EXPERT SYSTEM VERIFICATION AND VALIDATION STUDY

FINAL REPORT

Scott W. French
David Hamilton

International Business Machines Corporation

August 1992

Cooperative Agreement NCC 9-16
Research Activity No. AI.16

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division

Research Institute for Computing and Information Systems
University of Houston-Clear Lake

TECHNICAL REPORT
The University of Houston-Clear Lake established the Research Institute for Computing and Information Systems (RICIS) in 1986 to encourage the NASA Johnson Space Center (JSC) and local industry to actively support research in the computing and information sciences. As part of this endeavor, UHCL proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a continuing cooperative agreement with UHCL beginning in May 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The UHCL/RICIS mission is to conduct, coordinate, and disseminate research and professional level education in computing and information systems to serve the needs of the government, industry, community and academia. RICIS combines resources of UHCL and its gateway affiliates to research and develop materials, prototypes and publications on topics of mutual interest to its sponsors and researchers. Within UHCL, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business and Public Administration, Education, Human Sciences and Humanities, and Natural and Applied Sciences. RICIS also collaborates with industry in a companion program. This program is focused on serving the research and advanced development needs of industry.

Moreover, UHCL established relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research. For example, UHCL has entered into a special partnership with Texas A&M University to help oversee RICIS research and education programs, while other research organizations are involved via the "gateway" concept.

A major role of RICIS then is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. RICIS, working jointly with its sponsors, advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research and integrates technical results into the goals of UHCL, NASA/JSC and industry.
EXPERT SYSTEM VERIFICATION
AND
VALIDATION STUDY

FINAL REPORT
RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Scott W. French and David Hamilton of the International Business Machines Corporation. Dr. T. F. Leibfried, Jr. served as RICIS research coordinator.

Funding was provided by the Information Technology Division, Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Christopher Culbert, Chief, Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representative of the official policies, either express or implied, of UHCL, RICIS, NASA or the United States Government.
Preface

This report satisfies deliverable number 5 of RICIS contract #069. The purpose is to document results of the four Expert Systems Verification and Validation workshops taught during the period of March 1992 to August 1992.
Background

Five workshops on Verification and Validation (V&V) of Expert Systems (ES) were taught during this recent period of performance. Two key activities, previously performed under this contract, supported these recent workshops.

1. Survey of state-of-the-practice of V&V of ES
2. Development of workshop material and first class

The next two sections describe these activities in more detail.

Survey

The first activity involved performing an extensive survey of ES developers in order to answer several questions regarding the state-of-the-practice in V&V of ES. These questions related to the amount and type of V&V done and the successfulness of this V&V. The answers to these questions led us to two primary conclusions:

1. The state-of-the-practice in V&V of ES needs to be improved. This conclusion came from the lack of V&V techniques used, the fact that many systems did not meet expectations, and other results that indicated a relatively unstructured approach to V&V. This does not necessarily mean that the systems being developed were of poor quality or that the developers were not trying; in fact, many project spent a considerable amount of time doing V&V, in some cases up to 80% of the effort was spent on V&V activities. The results only indicate that many systems did not meet expectations and the V&V approach appeared inadequate in most cases.

2. A great improvement in ES V&V could be achieved by using existing V&V techniques and methods. That is, although ES V&V is still an active research area with many unsolved problems, there are still many existing methods and techniques that are just not being used. Furthermore, many of these methods and techniques are ones that are being applied to conventional (i.e., non ES) software. It seemed as though the large body of knowledge about general V&V was not being applied to ES.

These conclusions led us to believe that a class on V&V, with a special emphasis toward ES, would be of great value in improving the state-of-the-practice in ES V&V. Specifically, we wanted to inform developers about:

1. basic V&V concepts
2. the most useful V&V methods and techniques
3. differences between ES and conventional software
4. V&V techniques specifically for ES

Additionally, we hoped to provide some hands-on experience with these techniques.

Workshop Development

The next key activity involved developing an intensive hands-on workshop in V&V of ES. This activity

---

1 The full results of this survey has been delivered in a previous report under this contract.
involved surveying a large number of V&V techniques, conventional as well as ES specific ones.\footnote{In particular, we capitalized on previous work done by Lance Miller of Science Applications International Corporation in support of a contract to the Electrical Power Research Institute and the Nuclear Regulatory Commission.}

In addition to explaining the techniques, we showed how each technique could be applied on a sample problem. References were included in the workshop material, and cross referenced to techniques, so that students would know where to go to find additional information about each technique.

In addition to teaching specific techniques, we included an extensive amount of material on V&V concepts and how to develop a V&V plan for an ES project. We felt this material was necessary so that developers would be prepared to develop an orderly and structured approach to V&V. That is, they would have a process that supported the use of the specific techniques.

Finally, to provide hands-on experience, we developed a set of case study exercises. These exercises were to provide an opportunity for the students to apply all the material (concepts, techniques, and planning material) to a realistic problem.

**First Class and Review of Workshop**

Through our previous work, we had made contacts with a number of leading researchers in the area of ES V&V. We sent copies of our workshop material to these researchers and solicited their feedback and we updated the material based on their feedback. It is worth noting that we received many positive comments about the workshop material and our plans to teach it to ES developers. Based on this review, we felt confident that we had a solid and comprehensive set of course material.

The next review step was to present an overview of the material to a group of people knowledgeable in ES but not experts in ES V&V. This review gave us an opportunity to find out how well we could communicate the information to practitioners. A number of concerns were raised at this review. We addressed these concerns primarily by adding material to explain the role and purpose of V&V in ES development. This review was at the end of the previous phase of this contract (February, 1992) and constituted the first class.
Teaching the Workshop

The first full class was taught in March of 1992 to a select group of ES developers and software V&V professionals. The students were arranged by the NASA JSC Software Technology Branch (STB). This class lasted three full days.

At the end of this first class, we, along with Chris Culbert and Bebe Ly of the STB conducted an interactive discussion with the students on the value of the material and suggestions for improvement. It was felt that the material was very valuable and did need to be taught to others. For example, one student said that they had been doing V&V for six years (and had to learn things on the job), but still learned some new things in the class; “I just wish I had had this class six years ago”, they said.

We also learned many ways the material could be improved and the way it was presented could be improved. We learned that more time was needed for exercises and the order of presentation needed to be revised. Based on the recommendations and further analysis of the course material, an improved version of the workshop was produced. The new version had a new outline that allowed us to interleave lectures and the case study exercises, rather than having the students do all the exercises at the end of the class. We also added some videotaped demonstrations to further break up the lectures and make the class more interesting. This new version required four days to teach, instead of three.

To solicit students for additional classes, we developed a flyer that the STB circulated for us. We had scheduled three additional classes and received enough responses to hold each class.

The additional classes did not indicate a need to modify the existing material, though some new material was added. The new material was in the form of worksheets that walked the students, step by step, through applying several of the more advanced techniques. Examples of the kind of work performed by the students is in “Attachment B” on page 11.

Computer-Based Training Prototype

To support the advertisement of the workshop, IBM prototyped a multimedia introduction to the course. This presentation is centered around a problem situation that occurred during the Apollo 11 lunar landing. This problem is an intriguing example that can be used to discuss many V&V concepts and motivate people to attend the workshop.

The development of this prototype resulted in a new item being added to the contract. This item involves providing assistance to the STB in the development of an expanded multimedia computer-based training tool that illustrates concepts from the workshop.

Management Overview

One major goal of the workshop was to “sell” developers on the importance of V&V activities. In general we were very successful. Most students indicated a belief that the ideas taught were the “right way to go.” However, they also indicated that teaching their managers these same concepts would increase the likelihood of their success in applying them.

---

3 Refer to “Attachment C” on page 13 for a complete list of workshop attendees.
Therefore, we developed a two hour management overview. The management overview covers the same basic concepts of V&V without focusing on techniques. The idea is to tell managers why V&V is beneficial and how they can help their ES project developers as they try to apply the ideas taught in the workshop. At this point, however, the management overview has not been taught.

---

4 Refer to “Attachment D” on page 15 for a copy of the management overview.
Results

Overall, the workshops were very successful. The evaluation scores shown in "Results of Student Evaluations" on page 7 reflect that success. We also received many positive comments about the value of the material. However, it is still too early to detect any long-term impact of the class on current V&V practice. Activities are underway, however, to begin assessing this long-term impact.

As with any successful job, however, there is always room for improvement. The remainder of this report will focus primarily on areas where the workshop could be improved. These areas of improvement have been derived from informal discussions with the students, student evaluations and instructor observations.

Student Profile

In order to put student comments/evaluations into perspective, it is helpful to consider the profile of the students that attended each of the workshops. This section describes that student profile and how it impacted the successes and shortcomings of the workshop.

The ES V&V workshop was taught five times during the period of March 1992 to August 1992. The workshop in March was substantially different from the others held from May to August. This was the case for several reasons: the class material discussed fewer techniques, the class was shorter (three days instead of four), and the students were "hand-picked" to attend and give valuable feedback on the first class. Since the students were "hand-picked" for the March class, the attendance was very good (a consistent twenty-one each day of the class).

