

Keys to Software Development Success

From NASA's Software Engineering Lab

Adapted from
fdd.gsfc.nasa.gov/ra_docs/RA_Cover.html



Keys to Software Development Success



- Understand the environment.
 - Determine
 - the nature of the problem
 - the limits and capabilities of the staff
 - the support software and hardware environment
 - Collect information necessary to determine all of the above.

Keys to Software Development Success



- Match the process to the environment.
 - Select and tailor the software process.
 - If you don't intend to or cannot enforce a standard or a procedure, do not include it in the plan.
 - Make data collection, analysis, and reporting integral parts of the development methodology.

Keys to Software Development Success



- Don't attempt to use excessively foreign technology.
 - The technology, and the risk that accompanies its adoption, must suit the local environment.

Keys to Software Development Success



- Experiment to improve the process.
 - Identify candidate areas for improvement.
 - Experiment with new techniques or extensions to the process.
 - Measure the impact.
 - Select candidate extensions with a high likelihood of success.

Keys to Software Development Success



- Be careful not to change too many things at once so that the results from each change can be isolated.
- Not all changes will lead to improvement; be prepared to back off at key points.

Do's and Don'ts for Project Success

Nine DOs for Project Success



1. Develop and adhere to a software development plan



- A software development plan
 - sets project goals
 - specifies the project organization and responsibilities
 - defines the development methodology that will be used
 - clearly documents any deviations from standard procedures

1. Develop and adhere to a software development plan



- This plan includes
 - project estimates and their rationale,
 - specifies intermediate and end products, product completion and acceptance criteria, and mechanisms for accounting progress
- It identifies risks and prepares contingency plans.

1. Develop and adhere to a software development plan



- The plan is a living document.
 - Record updates to project estimates, risks, and approaches at key milestones.
 - Periodically, audit the team for adherence to the plan.

2. Empower project personnel



- Allow people to contribute fully to the project solution.
- Each team member should have clearly identifiable responsibilities.
- Decision making should be delegated to specific team members.

2. Empower project personnel



- All of the team must have a precise understanding of project goals and limitations.
- The team must understand the development methodology and standards to be followed.

3. Minimize the bureaucracy



- Establish the minimum documentation level and meeting schedule necessary to fully communicate status and decisions among team members and management.
- Don't try to address difficulties by adding more meetings!

4. Establish and manage the software baseline



- Stabilize requirements and specifications as early as possible.
- Keep a detailed list of all TBD (to be determined) items.

4. Establish and manage the software baseline



- Classify them by severity of impact in terms of size, cost, & schedule.
 - set priorities for their resolution
 - assign appropriate personnel to resolve TBD items
 - ensure timely resolution
- Estimate and document the impact to costs and schedules of each specification change.

5. Take periodic snapshots of project health and progress, replanning when necessary



- Compare the project's progress, product and process measures against the project's plan.
- Compare the current project with similar projects and with measurement models for the organization.

5. Take periodic snapshots of project health and progress, replanning when necessary



- Replan when the management team agrees that there is a significant deviation.
- Do not hesitate to reduce the scope of the work when project parameters dictate.

6. Reestimate system size, staff effort, and schedules regularly



- More information is learned about the size and complexity of the problem as the project progresses.
- Do not insist on maintaining original estimates.
- There is nothing wrong with realizing that size has been under estimated or that productivity has been over estimated.

7. Define and manage phase transitions



- Much time can be lost in the transition from one phase to the next before the start of each new phase.
- Review progress to date.
- Set objectives for the next phase.

8. Foster a team spirit



- Maximize commonality and minimize differences among team members.
- Have everyone follow the same rules.
- Struggle through difficulties and celebrate successes together as a unit, helping and applauding each other along the way.

9. Start the project with a small senior staff



- A small group of experienced senior people should be involved from the beginning to determine the software development plan.

Do's and Don'ts for Project Success

Eight DON'Ts for Project Success



1. Don't allow team members to proceed in an undisciplined manner



- Developing reliable, high-quality software at low cost is a disciplined application of a set of defined principles, methods, practices, and techniques.

2. Don't set unreasonable goals



- Setting unrealistic goals is worse than not setting any.
- Setting solid reachable goals early in the project usually leads to continued success.
- Success leads to more success.

3. Don't implement changes without assessing their impact and obtaining proper approval



- Estimate the cost and schedule impact of each change.
- Little changes add up over time.
- Explore options with the change originator.

4. Don't “gold plate”



- Implement only what is required.
- Little items add up over time and can cause a delay in the schedule.
- Deviations from the approved design may not satisfy the requirements.

5. Don't overstaff



- Bring people onto the project only when there is productive work for them to do.
- A small senior team is best equipped to organize and determine the direction of the project at the beginning.
- However, be careful to provide adequate staff for a thorough requirements analysis.

6. Don't assume that an intermediate schedule slippage can be absorbed later



- Developers tend to assume that the team will be more productive later on in a phase.
- The productivity of the team will not change appreciably as the project approaches completion of a phase, especially in the later development phases when the process is more sequential.

6. Don't assume that an intermediate schedule slippage can be absorbed later



- Don't assume that late pieces of design or code will fit into the system with less integration effort than the other parts of the system required.
- The developers' work will not be of higher quality later in the project than it was earlier.

7. Don't relax standards in an attempt to reduce costs



- Relaxing standards in an attempt to meet a near-term deadline tends to lower the quality of intermediate products and leads to more rework later in the life cycle.
- It also sends the message to the team that schedules are more important than quality.

8. Don't assume that a large amount of documentation ensures success



- Each phase of the life cycle does not necessarily require a formally produced document to provide a clear starting point for the next phase.
- Determine the level of formality and amount of detail required in the documentation based on the project size, life cycle duration, and lifetime of the system.