

White Paper
CMII 865B

**How to Write Requirements
for Development Programs**



Prepared by
CMII Research Institute

White Paper CMII-865B
How to Write Requirements for Development Programs

Revision Record

<i>Revision</i>	A	B	
<i>Released by</i>	WWG	WWG	
<i>Release date</i>	03/27/08	12/29/09	
<i>Authority</i>	062-WP	063-WP	
Page 1	A	B	
Page 2	A	B	
Page 3	A	B	
Page 4	A	B	
Page 5	A	B	
Page 6	A	B	
Page 7	A	B	
Page 8	A	B	
Page 9	A	B	
Page 10	A	B	
Page 11		B	

Purpose of This White Paper

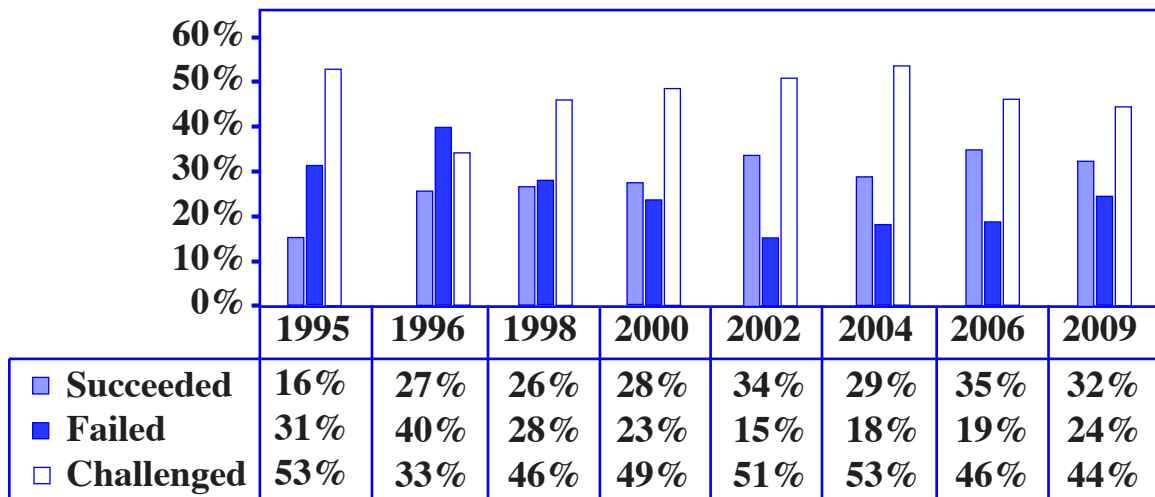
Software development programs and implementations of mature information systems both have a high failure rate. The leading causes are attributed to poorly defined requirements in both cases. This paper provides a recommended solution.

OUTLINE

- **Systems and Software Development Statistics**
- **Failure Rates When Implementing Mature Systems**
- **The Impact of Technological Advancements**
- **QS&S Guide for Developing Systems & Software**
- **DODI 5000.2 for Developing Automated Systems**
- **Development Planning and Plan Validation**
- **Information Systems, Procedures and Work Flow**
- **Summary. Conclusions and Recommendations**

Systems and Software Development Statistics

Statistics compiled by the Standish Group on cost and schedule performance of software development projects in the U.S. since 1995, are shown below. Opportunity for improvement is obviously huge and the possible reasons for failure are listed. Much of the problem is attributed to vague requirements, poor user input and failure to plan.



Reasons for failure are attributed to the following:

- *vague requirements*
- *poor user input*
- *failure to plan*
- *poor architecture*
- *inadequate skills*
- *stakeholder conflicts*
- *poor cost and schedule estimates*
- *communication breakdowns*
- *late failure warning signals*
- *cutting jobs and expecting the same work*

The Standish Group, "Chaos Database, 2009"

Projects fail for one or both of two reasons. Projects fail because the plan was bad or execution of the plan was bad, or a combination of both. Most of the reasons listed above indicate that the plan was bad.

How can you determine ahead of time if your plan is good or bad? How can you be assured that your plan is a good plan? If unsure, how can you proceed on an exploratory basis and keep risks to a minimum until you are sure? If the need for improvement is urgent, how can you accomplish the exploratory tasks quickly? The purpose of this white paper is to answer these questions.

Failure Rates When Implementing Mature Systems

The Standish survey results can be summarized very simply — "we struggled because our requirements were not clear." On the other hand, the following article reveals that a high rate of failure also occurs when implementing mature systems such as ERP. Nearly 60% of the failures are attributed to poor definition of functional requirements. Same conclusion — "we struggled because our requirements were not clear."

