has only just begun for the DEASEL project.

2.3 Representing the Rules

After collecting a preliminary set of knowledge, thought began on how to actually represent this knowledge. The initial work on knowledge representation for DEASEL was directed at using standard expert system techniques, including if-then production rules. But soon the discovery was made that knowledge regarding the assessment of a software project's development is more naturally represented in a different manner. In fact, the overall conclusion drawn from an assessment is quite different from that drawn by a traditional expert system. The difference lies in the type of question answered by DEASEL. The traditional medical expert system, such as the often cited MYCIN [9], answers a question like "What disease does patient X have?" Then, given some data on the patient the system determines the disease. DEASEL, on the other hand, must answer the question "How is project X doing?" Thus, it must give a rating to the system based on the facts given to it. The analagous question in the medical domain would be "How is patient X's health?"

In order for DEASEL to answer the question "How is project X doing?", it needs two different types of knowledge. The first type of knowledge is the assertions which relate to the specific

project in question. This includes the facts known about the project as it currently stands. The second type of knowledge is the detailed representation of how different facts affect the overall development process of a project. These are the more general "rules" on what affects the quality of a software project. These rules are set up based on the knowledge described earlier from the data base and the software manager. They are used to describe all of the factors which affect a software project's quality and all the sub-factors that affect those factors, etc. For this reason this system of knowledge representation, which is unique to DEASEL, is called factor-based. Each rule in the factor-based representation scheme specifies a system and its factors (sub-systems) and the weight (strength of the relationship) each factor has on the system. Thus, between the specific assertions about the project and the general rules concerning software development within the SEL environment DEASEL can rate a project.

2.4 An Example Rule

To explain how this rating process works, here is an example rule from DEASEL's knowledge base:

```
The factors that affect Computer_Environment_Stability are
        1) Operating_System_Stability        .3
        2) Software_Tool_Stabiltiy           .2
        3) Hardware_Stability                .4
        4) Computer_Env_Proc_Stability       .1
```

The number associated with each factor is a weight, and the sum of the weights must always total one. This rule states that the four listed factors have an affect on the quality of the Computer_Environment_Stability. The rule's weights indicate that Hardware_Stability is the most important factor in the assessment of Computer_Environment_Stability, while Computer_Env_Proc_Stability is the least important factor. DEASEL uses the ratings of all four factors to determine a rating for Computer_Environment_Stability.

2.5 Deriving Conclusions

DEASEL's overall assessment process consists of trying to assign a rating to each of the quality indicators specified via the knowledge base. Obviously just answering the question "How is project X doing?" will not give the manager specific enough information about his project. Therefore, the knowledge base specifies the top level factors DEASEL should rate. Currently, the knowledge base has four such quality indicators: reliability, predictability, stability, and controlled development. Thus DEASEL actually gives information (a rating) on each of these four indicators which gives the manager an assessment of how his project is doing in these areas. In order

to rate these four factors DEASEL must find the rules which relate to these factors and assign a rating to these rules. That is, DEASEL reaches a conclusion on what it believes is the rating of these indicators. For DEASEL to do this it must first reach the conclusions on the factors which affect these indicators. Of course, these factors may have rules which specify their assessment, so this process continues until all of the necessary conclusions are reached.

DEASEL reaches conclusions in one of three ways:

1) The conclusion can be an assertion from the knowledge base.
2) DEASEL can infer the conclusion based on other conclusions and its rule base.
3) If both 1) and 2) fail, it can ask the user to supply the conclusion.

The three types of conclusions combine to allow DEASEL to make its assessment of the supplied quality indicators. The basic process is to first find a rule for one of the quality indicators then to resolve all of the conclusions necessary to reach a conclusion for that indicator. This process continues by reaching conclusions in each of the three ways, until all the conclusions are resolved.

To fully understand the rating process one must also understand how these conclusions are reached. A conclusion is reached when a rating has been assigned to a factor in the knowledge base. A rating is defined as a number between zero and one, the higher the rating the better the factor's quality. A rating of .5 would be average or normal. Note that the ratings always indicate quality, for example a rating of .7 for error rate as a factor would indicate a lower than normal error rate. In addition, every conclusion has an associated certainty. A certainty is the probability that the conclusion's rating is correct within some fixed delta. Currently, DEASEL sets delta at 0.1.

