An Assessment of SEES Based on Operational Experiences*

A Final Report
to
Kelly Hayhurst and Mike Holloway
NASA Langley Research Center

by

James D. Arthur
Markus K. Gröner

10 July 1996

*This work is funded by NASA - Langley Research Center under contract NAS1-19610, Task 17

EXECUTIVE SUMMARY

During the Fall of 1995 a modified form of SEES (Software Engineering Evaluation System), denoted SEES', was defined and used in a study to examine the value added of Independent Verification and Validation*. A follow-on study, and topic of this report, focuses on an assessment of SEES based on operational experiences gained from the 1995 study. The report partitions it findings relative to phases of the software development process. Those findings are outlined below.

Requirements Analysis: During the requirements analysis phase SEES employs two major forms and defined activities for completing those forms: the Software Requirements List (SRL) and the Requirements Analysis Diagram (RAD). The SRL and attendant activities focuses on the examination of all requirements, while RAD activities emphasize assessing only functional correctness. Comparatively, RAD activities account for 60% of analysis effort and SRL activities for 40% of the expended effort. Because both sets of activities addresses functional requirements, substantial duplication of effort is evident. We conjecture that if only RAD activities are used, the effectiveness of error detection will be at 40%; if only SRL activities are employed, error detection effectiveness could approach 95%. Clearly, there exists an imbalance in effort and error detection effectiveness.

To address this imbalance we offer two alternatives:

(1) Make RAD activities optional and, perhaps, expand functional assessment activities associated with the SRL, or
(2) Redefine the RAD and SRL activities to make them disjoint.

Our experiences also reveal a crucial need to automate the recording of information generated during SRL and RAD activities and to ensure that such activities are applied to only mature sets of requirements.

---

*Arthur J.D., *et.al.*, "Report on Quasi-Experiment for Evaluating SEES - (Requirements Analysis Through Acceptance Testing) Final Report (Revised)," 15 January 1996.

Preliminary and Detailed Design Verification: During both the preliminary and detailed design verification, SEES employs the Fagan Inspection process. We believe that the preliminary and detailed design verification effort the Fagan approach is ideal and has much to offer. Moreover, SEES provides a good set of checklists that help focus the verification activity.

During the verification process, however, SEES excludes all development personnel from participating in associated activities. We found that having a development team member lead the *overview session* of the inspection process is beneficial. In particular, the development team member is in a better position to present system design more accurately and to provide insights into the decisions leading to that design.

Like SEES wee, too, advocate the exclusion of development team members form participating in the inspection session. Inclusion leads to "defensive posturing" and rebuttals when potential defects are identified.

Additionally, experiences during preliminary and detail verification leads to the following observations that significantly enhanced our verification capabilities.

(1) Inspectors must focus on ensuring the consistent use of terms and parameters in design documents, and
(2) The partitioning of inspection segments must be defined such that each interface between components lies (a) within the scope on a single inspector and (b) that same interface lies on the boundary between two other inspectors. This approach increases the effectiveness of interface assessment.

As a final observation we note that significant overlap of information exists in the Inspection Defect List and the DTR forms. Again, automation can assist in minimizing the necessity of detail duplication and reduce the possibility of information inconsistencies.

Code Inspection and Verification: Code inspection and verification is minimal in SEES. More specifically, SEES does not require the verification of code back to the detail design. Moreover, assessment is performed using only an automated code analyzer which focuses primarily on quality characteristics, and to a much lesser degree, adherence to specified coding standards and practices. Consequently, no direct assessment of implementation details is performed.

While such an approach is cost effective, no process exists to determine if mandated algorithms have been implemented, nor can algorithmic correctness and efficiency be judged (at this phase of the development process).

During our study, the IV&V group did perform code verification and found errors that were not discovered by the development group until Integration and Testing (I&T).

Integration and Testing: Although our approach to I&T differed somewhat from that defined by SEES, our experiences are still relative. In general, we find SEES multi-step approach to I&T to be systematic and rigorous. It is well-defined and guided by the Software Test Plan (STP) and Description (STD).

The one concern is that testing is guided almost *exclusively* by the dictates of the STP and STD - no provisions appear to exist to encourage nor provide any latitude for independent testing by the V&V personnel.

