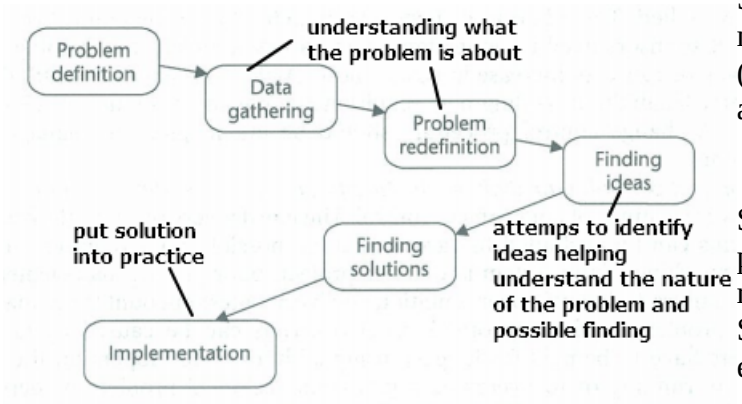
Project Lifecycles



Subdividing problems into more tasks: easier to manage, there can be many ways but the core activities (understanding the problem, choosing and designing, and building the solution).

Software development project = focused solely on producing a software system that satisfies the user requirements
System development project = wider cope, may not even include software.

Phase	Output deliverables
System engineering	High-level architectural specification
Requirements analysis	Requirements specification Functional specification Acceptance test specification
Design	Software architecture specification System test specification Design specification Subsystem test specification Unit test specification
Construction	Program code
Testing	Unit test report Subsystem test report System test report Acceptance test report Completed system
Installation	Installed system
Maintenance	Change requests Change request report

Waterfall lifecycle = systematic approach, good when requirements are not likely to change during the development process and nothing has to be delivered during the dev. proc.:
+ the phases have explicitly defined products or deliverables which can be manages individually.
- Real project rarely are that simply structured
- Task have to be repeated
- requirements may change (when lot of time between start and end)
- difficult to change, inflexible (f.ex when new technologies comes up)

Prototype:

Disadvantages

- The client may perceive the prototype as part of the final system, may not understand the effort that will be required to produce a working production system and may expect delivery soon.
- The prototype may divert attention from functional to solely interface issues.
- Prototyping requires significant user involvement, which may not be available.
- Managing the prototyping lifecycle requires careful decision making.

Prototyping has the following advantages:

- Early demonstrations of system functionality help identify any misunderstandings between developer and client.
- Client requirements that have been missed are identified.
- Difficulties in the interface can be identified.
- The feasibility and usefulness of the system can be tested, even though, by its very nature, the prototype is incomplete.

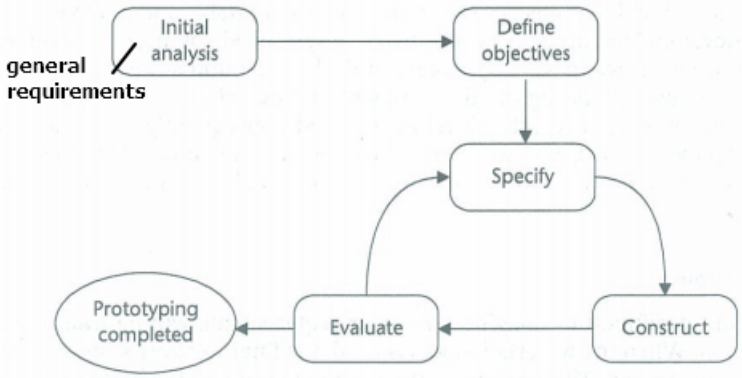
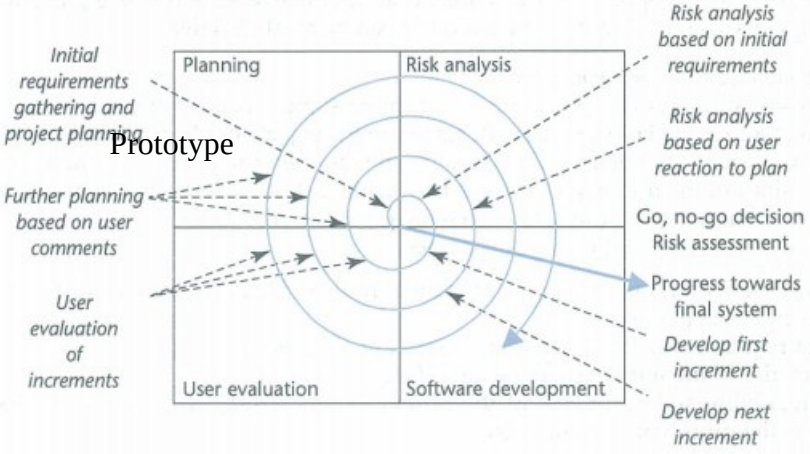


Figure 3.6 shows how Boehm's spiral model can be adapted to suit incremental delivery. Note that prototyping may be used either during the risk analysis or during the software development part of the development cycle.