When implementing a mature system, how can this be true?

Twelve Cardinal Sins of ERP Implementation

Enterprise Resource Planning (ERP) is an outgrowth of Material Requirements Planning (MRP) initiated in the 1970's. MRP evolved to MRPII (Material Resource Planning) in the late 1970s and software companies began to develop software packages around the MRPII concept. Database management systems emerged at the same time and the integrated database became the engine for fully integrated software.

The early software packages were transaction-oriented and highly unfriendly to users. Then personal computers and client servers began to emerge. Base operating systems such as Windows allowed software packages to become more and more user-friendly.

Today, ERP systems have proliferated extensively and reached a stage where development has become industry specific. Yet, implementation success is very poor. Over 60% of ERP implementation starts have failed. Twelve leading causes for failure are listed below. Industry surveys indicate that inadequate definition of functional requirements is the leading cause.

- (1) Lack of top management commitment.*
- (2) Inadequate definition of functional requirements (accounts for nearly 60% of ERP implementation failures).*
- (3) Poor ERP package selection (inadequate functional requirements is a major contributor to this reason).*

Nine Inadequate resources, resistance to change, miscalculation of time other and effort, misfit of application software, unrealistic expectations, causes inadequate training, poor project design, poor communications and ill-advised cost cutting.

White Paper by Richard G. Ligus, Rockford Consulting Group

The Impact of Technological Advancements

An information system is an integration of technological components. Those components continue to evolve and their interfaces continue to change. Advancements in software technologies and architectures have continued to accelerate. IT departments throughout the universe are trying to stay ahead of the curve — even though much of the newly acquired and costly technology is never implemented.

Technology is making work easier and also changing how work is done. Consequently, systems experts and system users must jointly write the system requirements, user requirements and procedures.

The Rapidly Changing Pace of Technology

Unlike other fields that are more predictable, technology moves so so quickly, its development pace is exponential. Whereas some fields have changed little in hundreds of years, you can barely predict how technology will transform society in 20 years. The future is an unknown. Rapidly Changing Pace of Technology: Ten Year Forecast

An analysis of the history of technology shows that technological change is exponential. We will not experience 100 years of progress in the 21st century — it will be more like 20,000 years of progress (at today's rate). By Raymond Kurzweil

So what does all of this mean? Perhaps the causes for failure, as listed on pages 2 and 3, are misleading. Vague requirements and poor user input would imply that the user is at fault. This must be challenged. Perhaps the technology could not support the user-defined requirements. The following is an example wherein that was the case.

Evolution of PLM Tools

The leading obstacle to the implementation of CMII since 1995 has not been poorly defined requirements or lack of top-management commitment. CMII-trained professionals who use such systems know precisely what they want. The leading obstacle has been enabling software tools and their lack of CMII functionality. As of 2005, that obstacle has been overcome. Several PLM tools now provide CMII functionality that is quite robust. CMII Research Institute

QS&S Guide for Developing Systems & Software

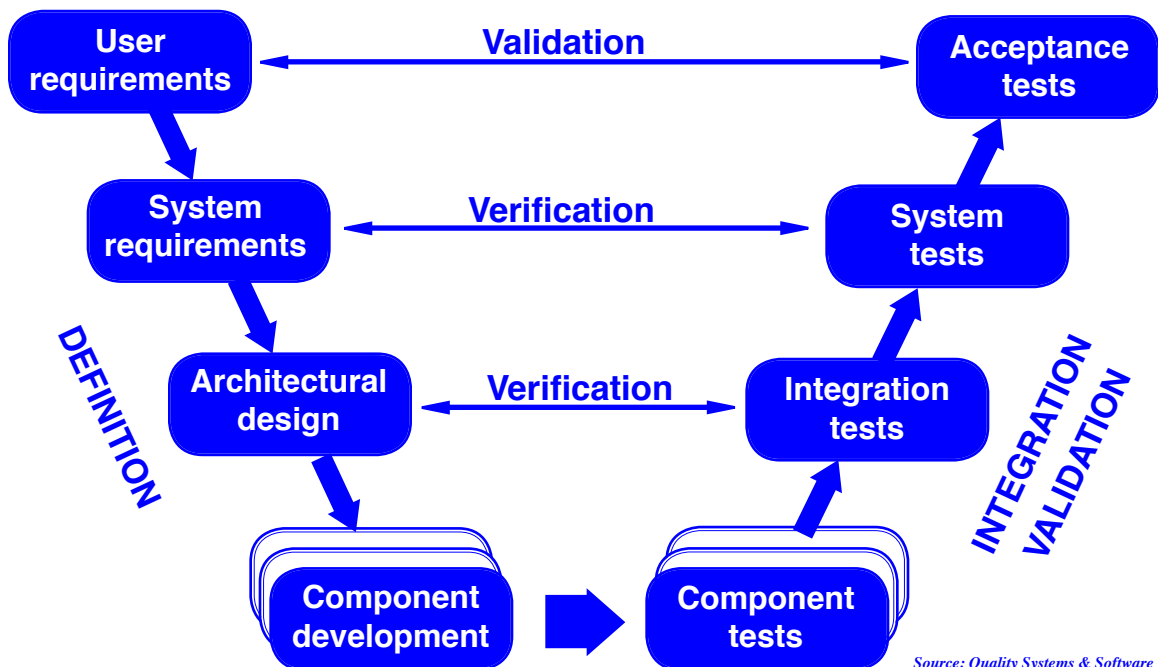
Quality Systems and Software (QS&S) publishes guidelines for how to develop information systems and their emphasis is on how to write requirements. They cite studies by the Standish Group which show (1) incomplete requirements and (2) lack of user involvement to be leading causes for failure in systems and software development projects.