A Summary of Major Observations and Recommendations:

(1) The SEES approach to V&V is well-defined and provides extensive guidance in the necessary activities underlying each phase of the V&V process.

(2) During requirements analysis, the activities associated with the generation of the SRL and RADs overlap significantly. This overlap has a detrimental impact on effectiveness and productivity.

(3) The use of automated recording and reporting can contribute significantly to the effectiveness of the process and to enhancing personnel productivity. Benefits can be readily recognized by examining the effort required to record similar data among the SRL, RADs, DTRs and ISR.

(4) There is no stated process for verification between code and the detailed design documents.

Perhaps the rationale behind this omission is that functional requirements are assessed through examination of the STD and its guidance during FQT, FCA and PCA. Clearly there is a cost benefit gained by omitting this step, but there are drawbacks, too.

(5) The correction of an error in one development phase might necessitate changes in a previous phases. Currently, SEES has no stated provision for tracking or confirming the implementation of such changes.

The Detailed Critique

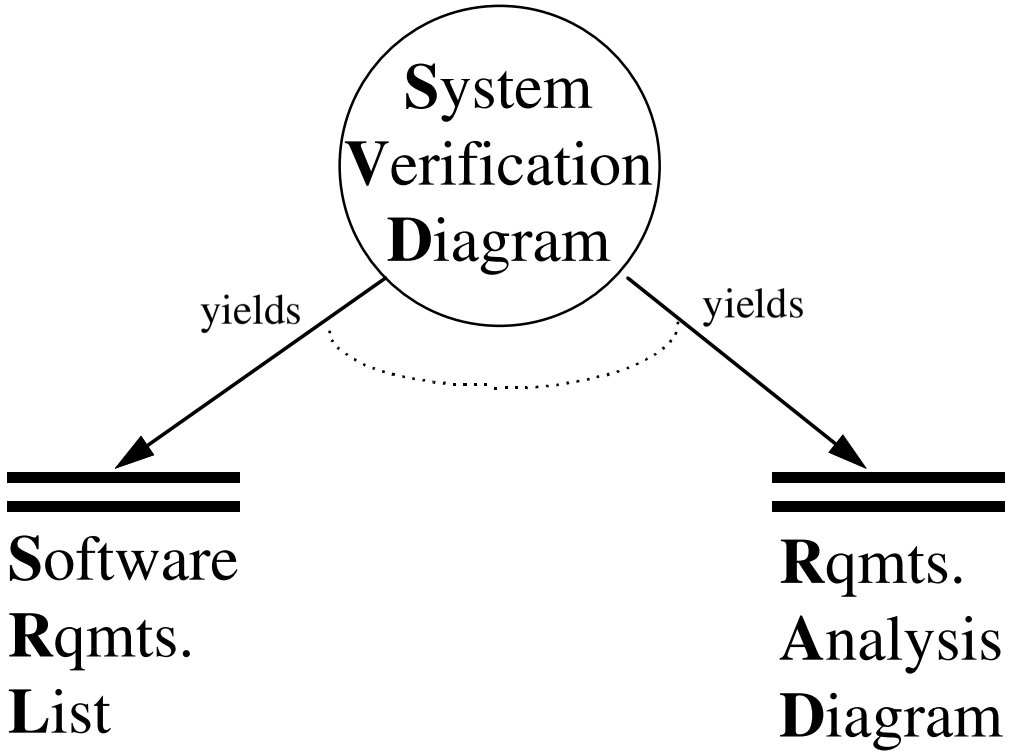
From the perspective of software requirements analysis, SEES derives its two major components from the System Verification Diagram (SVD)*. The SVD dictates the creation and use of (1) a Software Requirements List (SRL) and (2) the Requirements Analysis Diagram (RAD).

The SRL is an enumeration of each requirement found in the Software Requirement Specifications (SRS). Two primary objectives in stipulating the development of the SRL is (1) to ensure that each requirement is identified and recorded, and (2) to provide quick access and reference to each requirement throughout Verification and Validation (V&V) activities.

