The Iterative Development Process

Written by: Doug Lahti

Muslah Systems, Inc.

June 27, 2005

Last Modified: May 2, 2006

Printable Copy

Scope

This document describes the overview and technical issues associated with the development of a complex product definition.

Introduction

Conceptualization of a development process implementation that yields efficient production of a marketable product. The marketable product may have various forms: electroinic circuit, software system, electro-mechanical controls, a DSP algorithm, or a method as yet to be determined by today's technological capacity to fully understand. The iterative development process shall therefore describe an overview process flow implementable for all these development examples.

As an attempt to prevent confusion, some words used by this text may refer to other words used. The word product for example may refer to the system or its result. The word system may refer to the product or the process implemented to produce the system. The specific object of discussion shall therefore maintain accurate identity and prevent ambiguity. For this purpose we declare the system to identify the composition of materials that are the product. The product is therefore the outcome of the system. As a controllable system is observable, the iterative process shall therefore provide observability and therefore controlability for the system development process. The objective for the iterative development process has thus become observance and control for the development of a system.

As with the theorem of super-position, a system may be defined by its network of components having disticnt behavioral character. Correct identification of these components thereby becomes important, and necessary to evaluate. A software system would declare its components as modules with interactive behavior. An electronic circuit would declare its components as a network of energy impedence models coupled with other impedence models. The components of the iterative development process observe expected results that provide verification and product validation. We therefore have criterion, process, and evaluated results as tools to develop a set of metrics that quantify the iterative development process.

Criterion for a successful product development

A product development is most often accomplished with a project definition. The project definition has a mission to accomplish. Ground rules and restrictions that hinder completion of the project mission may exist.

Timeline constrainst, cost, realizability of the system may add risk for successful completetion of the project. The most frequently violated success criterion is apparently the timeline constraint. Without a comprehensive work breakdown, an accurate development schedule is less likely to exist. The success criterion for a product development has hereby become: Timeline, cost, and realizability. A product development becomes successful when it is accomplished within the expected timeframe without exceeding projected cost and satisfies all functional requirements.

There exists a model of interactive components composed of timeline, validation, and cost. Each of these components have an effect on the sucessful completetion of a product development. Implementing more cost does not necessarily imply reducing the timeline. As the satisfaction of functional requirements identify the end point of the timeline, identification of the functional requirement becomes critical to realizing project completion. Verification and validation thereby become our metric for project completion.

A mapping of product requirements to validate can be accomplished with a tracability matrix. This matrix identifies all product requirements for validation to the associate validation proceedure. Statistical distribution models may become developed to quantify effective performance for complete satisfaction of the functional requirements.

A product development process therefore contains the following componets:

- Project Document containing a mission, ground rules and restrictions, desired timeline, cost estimate, resource allocations, and functional requirements
- Development Plan
- System Requirements Document
- Systems Architecture, Design. amd component Specification
- System Construction and Initial Validation
- System Verification and Product Validation Plan
- System Verification
- Product Validation
- Project Performance Evaluation

The elements of the above list not mensioned previously are the architectural design, system construction, and project evaluation. The architectural design document is implicit with any product development, and therefore should not be disputed. As with the design document, the system construction is a necessary and obvious step with accomplishing a product development. The project performance evaluation is a useful tool to receive acknowledgement from the customer as to satisfaction of project delivery and accomplishment.

It may be important to note that this top-level list of the product development process entails several layers of intra-project tasks associated with each item on the list. The details of these intra-project tasks shall be discussed in a later text. At this time the above illistration is intended to present the reality and necessity for the iterative development process.

Necessity for the Iterative Development Process

As a marketable product definition comes into existence, the question of accurate and complete functional requirements develop. Initially we begin with a conceptualization of a product. We develop a description of the product having purpose and functionality. Every purpose has an observable outcome that defines its functionality. The system requirements leading to a design specification that orders and defines appropriate validation methods provide confidence with a reliable and maintainable system, and therefore a satisfactory product yield. Product yield performance is based on timeline, cost, and functionality performance. Therefore the iterative development process shall present quantitative evaluation with the objective to show comparitive performance with other more conventional development methods.

Every design specification questions the validity of it requirement to exist. When a design specification observes complexity that identifies confusion, ambiguity, or validity with a requirement, a design iteration is declared. During implementation of a design faulacies may occur for various reasons. An iteration during implementation could effect, and possibly require modification to, the design specification. A produce validation proceedure may reveal, or infer, product enhancements or alternative functionality that modifies the product definition and trickels down through the system requirement, design, and implementation. The iterative develop process anticipates the existence of these probabilities and attempts to order there occurance effectively optimizing the development process by minimizing the number of iterations required to accomplish the project mission. It is clear that unforeseen design flaw or problems accomplishing validation could hinder the sucess of a project. Therefore, the iterative development process shall anticipate the existence for the probability of completeness.

