Software DEVELOPMENT

methodology

Woodbridge Corporate Plaza
P.O. Box 158
Woodbridge, New Jersey 07095
732.842.2684

Software Development Methodology

The aim of HudsonIT's software development methodology is to produce a high quality product in a timely and cost-effective manner for ourselves and our customers. Since HudsonIT is in the business of providing customized software solutions for our customers, the methodology is directed towards creating products that address the characteristics of client driven software development efforts.

HudsonIT's Quality Management System (QMS) defines the processes and procedures that we utilize to delivery solutions to our customers. It focuses the basis for our ISO certification process. This methodology is one of a family of related methodologies employed by HudsonIT to deliver high quality OSS solutions to our customers.

Goals

Four basic goals govern the HudsonIT Software Development Methodology to insure a quality product:

- Produce documents about product requirements, specifications, etc. in a clear, concise, precise and complete manner. Through these documents, HudsonIT communicates product characteristics both to its clients and to HudsonIT's developers.
- Identify errors, omissions, and risks early in the development cycle so that correction costs are minimized.
- Produce the highest quality product possible within resource constraints (budget, schedule).
- Discover the most appropriate and cost effective technological solutions for the product (hardware, third party software, user interface, database organization, etc.).

Meeting the Goals

The HudsonIT Software Development Methodology is comprised of four major components which guide the engineering and development process toward meeting the goals:

1. A set of standard documents that are produced using pre-designed templates.
2. Quality Assurance through attribute definition and quantification.
3. A process based on inspections and reviews.
4. An evolutionary development and product delivery approach.

All development projects undertaken at HudsonIT involve understanding the requirements for the project. Requirements for these projects are more then just functional (i.e., *What* are we building), they include quality requirements that detail *How Well* the product will perform as well as resource requirements that detail *How Much* we can spend on this project. It is our belief that for any project to be successful, it must meet all of these requirements. Figure 1 illustrates how these requirements relate to one another.

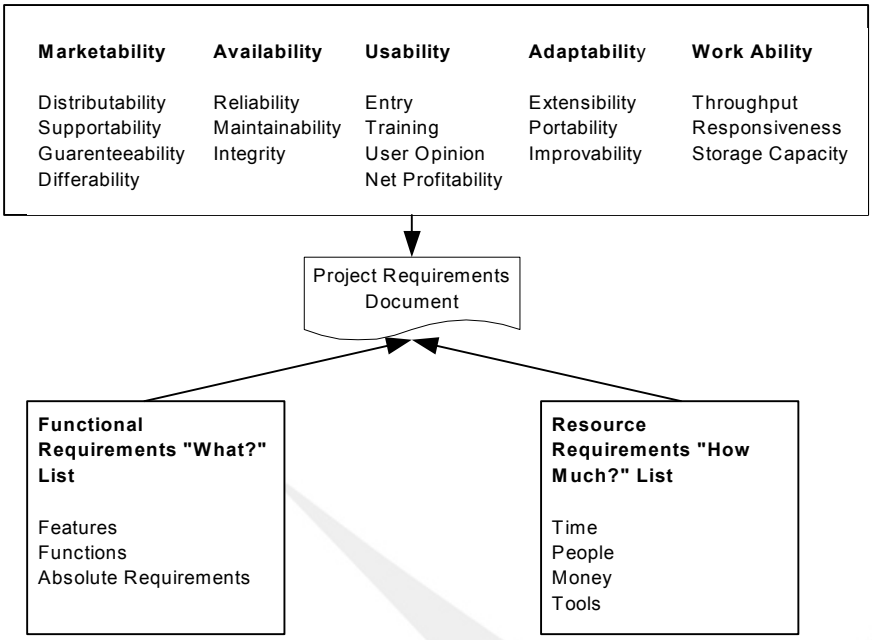

Figure 1. Requirements Hierarchy. Project requirements can be grouped into three categories: functional, quality, and budgetary. The most problematic centers around the Quality Requirements; which are not *what* we want, but *how well* we want it to perform.

Once the requirements phase of a project is completed, the evolutionary software development phase of the methodology is utilized. This process is described in Figure 2, below