The RAD is based on a form that, for each requirement, details (a) the function (or data transformation) being performed, (b) the list of identified inputs to the transformation process, and (c) the expected output of the transformation. The major objective of the RAD is to ensure that each transformation activity defined by a requirement is complete. That is, for each transformation the necessary and sufficient set of inputs are identified and available, the transformation process is well-defined, and that the intended output elements are enumerated. The RAD also serves as the "anchor point" for Defect Trouble Reports (DTRs) which are produced if a transformation activity is not completely defined.

*Dutsch, M.S. (1982) , *Software Verification and Validation*, Englewood Cliffs, NJ: Prentice Hall.

SEES Methodology



A closer examination of the activities underlying the development of the SRL and RAD reveals a substantial duplication of effort and overlap of objectives. In particular, both

- enumerate the set of functional requirements, and
- attempt to determine the extent to which each functional requirement is complete and correct.

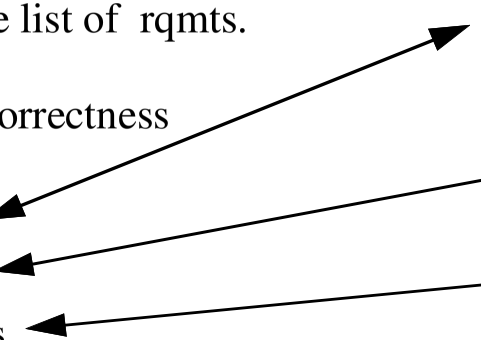
More specifically, the SRL focuses attention on assessing the correctness of **all** requirements, including those which are functionally oriented. The RAD, on the other hand, directs its investigative activities toward assessing the correctness of functional requirements only.

As a consequence of somewhat their differing objectives, assessment activities associated with the development of the SRL are more inclusive (because they include checking for ambiguity, consistency, traceability and testability), but address functional correctness and completeness from a more shallow (or breadth-oriented) perspective.

In effect, the RAD activities (a) subsume all SRL activities devoted to assessing functional completeness and correctness, and (b) includes additional assessment procedures that provide a more in-depth examination of the functional requirements than those dictated by the process of SRL development.

SEES Methodology
SRL & RAD Objectives

- SRL
  - identify & create list of rqmts.
  - check SRS for correctness
    - » ambiguities
    - » functional
    - » correctness
    - » completeness
    - » consistency
    - » traceability
    - » testability
  - create DTRs for errors



- RAD
  - create list of functional rqmts.
  - check SRS functions for correctness
    - » missing input
    - » missing processes
    - » missing output
  - create DTRs for errors

The percentages shown below are conjectures based on our experiences in applying SEES to two IV&V projects. We readily acknowledge that project attributes and personnel expertise can have a significant impact on their fluctuation.

The following slide attempts to underscore three points:

(1) If one examines the total effort required to perform the software requirement analysis (as dictated by SEES), activities associated with producing the RADs account for 60% of the expended effort, while those associated with producing the SRL fall in 40% range.

(2) Assuming that only one set of activities were to be performed during the requirements analysis phase, we would conjecture that those associated with the RAD would be 40% effective in detecting all requirement defects. Alternatively, if only those activities associated with the development of the SRL were performed, then we would conjecture that 95% of all defects would be detected.

(3) Hence, the question that must be asked is:

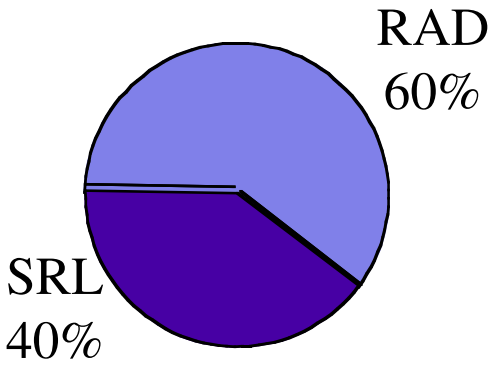
"When is 5% increase in effectiveness worth a 60% increase in effort cost?"

In part, the answer to this question lies in (a) the critical nature of the software system being developed, (b) the ramifications of system failure, and (c) the amount of funds allocated for system development.