Many helpful suggestions were implemented as a result of the March workshop. In addition, a flyer was distributed by the Software Technology Branch (STB) to advertise the next four workshops (May through August). People wishing to attend the workshop were asked to either call IBM or the STB to register for the workshop. As we learned later, more time should have been spent screening people requesting to attend the class. The flyer indicated that the workshop would be of primary interest to those currently working ES problems. However, the background of the students actually attending the workshop reflected a broader scope of interest. The following statements reflect comments made to the instructor by students when describing their reason for attending the workshop:

• "I am here because my manager signed me up"
• "I am here to learn about ES; I do not care about V&V."
• "I am here to learn about V&V; I do not care about ES."
• "I am here to learn how to use ES to do V&V"
• "I do not know why I am here; I am a programmer. I do not do testing."

Unfortunately, the difficulty in finding the right students for the workshop meant that student attendance was not as good as the March workshop. With the exception of the June workshop, each of the workshops started with 15 or more students (our goal was to have at least 20). Attendance, however, steadily dwindled each day to the point where the workshop usually ended with only one-half to two-thirds of the original students still in attendance. It should be noted that many who stopped attending did so because of demands on the job. Yet, many left because they thought the workshop was going to be something that it was not. We have attempted to address this issue by asking each student to fill out a questionnaire, indicating what information they would most like to get out of the class. We then use this information to focus the lecture time on information of the most interest. However, the best solution in the future would be to better communicate the goals of the class to each prospective student, so we can be sure that the class meets the students needs and expectations.
An example of this latter point is evident in the number of students who had no idea what rule-base programming was all about. In the "Basic Concepts" section of the workshop, an example of a rule-based program and a procedural program are contrasted to illustrate key impacts of "AI" languages on V&V. In the March workshop the rule-based example was written in CLIPS. Roughly one-third of those students knew nothing about CLIPS. Therefore, we spent about an hour explaining the basics of CLIPS. To avoid this problem in future classes, the CLIPS examples were modified to use a English-like pseudo language (still rule-based). The idea was to remove the requirement to know CLIPS syntax. This only partially solved the problem, because anywhere from one-third to two-thirds of the workshop students in the next workshops did not know anything about rule-based programming or any AI language (in the case of the June workshop, there was only one student in the class that knew anything about rule-based programming). This increased the difficulty in contrasting ES and procedural V&V issues.

Some of the dissatisfaction with the material on techniques may be due to the state-of-the-art in ES V&V. As with software in general, there is no "silver bullet" as some would like to find. Instead, there are only techniques that require discipline and skill to apply. Students that worked hard to apply the techniques on the team exercises generally had a more positive response to the workshop. However, some students (as evidenced by some of the responses in "Attachment B" on page 11) expended minimal effort on the exercises. The reasons for this are not clear but may be due to the discipline required to apply many of the techniques. For example, one of the better techniques for evaluating rule-based programs is to generate "connectivity" matrices. This technique is good because it is relatively easy and could easily be automated with off-the-shelf matrix operations routines. But it is somewhat tedious and only finds anomalies that must be analyzed to determine if they are faults or not. Thus, it requires discipline and definitely is not a "silver bullet" solution. One student indicated that they had learned about matrices in school and did not want to have to use them anymore. We recognize that some of the techniques are somewhat unpleasant and not "fun" to use. It is for this reason that we are recommending that a significant effort be made to select the best techniques and then automate them.

We attempted to teach about techniques and also about developing a V&V approach. The information required to develop a V&V approach is much less than what is required to master the techniques. So we were able to thoroughly cover V&V planning, but could only introduce each technique. To thoroughly cover each technique would have required an order of magnitude more time, something on the order of a one-semester college course. We attempted to overcome this shortcoming by providing extensive references which were cross referenced to techniques. We also provided a set of worksheets that would help a student follow each technique in a step by step fashion. Still, many students would have rather covered fewer techniques, but covered them more in depth. We feel the best way to resolve this issue is to teach a class that covers one or two development methodologies that support V&V. The methodology would be composed of a few techniques that work together well and address a wide range of V&V.

Students who are technical project leads found the class very exciting and rewarding. This was because they did not need as much detailed information. Instead, they just needed a general overview of issues that they need to consider in leading their projects. Many of these students explicitly said, "I am going to start using this information on my project." Others indicated that this class will be very helpful as a "reference" when "planning for V&V" on their project.

On the other hand, students who came to get specific help on specific problems found the experience less rewarding because the workshop was not geared for that. For example, one particular student attended the class wishing to receive specific help in verifying a G2 rule-base. Rather than focusing on specific problems like this one, the workshop tried to address broader information about V&V concepts, issues and techniques. In some cases, though, the information was too broad, because many students had a stronger background in the concepts of V&V than we anticipated. This was not necessarily a big problem (i.e., some

---

3 The survey results indicated that most ES projects were built by people with little training in V&V (i.e., they were engineers, flight controllers, etc. — not programmers).
enjoyed the review of V&V concepts), but reflects on why some of the evaluation comments indicated there should be less emphasis on "basic concepts."

The workshop was definitely geared to students whose jobs are in the technical area of systems development. However, it soon became clear that many approaches being advocated in the class would be difficult for the student to do without significant commitment from their management. Many students brought this concern to the attention of the instructors. A good example of this is found in the team exercise solutions of "Attachment B" on page 11. One group indicated that they would not use the technique of inspections as part of their V&V approach. Their reason? "My management will not pay for inspections." This answer was given despite a thorough presentation on the benefits of inspections.

In summary, a better job of "screening" students and a better job of communicating the course intent should solve many of these problems. The objective of each workshop was clear: provide information on V&V and encourage people to use V&V techniques on ES (i.e., V&V of an ES can be done, despite what a student may have heard to the contrary). The purpose was not to make the student an expert on using each technique or to teach expert system programming techniques/languages. To this end, the class, at least so far, has been very successful.

### Results of Student Evaluations

<table>
<thead>
<tr>
<th>Rating (1 = High, 5 = Low)</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9</td>
<td>Quality of course material</td>
</tr>
<tr>
<td>1.6</td>
<td>Effectiveness of Instructors</td>
</tr>
<tr>
<td>2.1</td>
<td>Depth of course content</td>
</tr>
<tr>
<td>1.7</td>
<td>Degree to which the course met its stated objective</td>
</tr>
<tr>
<td>1.9</td>
<td>Effectiveness of the delivery method</td>
</tr>
<tr>
<td>2.0</td>
<td>Relevance of the course to my job requirements</td>
</tr>
<tr>
<td>1.9</td>
<td>Confidence in my ability to apply the course content to my job</td>
</tr>
<tr>
<td>2.0</td>
<td>Course exercises</td>
</tr>
<tr>
<td>2.3</td>
<td>Length of course</td>
</tr>
</tbody>
</table>

### Student Comments/Suggestions

The following items are a condensation of comments received from students at the bottom of the class evaluation form (see "Attachment A" on page 9).

- Too much standard s/w engineering basics; not enough knowledge-based related material
- Less emphasis on basics and more in depth exploration of techniques/guidelines
- Course should probably be longer to allow a bit more detail on techniques
- More abstract examples
- More examples of what an ES is. This hard for one to pick up with no previous experience in ES.
- Preferred a shorter class focusing on specific techniques in great detail
- Split the workshop into two pieces: basic concepts and techniques
- Course is more of an overview - more detail on techniques is needed

---

6 Students learned that, on average, inspections find roughly sixty percent of all errors.
• Course was too long and the team exercises too drawn out
• The "References" at the end of each section make the course excellent and very useful
• Improve the quality of video presentations
• Present instructor “solutions” to team exercises on the last day
• Better student attendance would have helped when applying techniques to team exercises
• Course should have had a real-world problem (using G2) with terminals for each student where students could apply specific analysis techniques

Observations/Recommendations of the Instructors

The following are suggestions for improving on the work done with the ES V&V workshop:
• Do a better job of “screening” people wishing to take the class
• Learn about G2 and have examples on how to apply V&V techniques to a G2 rule-base
• Advertise the workshop as two two-day workshops: “Basic Concepts” and “Techniques/Guidelines”
• Spend more time on techniques with improved examples and more detailed discussion
• Teach a management version of this workshop
• Fund future work in automating some of the better techniques; many of the techniques lack automation and therefore will be minimally used
• Improve the use of video during the workshop. For example, a video could be made illustrating (via some “role-play”) a knowledge acquisition process and application of knowledge correctness techniques to that process.
• Develop a “corollary” workshop on how to build verifiable ES (using techniques such as “cleanroom”)
• Spend some time discussing what to do when you already have the system done and that system was not built using a V&V approach
## CLASS EVALUATION

Your input helps us improve course offerings. 

**Course Name:** 

**Instructor:** 

**Your Name (Optional):** 

(circle one response)  

1 = Very Satisfied  2 = Satisfied  3 = Neither Satisfied  4 = Dissatisfied  5 = Very Dissatisfied  

### OVERALL 

1 2 3 4 5 

How satisfied are you with this educational experience? 