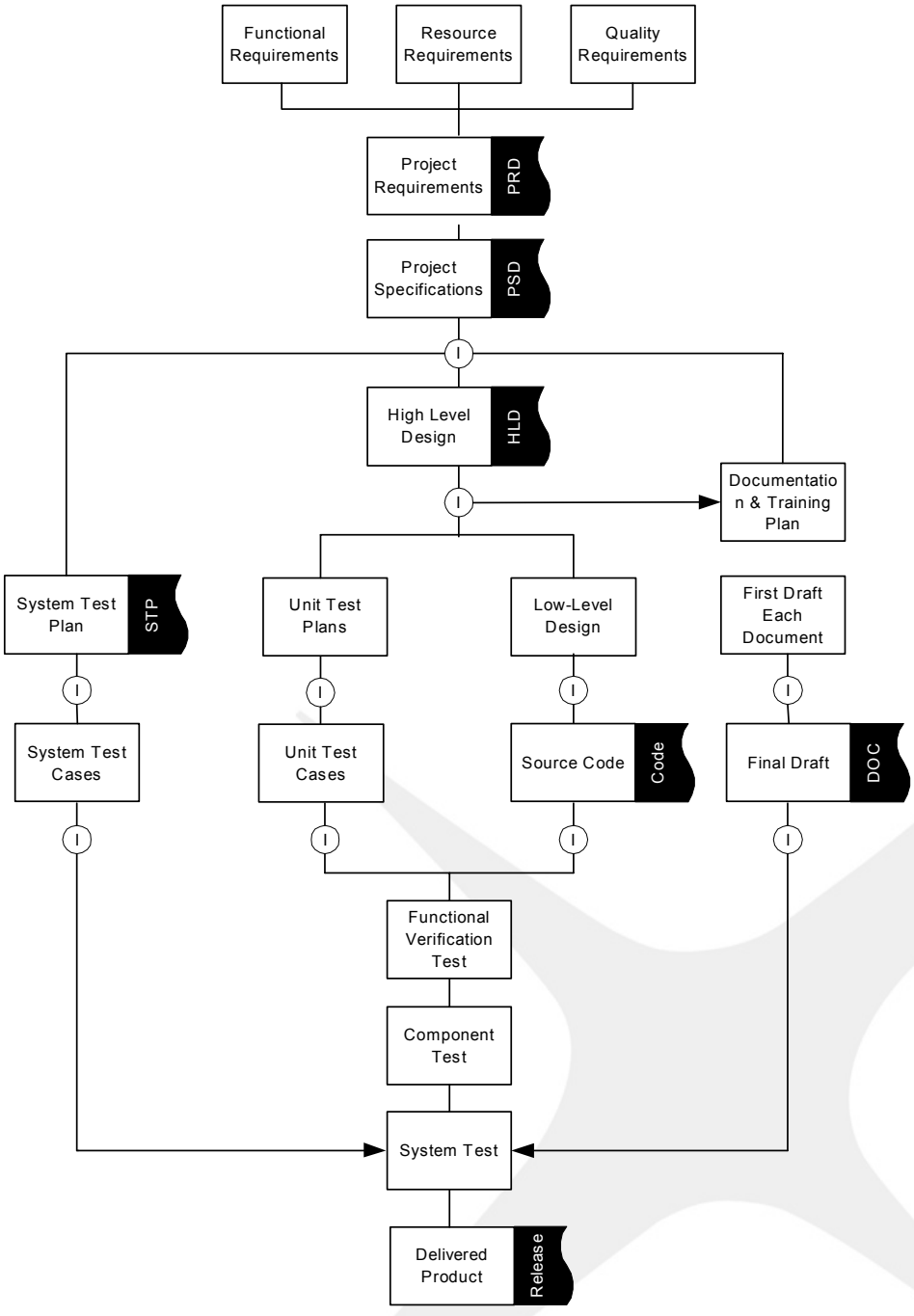

Figure 2: Software Development Methodology

Documentation

Standard documents regularize the process and insure that all needed steps are included.

Pre-designed templates, through which the documents are created, prescribe the document organization, but not product specifics. They insure that each document conforms to a standard and that all needed areas are covered. However, templates change and evolve, sometimes motivated by HudsonIT's own experience using the templates and at other times to adapt to the development practices of the client organizations. Client motivated changes are sometimes incorporated into a standard template when they prove to enhance the process.

HudsonIT uses most, if not all, of the following documents in the course of a development project:

- **Project Requirements Document (PRD)**
  A description of project requirements including Functional Requirements (what the product will do), Quality Objectives (how well the product will perform), and Resource Limitations (constraints in terms of time, money, people, etc.). Quality objectives are discussed in more detail under Quality Assurance.
- **Project Specification Document (PSD)**
  A complete specification of all aspects of a product that are visible to the user (either a human or a machine user). The PSD also describes key internal components when they highlight the product's value to the user, for example, a forecasting algorithm.
- **System Features Demonstration (SFD)**
  A software package that demonstrates to the user the look and feel of the product. The SFD is typically developed with the user through a series of Work in Progress reviews and Joint Application Development sessions.
- **High Level Design (HLD)**
  Details the solution used to create a product.
- **Test Plans**
  Documents the test plans, test methods, and test cases that will be used to insure that product requirements and specifications are met.
- **User Documentation**
  One or more documents designed to aid the user to utilize the product. User documentation should be directly derivable from the PRD and PSD.
- **Miscellaneous Documentation**
  Training and job aids, sales tools, etc.
- **Computer Code**
  While the computer code is not formal documentation, it is included in this list because it is subject to inspection.