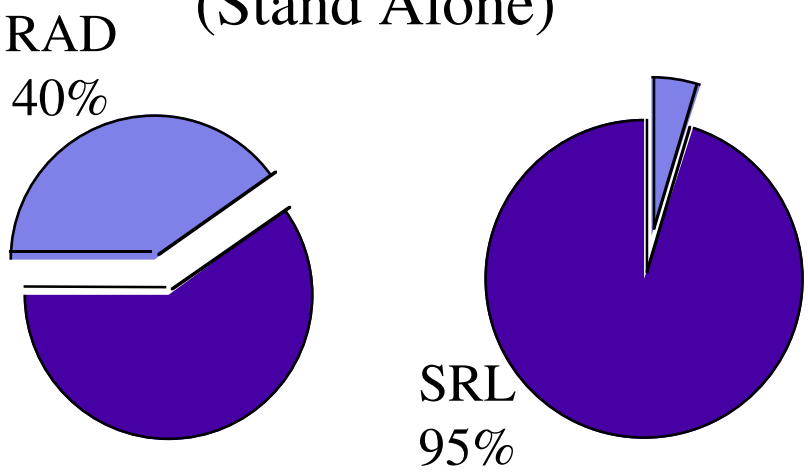
SEES Methodology

Comparing SRL & RAD Activities

Relative Effort (Combined)



Relative Effectiveness (Stand Alone)



When is a 5% increase in effectiveness worth 60% increase in effort cost?

- mission critical systems?
- safety critical systems?
- others?

In addressing the overlapping of activities and duplication of effort associated with the development of the SRL and RADs we offer two alternatives.

Alternative 1 focuses on making the RAD an optional activity.

> We believe that for non-critical systems this alternative presents a viable solution because activities associated with the development of the SRL do emphasize that all requirements be examined for correctness, including functional requirements. In an effort to minimize the impact of skipping RAD activities, one might consider enhancing the verification activities focusing on the analysis of functional requirements.

The benefit of Alternative 1 is that RAD activities are employed only when judged critical -- cost savings are realized when RAD activities are omitted.

The disadvantage of Alternative 1 is that one must assume a higher probability of overlooking a functional error.

Recommendations
Alternative 1

- Strive to make RAD an optional activity
  - SRL
    - » create list of all requirements
    - » verify all requirements
  - RAD
    - » create RADs for functional requirements only
    - » cross-reference functions in SRL
- Benefits
  - RAD used only when necessary
  - savings in effort cost
- Disadvantage
  - higher probability of overlooking functional error

Alternative 2 focuses on making activities associated with developing the SRL and RADs disjoint. In particular, if both the SRL and RADs are to be developed, then focus the the verification activities associated with SRL development on non-functional requirements only.

Alternative 2 removes the duplication of effort (and attendant costs) associated with the current approach defined by SEES. Properly defined and implemented, Alternative 2 maintains the same level of verification effectiveness, but at reduced effort.

The benefits of Alternative 2 are

(1) a reduction in effort (and cost), with minimal (if any) reduction in verification effectiveness,

(2) less duplication in identifying and enumerating similar information, and

(3) an enhanced ease of cross referencing through (2) above.

Recommendations
Alternative 2

- Strive to make SRL & RADs disjoint in activities
  - SRL
    - » create list of all requirements
    - » only verify non-functional requirements
  - RAD
    - » create RADs for functional requirements only
    - » cross-reference functions in SRL
- Benefit
  - less overlap in data produced
  - less effort to create both
  - ease of cross-referencing

Independent of which (if either) alternative is chosen, we recommend automating the process of generating the SRL, RADs and DTRs. In particular, we suggest an implementation that employs screen oriented forms and hyperlinks connecting equivalent fields.

The automated approach facilitates the "duplication" of requirement statements and the cross-referencing of fields, e.g. DTR numbers.

Further Recommendation

- automate process of generating SRL and RAD
  - allows cross-referencing

During the past year we employed SEES in two separate IV&V efforts. The first effort focused (where applicable) on applying (where applicable) the SEES approach "after the fact", that is, applying parts of SEES to a system that had already been developed. The second effort employed SEES throughout the development effort. Experiences gained from both efforts are reflected in the observations provide below.