What influenced your answer? (circle one response per factor)  

1 = Very Positively  2 = Positively  3 = No Influence  4 = Negatively  5 = Very Negatively 

<table>
<thead>
<tr>
<th>Course Content/Delivery Factors</th>
<th>1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Degree to which the course met its stated objectives</td>
<td></td>
</tr>
<tr>
<td>b. Depth of the course content</td>
<td></td>
</tr>
<tr>
<td>c. Length of the course</td>
<td></td>
</tr>
<tr>
<td>d. Effectiveness of the instructor(s)</td>
<td></td>
</tr>
<tr>
<td>e. Course exercises/labs</td>
<td></td>
</tr>
<tr>
<td>f. Effectiveness of the delivery method (e.g., Classroom, Satellite, etc.)</td>
<td></td>
</tr>
<tr>
<td>g. Quality of the course materials</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job Application Factors</th>
<th>1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>h. Availability of the course when needed</td>
<td></td>
</tr>
<tr>
<td>i. Relevance of the course content to my job requirements</td>
<td></td>
</tr>
<tr>
<td>j. Confidence in my ability to apply the course content to my job</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Administrative/Environmental Factors</th>
<th>1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>k. Enrollment process</td>
<td></td>
</tr>
<tr>
<td>l. Class administrative services</td>
<td></td>
</tr>
<tr>
<td>m. Education facilities (e.g., classroom climate, equipment, etc.)</td>
<td></td>
</tr>
<tr>
<td>n. Travel (when applicable)</td>
<td></td>
</tr>
<tr>
<td>o. Lodging (when applicable)</td>
<td></td>
</tr>
</tbody>
</table>

**Other factors:**  

1 2 3 4 5 

p. Please specify 

Please provide any additional comments if appropriate. 

---

---
1. Define the "black box" view for your system.

```
room #     ---
room class ---
time of Request ---
Requested Time ---
Special Priority ---

lateness
Current Time
Room Priority
Call Time
Assigned

→ wake up call
```
2. Identify key terms from the problem description.

room &
room Class
Current Time
DEL Time
Cleared Time
Inspect Time Time of Req.
Spec. priority

3. Which of the following techniques would you use? Explain your answer.

- Prototyping — proof of concept, test user interface
- Competing Designs
- Independent V&V
- Inspections — catch 90% of errors early.
4. Do a very high level specification for your system using one of the following techniques:
   - Decision Table
   - Cause-Effect Graph
   - State Diagram

---

Lab

Early req — higher prior — 5 min
early late — prob 6 x 3 min
late > 20 min — Spec Prior — higher prior in class
late > 20 min — Spec prior — higher priority in all classes
Handout #8: Exercises on System Test Techniques

1. Define 1 or more "realistic" test cases for your team exercise.
   
   avg # rooms  make avg # requests
   
   see if wake up calls made correctly

2. Define some attributes of your system. Define 1 or more test cases based on those attributes.
3. Define 1 or more test cases that do "boundary value" testing.

Have 6 hi class rooms request 7am

6:55 6:57 7:00

4. Define 1 or more test cases that "stress" test the system.

- Have 7 hi class rooms req at 7am
- Have 11 med class rooms req. 7am
- Have 7 hi class + 11 med class req. 7am

Handout #8
5. Define the external interfaces to your system. Define 1 or more test cases to test those interfaces.

   1. reservationist I/O interface
   2. dial up machine

6. Define 1 or more test cases to test the system's performance.

   Ask expert for normal peak load & during busy season.

   Run simulation - shouldn't fill up computer, watch hardware load
7. For each question, indicate how the results of each test case will be analyzed (i.e., how you will know the answer is correct).

1. Realistic
2. Attrib.
3. Boundary - syst. gives expected results

4. Stress - syst. survives

5. User 1/0 - acceptable response

8. Did the problem description provide enough detail to adequately perform the tests from questions 1-6?

No - did not answer what to do if wakeup call not answered
9. Develop a "certification" test for your system.

10. Identify system "disasters" (i.e., things that should not happen). Explain how you will test your system for these "disasters".

- power failure
- phone syst. out of order
11. Will your project need the aid of an expert (provide rationale)? If so, indicate the kind of expert required and the type of analysis to be performed.

   experienced operator needed to verify day to day operation

12. Define 1 or more models to aid in your understanding of the system. Document each model.
Handout #9: Exercises on Unit/Integration Test Techniques

1. Pick an implementation approach for your problem. Based on this choice, would you use:
   - Coverage techniques
   - Interprocedural data-flow analysis

Implement Approach

I. log calls (user input)
II. schedule calls (rule for each call, rnclean, prior)
III. Make wakup calls (user wp-call machine, rules aht wakeup call)
IV. Reschedule calls (laten rules, rnclean,...)

2. Identify "part" of the system that may impact reliability (HINT: you may have to define what reliability is). Define 1 or more test cases to test those "parts".

Wake Up Calls must be made w/in acceptable time frame

Clock - run a pgm to req. time - Check agmt watch
3. Document 1 or more expected sequences of actions for your system.
   - rm 100 calls w/ req. wake-up @ 7am
   - pgm schedule wake-up call for 7am
   - pgm makes call @ 7am

   Add rm 200 (send data) req. 7am
   pgm shud resched to 6:59


   prototype for use I/O

   mutation testing for sched & resched call Times.
5. Exchange your work with another team. Study the problem. Ask yourself the following:

- Does their implementation match the problem?
- Are there any "holes" or inconsistencies in their descriptions?
- Did they pick the right techniques for their implementation approach?
Handout #10: Exercises on Static Test Techniques

1. Identify and define at least 1 "object" in your system (remember, objects consist of both data and operations on that data).

A room has

room class

desired wake up time

Time wake up was requested

assigned call time

spec. prior

Operation: Assign a call time

pre: desired wake up / class / spec room / time A

if \{all rooms\} call time != new room, desired time

Then new room. call time = ""

post: call time is assigned a value
2. Write a pre-condition and a post-condition for each operation on the object.

Opn: call a room

pre: assigned call Time, curr.Time

if curr.Time == call Time
then call room is confirmed

post: put call in queue

Opn: Make calls from call queue

pre: class, sp.pr, Time Req, curr.Time, call To

post: call in 15:30
3. Describe any general properties your "object" must satisfy. Discuss how you would analyze your "object"'s implementation to "prove" those properties are always satisfied.

\[ \text{assigned \ call \ \( Tnew \leftarrow curr\ Time + 20 \)} \]

\( \text{class \ must \ be \ (closed, \ true)} \)

\( \text{spec \ on \ must \ be \ (true, \ false)} \)

4. Pick at least one operation and defined some rules that implement its specification.

see above #2
5. Select one of the following techniques for analyzing these rules. Explain your answer.

- Petri Nets
- Directed Graphs
- Connectivity Matrices

\[
\begin{array}{ccc|c}
   & r_i & r_j & r_k \\
\hline
r_i & 0 & 1 & 1 \\
r_j & 1 & 1 & 1 \\
r_k & 1 & 1 & 1 \\
\end{array}
\]

6. Identify 1 "hazard" in your system. Build a fault tree for that "hazard".

[Diagram of a fault tree with labels:
- one room not called
- phone dead
- prompt delay
- too explicit
- too many
- same rag
- too many
- kitchen]
7. Identity 1 "fault" in your system. Build a fault tree for that "fault".

```
No calls made
  /
Comp sys down
  /
  dead
  /
  too many calls at once
```

```
Comp sys down
  /
  overload
  /
  too many calls at once
```

```
Phone sys down
```
Handout #7: Exercises on General Techniques

1. Define the "black box" view for your system.
2. Identify key terms from the problem description.

- **Objects** (monkey, key, pillow)
- **Goals** (unlock, eat, climb)
- **Attributes** (color, location)
- **Actions** (hold, walk, jump)

**Initial Conditions**

**Goals and actions are closely related.**

3. Which of the following techniques would you use? Explain your answer.

- Prototyping - Test concepts and baseful knowledge representation
- Competing Designs - Wider range of implementation ideas
- Independent V&V - No expert or criticality to achieve optimum path
- Inspections - Are we trying to model the 'average' monkey or a genius?
4. Do a very high level specification for your system using one of the following techniques:

- Decision Table
- Cause-Effect Graph
- State Diagram
1. Define 1 or more "realistic" test cases for your team exercise.

Any scenario that involves a 'legal' situation. Given a solvable initial state, assert a goal and observe the actions.

2. Define some attributes of your system. Define 1 or more test cases based on those attributes.

Location - can the monkey get from one location to another.
3. Define 1 or more test cases that do "boundary value" testing.

   PLACE OBSTACLES ON THE ROOM BOUNDARY TO MAKE SURE THE MONKEY STAYS IN THE ROOM.

4. Define 1 or more test cases that "stress" test the system.

   SEE ABOVE.

   COMPLEX SITUATIONS (MULTIPLE ITEMS ON THE KEY)
5. Define the external interfaces to your system. Define 1 or more test cases to test those interfaces.

NONE, OUTSIDE OF ESTABLISHING THE INITIAL CONDITION.

6. Define 1 or more test cases to test the system's performance.

WIDE VARIETY OF COMPLEXITY, RANGING FROM BANANAS NEXT TO MONKEY TO ALL OBSTACLES BETWEEN MONKEY AND BANANAS.
7. For each question, indicate how the results of each test case will be analyzed (i.e., how you will know the answer is correct).

   **Primarily - Achievement of Goal.**
   **Secondary - Path taken.**

8. Did the problem description provide enough detail to adequately perform the tests from questions 1-6?

   **Yes!**
9. Develop a "certification" test for your system.

Full stomach!

