An Introduction to Systems Analysis and Modelling

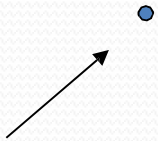
Dr Kristin Stock

Outline

- What, Why, When and Where?
- Different levels of modelling (conceptual, low level etc.).
- Modelling and Design – what is design, how does it relate to what we will cover in this course?
- The systems development lifecycle.
- Systems development methodologies.
- Systems analyst roles and skills.
- Project management, especially re CASE Tools.

What is Systems Analysis and Modelling?

- How do programmers know what to write in their code?



A typical 1st year programming assignment
Maybe 20 - 30 hours concentrated work?

A large industry project might involve about 10 000 software development hours
Additional hours for requirements, management, acceptance testing, …

What is Systems Analysis and Modelling?

- Systems analysis and modelling is a process of:
  - identifying the problem that needs to be solved
  - analysing the current situation
  - identifying requirements for the new system
  - modelling those requirements in a systematic way
  - creating a clear specification of what the system must do

What is Systems Analysis and Modelling?

- When we talk about **specifying what the system must do**, specifically this can include:
  - What data is needed
  - The rules that govern that data
  - The rules for how that data will be processed
  - The layout of user interfaces, reports, etc.
  - Rules about access constraints
  - Expectations about the performance of the system

Is it about designing a system?

- Well…..yes, but there are different definitions of design.

What is the difference between analysis, modelling and design?

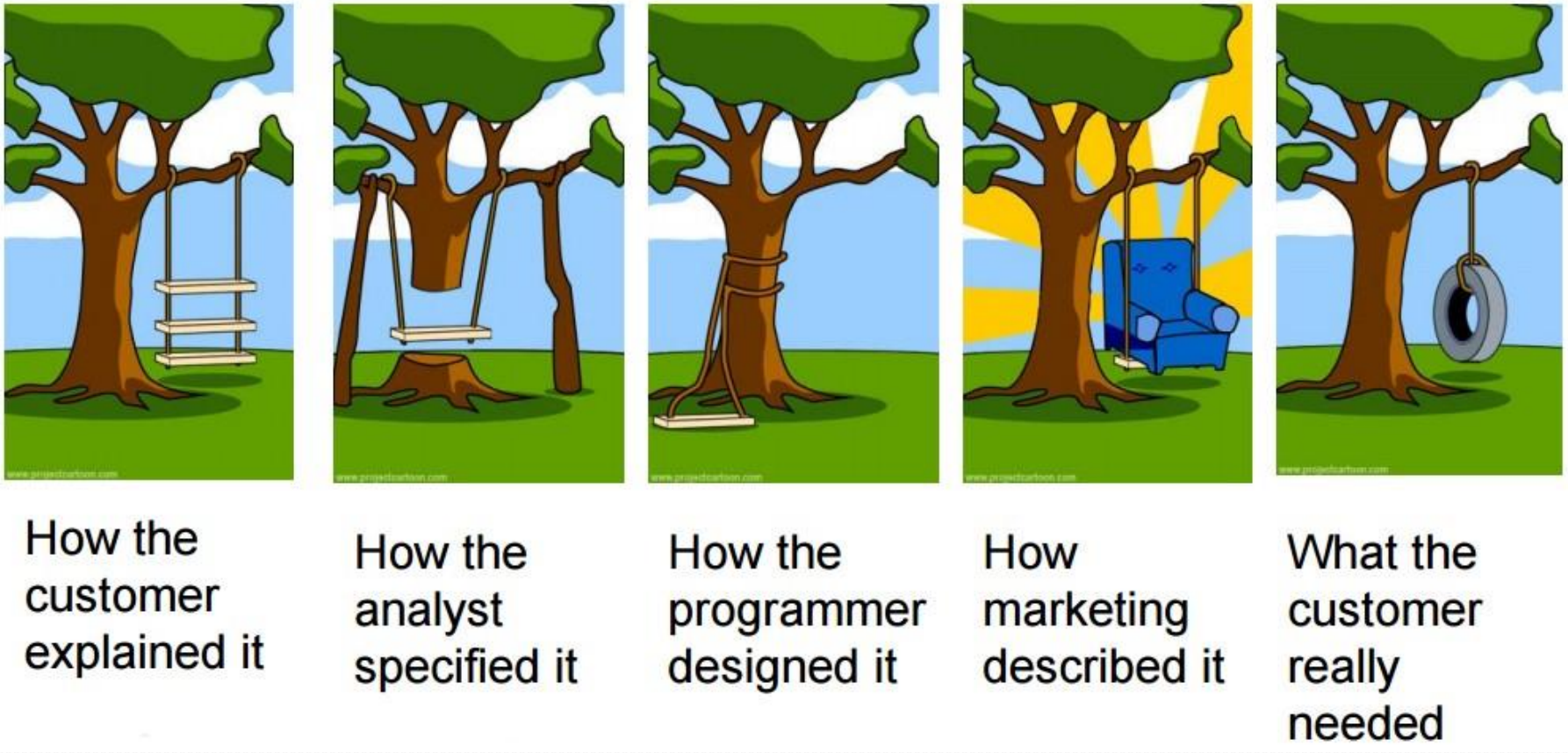
- Analysis is problem focussed
- Design is solution focussed
- But one morphs into the other through the process.
- The tools we use for analysis are also used for design.
- We model the existing process, and then we model our design of the new process.
- We can sometimes use this to generate code…

