Core software engineering process components

- **Specification –** what the system should do and its development constraints
- **Development –** production of the software system
- **Validation –** checking that the software is what the customer wants
- **Evolution –** changing code in response to demands

Critical success factors in a systems project

- Top management support
- A sound methodology
- Solid technical leadership by someone who has successfully completed a similar project

Interdependent Factors in Project Success

- Cost
- Quality
- Speed
- Risk

The two that are most important are risk and quality. The system must work and successfully meet user requirement. This leaves speed(time) and cost(money) to be adjusted accordingly.

Data migration and implementation

Two important factors, that are often forgotten are data migration and the system implementation itself. Data migration should be planned **early on** in the project.

How to prevent failure of a systems project

Use a specific methodology, coding isn't the only thing that matters

Using a structured systems development methodology is one of the critical success factors in a systems development project

Don't work backwards from exact project completion date

Unfortunately, this is an all too common practice in the IT community. Managers may make a decision about when a new or re-engineered system would be useful to have in production without the necessary technical knowledge to determine whether or not it is possible to accomplish successfully in the given time period. Projects never go smoothly from Strategy to Implementation. Working back from exact date can bring up these problems :

- Failure to perform careful analysis resulting in critical new requirements being uncovered late in the development process
- Failure to take data migration into account
- Failure to accurately assess the political climate of the organization vis à vis the project
- Failure to enlist approval at all levels of the user community

You must set a realistic timetable for the completion of a systems project
Expect that some deadlines will slip as unexpected setbacks inevitably arise.
Forcing a deadline to drive the schedule will cause corners to be cut, greatly increasing the risk of the project.

Build a data model for the database beforehand

The data model is the core of the system. Without a carefully constructed data model, any system is doomed to failure, or at least to not meeting the users' needs and requirements. Data models should be **directly under the control** of the **technical lead**. Each subsystem should be integrated into the whole prior to development. Models should be designed and audited, and re-audited, and re-audited until no more errors are uncovered. The audits should done by more people than just the technical lead

Hire a Technical Lead with experience in a similar system project.

The person in the role of project Technical Lead must be experienced. He/she should have completed other successful similar projects.

Less developers, more quality and focus.

More is not always better. This is especially true in the case of development teams. A project with a fewer, more skilled developers, carefully supervised by the appropriate technical lead has a much greater chance of success than one where hordes of developers are cranking out mountains of code each day.

Get the right tool for the job

This motto is as true for building systems as it is for building houses. You have to remember that even the basics of the development process are greatly influenced by the tools selected. But the chosen “right tool” can only be used if the necessary expertise is in place for it. Don’t aim for the newest, “most shiny” technology , go with the one that the team is comfortable working in, this will greatly increase the quality of the project.

Put more effort in to proper data migration

The data migration phase of a project can consume up to 30% of the total project resources and yet considerably small amount of attention is paid to it. The cost of data migration tends to go unforeseen, until far too late in the project schedule. The most common flaw is that too few resources are invested in it. Many project schedules will list data migration as a single task in a development effort. However, by project’s end it will be evident that data migration was indeed an entirely separate project, highly dependent upon other parts of the development project.

For these reasons, it is very important to plan ahead for data migration on any project. In particular, implementing generic data structures, which are desirable for increasing the lifetime of the system, can create significantly more complex migration issues than implementing traditional system designs.

Never skip the testing phase.

No system was ever created completely bug-free. The time spent to thoroughly test any system before placing it into production can save much more time in the long run.

Test plan should describe not only the tests to be run but also how test failures or variances will be handled. Especially for a large system, it is not practicable to wait until the end of the Build phase to start developing this test plan. There must be some overlap. As each module is tuned and passes unit-level testing, it is stable enough so that formal tests can be planned.