10. Identify system "disasters" (i.e., things that should not happen). Explain how you will test your system for these "disasters".

Key locked in the cabin that it opens, or key not in room.

(Any invalid initial state.

Limit on maximum number of actions taken.)
11. Will your project need the aid of an expert (provide rationale)? If so, indicate the kind of expert required and the type of analysis to be performed.

No.

12. Define 1 or more models to aid in your understanding of the system. Document each model.

Actions of a pet or young child.
Robot with limited knowledge.
Handout #9: Exercises on Unit/Integration Test Techniques

1. Pick an implementation approach for your problem. Based on this choice, would you use:
   - Coverage techniques
   - Interprocedural data-flow analysis

   1) Define actions
   2) Define states
   3) Define transitions between states
   4) Determine knowledge representation
   5) Combine transitions to achieve more complex goals.

2. Identify "part" of the system that may impact reliability (HINT: you may have to define what reliability is). Define 1 or more test cases to test those "parts".

   All parts are critical to achieve the goal.

   Resoulation of each action must be followed by selection of an appropriate next action.

   Any test case involving pairs of actions will help verify this capability.
3. Document 1 or more expected sequences of actions for your system.

MONKEY PICKS UP, HOLDS AND EATS BANANAS.

MONKEY PICKS UP AND HOLDS KEY, WANTS TO AND LOCKS CHEST.


PROTOTYPE EVALUATION - YES TO DETERMINE SOUNDNESS OF IMPLEMENTATION APPROACH AND KNOWLEDGE REPRESENTATION

MUTATION TESTING
2. Write a pre-condition and a post-condition for each operation on the object.

**OPERATION**  
**Hold**

**PRE**  
Goal to hold  
Monkey at same location  
Monkey holding nothing

**POST**  
New goal  
Monkey holding key
Handout #10: Exercises on Static Test Techniques

1. Identify and define at least 1 "object" in your system (remember, objects consist of both data and operations on that data).

   **KEY**

   **Data - Location, Color, Weight, Position**

   **Operation - Hold, Drop, Unlock**
3. Describe any general properties your "object" must satisfy. Discuss how you would analyze your "object"'s implementation to "prove" those properties are always satisfied.

**CAN BE HELD**

**UNLOCKS A PREDETERMINED CABINET.**

**STATE DIAGRAM.**

4. Pick at least one operation and defined some rules that implement its specification.

**HOLD - SEE PRE AND POST CONDITION.**
5. Select one of the following techniques for analyzing these rules. Explain your answer.
   - Petri Nets
   - Directed Graphs
   - Connectivity Matrices

6. Identify 1 "hazard" in your system. Build a fault tree for that "hazard".

   Key locked in the cabinet that it unlocks.
Monkeys and Bananas

7. Identify a "fault" in your system. Build a fault tree for that "fault".

- Red Chest
- Pillow
- Red Couch
- Red Key
- Ladder
- Blue Chest
- Blue Couch
- Green Coat
- Monkey
- Couch
- Banana

```
```

Handout #10
Handout #11: Exercises on Guidelines

1. Determine whether the recommended approach fits your problem. Identify additional issues that need to be considered.

   YES
   NONE

2. Generate a detailed development plan for your problem. Try to include specific milestones and how they will be achieved.

   Analyze - Study problem description, analyze holes.

   Design - Define actions, states, transitions between states, knowledge representation, model of actions, states and transitions.

   Prototype - Implement representation, completing simple tasks, refinement of representation.

   Implement - Combine simple tasks to form more complex tasks and achieve goals. Add control structure to determine n.e. goals and user interface.
3. Define specific development increments. Update your plan to reflect those increments.

See previous slide.

4. Consider the test cases you have selected so far. Are there any other kinds of testing you need to do? When will you know when to stop testing?

Stop testing when:

1) Each valid action can be performed. (Unit testing)

2) Each pair of actions can be performed and are correctly determined (Integration testing).

3) Valid/invalid initial conditions to see that goal is achieved. System degrades gracefully (System testing).
5. Build a high-level requirements outline for your system. How well does the original problem definition map to your outline?

Find Bananas
Get Bananas
Eat Bananas
Take a nap
Handout #7: Exercises on General Techniques

1. Define the "black box" view for your system.

```
7 user inputs

engine turns over
gas tank empty
dim headlights
clicking sound
spark
engine runs for short while
strong gas flow
```

Most likely failed object
2. Identify key terms from the problem description.
   - Gas gauge empty
   - Clicking sound
   - Gas strongly squirts
   - Check Spark
   - Headlights not shining brightly
   - Intermittent running
   - Engine turns over / doesn't turn over

3. Which of the following techniques would you use? Explain your answer.
   - Prototyping - user interface
   - Competing Designs
   - Independent V&V - test requirements are met
   - Inspections - code
4. Do a very high level specification for your system using one of the following techniques:

- Decision Table
- Cause-Effect Graph
- State Diagram

<table>
<thead>
<tr>
<th>Cause</th>
<th>Effect (failed item)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Gas</td>
<td>Gas</td>
</tr>
<tr>
<td>Turnover</td>
<td>Battery</td>
</tr>
<tr>
<td>No Turnover</td>
<td>Battery</td>
</tr>
<tr>
<td>dim lights</td>
<td>solenoid -- starter</td>
</tr>
<tr>
<td>no Turnover</td>
<td>clicking sound</td>
</tr>
<tr>
<td>Gas</td>
<td>spark plugs</td>
</tr>
<tr>
<td>Battery</td>
<td>distributor</td>
</tr>
<tr>
<td>turnover</td>
<td></td>
</tr>
<tr>
<td>spark</td>
<td></td>
</tr>
<tr>
<td>turnover</td>
<td></td>
</tr>
<tr>
<td>no spark</td>
<td></td>
</tr>
<tr>
<td>intermittent run</td>
<td>carburetor</td>
</tr>
<tr>
<td>fuel pumping</td>
<td></td>
</tr>
<tr>
<td>intermittent run</td>
<td>fuel pump</td>
</tr>
<tr>
<td>not pumping well</td>
<td></td>
</tr>
</tbody>
</table>
Handout #8: Exercises on System Test Techniques

1. Define 1 or more "realistic" test cases for your team exercise.
   
   Test - Gas tank/battery
   
   User ELF inputs for questions:
   
   Does engine turn over?
   
   If yes - is gas gauge reading empty? Yes - gas
   
   If no - do headlights shine brightly? Yes - next
   
   No - battery

2. Define some attributes of your system. Define 1 or more test cases based on those attributes.

   Reliability
   
   Use statistical record keeping
3. Define 1 or more test cases that do "boundary value" testing.

N/A

4. Define 1 or more test cases that "stress" test the system.

Test for cases with more than 1 problem.
5. Define the external interfaces to your system. Define 1 or more test cases to test those interfaces.

6. Define 1 or more test cases to test the system's performance.

Time from when final input is received to when output is displayed.
7. For each question, indicate how the results of each test case will be analyzed (i.e., how you will know the answer is correct).

Use expert for comparison or simulated failures (where we input failure to see if system can detect it).

8. Did the problem description provide enough detail to adequately perform the tests from questions 1-6?

Yes.
9. Develop a "certification" test for your system.

Simulate all possible failures and compare with expert's response.

10. Identify system "disasters" (i.e., things that should not happen). Explain how you will test your system for these "disasters".

- Distributor shake
- Fuel pump fire

Test these two conditions and insure adequate warnings are given to user when testing.
11. Will your project need the aid of an expert (provide rationale)? If so, indicate the kind of expert required and the type of analysis to be performed.

   Yes. Mechanic needs to verify system's logic.

12. Define 1 or more models to aid in your understanding of the system. Document each model.

   - Model user interface.
   - Cause-Effect diagram.
Handout #9: Exercises on Unit/Integration Test Techniques

1. Pick an implementation approach for your problem. Based on this choice, you would use:
   - Coverage techniques
   - Interprocedural data-flow analysis

Rule-based system.

Coverage techniques:
- Path coverage (cover all possible outcomes)
- Structural (verify each rule fires)

2. Identify "part" of the system that may impact reliability (HINT: you may have to define what reliability is). Define 1 or more test cases to test those "parts".

   External inputs.

   TC: Simulated inputs
3. Document 1 or more expected sequences of actions for your system.

1st: Gas/Battery
2nd: Starter/Solenoid
3rd: Spark plug/distributor
4th: Carburetor/Fuel Pump


Prototype the user interface.
5. Exchange your work with another team. Study the problem. Ask yourself the following:

- Does their implementation match the problem?
- Are there any “holes” or inconsistencies in their descriptions?
- Did they pick the right techniques for their implementation approach?
Handout #10: Exercises on Static Test Techniques

1. Identify and define at least 1 "object" in your system (remember, objects consist of both data and operations on that data).

   Ignition system consists of ignition and ignition.
2. Write a pre-condition and a post-condition for each operation on the object.

Precondition: Gas tank/battery and starter/solenoid must be "good."

Post-condition: Fuel/air to ignition system
3. Describe any general properties your "object" must satisfy. Discuss how you would analyze your "object"s implementation to "prove" those properties are always satisfied.

- Determining failed component.
- Analyze with simulated inputs.

