**TOOL: Tipsheet: Iterative Software Development**

*Author: Shannon Turlington (IntraHealth International)*

Objective: Develop any new software features using an iterative approach.

**Iterative development** is a software development process that supports development of a system incrementally. The process builds in regular and frequent cycles for feedback from end users to a working, although incomplete, system. This enables the development team to deepen their learning and understanding of the system as development progresses and adapt to changes in requirements as they come up. This process helps ensure that the end result will be a system that best meets the current needs of all stakeholders, yet can change or grow to adjust for new and unforeseen needs.

In this way, the iterative development process reflects what most of us actually experience as the reality of software development, that requirements cannot be frozen and new requirements are frequently discovered once programming is under way. This methodology enables the software development team to remain flexible as new needs come up or the unforeseen happens, which they always do. Instead of trying to impose artificial deadlines and milestones on a project, as is common in traditional models of software development, the team can tailor the development process to conform to the situation as it actually exists, and therefore increase the chances for success.

Principles to Follow

* Manage requirements not tasks, based on user goals and nonfunctional requirements.
* Manage to meet business goals, due dates, and budgets. Be willing to change requirements to fit these, not the other way around.
* Practice the art of doing only what is necessary and no more. Every requirement should satisfy a user goal.
* Learn as you go; be adaptable to new or changing business needs that become clear only after development begins.
* Analyze existing implementations frequently to assess whether they are meeting business goals.
* Begin with a simple implementation of a subset of the requirements that demonstrates key aspects of the system.
* Design around isolated, easy-to-find modules that group small sets of related requirements. Complete or re-code one module per iteration.
* Work in short cycles composed of overlapping phases: requirements, design, programming, testing.
* During the iteration, the external customer or project manager cannot change the scope for that iteration, but the development team may change the scope by dropping features if the end date will not be met.
* Build on what was built before and produce a working product at each stage. Gather customer feedback at the end of each iteration. Plan for a release every 3-4 iterations.
* Any difficulty in design, coding, and testing a module should signal the need for redesign or re-coding.
* Modifications should become easier to make as the iterations progress. If not, redesign is needed.

# The Iterative Development Process

A couple of decisions have to be made before plunging into development. The first is how long the iterations will be. Depending on the complexity of the development, they can last anywhere from one to six weeks. The important thing to remember is to try to produce something that works at the end of the development cycle, even if it’s not final, and to work on only the number of requirements that can be completed within that timeframe. In other words, don’t extend the deadline to accommodate more requirements; rather, scale back the requirements to meet the deadline.

The second decision is how formal or informal the development process should be. This will impact the kinds of documentation that need to be produced and the intensity of testing. It may be easiest to document as you go, even if it’s just notes in a wiki that can be cleaned up later. At the very least, keep an ongoing list of changes made, features to be addressed, and bugs or issues that arise during development. A more formal process may require a full-time documentation manager who can manage changes to the use cases and issues list, as well as automated and human testing from test cases. The key is flexibility; adapt the processes to fit the project, not the other way around.

Here are the general steps in each iteration of development:

1. Planning: Before starting the iteration, sit down with the development team and ask these questions: What have we learned from the previous iteration? Have the vision, goals, and/or scope of this project changed? What feature subset will be coded in this iteration?
2. Design and develop in order to produce a working end product.
3. Test and correct bugs; this phase can overlap the previous phase.
4. Update documentation and generate user help materials, if needed; this phase can overlap the previous phase.
5. Collect feedback from the user group, which leads to planning for the next iteration.
6. Update the issues lists and related documentation with changes based on the user group’s feedback. Refine the project plan and related estimates, such as cost estimates.

# Guidelines for Making Iterative Development Work

These guidelines are intended to combat the typical issues that arise in software development projects:

1. **Identify a project point person:** The Project Manager should identify a point person from the Stakeholder Leadership Group, who must agree before work begins to be available to attend meetings, answer questions, and review work, often with quick turnaround times. The project point person needs to have decision-making authority, including approval of budgets. If the project point person does not fulfill this obligation, this should be documented by the Project Manager as early as possible in the project, and a new project point person should be identified or the project put on hold.
2. **Generate a specific requirements list:** For each iteration, a clear set of requirements should be generated that can be translated into specific actionable items for the software developers to work from. Often, it is the Project Manager’s job to generate these requirements and get approval of them by stakeholders.
3. **Set a definite start and end date for the iteration:**  The end date should be immovable; requirements should be changed to accommodate the end date, not the other way around. While working with the developers, check in and request submissions of completed work daily to ensure they stay on track.
4. **Institute a change request process:** During the development time, any new feature requests or changes to the pre-approved requirements should be submitted as change requests. The change request should result in a change to the scope of work to ensure that the project can still be completed in the allotted timeframe; for example, a lower priority feature may be dropped or postponed in favor of a new request. Change requests that cannot be accommodated should be placed on the requirements list for the next iteration.
5. **Treat each iteration as a new project:** Once the end date arrives, a new requirements list should be generated and end date set for the subsequent iteration. It should not be assumed that the same developer will complete subsequent iterations or that the developer will be “on call” to make minor changes once the project has ended.

**Tags:** guidelines;iterative development; pilot;software and systems; software development; tipsheet