It is not necessarily true that every test failure or variance leads to modifications of the system prior to production. Within the Test phase, it will be necessary to re-audit the design process, perform system and user-level tests, audit the quality of the documentation, test the internal controls and backup and recovery procedures, and in all ways ascertain the fitness of the system to move into a production environment. When the bug filled system is put into production, the users will get furious.

Decide at the start that you are going to re-engineer your business to fit the limitations of the COTS package.

Commercial,off-the-shelf packages usually imply a specific method of doing business with a corresponding set of capabilities. A managerial decision must be made that whatever the package does is good enough and that customization is not an option. If you do not follow this philosophy, implementing a COTS package is no cheaper and no less risky than building a system from scratch.

Conclusion

- Don't cut corners, methodologically. In the long run, this results in system failure or an inadequate system that doesn't meet the users' needs.
- Audit each major deliverable and step along the way for accuracy and correctness.
- Carefully monitor top management support for the project. Make sure that managers are aware of the progress of the team.
- Secure the correct technical lead for the project.

Software Engineering Process

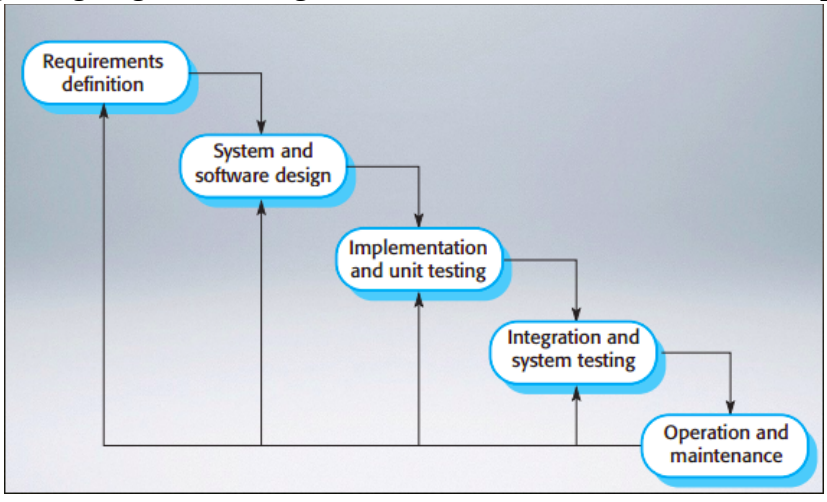
Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