4. Pick at least one operation and defined some rules that implement its specification. (Ignition System)

(defined spark-plug)
F1 (engine turns over)
F2 (engine dies)
F3 (distributor sparks)
⇒ (problem spark-plug) (conclusion reached)
R1

(defined distributor)
F4 (engine turns out)
F5 (engine dies)
⇒ (problem distributor) (conclusion reached)
R2
5. Select one of the following techniques for analyzing these rules. Explain your answer.
   - Petri Nets
   - Directed Graphs
   - Connectivity Matrices

6. Identify 1 "hazard" in your system. Build a fault tree for that "hazard".

   ![Fault Tree Diagram]

   - Shocking Yourself
   - Person Doesn't Wear Gloves
   - System Fails to Adequately Warn of Hazard
7. Identify 1 "fault" in your system. Build a fault tree for that "fault".
Handout #11: Exercises on Guidelines

1. Determine whether the recommended approach fits your problem. Identify additional issues that need to be considered.

   YES

2. Generate a detailed development plan for your problem. Try to include specific milestones and how they will be achieved.

   GATHER REQUIREMENTS
   AND EXPERT KNOWLEDGE
   - REQ. REV.
   PUT DESIGN TOGETHER
   - DES. REV.
   WRITE CODE
   - CODE REV.
   UNIT/INTEGRATE TEST
   I.V.V

   SOME
3. Define specific development increments. Update your plan to reflect those
increments.

do your minimum test

do each part of the
code and test
after each increment

4. Consider the test cases you have selected so far. Are there any other kinds of
testing you need to do? When will you know when to stop testing?

when all problems
have been proved
to be findable
5. Build a high-level requirements outline for your system. How well does the original problem definition map to your outline?

**Find the most likely part to have failed**

**Causes:**

- Battery
- Starter Motor
- Starter Solenoid
- Spark Plugs
- Distributor
- Carb.
- Gas Tank
- Fuel Pump
Handout #7: Exercises on General Techniques

1. Define the "black box" view for your system.

Diagram:
- System Clock
- Timer
- Biased Valve
- PC Voltage

Secondary Thrust
Primary Thrust
2. Identify key terms from the problem description.
   
   **Nominal launch sequence functions:**
   - Main Engine Ignition
   - Secondary Engine Ignition
   - Terminate direct ground link

   **Monitor error conditions:**
   - Engine Communication Failure
   - Engine Failure
   - PIC ignition voltage

3. Which of the following techniques would you use? Explain your answer.
   
   - Prototyping
   - Competing Designs
   - Independent V&V
   - Inspections

   Because there is strong detail for nominal cases.
4. Do a very high level specification for your system using one of the following techniques:

- Decision Table
- Cause-Effect Graph
- State Diagram
Handout #8: Exercises on System Test Techniques

1. Define 1 or more "realistic" test cases for your team exercise.

   Constrain Secondary Thrust to 85%.

   This should abort.

2. Define some attributes of your system. Define 1 or more test cases based on those attributes.

   Criticality:

   Comm word bit not reset at T+1 and T+2 should generate Hold message.
3. Define 1 or more test cases that do "boundary value" testing.

Structure Data to launch within 2.5 intervals at 4.0 intervals.

4. Define 1 or more test cases that "stress" test the system.

110% thrust on Secondary at T + 0.5
5. Define the external interfaces to your system. Define 1 or more test cases to test those interfaces.

Comm Word is external output all events should affect Comm Word.

6. Define 1 or more test cases to test the system's performance.

Initializing event (push button) should result in launch if no error conditions abort it.
7. For each question, indicate how the results of each test case will be analyzed (i.e., how you will know the answer is correct).

Cases are very specific, can use pass/fail.

8. Did the problem description provide enough detail to adequately perform the tests from questions 1-6?

Yes but current system description is not "terminal. Not all possible outcomes are defined (i.e., overflow..."
9. Develop a "certification" test for your system.

See answers 1, 2, and 6. Test nominal scenario.

10. Identify system "disasters" (i.e., things that should not happen). Explain how you will test your system for these "disasters".

Main should never proceed. Secondary. Secondary should never exceed 100% through
11. Will your project need the aid of an expert (provide rationale)? If so, indicate the kind of expert required and the type of analysis to be performed.

Yes, because boundary values are ill defined and assumptions need validation.

12. Define 1 or more models to aid in your understanding of the system. Document each model.

Build computer simulation.
Handout #9: Exercises on Unit/Integration Test Techniques

1. Pick an implementation approach for your problem. Based on this choice, would you use:
   - Coverage techniques
   - Interprocedural data-flow analysis

   **Expert System (less code) Combination because this is safety critical.**

2. Identify "part" of the system that may impact reliability (HINT: you may have to define what reliability is). Define 1 or more test cases to test those "parts".

   Clock
   Timer
   Command Word
3. Document 1 or more expected sequences of actions for your system.

Nominal ➔ launch


Prototype testing is possible because of detail describing Nominal versus error conditions.
5. Exchange your work with another team. Study the problem. Ask yourself the following:
   
   - Does their implementation match the problem?
   - Are there any "holes" or inconsistencies in their descriptions?
   - Did they pick the right techniques for their implementation approach?
Handout #10: Exercises on Static Test Techniques

1. Identify and define at least 1 "object" in your system (remember, objects consist of both data and operations on that data).

![Diagram]

If (Secondary ignition) and (Thrust ≥ 90%) and (Timer ≤ 2)
Then (Start main Engine)

If (Secondary ignition) and (Thrust < 90%) and (Time > 2)
Then (Abort)
2. Write a pre-condition and a post-condition for each operation on the object.

Pre-condition
Propellant bleed valve (closed)
and $\leq 2$ sec from initiate

Post-condition
main engine ready to fire
3. Describe any general properties your "object" must satisfy. Discuss how you would analyze your "object"'s implementation to "prove" those properties are always satisfied.

Tight constraints on time and minimum Thrust.

Boundary Value Testing.

4. Pick at least one operation and defined some rules that implement its specification.

\[
\text{If (Secondary Ignition)}
\]
\[
\text{If (Thrust < 90)}
\]
\[
\text{If (Time > 2) then (Abort)}
\]
\[
\text{Else If (Thrust > 90%)}
\]
\[
\text{If (Time > 2) then (Abort)}
\]
\[
\text{Else (Success)}
\]
5. Select one of the following techniques for analyzing these rules. Explain your answer.
   - Petri Nets
   - Directed Graphs
   - Connectivity Matrices

6. Identify 1 "hazard" in your system. Build a fault tree for that "hazard".
7. Identify 1 "fault" in your system. Build a fault tree for that "fault".

```
  thrust
     /\     /
    /  \   /  \
   /    \ /    \\
  bad bleed valve

  bad timer
    /\     /
   /  \   /  \
  premature check
```
Handout #11: Exercises on Guidelines

1. Determine whether the recommended approach fits your problem. Identify additional issues that need to be considered.

   Approach is alright. Initial analysis indicates constraints, rules, and facts are incomplete.
   Overthrust, over voltage, non-adjustable measurement sampling rates.

2. Generate a detailed development plan for your problem. Try to include specific milestones and how they will be achieved.

   Modular approach, top down
   Expert clarification
   Final requirements
   Design/code/unit test
   Integration Test
   Validation Test (w Expert)
   Delivery
3. Define specific development increments. Update your plan to reflect those increments.

detailed development, assume all final requirements are baseline.

Comm Word
bleed valve/pic
Timer
Secondary
main

code/test are included for each line.

4. Consider the test cases you have selected so far. Are there any other kinds of testing you need to do? When will you know when to stop testing?

Most of our test cases are exhaustive boundary value tests. All combinations that should lead to shutdown must be tested and verified to lead to shutdown. There must be tests for normal functions.
5. Build a high-level requirements outline for your system. How well does the original problem definition map to your outline?

All activities must be controlled by a main scheduling activity. That is, they are started by acknowledge start, terminate, and report final status to the scheduling activity. The scheduling activity will use high order bits on the comm status word to acknowledge/not-acknowledge activity start. All status communications will update a permanent record so that the sequence of activities and their status can be verified after the fact.
CAR WON'T START

Relevant Parts of the Black Box

Input
Symptom

Gas Gauge Reading (Low or High) is Empty
Headlights Dim or Bright
Engine Turn Over
Clicking Sound from Engine (Not Turn Over)
Sparks from Spark Plug
Gas Squirt from Fuel Line

Output
Detective Part

Battery
Starter Motor
Starter Solenoid
Spark Plugs
Distributor
Carburetor
Gas Tank
Fuel Pump
Gas gauge empty and gas tank empty → COM1

Engine turns over → COM2

Headlights don't shine brightly and battery dead → COM3 (engine does not turn over)

COM2 (gas tank not empty) → COM4

COM2 (battery not dead) and COM3 → STARTER

COM3 (engine does not turn over) and engine makes clicking sound → COM5

Engine does not make clicking sound → SPOOL

COM3 (engine does not turn over) and poll plug → SPARK PLUG

See sparks or do not see sparks → DISTRIBUTOR

COM4 (engine turns over) → COM5

Engine runs a while and carburetor → COM6

Fuel squirts from the fuel line when removed from the fuel pump

OR

No fuel squirts → FUEL PUMP
CAR WON'T START

1. Analyze Problem
   - Areas requiring prototyping
     - Process involving engine not starting
       - Starter/Solenoid
       - Ignition system
     - Checking the electrical sy.
     - Checking the fuel system
   - High Criticality Areas
     - Elec. Sys. Check
     - Fuel Sys. Check
   - Expertise
     - Expert mechanic for KA system initially relies on expert's knowledge, then requires some anal. by user for subsequent inpu
   - Scenarios
     - Statistical values w/ parts
     - Test cases exercising all causes and parts
     - Boundary value test cases
     - Multiple symptoms
   - ES Aspects
2. Initial Planning
   - Initial system based on KA
   - Next iteration will
     update knowledge
     develop interfaces
     mutation testing
   - Criticality
     initially small
     W/ Sys. checking - greater
   - System size
     small because of # of options (parts)
     primarily pattern matching
     U4U incorporated into develop.
   - Milestones
     module development
     module testing
     UI revision
     update Sys. as necessary
   - Resources
     expert (begin & end)
     developer (throughout)
     user (begin & end)

- Each module will be tested as developed and then when integrated

- Utilize variable/function labels common to both the code & docs.