Operational roles
- Normal operations
- Maintenance
- Support
- Helpdesk
- Training
- Planning, scheduling
- Control, management, administration

Beneficiaries
- Functional beneficiary
- Social beneficiary, etc (if indirect benefits exist)
- Financial beneficiary
- Political beneficiary (not only professional politicians)

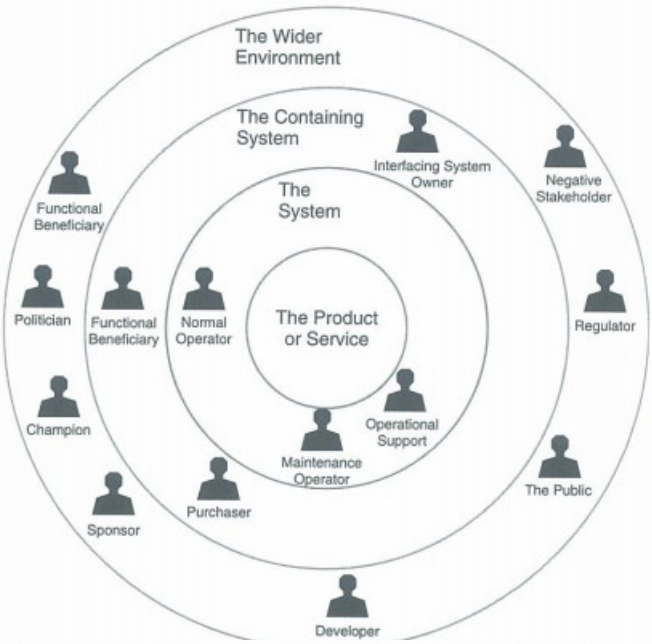
Interfacing roles
- Owner of interfacing system (sharing data, etc)
- Neighbouring business (mutual benefit)
  Interoperating service (sharing facilities/equipment)

Surrogate roles (representing or working on behalf of others)
- Champion
- Purchaser (roles here vary widely and often overlap)
  - Internal customer
  - Procurement
  - Project sponsor
  - Marketing (representing the consumer)
  - Product manager
- Developer (many roles possible here)
  - Requirements analyst
  - Designer
  - User interface designer
  - Programmer
  - Tester (very useful in requirements work)
  - Technical writer
- Manufacturer/subcontractor/supplier
- Expert/consultant (many more roles possible here)
  - Human factors/ergonomics
  - Safety engineer
  - Security engineer
  - Simulation/modelling expert
  - Legal opinion
  - Translator, cultural advisor, etc
- Regulator
  - Parliament, statute law
  - Government departments, regulations
  - Statutory regulators
  - Standards bodies (national and international)
  - Voluntary/industry regulators

Developers: Inside or Outside the System?

Developers are very close to the product while it is under development but may have little to do with it once it is in service unless they 'wear two hats' by also having a specific operational role, such as maintenance or helpdesk (operational support). If the onion is drawn for the product when it is in service, the developer therefore appears in 'the wider environment'; the maintenance operator appears as part of 'the system'.

You could draw a *different* onion model for the product under development, in which case your system will be 'the development system' (your software factory, for instance). In that case the developer will be inside, ounteract o we tend point.



System: people that deliver results to the world outside.
Beneficiary: who intend to benefit in some way (financial, political, functional, social)
Regulator: according to law, standards
Interfacing Roles: Responsible project / operation of systems that our service or project has to cooperate with
Negative Stakeholders: peaceful opposition to active hostilitiy, can threaten or harm project; f.ex. Competitors, security threats
Champion: really wants the project to work out, provides "political" support (Can be the same as sponsor but normally

Generic role	Work done by role	Example
Normal operator	Interacts with the product to deliver results (to functional beneficiaries)	Tram driver drives the tram
Maintenance operator	Services and repairs the product (i.e. carries out both planned and unplanned maintenance)	Mechanic services the tram
Support operator	Provides help and co-ordination to keep the other operators productive	Roster co-ordinator allocates drivers to trams each day

Hybrid roles
- User (= Normal Operator + Functional Beneficiary)
- Consumer (= Mass-market Purchaser + User)
- Service Provider (= Operational Support + Maintenance + Helpdesk, and sometimes Developer, Subcontractor, Manufacturer)
- Risk and Revenue-Sharing Partner (= Developer + Manufacturer + Financial Beneficiary), etc

Negative/hostile roles
- Vandal, graffiti artist
- Thief
- Hacker, virus writer
- Competitor
- Industrial espionage (via malware, etc)
- Fraudster
- Disgruntled employee
- Trades union
- Political opponent
- The public, residents' association, etc
- Activist, environmental pressure group, etc
- Military enemy, terrorist

2.3 Identifying Stakeholders

There are several ways of finding out who your stakeholders are, i.e. who should be involved or consulted on your project:

- by asking your sponsor or client;
- by examining what is and what isn't shown on an organisation chart;
- with a template such as the 'onion model';
- by comparison with similar projects;
- by analysing the context of the project.