- In SEES each DTR requires the generation of a separate form. We found that computerized forms allowing multiple DTRs (in a condensed format) per "page" significantly enhanced DTR recording. In particular, changes and corrections were easier to handle, and cross-referencing among DTRs, SRL and RADs was made easier through simple key-word searches.
- Participants in both efforts voiced the opinions that RADs were not applicable in all development efforts. More specifically, because the RADs are inherently manpower intensive, they should be reserved for mission critical systems or functions determined to be on a "critical path".

  Experience also indicates that if RADs are to be employed effectively, appropriate V&V staffing is crucial -- both in the numbers and expertise.

- Finally, those activities surrounding the generation of the SRL and RADs should be applied only to a mature set of requirements. Otherwise, the majority of time and effort will be spent detecting minor defects, .e.g. typos, which in turn, could have a detrimental impact on the detection of major defects.

Observations, Notes, Comments
Requirements Verification

- Individual DTRs
  - summarized in group computerized DTR
  - IV&V group preferred computerized forms and entry
  - changes/corrections easier, look-up easier, later evaluation much easier
- RADs not applicable to all development efforts
  - systems with non-complex (non-substantial) functions
  - small systems
  - small IV&V groups - not enough personnel to do RADs and SRL in timely fashion
- Application of SRL and RADs require a mature set of requirements

In general, SEES dictates an effective approach to both Preliminary and Detail Design verification. Within these two development phases, verification is modeled after the Fagan Inspection Process* which requires a systematic and structured assessment approach.

SEES provides preliminary and detailed design checklists, and requires the preliminary checklist to be applied also during the detailed design phase as well.

Because the verification process follows the Fagan Inspection approach, SEES stresses the importance of defining and distinguishing among the duties of the moderator, reader, recorder and inspectors. SEES also states the need for personnel to rotate among roles to reduce the undesirable consequences that often comes with repetitive processes.

Each of the above are laudatory stipulations.

*Fagan, M.E. (1976), "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, Vol. 15, No. 3.

Preliminary & Detailed Design

A General Critique of SEES

- Good approach taken for both design levels
- Check-lists useful and necessary for both levels
  - require use of preliminary & detailed checklist for detailed design inspection ALWAYS!
- Important to separate duties in inspection and rotate as stated

The standard Fagan Inspection Process allows one or members of the development team to participate in the overview and inspection sessions.

SEES excludes all members of the development team from participating in the Preliminary and Detailed Design verification -- presumably to minimize contention and maximize defect identification.

Our experience suggests that having a development team member participate only in the overview session is the best approach. More specifically, the development team member is in the best position to describe the overall system objectives and design -- hence, he/she should be able to provide the best overview and design insights. We advocate excluding the development team member from the inspection process to minimize “defensive posturing” and rebuttals (or justifications) when potential defects are identified.

Preliminary & Detailed Design Verification
Including Member of Development Team

- Fagan
  - allows development team member to participate in overview and inspection process
- SEES
  - excludes development team member from all verification activities
- SEES'
  - requires development team member to lead overview session only

Elaborating on the previous slide subject, we allowed a development team member to participate in both the overview and inspection sessions for the Preliminary Design verification, but restricted participation to the overview session only for the Detailed Design verification.

The overview session for Preliminary Design verification was led by a non-development team member; a development team member led the overview session for Detailed design verification. The latter resulted in better organized and more comprehensive presentation of system design.

However, the inspection session for Preliminary Design verification (which had a participating development team member) produced less than desirable results. In particular, inspectors not involved in development process were viewed by the development team member as attacking the design, which in turn, led to contentious discussions. Moreover, too much time was spent on justifying design decisions (by the development member) and finding solutions, rather than on identifying faults.

Learning from this mistake, development team members were excluded from the inspection session for Detailed design verification. Consequently, discussions proceeded in a more timely fashion, and without the presence of defensive posturing among inspectors. Although faults were more readily identified, the capability to suggest solutions was diminished.

Preliminary & Detailed Design Verification
Observations

- Requiring development team member to lead overview
  - well organized approach to presenting system components
  - design rationale enunciated
  - fostered better understanding by IV&V team
- Including development team member in inspection process
  - disaster
  - arguments between IV&V team and development team member was commonplace
  - too much focus on finding solutions, <u>not</u> identifying problems
- Excluding development team member from inspection process
  - smoother flowing discussions
  - less time wasted on defensive pastures
  - focused on problem identification
  - diminished ability to suggest solutions to problems

One of the four components of formal inspections is the inspection process, itself. The primary objectives of the inspection session is to identify design defects and to outline correction paths. Clearly, such a process can be time consuming.