Why Systems Analysis and Design?

- To ensure the system meets the needs it is intended to meet.
- There is a long history of expensive IT systems that failed to meet user needs.
- To ensure system quality.
- To ensure a good fit with the IT environment
- Ultimately… to provide value to the organisation

The Famous Cartoon …

- Communication is key!



Examples of Project Failures

- National Health Service, UK
  - http://calleam.com/WTPF/?p=2003
- Transit Ticketing Authority, Melbourne, Australia
  - http://calleam.com/WTPF/?p=455
- Department of Environment, Food and Rural Affairs, UK
  - http://calleam.com/WTPF/?p=1881

Contributing factors as reported in the press:
Underestimation of complexity. Inadequate requirements. Design that was "rigid and task based", and "unsuited" to the agency's needs. Choosing the most complex design possible while simultaneously attempting to complete the project in a compressed time frame. Lack of testing (due to compressed schedule).

The systems analyst:
translates,
connects
communicates

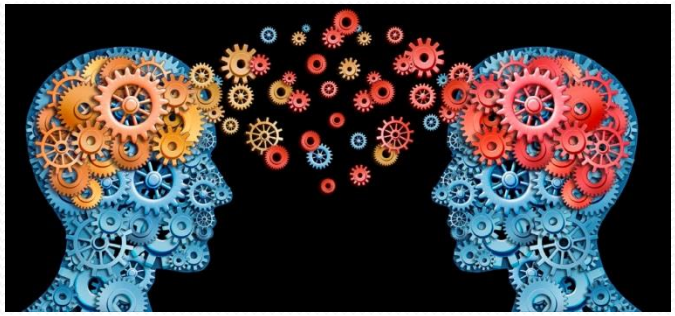
bridges the gap
between the
users/organisation
and the
technological
details

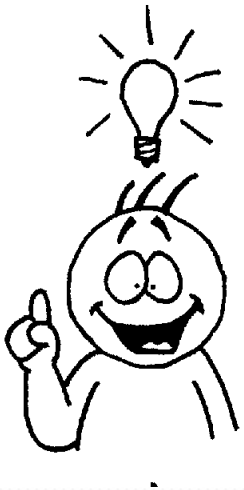
When?

- Right at the beginning!
- When the project idea is first formed, some systems analysis activities are appropriate.
- Can continue throughout, depending on SDLC methodology.