All three types of conclusions have both a rating and a certainty. Type 1 conclusions are really the assertions described earlier. Currently, the asssertions are entered by hand into the knowledge base. In the future this process will be automated and will be done by the DynaMITe tool, via the SEL data base. The certainties for these conclusions are generally very high (around .9) because the ratings are basically comparisons between real data and average or normal numbers. Conclusions of type 2 are computed using the following formulae:

$$\text{Rating} = \sum_{i=1}^{N} (\text{Rating of factor}(i) \times \text{Weight of factor}(i))$$

$$\text{Certainty} = \sum_{i=1}^{N} (\text{Certainty factor}(i) \times \text{Weight of factor}(i))$$

where n is the number of factors in the rule

Thus, a rule for a certain factor is given a conclusion by using these formulae to calculate its rating and certainty. The schema used here should look familiar to anyone with knowledge of

probability. In its typical application, however, each of the factors in the system being rated must be independent. In the complex and unfamiliar domain of software engineering, such an assumption may be incorrect. Our computations could therefore be slightly or grossly in error depending on how much the knowledge base violates this constraint. Future DEASEL knowledge engineers must keep this in mind when creating and modifying the rule base. Type 3 conclusions are necesssary when the system cannot use type 1 or type 2 conclusions. In order for the system to complete an assessment it must have conclusions for all the factors in the knowledge base. Since expert systems must deal with incomplete knowledge, whenever DEASEL cannot reach a conclusion for a factor it assumes a normal rating (.5) with a certainty of .2. Note that the .2 is the probability that the rating will be correct within + or - delta, which in effect makes for a meaningless conclusion. Whenever DEASEL is forced to do this, it makes a note to ask the user if the conclusion can be provided. Thus, the user can later provide the answers to questions about the incomplete knowledge. Once these questions are answered, DEASEL gives the rating supplied by the user a certainty of 1.0.

2.6 Current DEASEL Capabilities

The capabilities of the current DEASEL system include allowing the user to obtain an assessment of his project, if some assertions exist for that project. After the initial assessment is given the user has three options 1) asking for an explanantion, 2) answering questions about his project, and 3) playing what-if games. For any conclusion, the user can ask for an expalnantion of how the conclusion was reached. The explanation consists of the conclusions DEASEL reached about the factors of the original conclusion. That is, the user is able to ask DEASEL what caused it to reach any specific rating for any factor. This process can continue as the user asks for explanations of the factors previously reported on, and so on. Earlier we mentioned that DEASEL makes a note of type 3 conclusions. The user may opt to answer these questions as he wishes. He may also respond to the questions by indicating he does not know the answer. In this case, DEASEL maintains the meaningless conclusion reached earlier. Answering questions is encouraged because it leads to more certain conclusions. What-if games aid the manager in evaluating the effects of changes he may wish to make in his project. This process allows the user to enter controls into the system, by actually changing conclusions. That is, the user can see what will happen if he changes certain conclusions in the knowledge base. After changing one or more conclusions he can then reassess the project, to determine the affects of these changes. This is an important feature of the DEASEL system, because it allows the manager to determine how he might be able to improve his software project.

3.0 Summary

Although the current version of DEASEL does begin to attack the problem of project assessment, much more work is needed to make the system a useful tool. Three potential directions exist for future work: adding to and verifying the rule base, verifying the accuracy of the assessment process, and automating the creation of the assertion portion of the rule base. All of these areas will require time and effort to complete, but are necessary for successfully determining the validity of this project. Obviously, DEASEL is but an initial attempt at solving the problem of automating the process of assessing the state of an ongoing software project. DEASEL has, however, given some insight into the problem and ways to solve it. Hopefully this initial work will lead to techniques for solving the problem more completely.

