

7

Data Validation

Models are created by people and people make mistakes. For this reason it is necessary to validate both the contents and the utility of a model. As has been noted in previous chapters, models should be complete, correct and effective. Accuracy and completeness are necessary attributes of a correct model, while utility and maintainability are attributes of an effective model. To ensure that a model has these attributes a thorough review and verification procedure should be conducted. There are at least four good techniques to validate a model: team reviews, simulation, testing and direct application.

If a model is constructed in stages (e.g. iterative development) then the verification should also occur in stages. Models consisting of multiple levels of abstraction, such as one that has many detailed views, can also be verified in steps by testing how well the model is comprehended at each of its defined levels. As the model's detail emerges, the organizational power of the next higher abstraction layer is tested to see if all of the new data is still expressed effectively. If the key abstraction is poorly defined or ill-chosen, then some of the relevant data will not integrate smoothly; elements will seem "forced" to fit in an inappropriate category. An excellent model is one that has a certain elegance of form – all of the model elements are found in their proper place. A model has achieved its purpose if a reviewer need not ask questions of the modeler or subject expert that cannot be answered directly by the model itself.

Team Review

Reviews are one of the best techniques for confirming information contained in the model. Team reviews may be conducted in either a formal or informal setting, and either localized or distributed [9]. A formal review is established by an arranged meeting of modelers and reviewers. The goal of such meetings is usually to accept or reject part or all of a specific model's contents. Meetings of this nature are usually rather complex to schedule and so are often only held at critical project points. The meetings are conducted by a mediator and recorded by a scribe as was described in chapter 6 for group meetings, but here the mediator's role is to ensure that the model is thoroughly considered. At the

conclusion of the meeting the results and suggestions are published and specific tasks for modification or rework of the model are assigned.

As an alternative to a formal review with a moderator, models may also be reviewed by walkthrough. In this form of semi-formal review, the modeler takes the lead role in the meeting and presents his/her model to the rest of the team. The team members then ask questions or make comments, with the modeler incorporating the comments into the final model. In this form of informal review, which is more geared toward the model author than the team needs, the modeler is presenting the model so there is a bias toward the sections of the model in which the modeler is most interested. This subtle bias is not present in a formal review with a moderator, because the model is presented independently of the creator. This form of review is excellent, however, in getting direct feedback to the author, who can use the information to make corrections and additions.

Informal review sessions are usually conducted more frequently amongst the team members with the end goal of providing continuous feed-back to guide model development. These meetings may occur over lunch, in hallways, at work-stations, or other casual settings. Both informal and formal of review should be used as frequently as possible during the model development, and not just at the end phase of a project when the model is submitted for acceptance.

Another key aspect to reviews is that they are intended to *find* problems rather than fix them [9]. By this I mean that a review session is not a design session; faults that are found with the model should be noted and then corrected after the meeting finishes. This problem often occurs when technical reviews are conducted (such as an architectural model for software development, or the review of a device blueprint), rather than during specification reviews (e.g. system requirements). If the meeting participants begin to suggest fixes to the model, then the meeting will likely derail into a debate session rather than a review session. In a formal review it is the moderator's task to prevent this digression, however in informal reviews it is up to all of the participants to be responsible for policing their own behavior.

Team reviews may also be held in a distributed format. While this approach is significantly less efficient than a local group meeting, it may be the only avenue available to solicit input from a geographically dispersed team. Such is often the case with user interface reviews for software systems where the end users are located at sites separated from the development effort. In these instances the review team should be increased to 8-10 people compared to the 3-5 members considered optimal for a group review session [3]. The larger number of reviewers accounts for the reduced response rate due to oversight, neglect, conflicting priorities, etc. Reviews of this nature are also best conducted on a focused, critical section of the model to maximize the return value.

When conducting a review session it is often helpful to use an *active* style [4]. This involves the direct participation of all review members, rather than permitting a more passive role for the reviewers while the model presenter does most of the talking. This is accomplished by having the presenter/facilitator ask questions of the reviewers rather than the other way around. Some questions that can be posed are listed in Table 7-1.