Major control documents (e.g., PRD, PSD, HLD, Test Plans) use *tags* to identify, each function, concept, attribute, solution, and evolutionary step of the product. Each tag is a unique identifier, such as "User Interface." Tags are used to cross reference elements between documents and throughout the development process to insure that each element is inspected, designed, built, and tested. They and their associated elements are often hierarchically organized as in the following example:

UserInterface
UserInterface.Graphics
UserInterface.Graphics.Mouse
UserInterface.Graphics.Mouse.KeyClick

Because the Project Requirements Document (PRD) defines the major characteristics of a product, it is the critical document in the product design and development cycle. The PRD differs substantially from a Function Requirements Document since it contains, not only functional requirements, but also resource limitations and quality objectives. Resource limitations define the possible "solution area" in terms of the available budget, personnel and other limiting factors.

Quality Assurance

Quality objectives are defined in terms of attributes. An attribute is a quantitative description of how well a project task will perform. Attributes define simple characteristics such as load, response time, and reliability, which are expressed in straight-forward metrics (percent availability, transactions per second. etc.).

However, attribute definition is most important in terms of dealing with a product's qualitative aspects, which might otherwise be defined vaguely or not at all. By quantifying such qualitative goals as "increased productivity" and "ease of use," they can be measured and tested. For example, "ease of use" can be quantified by specifying the amount of user training time needed for a user with a pre-assessed proficiency level to attain a specified proficiency level.

All attributes must be quantifiable and measurable. In order to accomplish this, each is defined in a table the template includes the following:

- ***Scale -*** indicates the unit of measure of what we are trying to measure (e.g., minutes, bytes, transactions).
- ***Test*** - indicates some way to measure along the scale specified in a practical, economic and accurate way.
- ***Worst Limit*** - indicates the worst acceptable level of performance (i.e., the border between failure and non-failure).
- ***Plan*** - describes the design goal of the systems.

- ***Best Limit*** - identifies the best results that can be achieved for the attribute in the context of comparable products and known technology (it also indicates an unexploited potential for the system or an indication of excessive risk).
- ***Now*** - indicates the level achieved with a current product or procedure (if there is one).

To continue with our ease of use example from above, the following attribute specification might be developed for a product:

Tag	Definition	Scale	Test	Worst	Plan	Best	Now
TRAIN	Training need	Hours to solo ability	Sample of 20 new users	2	.5	0	N/A

In this example, the worst acceptable performance on the TRAIN attribute would be 2 hours of training to reach the level of proficiency where the user can operate the system on their own. We are planning for a half hour training period. The best case would be that the users require no training at all. Any attribute where the Planned measurement is difficult to attain than the Best measurement may indicate that excessive risk is being taken on the project with respect to that attribute.

These attribute characteristics provide several benefits. For our Clients, the advantages are:

- Certainty about what they want;
- Clarification of what is required; and
- Testability to insure that what is delivered matches what is specified.

For HudsonIT, it offers numerous advantages, including the following:

- Fail safe mechanism for our projects to insure that requirements are understood;
- We are in a better position to identify when a client changes requirements;
- We can speak to our clients on the basis of results to be delivered and the benefits;
- We can perform better estimates of the cost of producing the product desired; and
- We can be better focused on achieving results and freer to select appropriate technology for meeting the goals.

Inspection

Inspection occurs at several stages of development and is designed to insure that requirements, specifications and design are:

- Complete;
- Consistent across the project;
- Consistent with client goals;
- Realistic; and
- Clear and unambiguous.

HudsonIT's Quality Management System's Inspection Process has the following characteristics:

- ***Independent Audit*** – An author's contribution to a document is inspected by someone else. An author can be present during a review.
- ***Trained Leaders*** – Inspections are moderated by someone trained in the methodology and the inspection process.
- ***Strict Rules*** – There are established rules for how an inspection is conducted. Such rules determine the length of the inspection, amount of the discussion permitted, etc. An important rule is the "Exit Criteria." These are preset and determine whether a document has passed an inspection or whether re-inspection is needed.
- ***Specialized Roles*** – Each inspection participant has a specialized role. The moderator and the scribe (person who records defects and statistics) are participants with pre-defined roles. Other roles may include, for example, Documentation, Quality Assurance, and Marketing. A participant may bring specialized experience or a specialized concern to an inspection. Alternatively, a participant's role may be functional in relation to a document under inspection. A participant playing a functional role might insure explicit consistence with other documents or might inspect a document from back to front to insure that the entire document is well inspected.
- ***Peer Review*** – Inspections are carried out by HudsonIT personnel (and sometimes, clients) with different experience and responsibility levels. Within the context of an inspection, all inspectors are peers.
- ***Management Data*** – Statistics are kept regarding time spent, defects found, etc. within inspections. These statistics are used as feedback to improve the process.