- Experts involved w/ QA of Info, initially & during system updates.

- Static Testing

  Connectivity Matrix
4. Increments

- Screens/Windows
  Expert for info
  User for look & feel

- Files/Functions
  Test logic & paths

- Output
  Expert's verification
Launch Sequencing

Analyze Problem

Complexity occurs in Main Engine Ignition Command. This depends on Secondary engine thrust buildup and is time critical.

Communications with ground to monitor critical parameters.

The problem is to be solved by examining Secondary and Main engine interaction and building a simulation model of system to point in time and thrust parameters.

Operational scenarios follow start to point in time to progress.

Engine, value and thrust parameters will be at each critical point obtained. Check requirements to test results.
2. **Initial Planning**

- Build Engine model - Prototype
  - Fuel system
  - Time system
  - Thrust equations

- Develop simulation of engine model with ground control

- Update engine model with manufacture specification

- Verify each step with manufacture specification value

Milestones are:

- Requirement Anal.
- Preliminary Design
- Prototype
- Critical Design
- Model Testing
- Integrated Test
- Final Verification
- Ready for Training
Perform:

Cause-Effect Graphing
Boundary Testing
Defect and Disaster Testing
Hazard Analysis

Use Object Oriented Analysis

**ENGINE**
- Time
- Thrust
- C word
- Error C

**VALUE**
- Time
- Position

**Launch System**
- Time
- Ground Control
Problem:

1) System to Perform the NLSEF. Designed System.

Verify Status of Engine through Time up to Engine Start or Simulation

Develop Prototype to Test Initial Integrate, fir1 (costly)

Prototype - Full Simulation - Flight Hardware.

Realistic Considerations Data from Flight Hardware

- Must have engine module
- Concurrency

Key Terms:

- Sign
- Time Interval
- Parameters
- Error Processing
- MET
Prepare a space vehicle for lift off

- Use on-board functions
- Perform pre-launch activities & checks quickly

Develop a test approach
Consider Safety
Consequence of a wrong decision

<table>
<thead>
<tr>
<th>Resources</th>
<th>Implementation</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Astronauts - Start sequence - Clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Launch Experts - Engineers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Programmers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Schedulers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Ground Communication</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- ECC Input Count
- Astronauts Start Sequence
- Sensors Input
- Start Time
- Command Word
- Secondary Ensen Ignition (Initiate) 
- Propellants + Black
- Vehicle Escape Structure
- Main Engine Ignition
- Main Gasline Shutdown
- Final Response
Decision Table

Cause - Effect Graph:
State diagram

Rules

1) If main 20.0 and secondary asin has ignition between 2 and 2.3 sec, then issue command to "Main Engine Ignition".

2) If "propellant bleed valve closure" and "Secondary Engine Ignition" 2 seconds has not occurred and "propellant bleed valve closure has occurred" then issue "Secondary Engine Ignition".

3) If "Secondary Engine Ignition threshold > 80 %" command then record "Secondary Engine Ignition complete".

4) If "main engine ignition complete" then "terminate direct ground link".

5) If "terminate direct ground link command issued" and "prop link termination continued" then "Nominal Launch Sequence".

6) FF command to "Main Engine Initiator" occurred and "not more of 1 sec occurred" and "Engine Command Word BUT I reset" then note "Engine command failure".
Exercise IV-12

1) Implementation approach

- Use ADA

The Launch Segment problem involves functions occurring in proper order and testing parameters in order to determine The next step.

2) Parts of the system: elements of

The system consists of Valves, two types of Engines and Communication with ground. Each element must work reliably (no failures) for the total system to work. Engine Ignite. Each part will be tested separately. Subsystems are integrated together (System, Order in input Subsystem, System + Order + Testing).

3) Valve OPEN

Secondary Engine Ignite

Thrust + Time buildup

Main Engine Ignite + Throttle buildup

Ground Control Terminate

4) Prototypes in the way to go (can't distinguish between Engine by testing). No Model Test. Test to satisfy with manufacturing specifications.
7. IF "Engine Thrust Lower Than 900%" and command issued within 2 sec note one manual "Engine Failure".

If communication failure occurs

8. IF "Engine Bit 1 not reset" and "two consecutive commands issued" then "Inhibit Engine Communication Failure Recovery".

9. IF "In Engine Communication Failure mode" perform the following:

### Table 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Diagram

- Terminal Group Bit 1
- Not in 1 sec
- Bit 1
- Not set

Original page 10 of poor quality.
1. Realistic Test Case

- Secondary Engine Ignition within 2 sec of propellant bleed valve closure
  - Check failed - no start
  - Check propellant bleed valve open close
- Main Engine Ignite within time limit of Secondary Engine
- Reliability of return - always work
- Main and 2nd Secondary Engine
  - Position of blood valve
    - Time Indicator -von electron
- At earth top time bracket, the Engine Ignite
- Monitor indication of termination of ground link at engine ignition
  - Continue bleed valve
  - Engine pressure up to 90% in 5 sec
  - (Continue test of thrust to record)

5) Time pulse

Fuel System

Ground Control

User Monitor Control

6) Engine ignition with limits and need Thrust of 90% in 5 sec

7) Graphics

Secondary Engine

Main Engine

8) No Next menu White Box promot"in


10) Distra Mode 70% Thrust for start to check - secondary

  - Max limit 70% 
  - Shut down & get behind
  - Engine wall
  - (Only simulate disaster)

11) NW: Highly NW intensive, Need Manufacture Report to interpret multi.

Engine warm up 11
Pre Condition: OFF

Post Condition: 90% Thrott

Pre Condition: Engine Start

Post Condition: Engine Control word not - No Thrust

Pre Condition: No Error - Engine Control Word OK

Post Condition: Launch held, Engine Shutdown.
3. Command word 817 set
   Thrust 1 to 90% 2 seconds after start
   Thrust building to 90% during 2 seconds.

4. Ignition
   Rule 4.
   IF MET 0.0 and secondary engine has ignition between 2:30 and 3:00
   Then: Issue Command to "Main Engine Ignition"

Rule 5.
IF 2 seconds has not occurred and "propellant bleed valve closure has occurred" then issue "Secondary Engine Ignition"

5. Petri Net

<table>
<thead>
<tr>
<th>Fact</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fi</td>
<td>Rule 4: Time built up 2 seconds</td>
</tr>
<tr>
<td>Fn</td>
<td>2 = If propellant valve open - Secondary engine</td>
</tr>
<tr>
<td>F3</td>
<td>MET = 0.0</td>
</tr>
</tbody>
</table>

6. Hazard - Check for failed "...
   Now failures. Do...
   Only check all electric circuits...
   Varin. Clock works but...
   Launch Sequencer start...
Uncertainty Requirement

Launch Sequence - M<1/T = 0.0
Build up Scour Engine Thrust 80% Within 2-2.3 Seconds
Secondary Engine Main Engine Ignition xamarin down gravity links

Blood Valve Closing - within 2 seconds
Engine thrust > 80% in .5

Secondary Engine Ignition

Monitor

Engine Command

Thrust < 80% at time = 2 sec =
Secondary Engine Failure
Fault Tree for Clock failure

Launch with not enough Thrust

No Clock

Secondary engine

Main engine

Block value stuck closed
1. Propellant bleed valve release fuel into Secondary Engagement within two seconds after Propellant bleed valve closure the Secondary Engine shall Ignition.

2. Secondary Engine shall obtain 790% thrust.

3. After Secondary Engine thrust builds up to 790%.

4. Main Engine Command shall be issued after Secondary Engine thrust build up to 790%.

5. Main Engine Command shall be issued between 2 and 2.5 seconds after Secondary engine thrust builds up to 790%.

6. Ground link shall maintain contact with vehicle until main engine ignition.

7. Ground link shall terminate at main engine ignition and send the Engine Command Word Bit 10.

8. Ground link shall monitor Engine Command Word Bit when main engine ignition occur.

9. Ground shall arm PTC ignition voltage and monitor for.

10. Engine Communication Failure occur at Engine Command Bit 1 not reset on two consecutive commands.
As initially stated is (to yield a banana to a monkey). The information supplied by the customer suggested a much too complex system but discussion showed that the real requirement is to measure the relative intelligence of a monkey. The system constructed from the requirements just submitted was invalid provided that some performance time measurements were added.