As we have discussed previously, our criterion for project completion is declared by our product validation that verifies satisfaction of the associate functional requirement. These functional requirement validations are mapped by the tracability matrix that stipulates 100% conformance for accredible accomplishment. As the functional complexity increases, the more likely our conception of completeness will fail. Therefore, the probability of completeness becomes less likely. Iterations that develop evolutionary progression toward the desired product thereby become more econically feasible..

Optimization of the timeline for a product development can be accomplished with iterative develop phases that miniize development time. As the construction of a system progresses we observe the issue to resolve multiple failures existing simultaneously. By accomplishing the system construction incrementally with preliminary validation process we elimininate the interactions associated with their probable existence. Therefore, development time is reduced by eliminating time not used to resolve such issue.

The issue of failure resolution is a reality of all project developments. The probability of inhibiting issues exists as declared by Murphy's Law. The anticipation for these events is therefore a more optimal preparation for success than to ignore this reality.

The feasibility, and necessity, for the iterative development process as described above seems more practical for complex product definition having vague requirements. A product definition is initially declared using the information currently known to develop its functional requirements. As the system requirement develop a design specification for implementation new information is developed that modifies the product definition. The iterations associated with the development of such a system have potential to take various form, but always with the objective to progress to a desired outcome. Finalizing the desired outcome thereby leads to a stable system requirement that a design specification can successfully accommodate. Optional operational configurations would further increase system complexity that can likewise be accommodated by appropriate iterations of enhancement.

Overview

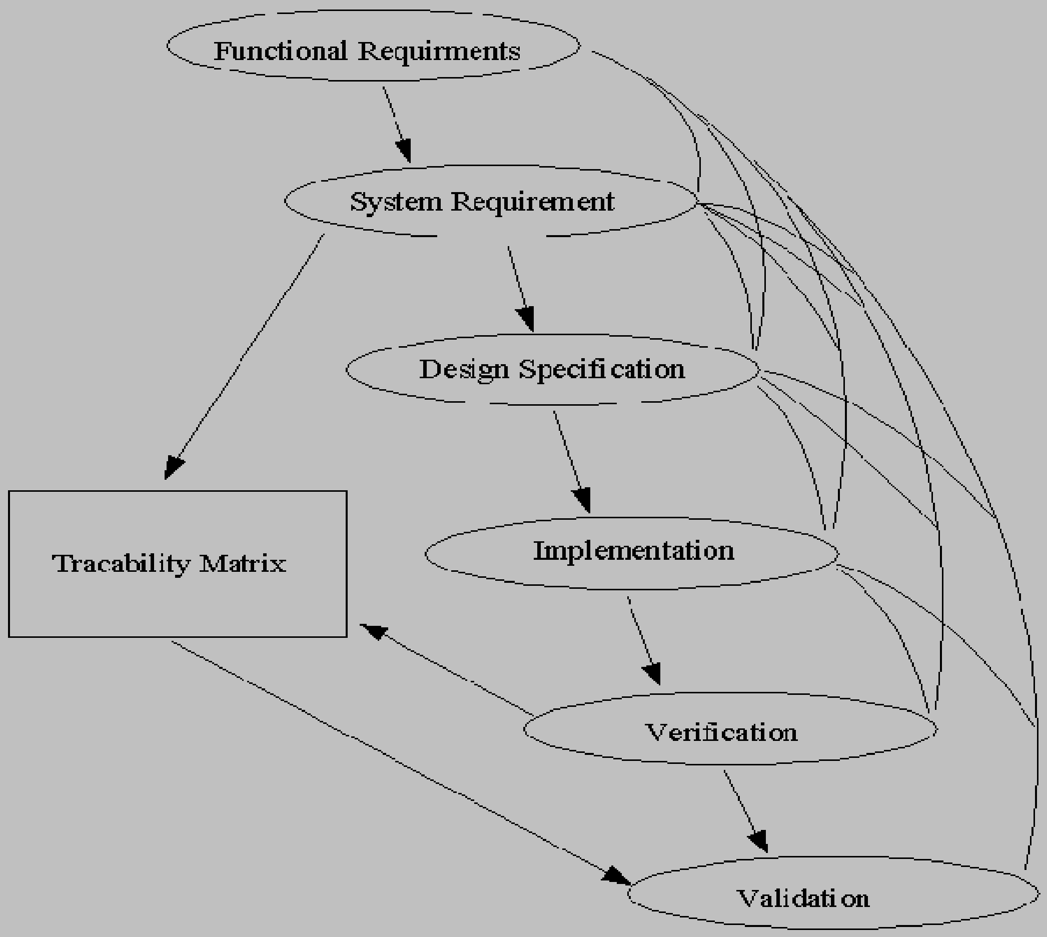
As this development for the definition of the iterative development process progresses the generic development process is reviewed. From conception we hypothesize realizability for functional requirements to be specified through design and validated from miplementation. The initial iteration for a project development is conceptualization. The second iteration becomes definition and declaration of functionality. As anticipated the development iterations follow the sequencial process of the project development. An ordered development sequence is presented in figure ??.