The Fagan approach suggests that no time limit be placed on such discussions.

SEES, on the other hand, limits session times to two-hour blocks.

Our experience indicates that time limits are beneficial in that they

- help keep inspection members focused on the specific task of identifying defects and outlining corrective actions, and
- tend to inhibit discussions of corrective actions evolving into complete formulations of solutions that may or may not be applicable.

Preliminary & Detailed Design Verification
Setting Time Limits for Discussions

- Fagan
  - No strict time limit
- SEES
  - Strict time limit on discussions
- SEES'
  - Strict time limits on discussions

<u>Observations</u>

- Strict time limits keeps team members focused
- Strict time limits inhibits discussions that can lead to formulations of solution approaches

Formal inspections provide a systematic approach to design verification and entails many well-defined activities. Often, however, crucial detail are omitted in the outlining of those activities.

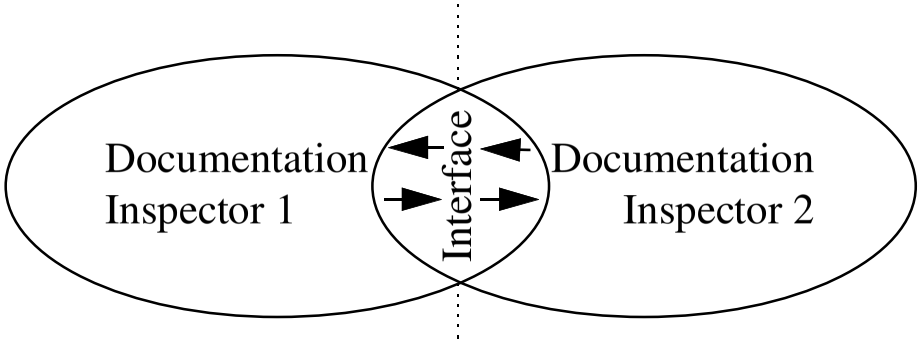
In performing the SEES' experiment, insights and observations led to two such details that significantly enhance the effectiveness of the inspection process.

(1) Inspectors must ensure consistent use of terms and parameters both internal to the design subsystem being evaluated and to those referencing external interfaces.

(2) When the moderator partitions the design into inspection segments, the partition should be determined such that each interface between two components lies (a) within the scope of one inspector, and (b) that same interface interface lies on the boundary between two other inspectors. This ensures that each major interface is checked for correct output specifications, input specifications, and for consistency between any two interfaces.

Preliminary & Detailed Design
Inspection Session MUSTS!

- Partition documentation and assign inspectors so that no design interface boundary is the same as documentation/inspector boundary



- Instruction to inspectors
  - check for internal consistency
  - check external interfaces

In addition to the previous focused set of comments, the following and more general set observations are deemed pertinent:

- The checklists provided by SEES are very helpful for orienting design verification. They should not, however, be considered as the only guidelines for design verification. An understanding of the project scope, its characteristics, and the intended use of the product can lead to the identification of additional checklist elements necessary for and effective verification process.
- A significant duplication of detail exists in the production of the Inspection Defect List (IDL) and the DTRs. An automated process for producing the IDL, DTRs and the Inspection Summary Report (ISR) is advised.

Preliminary & Detailed Design
Some Minor Observations

- Forms:
  - checklists very positive, could be automated
  - Inspection Defect List and DTRs have overlap:
    - » automate process of note taking during inspection session
    - » allow for IDL items to be transferred automatically to DTR and make modifications on-line
    - » include reference back to Individual Preparation Log (IPL) on both IDL and DTR forms
      - > makes problem statements clearer
    - » generate Inspection Summary Report (ISR) automatically

In SEES the principal method of “verification” is through automated code analysis. In particular, SEES mandates the use of a tool like Adamat to examine adherence to standards and to provide an assessment of quality characteristics relative to predetermined threshold values.