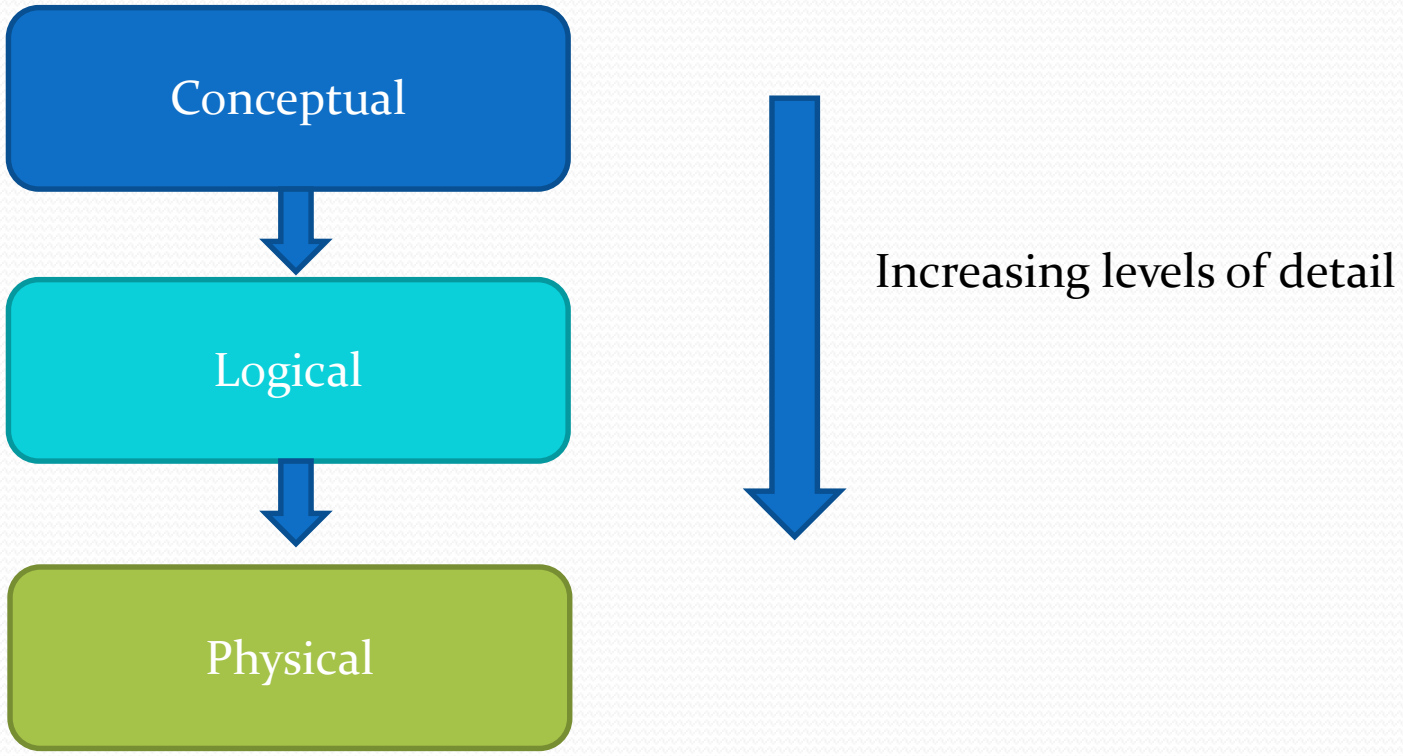
Where?

- The systems analyst needs to spend time with the users.
- Need to observe and understand the current situation, systems and environment.
- This is the foundation for a good system specification.



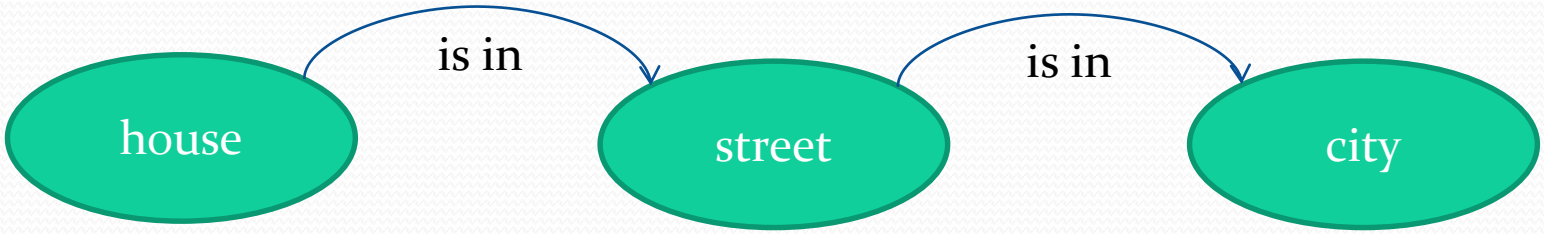


Different Levels of Modelling



Conceptual Analysis and Modelling

- High level.
- Sketches out broad concepts.
- Can work with users to define high level concepts.
- e.g. Concept Mapping.