**Life Cycle Model**

The life cycle model chosen for the development was the European Space Agency Model.

```
IF
((\$o-red) (holds-key) (ea-object-key))
```

...
Build prototype that assumes cooperative intelligent monkey.

*Note: Will need an expert to motivate the monkey to move.

A prologue trace could provide much the same information as the connectivity matrices in that all paths would be shown over whatever their length.
Monkey on top of pillar holding red key.

On floor at 72-2.

*Note: monkey on top of pillar holding red key.

Monkey unclimbed chest.

Monkey on top, pierce ladder, red keep.
Red chest unclimbed and ladder on top of red chest.

No specification for the weight of the ladder.

The monkey drops the ladder in the floor, monkey drops handles on monkey jump on floor.
Move monkey from 72-2.
Move monkey to T8-8 holding ladder.
Monkey climb on blue couch holding ladder.

Monkey climbs ladder on blue couch, monkey jumps on monkey climb on ladder to green chest floor.

Monkey moves back to T2-2 obtain red key.
Move back to T8-8.

Climbs on top of blue couch holding ladder, holding red keep.
Climbs back up ladder to green chest, opens it and gets the blue key.
Climbs back down to blue couch, jumps to floor.

Picks up blue keep, returns to T7-7, climbs ladder, opens blue chest and take banana.
1. The monkey is the object. The operation is to lower the monkey and banana to the same location.

Operation

More monkey to floor

Pre-condition

Monkey on top of green couch at T5-7

Post-condition

Monkey on top of floor at T5-7

More monkey to location T1-3

Pre-condition

Monkey on top of floor at T5-7

Post-condition

Monkey on top of floor at T1-3 holding the red key

Operation

Move monkey to T2-2

Operation

Move monkey to big pillar

Monkey on top of floor at T1-3

Post

Monkey on top of floor T2-2
The operation and its defined rules that implement its specification.
<table>
<thead>
<tr>
<th>Name</th>
<th>Org</th>
<th>Phone #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jerry Klumos</td>
<td>CAE-LINK</td>
<td>280 - 4559</td>
</tr>
<tr>
<td>Paul Sherman</td>
<td>CAE-LINK</td>
<td>280 - 4560</td>
</tr>
<tr>
<td>DANNY STAMRSS</td>
<td>INTECMETRICS</td>
<td>480-4101</td>
</tr>
<tr>
<td>Robert St John</td>
<td>Interactics</td>
<td>480-4101</td>
</tr>
<tr>
<td>TOM CALLAGHAN</td>
<td>Lockheed</td>
<td>333-7820</td>
</tr>
<tr>
<td>Ulysses Hunter</td>
<td>LOCAL</td>
<td>335-6733</td>
</tr>
<tr>
<td>Tyrau Pham</td>
<td>LOCAL</td>
<td>335-6573</td>
</tr>
<tr>
<td>Darrell Harlan</td>
<td>Unisys (Paramax)</td>
<td>282-4679</td>
</tr>
<tr>
<td>Debra Haddox</td>
<td>Paramax</td>
<td>282-4689</td>
</tr>
<tr>
<td>Glenn Lewis</td>
<td>Paramax</td>
<td>282-4672</td>
</tr>
<tr>
<td>Kahi Rogers</td>
<td>Paramax</td>
<td>282-4531</td>
</tr>
<tr>
<td>William Morris</td>
<td>McDonnell Douglas</td>
<td>283-1944</td>
</tr>
<tr>
<td>Patrick Kenney</td>
<td>McDonnell Douglas</td>
<td>283-4332</td>
</tr>
<tr>
<td>Name</td>
<td>Company</td>
<td>Phone #</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Al Brown</td>
<td>Lockheed</td>
<td>323-6853</td>
</tr>
<tr>
<td>Ray Runnels</td>
<td>UNISYS/ARAMAX</td>
<td>282-4665</td>
</tr>
<tr>
<td>Dan Drew</td>
<td>Paramax</td>
<td>282-3669</td>
</tr>
<tr>
<td>Ross Pettingar</td>
<td>Lockheed</td>
<td>323-7341</td>
</tr>
<tr>
<td>AL Carlton</td>
<td>CARLTON &amp; ASSOCIATES</td>
<td>780-4351</td>
</tr>
<tr>
<td>Leslie Ambrose</td>
<td>M器材</td>
<td>323-0494</td>
</tr>
<tr>
<td>Gian Ho</td>
<td>UNISYS</td>
<td>482-2941</td>
</tr>
<tr>
<td>Russell Sellers</td>
<td>GHG</td>
<td>488-8806</td>
</tr>
<tr>
<td>Dave Jennings</td>
<td>GHG</td>
<td>482-5206</td>
</tr>
<tr>
<td>Shalom Tifani</td>
<td></td>
<td>482-5206</td>
</tr>
<tr>
<td>Kim Bradt</td>
<td>Paramax</td>
<td>282-2534</td>
</tr>
<tr>
<td>Holly Gibbons</td>
<td>Intermetrics</td>
<td>480-4101</td>
</tr>
<tr>
<td>Michael Freeman</td>
<td>Loral</td>
<td>335-6775</td>
</tr>
<tr>
<td>Tom Thomsen</td>
<td>Loral</td>
<td>335-6714</td>
</tr>
<tr>
<td>Steve Kwan</td>
<td>Paramax</td>
<td>483-5234</td>
</tr>
<tr>
<td>Mark Pettingay</td>
<td></td>
<td>480-4101</td>
</tr>
<tr>
<td>Alice Lee</td>
<td>NASA/JSC</td>
<td>483-5234</td>
</tr>
<tr>
<td>Blake AFREERE</td>
<td>Intermetrics</td>
<td>480-4101</td>
</tr>
<tr>
<td>Richard Nguyen</td>
<td>Loral</td>
<td>335-2529</td>
</tr>
<tr>
<td>Ingrid Ku</td>
<td>CSC</td>
<td>280-2654</td>
</tr>
<tr>
<td>Name</td>
<td>Company</td>
<td>Phone #</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Tod Sandman</td>
<td>Lockheed</td>
<td>333-6712</td>
</tr>
<tr>
<td>Nina Tammi</td>
<td>Local</td>
<td>335-6911</td>
</tr>
<tr>
<td>Sunil Fazadan</td>
<td>MDSSC</td>
<td>283-4329</td>
</tr>
<tr>
<td>Ruth (First)</td>
<td>Lorac</td>
<td>335-6583</td>
</tr>
<tr>
<td>William Morris</td>
<td>MDSSC</td>
<td>283-1294</td>
</tr>
<tr>
<td>Arland Actkinson</td>
<td>Dual</td>
<td>486-1984</td>
</tr>
<tr>
<td>Satyanarayanan Seshan</td>
<td>GHG</td>
<td>468-5806</td>
</tr>
</tbody>
</table>
## Attendance List

<table>
<thead>
<tr>
<th>NAME</th>
<th>ORG</th>
<th>PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huy Nguyen</td>
<td>IBM</td>
<td>282-9340</td>
</tr>
<tr>
<td>Charles Greger</td>
<td>IBM</td>
<td>282-7228</td>
</tr>
<tr>
<td>Traci Brechmann</td>
<td>CSC</td>
<td>080-2335</td>
</tr>
<tr>
<td>Stephen Baarndstedt</td>
<td>CSC</td>
<td>280-2430</td>
</tr>
<tr>
<td>Lewis George Jr.</td>
<td>LORAL</td>
<td>335-6422</td>
</tr>
<tr>
<td>George Ward</td>
<td>LORAL</td>
<td>335-6507</td>
</tr>
<tr>
<td>Tom Jones</td>
<td>LORAL</td>
<td>335-6424</td>
</tr>
<tr>
<td>Ken Harris</td>
<td>LORAL</td>
<td>335-6829</td>
</tr>
<tr>
<td>Russ Walker</td>
<td>LORAL</td>
<td>335-6631</td>
</tr>
<tr>
<td>Celia Taylor</td>
<td>LORAL</td>
<td>335-6520</td>
</tr>
<tr>
<td>Chris Gilbert</td>
<td>NASA/PT4</td>
<td>483-8080</td>
</tr>
<tr>
<td>Joe Nielsen</td>
<td>GHG</td>
<td>488-8806</td>
</tr>
<tr>
<td>Bob McNenny</td>
<td>MDSSC</td>
<td>283-4325</td>
</tr>
<tr>
<td>Laurie Howard</td>
<td>NASA/ER23</td>
<td>244-5132</td>
</tr>
<tr>
<td>José Escobedo</td>
<td>GHG</td>
<td>488-8806</td>
</tr>
<tr>
<td>Don Culp</td>
<td>RSOC</td>
<td>483-0891</td>
</tr>
<tr>
<td>Sherry Land</td>
<td>JG4ER2</td>
<td>483-2064</td>
</tr>
<tr>
<td>Pratibha Buloor</td>
<td>UHCL</td>
<td>283-3724</td>
</tr>
<tr>
<td>Bob Ooss</td>
<td>LORAL</td>
<td>335-6550</td>
</tr>
<tr>
<td>Grace del Puerto</td>
<td>Unisys</td>
<td>282-4657</td>
</tr>
<tr>
<td>Name</td>
<td>Institution</td>
<td>Phone</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>David Hamilton</td>
<td>IBM</td>
<td>282-853</td>
</tr>
<tr>
<td>Tim Sarge</td>
<td>CSC/PAC</td>
<td>483-9246</td>
</tr>
<tr>
<td>Chris III</td>
<td>NASA/1074</td>
<td>483-17</td>
</tr>
<tr>
<td>Bela Li</td>
<td>NASA/PTY</td>
<td>483-805</td>
</tr>
<tr>
<td>Laci WAHL</td>
<td>NASA/ITT</td>
<td>380-75</td>
</tr>
<tr>
<td>Brian Dammell</td>
<td></td>
<td>380-75</td>
</tr>
<tr>
<td>Pratikha Bello</td>
<td>UHCL</td>
<td>283-3258</td>
</tr>
<tr>
<td>Dan Bowman</td>
<td>IBM</td>
<td>282-7665</td>
</tr>
<tr>
<td>Scott French</td>
<td>IBM</td>
<td>282-</td>
</tr>
</tbody>
</table>
Overview of Verification and Validation of Expert Systems