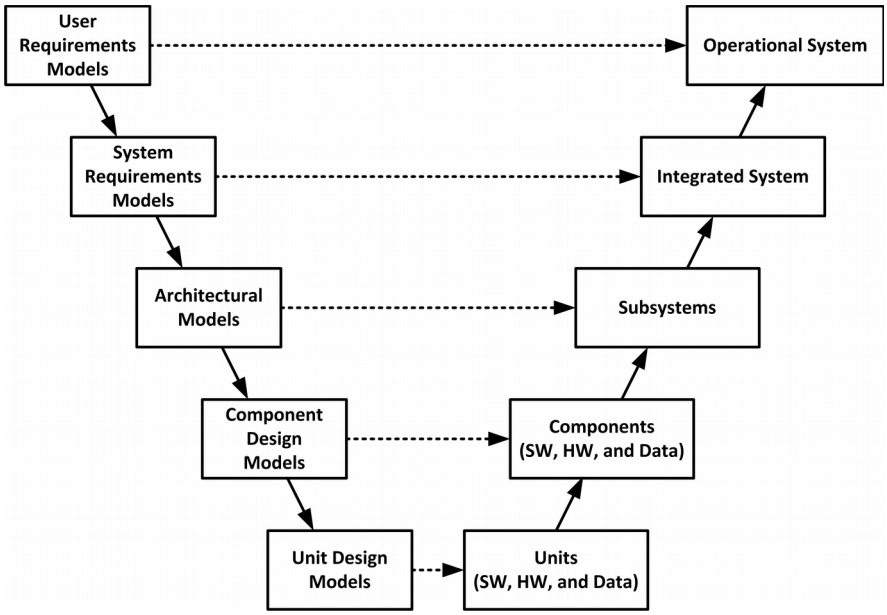
When software engineering

- You are not building something **for yourself.** Need to understand/agree what someone else wants
- You are not building something **by yourself.** You have to work with other people
- You are not building it for people **like yourself**
  - It's not just "computer experts" that will be using your program. Design it in such way that even your grandmother could learn how to use it.
- You are not working on code **you wrote yourself**
  - You will be working with a lot of different programmers, who might not understand your variable naming, your code style. Don't write "spaghetti" code, use standards.

Before you start coding

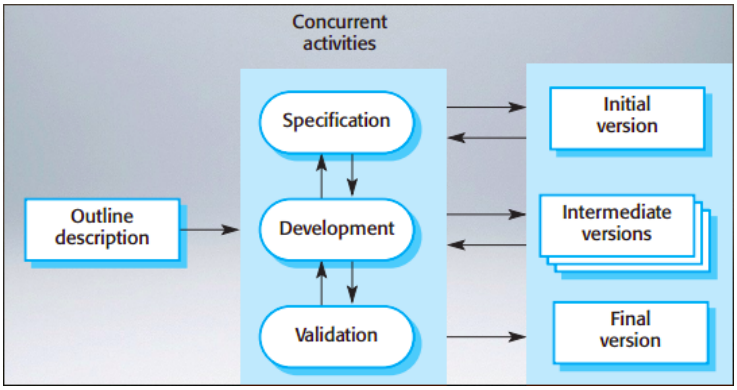
- Agree what you are building
- Decide how to tell that you did it right
- Design it in such way, that everyone knows what they are working on
- Coordinate how will everyone build it
- Plan testing phases for the project
- Expect change in schedule
  - Not everything will go right, there might be some setbacks, that need to be adapted to

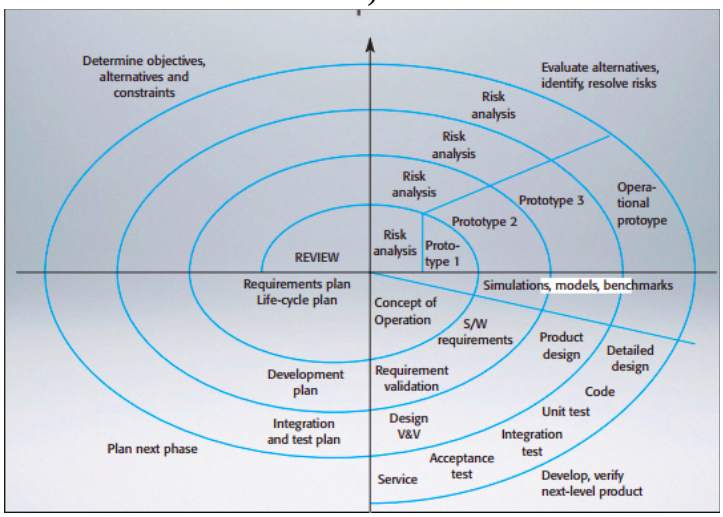




Problems with Waterfall and V models

- Needs stable and “perfect” requirements
- Can’t always anticipate what you are going to have to do
- Does not account for revision or refractoring
- Too inflexible and static
- Depends on getting each stage exactly right.(Changes can cause knock-on effects)





Problems with iterative processes

- Lack of process visibility(Systems are often poorly structured)
- Too much doing, not enough planning
- Special skills might be required
- Lightweight documentation taken to mean no documentation

SE & Quality Conclusions

1. Coding is only one stage of many
2. Having an appropriate SE Process/Methodology is really important
3. Quality means getting the right process
4. Quality means getting it right, not fast.
5. Mistakes can cost more than delay in schedule.
6. Maintaining/Evolving software often costs more than producing it in the first place.