Logical Analysis and Modelling

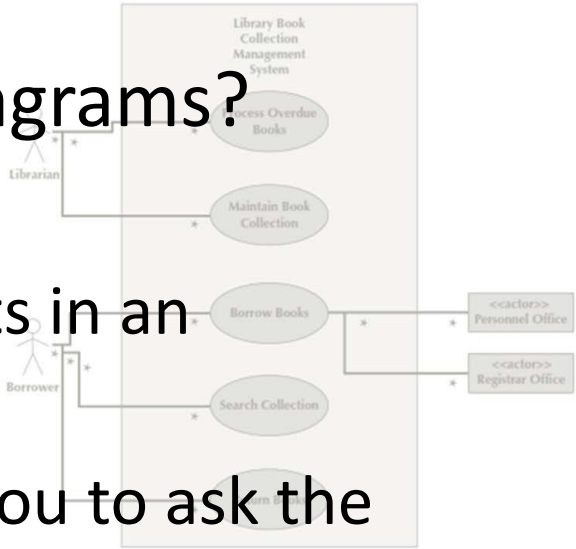
- Much more detailed.
- Independent of implementation.
- Does not depend on a particular operating system, software or hardware.
- Implementation decisions can be made separately, later.
- Changes in the implementation do not affect the logical design.
- PIM = platform independent model

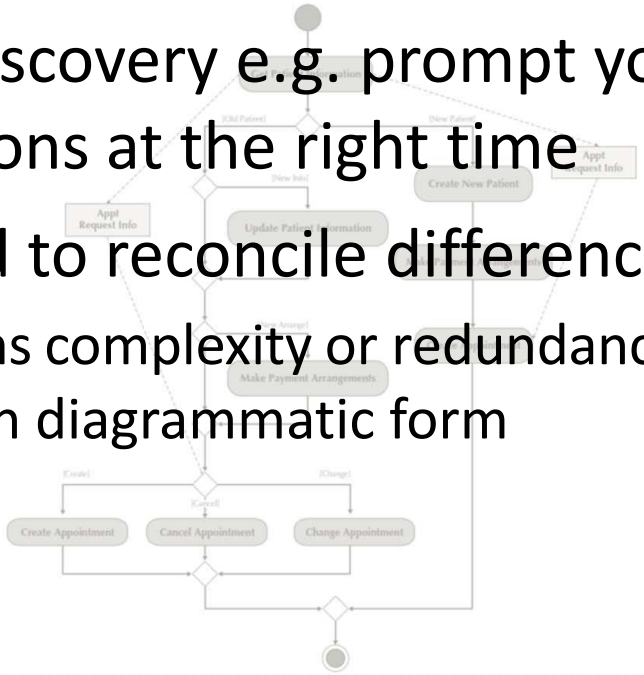
Physical Analysis and Modelling

- Moves from logical design into design on a particular operating system, hardware, software.
- E.g. Creating the actual database tables, creating software classes etc.
- Some tools are available to generate some parts of the physical design from the logical design models.
- E.g. logical database design can be used to generate specific Oracle, SQL Server databases.

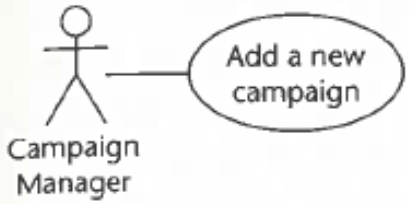
Why modelling?

- What's the point of all these diagrams?
  - For developers!
  - Get across a systems requirements in an unambiguous, standardized way.
  - Can drive discovery e.g. prompt you to ask the right questions at the right time
  - Can be used to reconcile differences in opinion
    - E.g. systems complexity or redundancies may be more apparent in diagrammatic form