REFERENCES

1. SEL-81-104, The Software Engineering Laboratory, D.N. Card, F.E. McGarry, G. Page, et al., February 1982

2. SEL-83-002, Measures and Metrics for Software Development, D.N. Card, F.E. McGarry, G. Page, et al., March 1984

3. SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freuberger and V.R. Basili, May 1979

4. McGarry, F.E., Valett, J., and Hall, D,. Measuring the Impact of Computer Resource Quality on the Software Development Process and Product, Proceedings of the Hawaiian International Conference on Systems Sciences, January 1985

5. SEL-85-001, Comparison of Software Verification Techniques, D. Card, R. Selby, F.E. McGarry, et al., April 1985

6. SEL-82-004, Collected Software Engineering Papers: Vol I, July 1982

7. SEL-83-003, Collected Software Engineering Papers: Vol II, November 1983

8. SEL-85-003, Collected Software Engineering Papers: Vol III, November 1985

9. Shortliffe, E.H., Computer-Based Medical Consultations: Mycin, Elsevier, North Holland, New York, 1986

THE VIEWGRAPH MATERIALS

for the

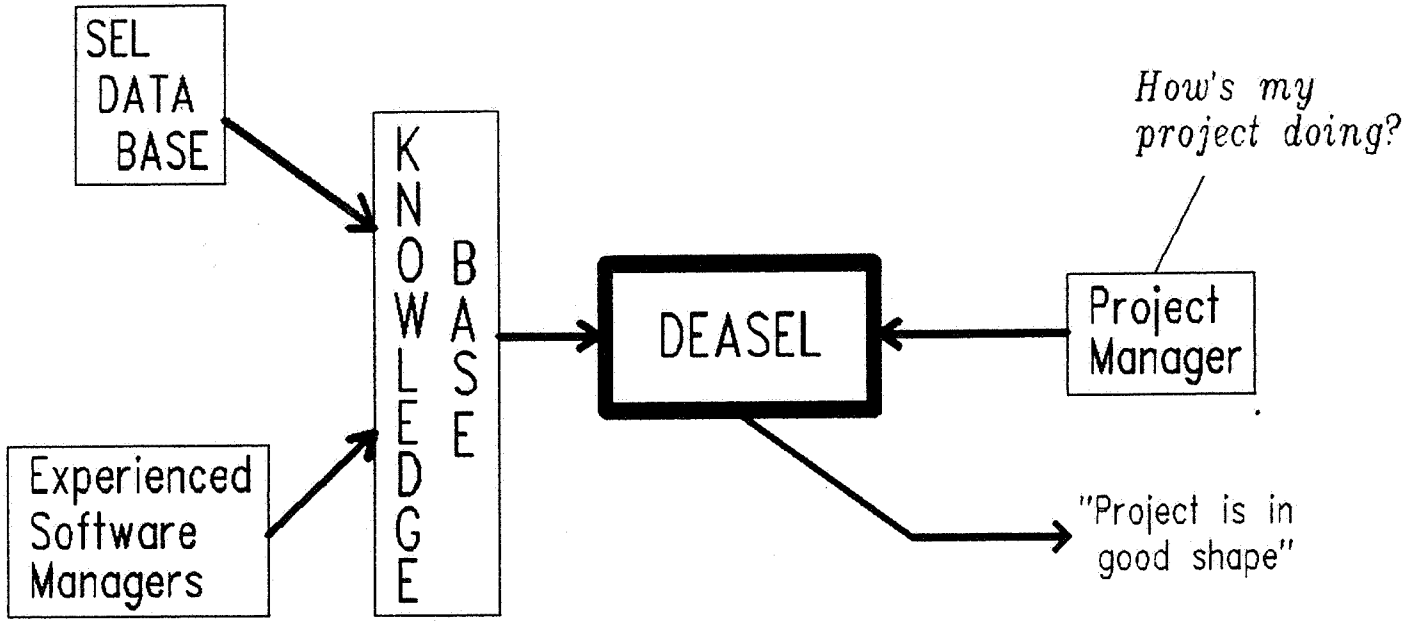
J. VALETT PRESENTATION FOLLOW

DEASEL : An Expert System for Software Engineering

Jon Valett
and
Andrew Raskin

NASA / GSFC

DEASEL



DEASEL is an experimental system integrating corporate memory and the knowledge of software managers to automatically analyze and assess a current software project.