Their template for developing an information system is shown below. User requirements drive the system requirements which are enabled and constrained by architectural design capabilities and limitations.

The architectural design document defines what is to be built, how it behaves and how it is laid out. It defines key components, interfaces, control structures and external systems. It is the basis for all milestones. Cost estimates should be well definable at this stage.

The next page describes how to define a system requirement and how to write a detailed user requirement. It also describes how user priorities and system costs are used to make implementation decisions.

Requirements Ensure You Build the Right Project (Information System) the Right Way



Example of a System Requirement



Corresponding Architectural Design



Source: Quality Systems & Software

Detailed user requirements are to be written in singular increments. It must indicate who needs it and the results must be measurable.

Anatomy of a User Requirement

"The order entry clerk shall be able to complete 10 customer orders in less than 2 hours"

This requirement identifies a real user type and an end result. It also defines the success criteria in measurable terms.

The challenge is to seek out the user type, end result, and success measure in every requirement.

Source: Quality Systems & Software

Detailed user requirements are selected for implementation in accordance with their priority relative to cost, as shown below.

Implementation Decisions: User Priority Vs System Cost

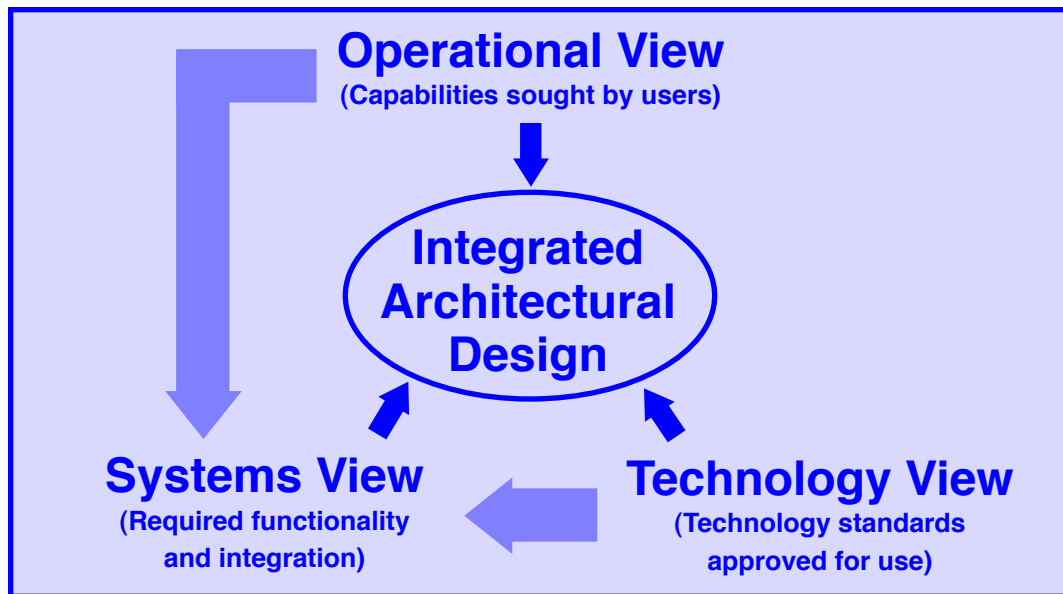
User Responsibility		Developer Responsibility	
Priority	Implement?	Cost	
User Requirements	High	Yes	System Requirements
	Medium	Yes	
	High	Yes	
	Low	No	
	Medium	No	

Source: Quality Systems & Software

DODI 5000.2 For Developing Automated Systems

This DOD instruction describes how automated systems are to be developed. Architectural design is an integration of three views.