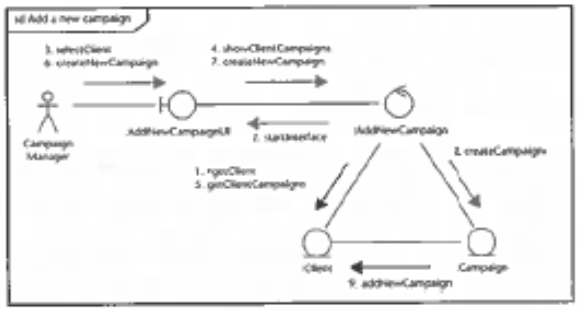


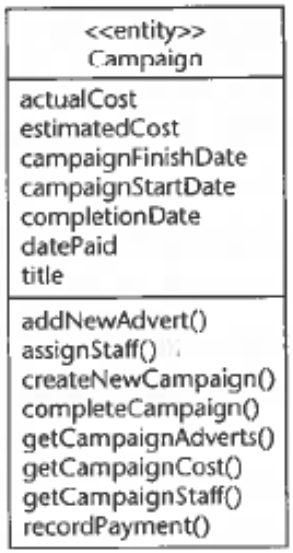
Requirements



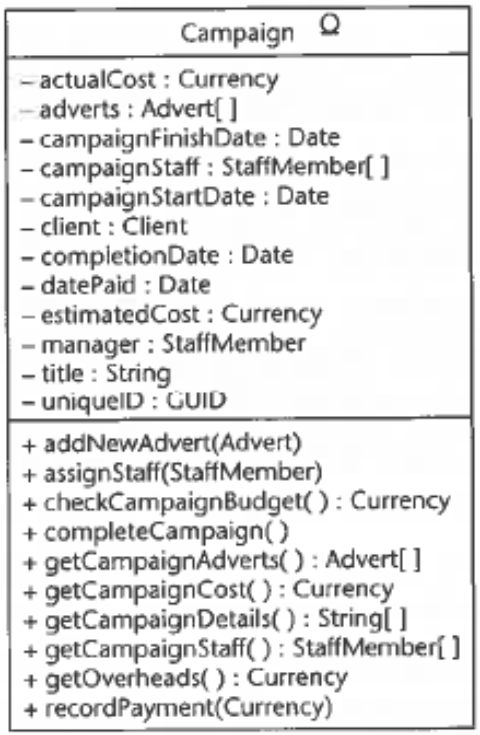
2. To record the details of each campaign for each client. This will include the title of the campaign, planned start and finish dates, estimated costs, budgets, actual costs and dates, and the current state of completion.