KEY ISSUES

1. *Can information relevant to assessing a software project be extracted?*

2. *Can we easily represent this information?*

3. *Can this information be integrated into a useful software tool?*

COLLECTING KNOWLEDGE

From Corporate Memory

- Look for relationships

  eg.

  increased tool use $\longrightarrow$
  increased productivity

- Strength of relationships
  (weights)

- "A lot" of knowledge

From Software Manager

- Subjective information

- Strength of relationships
  (weights)

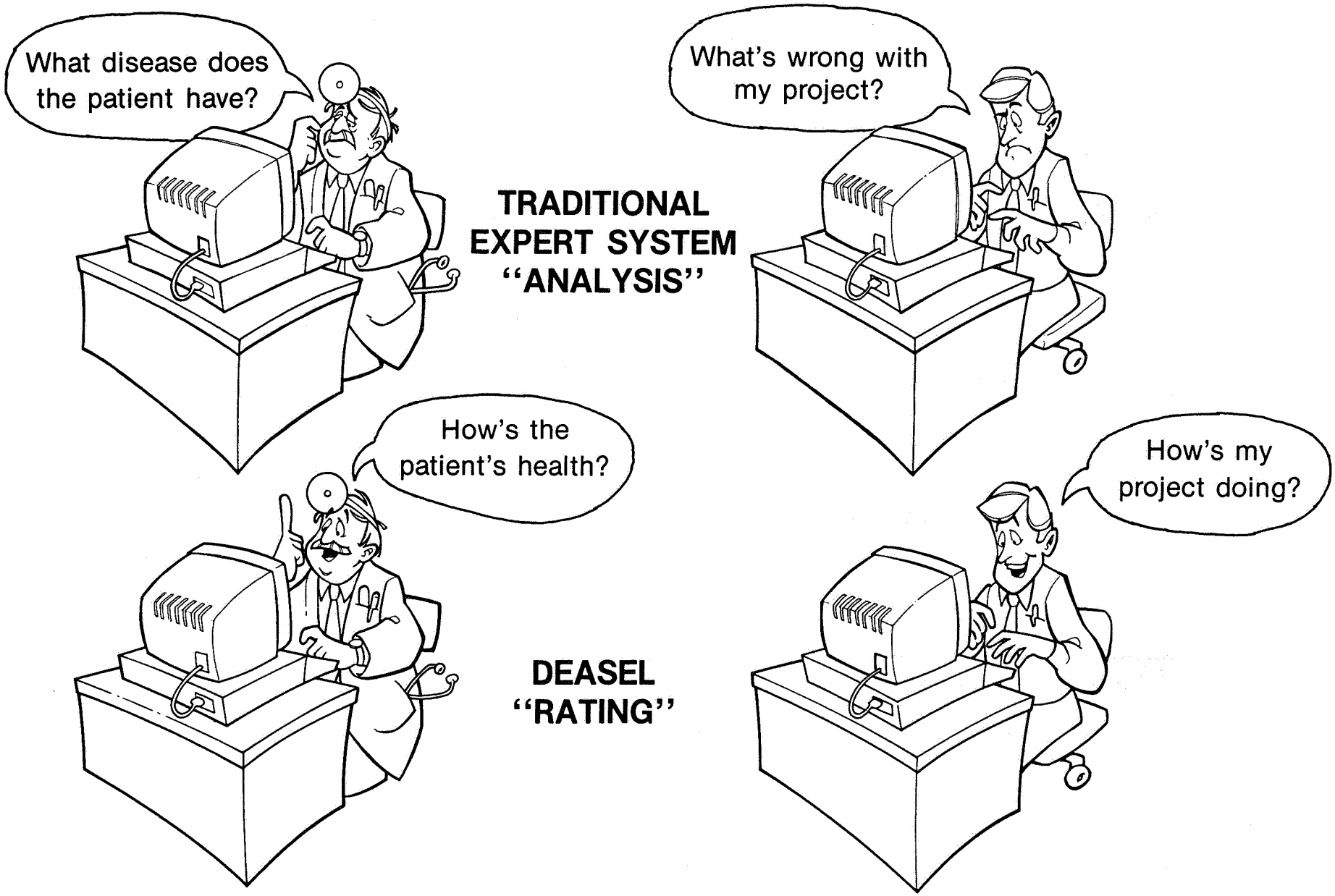
  eg.

  stability of a project affected by
  - no. of spec. changes minor importance
  - staffing stability very important
  - no. of design changes important

THE QUESTION

THE MEDICAL DOMAIN

THE SOFTWARE ENGINEERING DOMAIN



J. Valett
NASA/GSFC

TO ANSWER THE QUESTION . . .