Each Integrated Architecture Shall Have Three Views



3.3.1. Evolutionary acquisition is the preferred DoD strategy for rapid acquisition of mature technology for the user. An evolutionary approach delivers capability in increments, recognizing up front the need for future capability improvements. The objective is to balance needs and available capability with resources, and put capability in the hands of the user quickly.

3.3.2. Spiral or incremental approaches are both acceptable.

3.3.2.1. Spiral Development. In this process, a desired capability is identified but the end-state requirements are not known at program initiation. Those requirements are refined through demonstration, user feedback and technology maturation.

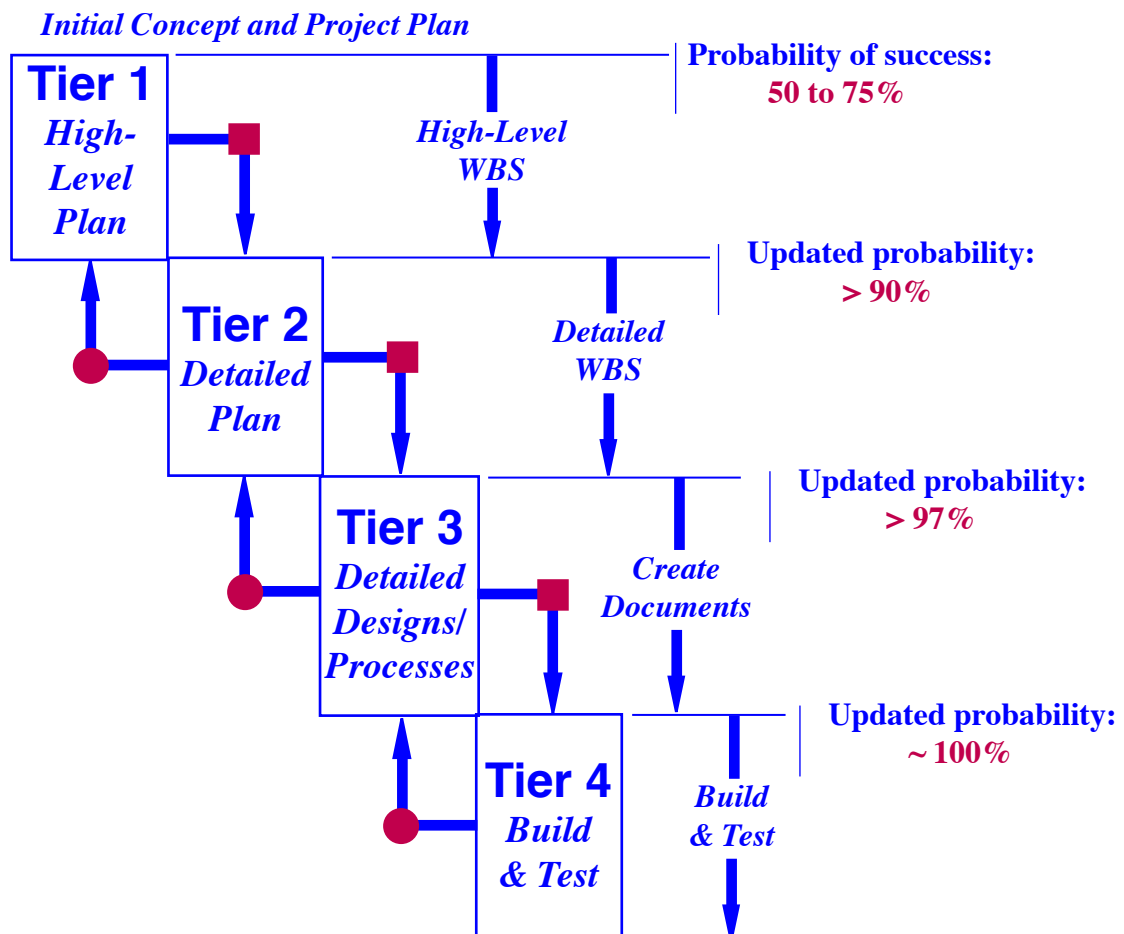
3.3.2.2. Incremental Development. In this process, a desired capability is identified, an end-state requirement is known, and that requirement is met over time by developing several increments, each dependent upon available mature technology.

Development Planning and Plan Validation

Any project to develop and/or implement an information system should include GO and NO GO decision points. Any uncertainty about project feasibility must be resolved in an early phase. Any major issues must be resolved while creating the high-level plan. Relatively minor issues that remain, if any, must be resolved while creating the detailed plan. By then, success is highly assured.