Table 7-1. Review questions

Stakeholder Needs

<p>Have the correct stakeholders been identified as consumers of the model?</p> <p>Are there any individual or group concerns that have not been considered?</p> <p>Are there sufficient model views to meet each stakeholder's needs?</p> <p>Are the model views well organized to permit rapid access to information?</p> <p>Does the model form adequately satisfy the purpose of the model?</p>
<p>Model Construction</p>
<p>Are there sufficient mappings between views to guide reviewers from one view to another?</p> <p>Have all assumptions been noted and justified?</p> <p>Is there a glossary of terms and a consistent nomenclature?</p> <p>Has all unnecessary redundancy been removed?</p> <p>What are the strengths and weaknesses of the model?</p> <p>Is there a consistent organization of information (adherence to form)?</p>
<p>Model Contents</p>
<p>Is the model internally consistent (e.g. no conflicting/ambiguous information)?</p> <p>Is the information current and complete?</p> <p>Do the defined views have a pivot?</p> <p>Does the model have a consistent theme?</p> <p>Is the presented information accurate and contain no distortions?</p> <p>Has any relevant data been overlooked or incorrectly interpreted?</p>

By having questions asked of reviewers there is a strong encouragement to think and reflect upon the nature of the model. This in turn will encourage a thorough and critical examination of the model form and contents leading to a stronger and more useful final product. For more information on reviews and walkthroughs please refer to the works of Weinberg [5] and Weiger [9].

Simulation

Validation of a model via simulation involves the creation of a set of environmental conditions to test the theories and assumptions that underlie the model. There are many models that can be verified in such a manner. A familiar example would be an aircraft flight simulator. Here a pilot may safely practice emergency procedures without risk to life or property. A well designed flight simulator allows for training and experimentation of emergency preparedness procedures in a controlled, safe environment.

Other forms of simulation can be used when there are no dire consequences in the event of model failure. One such involves the validation of computer user interfaces (e.g. graphical user interface or GUI) via usability testing. Here a candidate set of interfaces is simulated by one of two forms – a “click-through” model that permits viewing of scripted behavior, and a “wire-frame” model that provides a view to illustrate the static layout of web-page elements [6]. The first form facilitates conducting of user-experience interviews permitting the system user to comment on system behavior. The second form is useful when discussing the value of design trade-offs of different screen arrangements. In either case the user interaction model is validated before the full system is constructed.

Another example of model validation through simulation can be found for models of business workflows. For these model types it is useful to simulate a new or existing workflow. In this form of simulation a worker is asked to perform all of the steps for a particular business operation without actually conducting (or committing) any real business activities. A captured workflow model, or one developed to improve a current business process can be tested and corrected without placing actual customer accounts or orders at risk.

Finally, a potential software design model can be simulated by prototype development. Prototypes are a time honored approach to prove a design assumption. If a design choice is questionable, for example the selection of a particular application server that may cost hundreds of thousands of dollars, then the possible choices can be tested by creating a small simulation (e.g. a minimally functional program of core, complex system flows) and hosting the prototype on different application server platforms. A series of performance tests can then be conducted to select the best choice for the current need (Figure 7-1).