As functional requirements develop and increase in number, more complex architectures evolve. More complex architecture implies more suffisticated methods for product validation. Development iterations are thereby declared to accommodate implementation of the associate requirement. Additionally, the system architecture and component design stipulate satisfaction for system requirements associated with a reliable and maintainable system. Implementation of these requirement sets are more effectively accomplished sequentially with appropriate ordering. Essential architectural issues should be validated first. Each iteration may be validated with associate process, but should be considered extraneous if not useful for validation of future iterations. Therefore, for every requirement validation signifies completetion of a development iteraton. Conformance to these validation procedures should persist as the product development evolves.

As we describe development iterations that satisfy functional and system requirements. We declare validation processes that complete a development iteration. This approach seems very useful during system construction, but the concept can be extended to requirements and design iterations that segment the completion of the product definition. For example the system architecture specifies a network of tasks communicating with each other using a particular protocol. The mechanisms implemented to construct the tasks, and the methods used to transport communications are essential building blocks to accomplish the implementation. Furthermore communications protocol can be validated as an additional iteration, but only after the tasks are constructed and the communication mechanism is established. These orderings are therefore logically constructed, and eventually parrallel efforts can be declared. The feasability to implement parallel task developments depends on the availability of the essential resource requirements, and cost.

As this description of the iteratvie development process evolves, figure ?? is generated.We observe that the ordered sequence of events are the establishment of the project delieverables (sysetm requirement, design specification, implementation, verification, and validation). Ordering of the requirement set is important and neceissary. The initial architectural design must accommodate all functional requirements, but only sufficiently to allow implementation to validate the set of requirements declared as primary. Subsequent iterations of validation efforts occur as the system construction implements the associate requirement. The final iteration with this development process declares all requirements stipulated by the tracability matrix are validated.

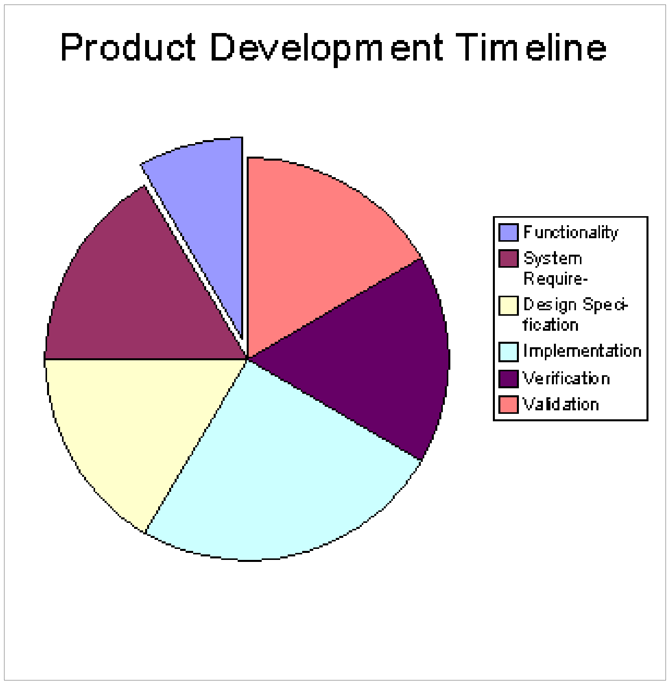
The Iterative Development Process



Timeline Considerations

At first glance, the iterative development process appears to have an indeterminate timeline, and a mesh of possiblly undesirable project progressions. It is always prefered to accomplish the project mission with one development iteration. The complex system definition suggests, however, that an interative development is more practically implementable. Therefore, the object of the iterative development process is to minimize, or prevent undesired corrective iteration without hindering the reality of there existence.