Use

ASSERTIONS about the specific project

eg.

Change rate of code is above normal

Number of design changes is normal

And

RULES on software management

eg.

The main factors that influence reliability are

1. Change rate of code
2. Design stability

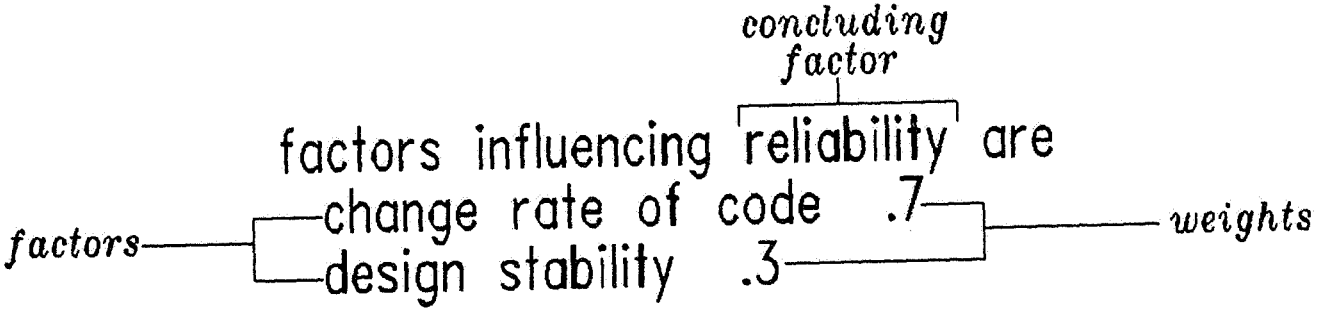
To

ASSESS the project on

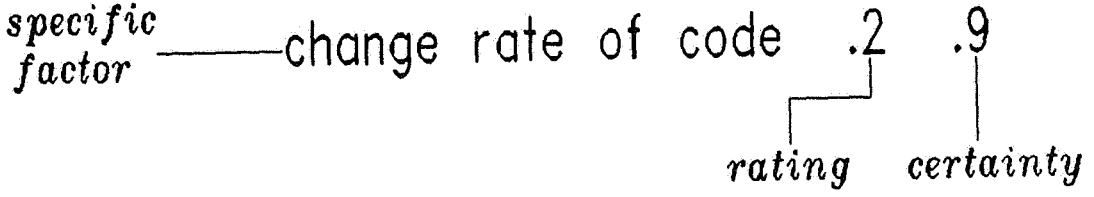
- Reliability
- Predictability
- Stability
- Controlled Development