Authors:
Scott W. French
FRENCHS@HOUVMSCC.VNET.IBM.COM

David Hamilton
HAMILTON@HOUVMSCC.VNET.IBM.COM

IBM Corporation
3700 Bay Area Blvd.
Houston, TX 77058
Welcome

Welcome to the Overview on Verification and Validation (V&V) of Expert Systems

This Overview provides

- A view of the "state of the practice" in V&V of Expert Systems
- Insight into what we have taught developers about V&V
  » Many Techniques
- Guidelines for management's role in V&V
  » Developers stressed need for management involvement
"State of the Practice" in Expert Systems V&V

Significant research has been done in Expert Systems V&V

- Developed conceptual approaches
- Proposed various techniques

No significant case studies or field demonstrations

Research is based on many conjectures about how Expert Systems are built

- Expert Systems have no requirements
- Small V&V effort for Expert Systems compared to other software
- Testing an Expert System is hard
"State of the Practice" in Expert Systems V&V ...

Survey state-of-the-practice in ES V&V
- Determine real issues in V&V of ES
- Assess accuracy of conjectures
- Impact future work in V&V of ES

60+ projects were asked questions such as:
- V&V activities done, not done
- Issues that occur in practice
- Extent to which V&V impacts issues
- User views of quality/reliability

Caveats
- Results are not statistically valid
- Responses reflect opinion

1The survey did not attempt to assess whether a given system was good or bad. Our goal was to uncover issues encountered.
"State of the Practice" in Expert Systems V&V ...

Major difficulties developers experience when building Expert Systems

- Determining when to stop testing (63%)
- Validating knowledge acquired from the expert (60%)
- Managing the complexity of the problem being solved (40%)

Process used in building an Expert System

- 22% followed no life-cycle model
- 43% built *operational* prototypes
- 14% cited Configuration Management as an issue
  » One-on-one interviews indicated greater concern
"State of the Practice" in Expert Systems V&V ...

Methods used in verifying and validating an Expert System

- 57% operational systems had no requirements
- 52% used only one technique

Resulting quality of the Expert System

- Considered both developer and user perspective

<table>
<thead>
<tr>
<th>Item</th>
<th>Dev</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation is difficult</td>
<td>27%</td>
<td>100%</td>
</tr>
<tr>
<td>Less accurate than Expert</td>
<td>44%</td>
<td>80%</td>
</tr>
<tr>
<td>Did not meet expectations</td>
<td>49%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Addressing the Issues

Survey indicates ES projects need help

Two presentations address those needs:

- Management overview of V&V
- Developer instruction in doing V&V (Workshop on V&V of ES)

These presentations seek ...

Target

To help project members "pull together" for higher quality results ...

And avoid this!

Additional work still needed in many areas
The Apollo 11 Scenario
ES Developer Workshop on V&V

Workshop is taught over a 4 day period
Goal is to help developers do their job better

Many topics covered
- Theoretical basis for V&V
- Planning for V&V
- 47+ techniques covered
- Guidelines for doing V&V

Developers learn by

6/5/92
Key Points Developers Learn

What is Verification and Validation?

**Verification**: Am I building the product right

- Did I do what I was told to do
- Most techniques address this
- Therefore, generally easier to satisfy

**Validation**: Am I building the right product

- Was I told the right thing to do
- Few techniques help with this
- Therefore, generally more difficult to satisfy
Key Points Developers Learn ...

Focus on finding errors early

- Follow a "test as you go" approach
- Emphasize human analysis

Phases of Correctness Testing

"It is not uncommon to spend 30 to 50 percent of the ... cost ... for the verification effort using the after-the-fact approach"6
Key Points Developers Learn...

Spend more time analyzing the problem

- A complete understanding of the problem is never initially possible
- Will use prototyping to model their understanding of user needs
- **Translation**: "Pay me now or pay me later"

Insist on following a development life-cycle

- No more "operational" prototypes

"Building large programs is NOT like building small ones and software engineering is different from most other engineering disciplines"
Key Points Developers Learn ...

Plan for V&V

- Match implementation to problem
  » Solution-oriented vs. Technology-oriented

"Many problems that occur ... are the result of ... generating code without thinking about the design."6

Survey indicated 45% of Expert Systems mix conventional and procedural code.

- Identify required resources
  » Hardware, Software, expertise, ...
  » All impact the project's feasibility
  » Should be done early instead of later
Key Points Developers Learn ...

Plan for V&V ...

- Prioritizing tasks (e.g., do the critical things first)

"A comprehensive test management approach recognizes the differences in objectives and strategies of different types of testing."¹⁰

- Remembering that the system will have to be maintained

For every dollar spent in development, two dollars is spent on maintenance.²
Key Points Developers Learn ...

Build a description of the problem

- Have something to test against

"If an expert system starts with vague objectives, some may conclude that it doesn't matter what the eventual system does, because anything is better than nothing."\(^4\)

- Must be a "crisp" definition

"Knowledge-based systems have a greater likelihood of succeeding - and, in a sense, of being valid - when they address a narrowly defined problem."\(^7\)
Key Points Developers Learn...

Focus on doing "smarter" testing

- Matching the right technique to the right problem in the right situation

The Verification Puzzle

- Develops an understanding of why the system is correct
- Know when to stop testing
Key Points Developers Learn ...

Expect the system to work

- Confidence in applying techniques
- Confidence that the appropriate issues have been considered
- Confidence that the problem can be solved

"A good programmer understands what his program is supposed to do and why he expects his program to do it."\(^5\)

"The difficulty with low expectations is that they become self-fulfilling."\(^5\)
Guidelines

The following are guidelines to be followed when applying V&V to your project.

You may want to do these yourself or delegate them to members of the development team.

Just make sure they happen.

Guidelines apply to the following steps:

1. Project Management
2. Planning
3. Requirements
4. Design
5. Test
Guidelines

Project Management

- Include V&V as part of the cost of developing software
  - Spread throughout the development cycle
  - Not all at the end

- Allocate resources for V&V
  - Be prepared to postpone a project if the resources are not available
  - For expert systems, you will need the expert's time for V&V

- Make sure you follow a systematic development approach
  - Best bet is to use a life-cycle model that includes major testing phases
  - Focus on a "test as you go" approach
Guidelines ...

Project Management ...

- Make sure your plan is based on the system's characteristics
  - What problem is to be solved
  - Complexity of that problem
  - Effort required to generate a solution
  - Types of correctness that matter
- Include prototyping to help validate understanding of the user's needs

"The only question is whether you or your customer will discover them (errors)"8

"... there is now less excuse than ever for not involving users early on ..."1

» Separate prototyping from complete system development
Guidelines ...

Problem Analysis

- Narrow the scope of the problem as much as possible
  » Better to have a system that solves one problem really well than a system that solves many problems poorly

- Do not force the solution to be an expert system

Requirements

- Write Requirements
  - Impossible to prove anything about the system without it
    » Consider all kinds of correctness
  
- Make sure that (at a minimum) the expected use of the system is defined
Guidelines...

Design

- Pick methods that make static analysis easier
  » Easier and less costly system test
- Map requirements to design
  » Helps decide if anything is missing
- Pick a reasonable design notation and stick with it

"... conceptual integrity is the most important consideration in system design. It is better to have a system ... reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas."³
Guidelines ...

Test

- Consider using an independent organization for final V&V
  » A "fresh look" can often find additional errors

- Use test techniques that find errors as early as possible

- Do not forget to do regression test
  » Easier when following a "test as you go" approach

- Prioritize the test approach
  » Focus on critical functions first
  » Test others later as resources permit
Conclusion

Doing the right things will produce the right results

- Reduced Maintenance and System Test cost
- More "up-front" work
- Identifying resources, milestones, techniques, and objectives
- Realistic schedules and cost (knowing when you are done and why it is right)
- Systems that work and solve the user's problem
- No "operational" prototypes
- Requirements Defined
Conclusion ...

Just remember

- **Software engineering is not easy**
  
  "Software engineering is harder than you think. I cannot emphasize strongly enough how true this statement is." \(^9\)

- **Expert Systems are software**

  "AI entails massive software engineering." \(^9\)

- They do not work "like magic"
References


References ...