Several defect levels are recognized:

- **Minor** – An error or omission that is not considered serious. Minor defects are often textual.
- **Major** – A serious error or omission (e.g., a missing condition in an algorithm).
- **Super Major** – An error or omission which is so serious that, if not corrected, endangers the entire project. A missing or needed sub-system is an example of a super major defect.
- **Question** – A question may or may not represent a defect. It represents an area for which further information is required in order to determine if a defect exists. Questions are usually evaluated during the Casual Analysis step of the inspection process.

An established process leads inspections. The following steps are utilized:

1. **Document Preparation**
2. **Kickoff –** A short meeting where the moderator explains the nature of the document to be inspected, distributes the document, and sets any special rules for the inspection.
3. **Inspection Meeting Preparation –** Time spent by each inspector in inspecting the document.
4. **Inspection Meeting –** Review inspection findings. The meeting is usually limited to two hours.
5. **Casual Analysis –** Additional time, usually immediately following an inspection meeting, when results are analyzed. During this analysis, questions are discussed and a determination is made about whether the document meets the inspection exit criteria.
6. **Re-Work –** The document is re-worked to address defects. It is not required (nor desirable) that each defect result in changes to a document, but if it does not, it must be because of a valid reason.
7. **Re-Inspection –** Occurs only when a document failed the original inspection.

Time and resources for the inspection are built into the schedules and project plans.

Evolutionary Development & Product Delivery

The final component of HudsonIT's Software Development Methodology is evolutionary development and product delivery. Where possible (in accord with requirements, expectations, and needs of the client) HudsonIT analyzes, designs, builds, tests, and delivers a product in an evolutionary manner.

Evolutionary development provides the user functionality early in the development stage; it allows for corrections to be made to address problems which were not or could not be discovered in the requirements and specifications stages; it permits the product to adapt to changing technological and business environments; and it takes advantage of the user's greater understanding of their own requirements which come with use and experience.

Traditional software development methodologies set the course at the end of the requirements phase and do not provide another point for feedback until the system is delivered (See Figure 3.) This approach does not allow the system to adjust both the design and the objectives of the project based on observed realities.

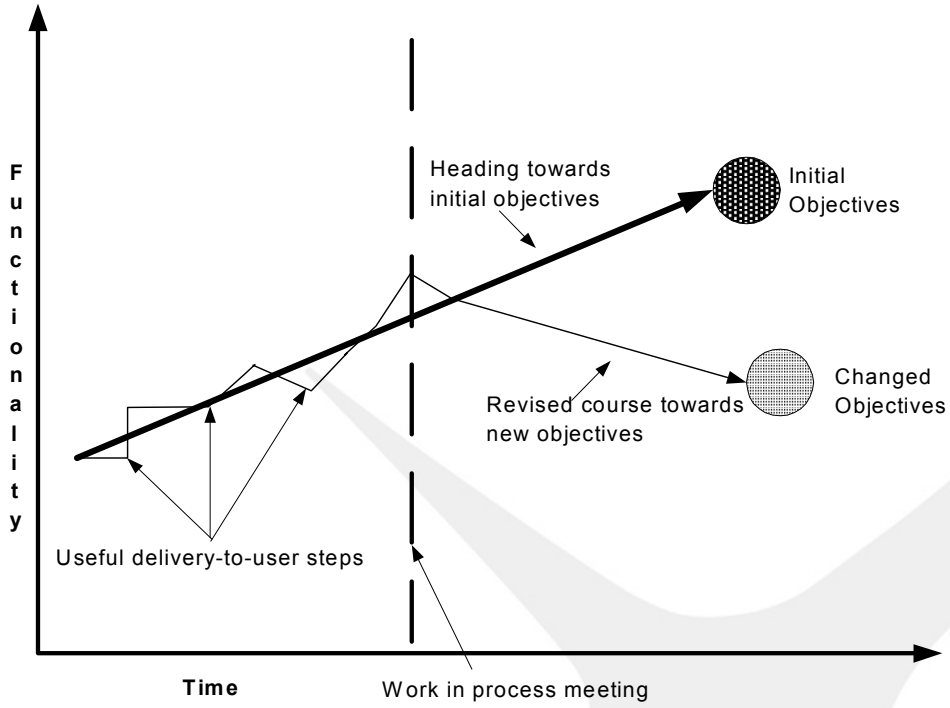

Figure 3. Evolutionary delivery method provides much better control over definition and measurement of objectives