SEES does not, however, require any verification of Code to Detailed Design. Hence, a direct assessment of adherence to design specifications cannot be judged. Nonetheless, the extent to which code accurately reflects functional specifications is achieved through Technical Assessment Procedures (TAPs) guiding Test Plan and Test Procedure development and usage.

One deficiency in this approach is that algorithms specified in design may or may not be implemented in the code. Additionally, no assessment process exists for judging the correctness and efficiency of implemented algorithms.

Code Inspection
Automated Quality Checking

- Code quality checked through
  - Adamat
    - » adherence, standards, and indicator analysis
- Algorithm correctness not checked
  - usage
  - correctness
  - efficiency
  - algorithms specified in requirements and design

Although code verification is not required in SEES, it was still performed by the IV&V team. The results were not, however, distributed to the development team.

During code verification one major error and one minor error were found. They were eventually discovered by the development group, but not until Integration and Testing (I&T).

These discoveries underscore the benefit of early error detection.

Code Inspection/Verification

Errors found by IV&V, not SEES

- IV&V Team found two errors
  - one major -> time to fix 28 minutes
  - one minor -> time to fix 3 minutes
- Benefit of code IV&V
  - find and fix errors early in CSU development
  - no “false” error tracking through cascading errors during or after I&T
- Costs
  - early detection savings

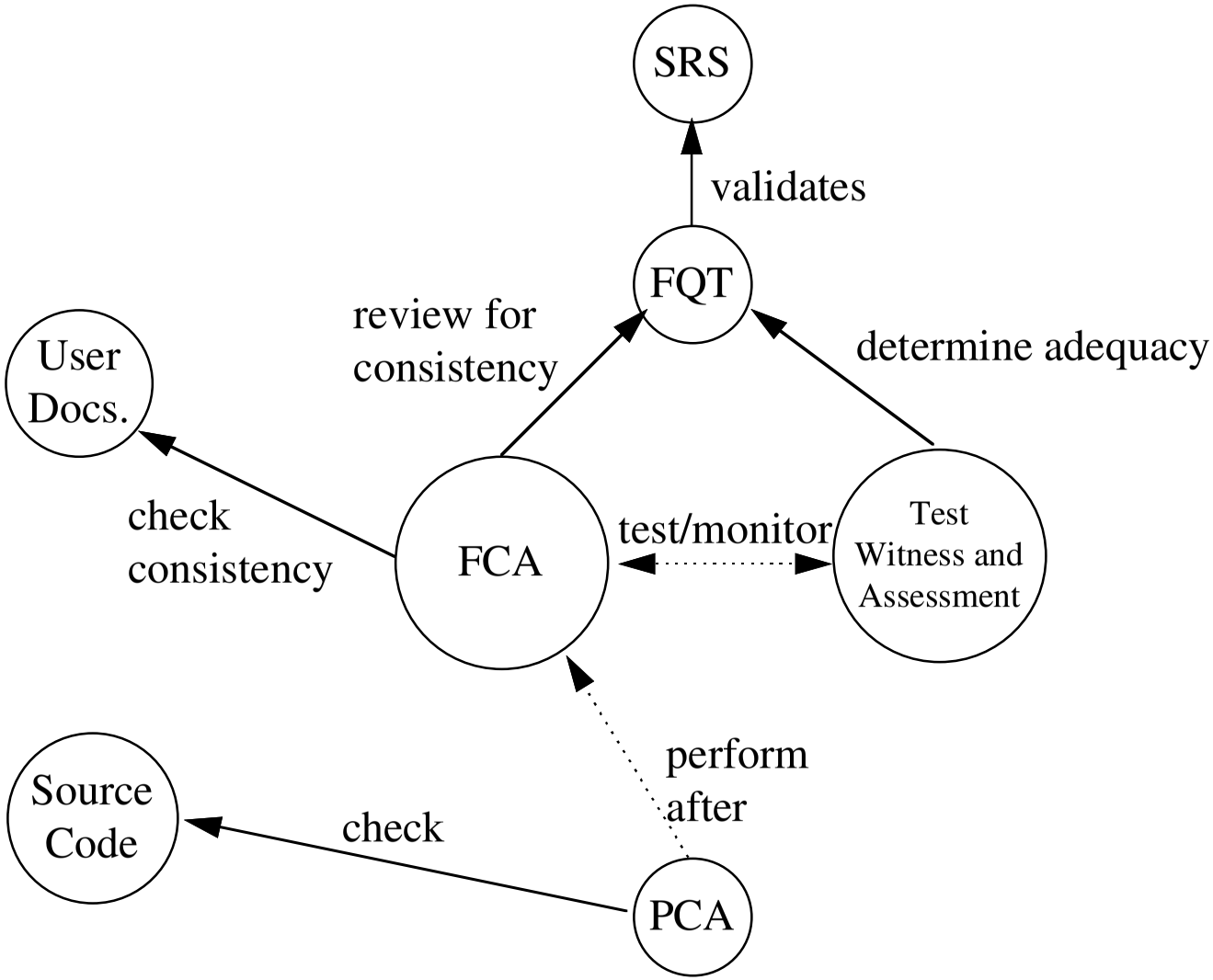
SEES approach to I&T is based on a comprehensive multi-step procedure. The Formal Qualification Test (FQT), Functional Configuration Audit (FCA), Test Witness Assessment (TWA) and the Physical Configuration Audit (PCA) are all instrumental components. The focus of each activity is outlined below.