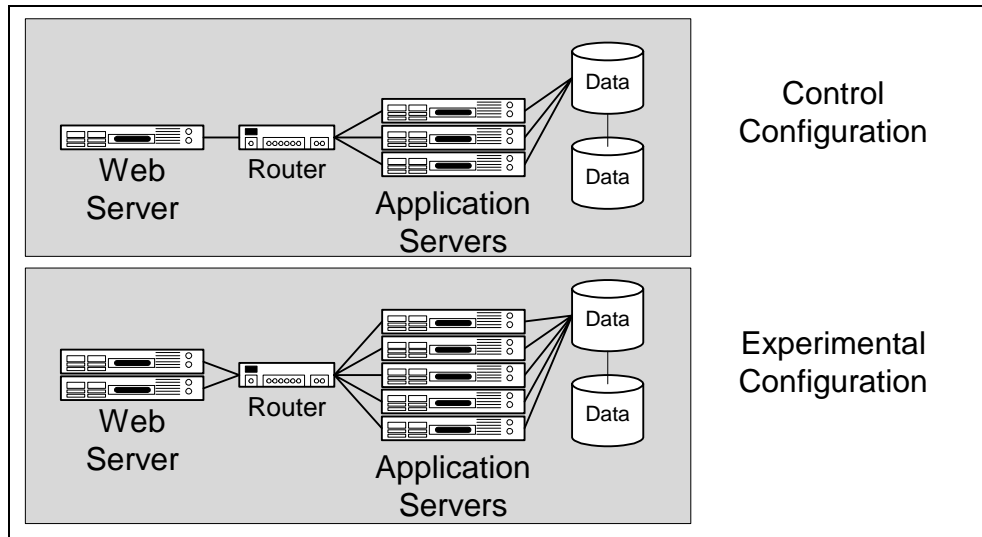


Figure 7-1. Performance Experiment for Web Server Configuration

Naturally, these results and conclusions are only reliable if the simulation is an accurate representation of reality. At first glance this would seem a paradox – we are using a model of reality to validate another model of reality! Fortunately, a simulation can be independently checked for accuracy either by inspection (e.g. review by domain experts), by mathematical proof (e.g. financial or engineering models), or by testing using a previously established and validated model as a control. If a suitable simulation is found and confirmed then this technique offers an excellent and inexpensive way to perform model validation.

Direct Application

One of the most conclusive ways that models may be verified is by *direct application* of the model. Needless-to-say, such models require the ability to be “executed” or otherwise exercised on an actual problem. This technique is also best utilized on a model that is built in stages illustrating key features of a particular solution. Such is the case with most models describing the construction of software systems. A best practice for system construction is to establish a base-line structure (also known as a reference or candidate architecture [4, 7, 8]). The model describing this structure is validated by implementation into a functional, albeit incomplete system. If the model fails to define an adequate solution it is still early enough in the project for corrective action.

Other model forms can be validated through execution, including business marketing models. The purpose of this form of model is to describe the likely response for a particular customer population to an advertising campaign. However, since a full campaign may involve many millions of dollars, a smaller “test market” may be selected to verify key assumptions regarding customer buying habits. If the test case meets expectations, then full funding may be approved for the project. Note that a group review theme (i.e. “focus groups”) may be combined with this approach to provide a more accurate result.

The main drawback to the direct execution approach for model verification is that not all models can be safely executed in a production (i.e. “live”) environment. A nuclear reactor control model would be far too dangerous to test on a real reactor system. For such models a form of simulation as described previously is a much less risky way to validate the model.

Test-Based Verification

Another time honored way to validate a model is via testing. Although similar in many ways to prototype development, validation through testing goes one step further by requiring more rigorous acceptance criteria. For test-based validation there can be *no* test cases that are in conflict with the model; the model must either be modified to account for the discrepancy or discarded as inadequate. This approach is often used in research to provide support for theoretical models. Since these models are constructed to be predictive, they can be directly tested by experimental procedures based on those predictions. Many scientific theories have failed to meet this high standard and as a result have been abandoned.

Testing is often used in software development to verify that the functionality has been delivered as specified in the requirements. In this case the testing is of the software model, rather than the requirements model, since it is the delivered functionality that is being verified against the original requirements. For this form of testing to be effective it is necessary that the source for the test model (e.g. requirements) be as accurate as possible. Typically, requirements are reviewed by one of the team review techniques as described above, while system testing is performed by direct use of the system in a suitable testing environment.

This form of verification also requires access to an unchanging base-line for comparison of the obtained results known as a *control or baseline*. Without internal controls it is often impossible to judge the effects of test manipulation. For example, consider a unit test where a developer is attempting to verify that a particular section of code performs as expected. If an error is found only one change should be made before the test is re-run. This is done to hold *variables constant* so that only one variable in the test is changing. This is a requirement for all forms of experimental testing. The reason is quite simple, the more variables that are allowed to influence the result the more complex becomes the interpretation. In the worst case scenario a proliferation of dependencies will make determination of a root-cause impossible.

Even with these limitations, experimental validation is the strongest of all system validation forms. Since the model must explain all of the test results there is no room in the model for error or ambiguity. Not even the direct execution of the model is as powerful a technique for ensuring that a model is robust and complete.

Summary

1. Model validation and verification is a necessary part of model construction in order to assure that the model is effective and correct. Well built models will embody an elegance and simplicity of form while maintaining a high degree of accuracy.