Analysis



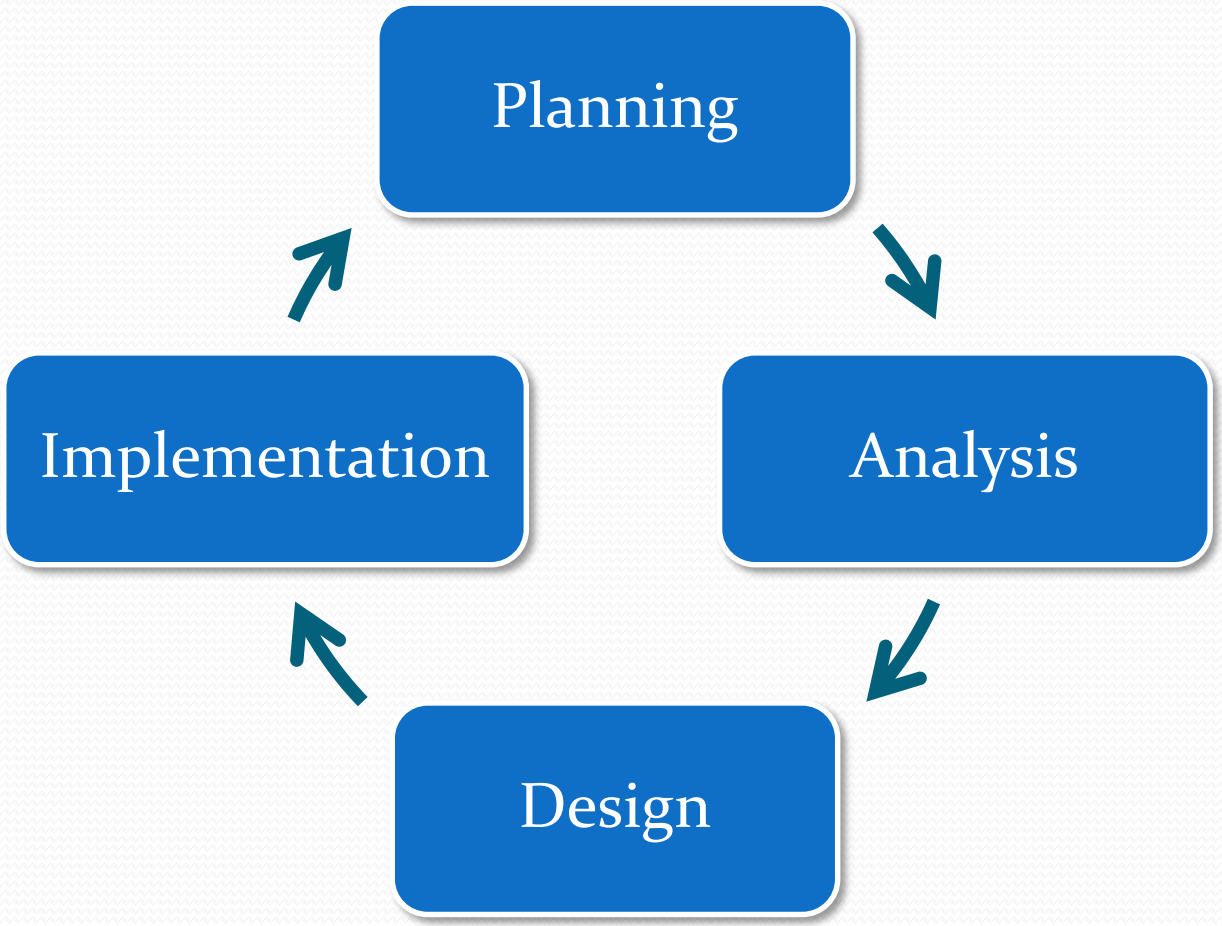


Design




```
CREATE TABLE Campaigns
  (VARCHAR(30) uniqueID PRIMARY KEY NOT NULL,
  FLOAT actualCost,
  DATE campaignFinishDate,
  DATE campaignStartDate,
  VARCHAR(30) clientID NOT NULL,
  DATE completionDate,
  DATE datePaid,
  FLOAT estimatedCost,
  VARCHAR(30) managerID,
  VARCHAR(50) title);
CREATE INDEX campaign_idx ON Campaigns
(clientID, managerID, title);
```

Systems Development Life Cycle (SDLC)



The SDLC Process

- The process consists of four phases
- Each phase consists of a series of steps
- Each phase is documented (deliverables)
- Phases are executed sequentially, incrementally, iteratively or in some other pattern

---> process of *gradual refinement*

Questions to be Answered

- Planning phase
  - Why should we build this system?
  - What value does it provide?
  - How long will it take to build?
- Analysis phase
  - Who will use it?
  - What should the system do for us?
  - Where & when will it be used?
- Design phase
  - How should we build it?

Increasing levels of detail

SDLC: The Planning Phase

1. Project Initiation
   - Develop/receive a system request
   - Conduct a feasibility analysis
2. Project Management
   - Develop the work plan
   - Staff the project
   - Monitor & control the project

SDLC: The Analysis Phase

1. Develop an analysis strategy
   - Model the current system
   - Formulate the new system
2. Gather the requirements
   - Develop a system concept
   - Create a business model to represent:
     - Business data
     - Business processes
3. Develop a system proposal

Deliverable: Analysis and High-Level Design

SDLC: The Design Phase

1. Develop a design strategy
2. Design architecture and interfaces
3. Select hardware, software, network infrastructure)
4. Develop databases and file specifications
5. Develop the program design to specify:
   - What programs to write
   - What each program will do

Deliverable: Detailed Design Specification

A word about specifications…

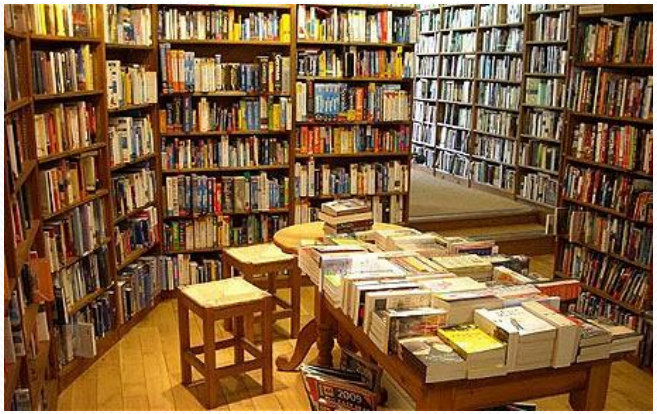
- Requirements specification.
  - What is required of the system?
  - High level.
- Functional specification.
  - What functions must it provide?
  - What business processes must it support?
- Design specification.
  - Details of design, including user interfaces, what each button does etc.
- Do we need all three?