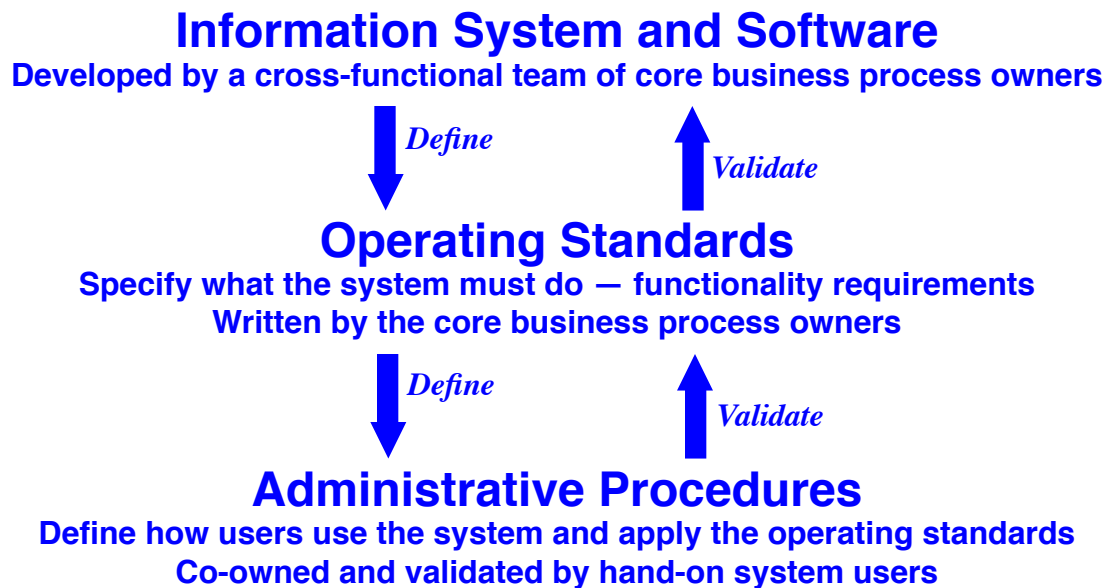
A work breakdown structure (WBS) is used to break a large project down into manageable work packages. A high-level WBS is used to create the high-level plan. A detailed WBS is used to create the detailed plan.

The high-level plan must be created by a cross-function team whose members have expertise in all facets of the project. Involvement by hands-on systems users should not begin until Tier 3.



Information Systems, Procedures and Work Flow

Operating standards serve to define what an information system must do. They define the functional requirements. Administrative procedures represent how system users utilize the system and achieve the operating standards in the performance of their work. Proper work flows for developing operating standards and administrative procedures, per the CMII model, are as follow:



It is noted that this work flow does not comply with the conventional approach for developing information systems and software. The first step of development, in the conventional approach, is to gather detailed requirements from those who will eventually use the system. The detailed requirements are then used to determine what the system functionality needs to be.

From a CMII point-of view, this conventional approach is backward, cumbersome and prone to error. With this conventional approach, systems developers are overwhelmed with unnecessary details.

Of course, user involvement is important, but there is a right time and place for their involvement. They validate the administrative procedures and request changes to operating standards as needed.

Summary

The title of this paper is based on the conventional approach to systems and software development wherein the high rate of failure is attributed to poorly defined requirements. This paper has challenged such conventional wisdom. The most significant cause of implementation failures is not because detailed user requirements were poorly written. It is because the high-level plan upon which the development program is based was deficient. The chances of success with any development program are directly proportional to the integrity of the high-level plan.

Conclusions

To enter into a major development without a sound high-level plan is to proceed without direction and without an early warning system. It is to begin a journey without a clearly defined destination. It is what happens when the first step of development serves to gather detailed user requirements rather than create and validate a high-level plan. The validated high-level plan includes ensuring that the selected technology is compatible with all other aspects of the development objectives.

"How to write requirements" is a question that is rarely asked by those who use work breakdown structures to subdivide projects into manageable work packages. Detailed requirements flow down from the higher-level requirements. How they are written is rarely an issue.

In the CMII approach to development, nothing is more important than the design basis. This is true whether using the spiral or incremental approach. The design basis is where any unknowns are identified.

Recommendations

Those who develop information systems and software must shift their initial focus away from gathering detailed user requirements and concentrate instead on the high-level plan. They must establish a cross-functional team whose members thoroughly understand all aspects of the development effort and the objectives. The cross-functional team must lead the development effort. User involvement must be delayed until the administrative procedures are to be validated.