To present a more practical, and predictable, project layout we begin with a work breakdown for the project milestones. The process to accomplish this task appears in figure *Iterative Development Timeline*. We use a generic software system as an example to show the associate content of each task declared by the work breakdown.

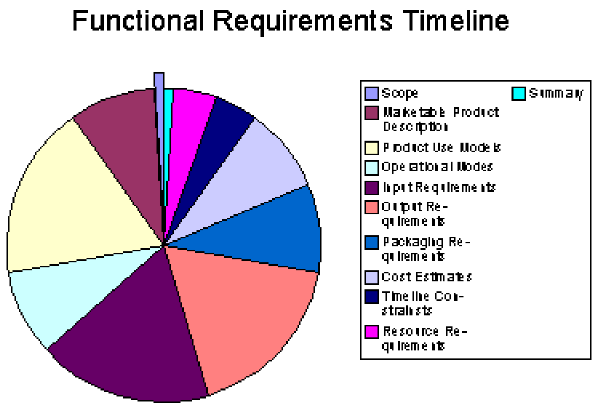


Product Development Outline

1. Functional Requirements
2. Systems Requirements
3. Design Specification
4. Implementation
5. Verification
6. Validation

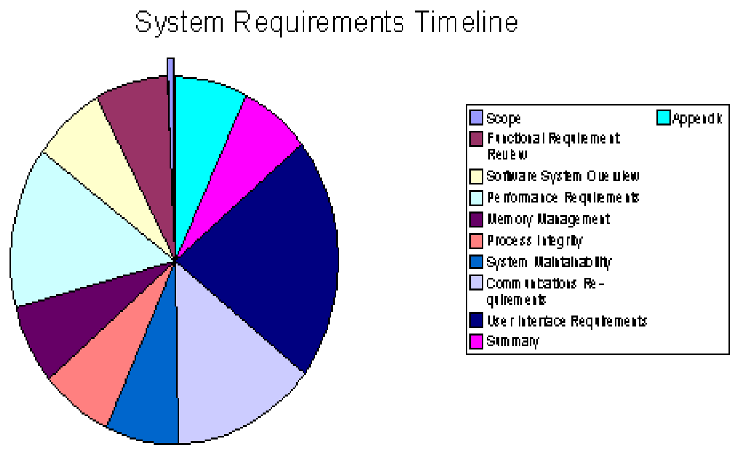
Initially, a functional requirement comes into existence. This functional requirement is developed with descriptive use models, input requiements, and desired results. Any information useful to develop the system requirement is requested. Characterization of the input data, algorithmic process requirements, and output specification are necessary to implement an appropriate system requirement. An outline for a functional requirement document follows:

Description	**Part**
Scope	0.01
Marketable Product Description	0.1
Product Use Models	0.2
Operational Modes	0.1
Input Requirements	0.2
Output Requirements	0.2
Packaging Requirements	0.1
Cost Estimates	0.1
Timeline Constrainsts	0.05
Resource Requirements	0.05
Summary	0.01

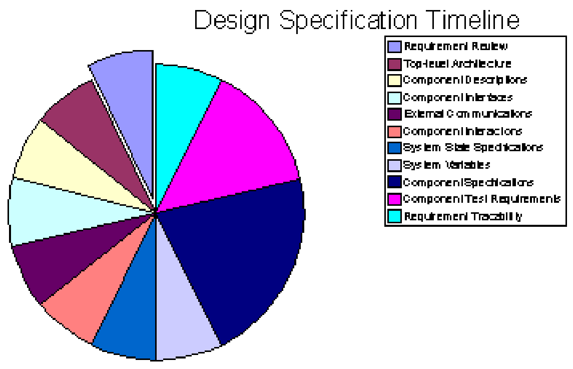


The requirements presented from the functional requirement is accommodated by the system requirement. The system requirement imposes additional structural and mechanical restrictions that provide a reliable, maintainable, as well as functional system. Memory management issues, inter-process communications, and device access are issues managed by the system requiement. Concepts involving debug and error handling are another requirement set declared by the system requirement. These concepts can be separated into sections of the system requiirement and developed separately. A sample breakdown of a system requirement is presented as follows:

Description	**part**
Scope	0.01
Functional Requirement Review	0.1
Software System Overview	0.1
Performance Requirements	0.2
Memory Management	0.1
Process Integrity	0.1
System Maintainability	0.1
Communications Requirements	0.2
User Interface Requirements	0.3
Summary	0.1
Appendix	0.1



Likewise the design specification, implementation, verfication, and validation milestones can be spearated into tasks declared by its work breakdown. A similar pie chart can be generated that represents the portion of time required to accomplish each task. The objective is to order the milestone tasks in such a maner as to incrementally construct the system that enhances parallel development efforts. By encouraging parallel task execution, the system will become assembled in less time (provided a finalized version of the functional requirements are available, and appropriate resource is allocated).