2. Model verification may be conducted in at least four ways: by review, by simulation, by direct application, and by test.
3. Reviews can be conducted in a formal or informal setting. Formal reviews are complex to arrange, but highly effective and require a smaller number of participants compared to distributed reviews. Informal reviews may be conducted with a minimum of preparation by walkthrough, and so permits rapid feedback to modelers. Active reviews are a very effective mechanism for group review sessions, where the moderator asks questions of the reviewers to solicit critical examination of the model.
4. Simulation offers an inexpensive and effective mechanism for the validation of model assumptions. Simulation permits the safe and controlled testing of a model using real-world parameters and conditions. However, the simulation must be verified as correct prior to its use as a test basis. Simulations have been successfully used in many environments where a direct test would be dangerous or impractical, and where model failure would have dire consequences.
5. Direct application of a model is a proven technique for model validation, but is best performed in stages. A best practice of software development is an iterative approach, where at each stage the development model is checked and verified to ensure correctness.
6. Testing represents the most rigorous form of model testing due to the high standard placed on the model to explain all results. In addition, this form of verification requires both a system that open to manipulation, and an set of requirements (controls) against which the test results can be compared.

Tips and Traps

Review sessions can often feel like a shark tank at feeding time; the attacks come fast and furious with little regard to the possibility of injury. While a critical review of a model's contents and form are expected, the manner in which those remarks are presented can make the difference between help and harm. The reviewers should be cautioned to focus on the work rather than the workers and to provide a solid reason for every comment. A response like "well I just don't like it" should only be allowed if the reviewer can support this position with details (e.g. color selection, organization categories, etc.).

Everyone likes to take pride in their work. This is especially true of models that are created to illustrate radical change. One of the most difficult tasks a modeler has is to disprove a dogmatic position that has become entrenched; the more the position is challenged, the stronger the adherents cling to the existing structure. When conducting verifications or validations of these kinds of models, be especially careful to present the evidence in a non-confrontational manner; after all there may be more than one possible interpretation for the model information. All conceivable views should be considered and debated before a final conclusion is reached.

Questions and Exercises

Question #1:

There are *at least* four verification techniques, can you identify others?

Question #2:

Are formal group reviews more effective than informal reviews? Why or why not?

Question #3:

Some distributed reviews take the form of a questionnaire. What are the advantages or disadvantages of this approach?

Question #4:

One of the major problems with simulations is the omission of critical real-world parameters. How might a simulation be constructed that is complete? Is it possible to have a simulation that is *completely* accurate to the real-world?

Question #5:

I have stated that test-based verification is more powerful than direct application of a model. Do you agree with this statement? What evidence can you provide for the opposite view?

Exercise #1: Critical Review

Select a recent technical article for review and analysis. Form a review group with a moderator and conduct a critical review session to illustrate the merits and failings of the article. When conducting the study consider the following questions about purpose, form and content:

- Has the author written the article with a consideration of the expected audience?
- Is the form of the article effective in meeting the needs of the reader?
- Has the information been presented in a clear and concise manner with examples?
- Are there critical omissions? Redundancy? Irrelevancy?

Prepare and present the groups findings to other groups for their assessment.

References

1. Tufte, E.R., *Visual Explanations*. 1997, Connecticut, USA: Graphic Press.
2. Tufte, E.R., *The visual display of quantitative information*. 2nd ed. 2001, Cheshire, Connecticut: Graphic Press.
3. Weinberg, G.M., *Quality Software Management: Volume 2, First Order Measurement*. 1993, New York: Dorset House Publishing.
4. Clements, P., et al., *Documenting Software Architectures*. 2003, Boston, USA: Addison-Wesley.
5. Freedman, D. and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*. 3rd ed. 1990, New York: Dorset House.
6. Meyhew, D., *Principals and Guidelines to Software Interface Design*. 1992, Englewood Cliffs, NJ: Prentice-Hall.
7. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. 1999, Reading, MA: Addison-Wesley.

8. Kruchten, P., *The Rational Unified Process, An Introduction*. second ed. 2000, Boston: Addison-Wesley.
9. Wiegers, Karl E., *Peer Reviews in Software: A Practical Guide* 2001, Boston: Addison-Wesley Professional.