THE KNOWLEDGE BASE

Rules – general



Assertions – specific project



THE RATING PROCESS

A Simple Example

Factors of Reliability are

Change Rate of Code .7
Design Stability .3

Key

RULES
ASSERTIONS
CALCULATIONS

THE RATING PROCESS

A Simple Example

Factors of Reliability are

Change Rate of Code .7 .14 .63 ←—— Change Rate of Code .2 .9
Design Stability .3

Factors of Design Stability are

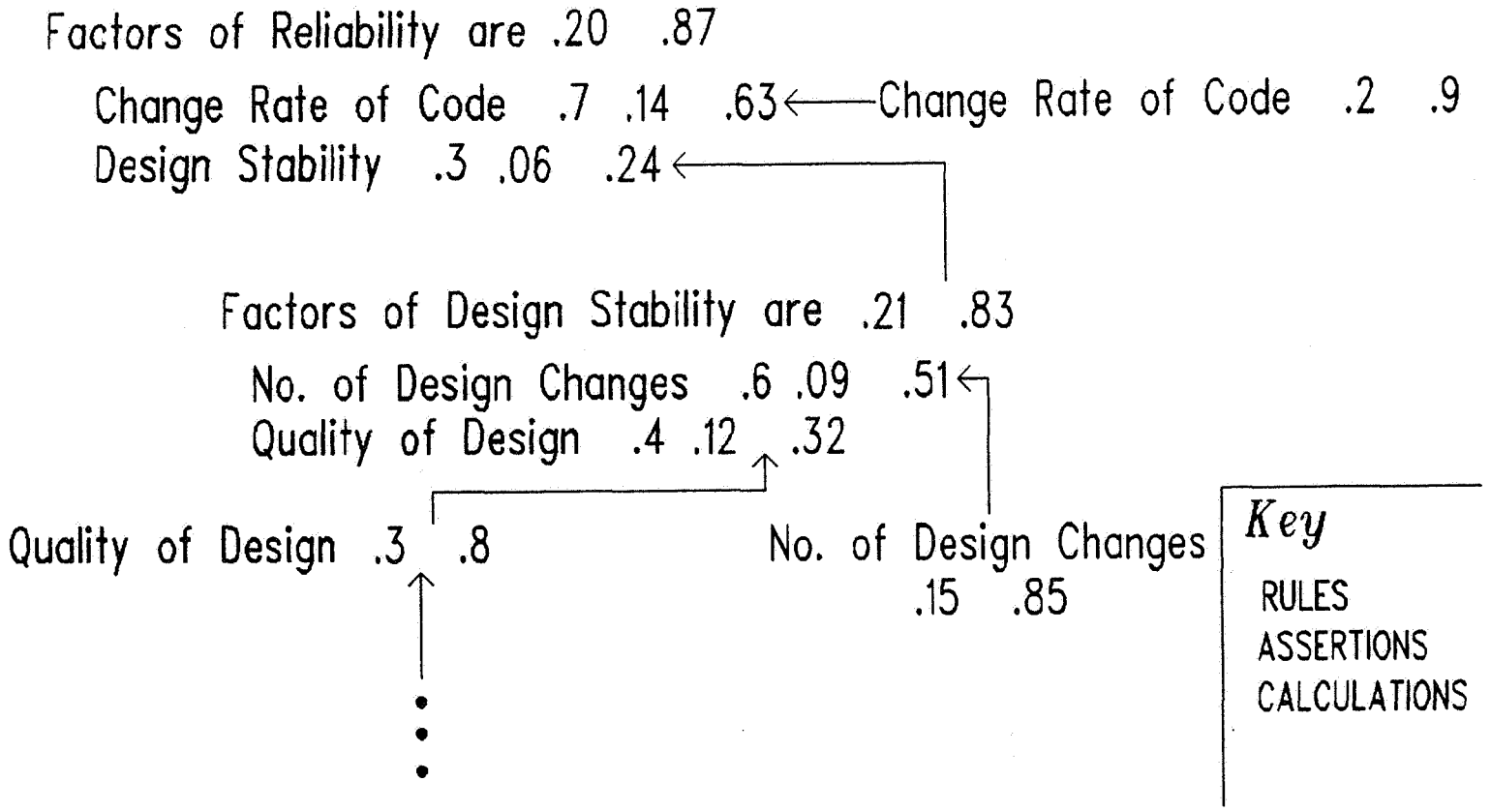
No. of Design Changes .6
Quality of Design .4

Key

RULES
ASSERTIONS
CALCULATIONS

THE RATING PROCESS

A Simple Example



CURRENT SYSTEM CAPABILITIES

- About 25 rules on code and unit test

Assess a Project *"How's my project doing?"* Reliability is Below Normal with High certainty	Explain a Rating *"Why is my reliability below normal?"* Change Rate is Very High and Design Stability is Normal
Answer Questions What is the quality of your project's design? *"Excellent"*	Play "What-if" Games *"What if my change rate was very low?"* Reliability would then be rated normal

PLANS

- *Add rules from other phases*
- *Validate existing rules*
- *Validate current assessment process*
- *Automate generation of assertions*

KEY ISSUES

1. *Can information relevant to assessing a software project be extracted?*

   - Yes, but is difficult and time consuming

2. *Can we easily represent this information?*

   - We think so, and hope our strategy is effective

3. *Can this information be integrated into a useful software tool?*

   - It can be integrated, but too early to determine usefulness