Description	***Part***
Requirement Review	0.1
Top-level Architecture	0.1
Component Descriptions	0.1
Component Interfaces	0.1
External Communications	0.1
Component Interactions	0.1
System State Specifications	0.1
System Variables	0.1
Component Specifications	0.3
Component Test Requirements	0.2
Requirement Tracability	0.1

The design specification declares a system architecture that satisfies the system requirements stipulated, and partitions the system into functional components. Each functional component declared during the design phase is restricted to conform to the system requirements as well as the product functional requirements. Therefore, the design document presents solution to resolve the functional requirement, and implements definition to develop methods to verify satisfaction of a requirement.

System Construction

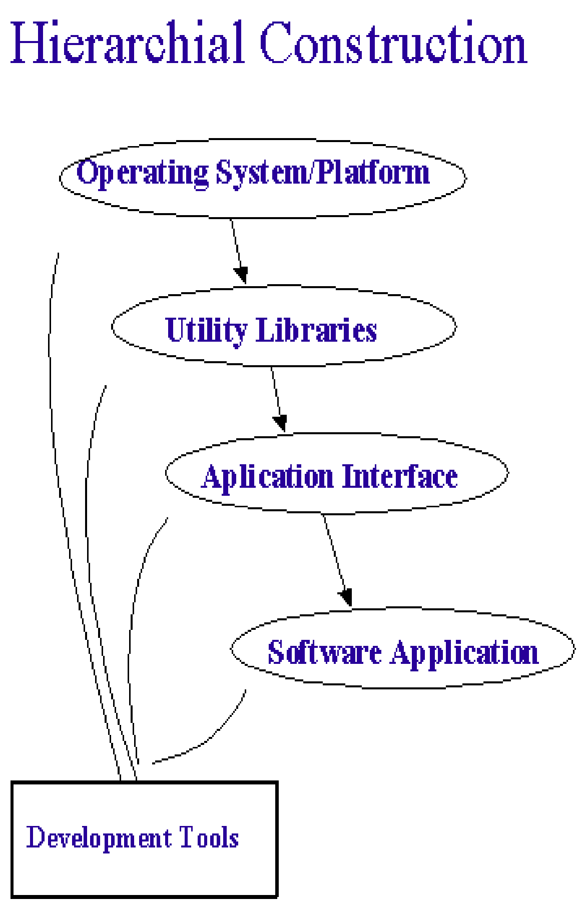
Reviewing the functional requirements document we observe that a functional requirement is associated with a command/action response. Each functional requirement can be represented as a functional block element with input, output, and purpose. Resolution of these functional requirements are stipulated by the design document that satisfies the additional system requirements. In other words, for each functional requirement there exists a set of system requirements (process integrity, memory management, communications, user interface, maintainability, etc.). These aspects of the system construction can be separated heirarichally from the lowest level being the operating system, then system utilities, then application interface, and finally the application. Compiler, debuggers, editors, and revision control utililities, as well as software utility libraries form the fundamental buillding blocks, and tools, to construct a marketable product. With these predefined resource, application programs can be constructed to implement software systems of asorted architectures. Selection of the appropriate utility libraries and development tools are a matter of preference, but should be influenced by the application/system requirements as well.

Thus far the system construction is implemented in heirarical fasion with the operating system at the lowest level. The figure reqpresents a process flow that constructs a software system. We observe that preliminary efforts to construct the apropriate product definition is to select the operating platform, the necessary utility libraries, possibly an application interface, and the needed development tools. These decisions are necessary prior to development of the product definition.

As the fundamental building blocks to construct the software system are acquired design speccification to construct the system application can begin. The top-level architecture represents the system components to implement. This top-level document describes any system device access, global memory, inter-process communications, or application specific objects appropriate at this scope of the application. I prefere to label this the skeleton for the softwre system.

With top-level view, segmenting the construction for the software system continues. Each system component declared has purpose and therefore functionlity. Any system interaction with another component are identified, and communication parameters are indicated. System variables, periipheral devices, shared memory, and system utiliities (such as sockets, pipes, fifo's, doors, threads, etc.) are presented.