- FQT stresses tests that provide a validation of the software relative to the functional specifications stated in the SRS.
- FCA is an audit process that focuses on consistency, accuracy and completeness of user documents and the results of the FQT and TWA.
- The objective of the TWA is to witness FQT procedures, assess their adequacy and completeness, perform additional tests (if necessary) and to compile results for the FCA.
- The PCA is generally performed after the FCA and ensures that problems detected in the previous steps are adequately addressed.

The experiment performed at Virginia tech did not distinguish between the procedures outlined above. With the exception of the PCA, all other steps were combined to form a single I&T step. An assessment of Operational Correctness implemented, in part, those activities imparted to the PCA.

Integration & Testing

The SEES Approach



In general we find the "I&T" process defined by SEES to be systematic and rigorous. It has a well-defined structure and sequence of test/review activities guided by the Software Test Plan (STP) and Software Test Description (STD).

The major disadvantage to the SEES approach is that testing is guided exclusively by the STP and STD -- there is no provision for or encouragement to perform independent tests. Furthermore, it appears that the software developers are the primary assessment agents -- we question whether some independence is needed.

As with many parts of the SEES approach to V&V, I&T recording and reporting activities can be significantly enhanced through automation.

Integration & Testing
Critique

- Benefits
  - good approach to final testing of system
  - well-defined set of test/review tasks
  - rigorous
  - guided by STD
- Disadvantages
  - only tests defined by the STD are emphasized
  - no independent set of tests
  - no independent assessment of results
  - some parts extremely redundant, calls for automation
    - » manual cut & paste of test witness and assessment forms!

In conclusion we reiterate the following major observations:

(1) The SEES approach to V&V is well-defined and provides extensive guidance in the necessary activities underlying each phase of the V&V process.

(2) During requirements analysis, the activities associated with the generation of the SRL and RADs overlap significantly. This overlap has a detrimental impact on effectiveness and productivity.

(3) The use of automated recording and reporting can contribute significantly to the effectiveness of the process and to enhancing personnel productivity. Benefits can be readily recognized by examining the effort required to record similar data among the SRL, RADs, DTRs and ISR.

(4) There is no stated process for verification between code and the detailed design documents.

Perhaps the rationale behind this omission is that functional requirements are assessed through examination of the STD and its guidance during FQT, FCA and PCA. Clearly there is a cost benefit gained by omitting this step, but there are drawbacks, too.

(5) The correction of an error in one development phase might necessitate changes in a previous phases. Currently, SEES has no stated provision for tracking or confirming the implementation of such changes.

SEES: An Overall Assessment

- Duplication of effort and recording of detail information
  - SRL and RAD generation
  - DTRs and ISR
- Lack of automated support
- Omitted verification between Code and Detail Design documents
- Substantiation of requisite changes to previous phase artifacts mandated by changes in current phase (“backward ripple effect”)
  - who tracks changes from one level to previous levels
  - Who confirms that such changes have been made