With each system component exists a description that declares the components construction. These descriptions are presented as individual descriptions that describe the internal construction of the associate component. Utiliity library usages, application layers, are required to accomplish implementation, and are presented with the component description. Any special compilation requirements specific to the component implementation should likewise be identified. Many mult-platform designs implement an application interface layer that provides a means to arbitrate the target operating environment. Implementation of an application interface layer is therefore a very useful, and practical, method for implementation of an application targeting multi-platform environments.



As the design specification becomes developed we conceptualize a work breakdown describing its implementation. Generally, work breakdowns are constructed with task definitions that require an average of two weeks to accomplish. These task items can likewise become prioritized, and ordered, appropriately to show the possible paths to accomplish the implementation. Depending on resource availability, and man power, the ordered breakdown can be implemented with parllel effort to reduce this time estimate for an accomplished implementation.

As with each system component declared during the design phase, methods to verify its implementation and functionality are required. A design implementation can not be declared as functional and correct until it has been thouroughly tested. All functional requirements stipulated by the functional requirements document must be satisfied. System integrity is also a major issue that requires validation. These verification/validation methods are declared in the Test Plan and Proceedures documentation, and are additional effort associated with the design implementation. Therefore, for every system component design there coexists a test plan and proceedure for its verification and validation.

The Iterative Processes

As the process for system construction evolves the clarity of definition converges to understanding. Initially, complex system definition may postpone aspects of an application that can be included as enhancements to the product at a later time. Another development iteration then continues the product development.

Product enhancement, and unresolved design issues, are reasons for implementation of an iterative development process. Multi-platform development may be another reason to implement an iterative development process. Many other reasons may exist the entail a product development iteration, and therefore the Iterative Development Process accommodates incremental evolution of a product definition. A set of criterion that help clarify the necessity for a product iteration is developed.

Iteration Criterion

As already mensioned, neccessity is our first criterion to identify a product development iteration. The necessity for continued development implies incomplete satisfaction of the product functional requirement. This condition may occur due to postponement of the initial requirement satisfaction, or additional functional requirements proposed as a product enhancement. For either case the system requirement, design specification, component verification/validation, and the tracability matrix are effected.

As the necessity for a product development iteration becomes identified, the feasability for the required satisfaction becomes a question.

- Feasability may include realizability of the requirement, if current state technological capabilities are insufficient, then the required functionality may not be realizable.
- Feasability, may include practical timeline issues as well. If resources are not available, or not enough man power, then it may be wise to postpone satisfaction of the object requirement for a later iteration.
- Feasability likewise includes the overall impact on the system performance, integrity, or maintainability. A functional requirement may require alteration to the system architecture that would require a major modification and delay with the product delivery. This condition is probably the most undesired and damaging to a project's successful delivery.

The feasability to realize a requirement, leads us to implementation that follow the classical development cycle: requirements, design, implementation, itegration, and validation. The remaining criterion are therefore cost, time, and available resource requirements.

To summarize the criterion to accomplish a product development iteration are listed below:

- Neccessity
- Feasability
- Cost, time, and resources
- Improved Product Performance or Functionality

If the product functionality, or performance, is not improved at the completion of the development iteration, then delivery of the released product does not show any improvement from its predecessor and the customer will observe time consuming product updates that do nothing for the customers welfare.

Process Flow

As discussed earlier, a development project has an initial development cycle that instigate a system architecture for implementation. The system requirements are developed from the product functional requirement. The design specification declares a system architecture, and its associated components. A tracability matrix is developed to map the identified requirement with its validation process. A work breakdown is generated to develop design specification, construction, and test criterion for accomplishing the initial product development cycle.

The initial product release, therefore, represents the completion of the initial development iteration. The initial product release is therefore satifies the minimal set of functional requirements, and the architecture developed to accommodate all requirements declared by the functional requirement. The incomplete aspect associated with the product development requires negotiation and agreement with customer to accomplish the order plan for product completion.

The stage is now set to accomplish the second development iteration. Progressive development iterations therefore satisfies incomplete functional requirements, or improves previous performance. A development iteration therefore declares the requirement set to be satisfied, or associate performance issue being addressed. The sections of the design specification stipulating the requirements to be satisfied by the development iteration are completed. The associate module test are completed, and the test procedures are finalized. The development iteration becomes complete when all associate documentation, system construction, implementaton, verification, and validation are complete.