SDLC: The Implementation Phase

1. Construct the system
   - Build it (write the programming code)
   - Test it
2. Install system
   - Train the users
3. Support the system (maintenance)

SDLC: Methodologies

- Methodology: a formalized approach to implementing the SDLC
- Categories
  - By area of focus
    - Process oriented
    - Data centered
    - Object-oriented
  - By sequencing and emphasis of phases:
    - Structured
    - Rapid action development
    - Agile development



Classes of Methodologies

- Structured Development
  - Waterfall Development
  - Parallel Development
- Rapid Application Development
  - Phased
  - Prototyping
- Agile Development
  - eXtreme Programming
  - SCRUM

Structured Methodologies

- Used commonly in 1980s
- Replaced ad-hoc approach.
- Formal, sequential, step-by-step.

Early methodology

Not very much of a model, more '**code-and-fix**'

Testing is restricted to testing the code (acceptance testing, involving the user and the requirements, is missing)

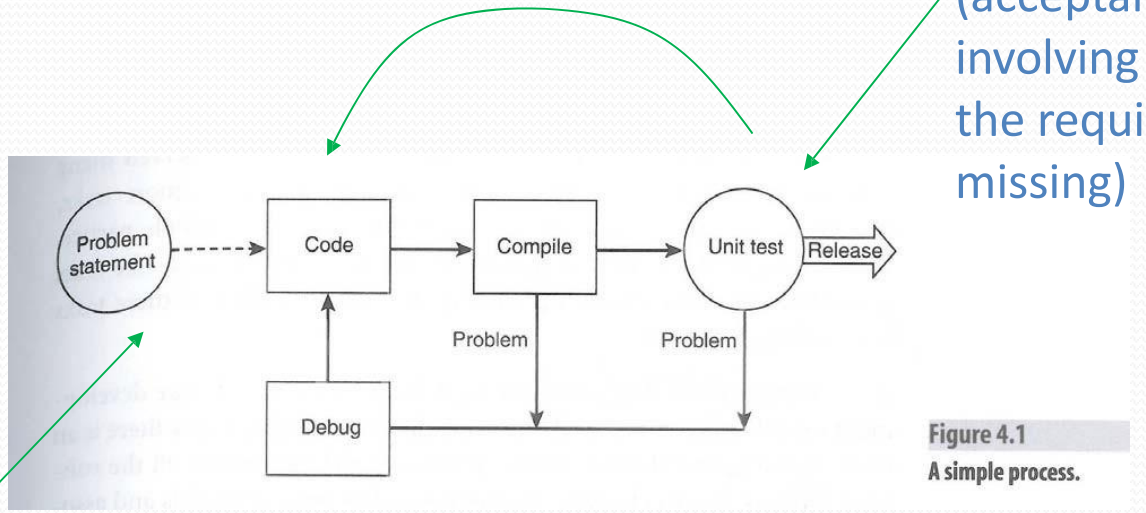


Figure 4.1
A simple process.

Requirements engineering, analysis and design are basically missing

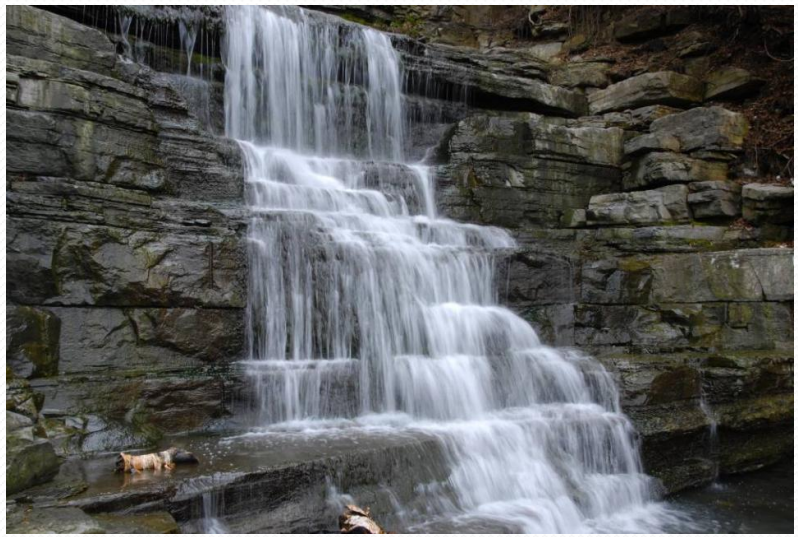
Focus on coding (and fixing problems with code)

Figure from Tsui and Karam, Essentials of Software Engineering, 2nd Edition, Ch. 4 'Software Process Models'

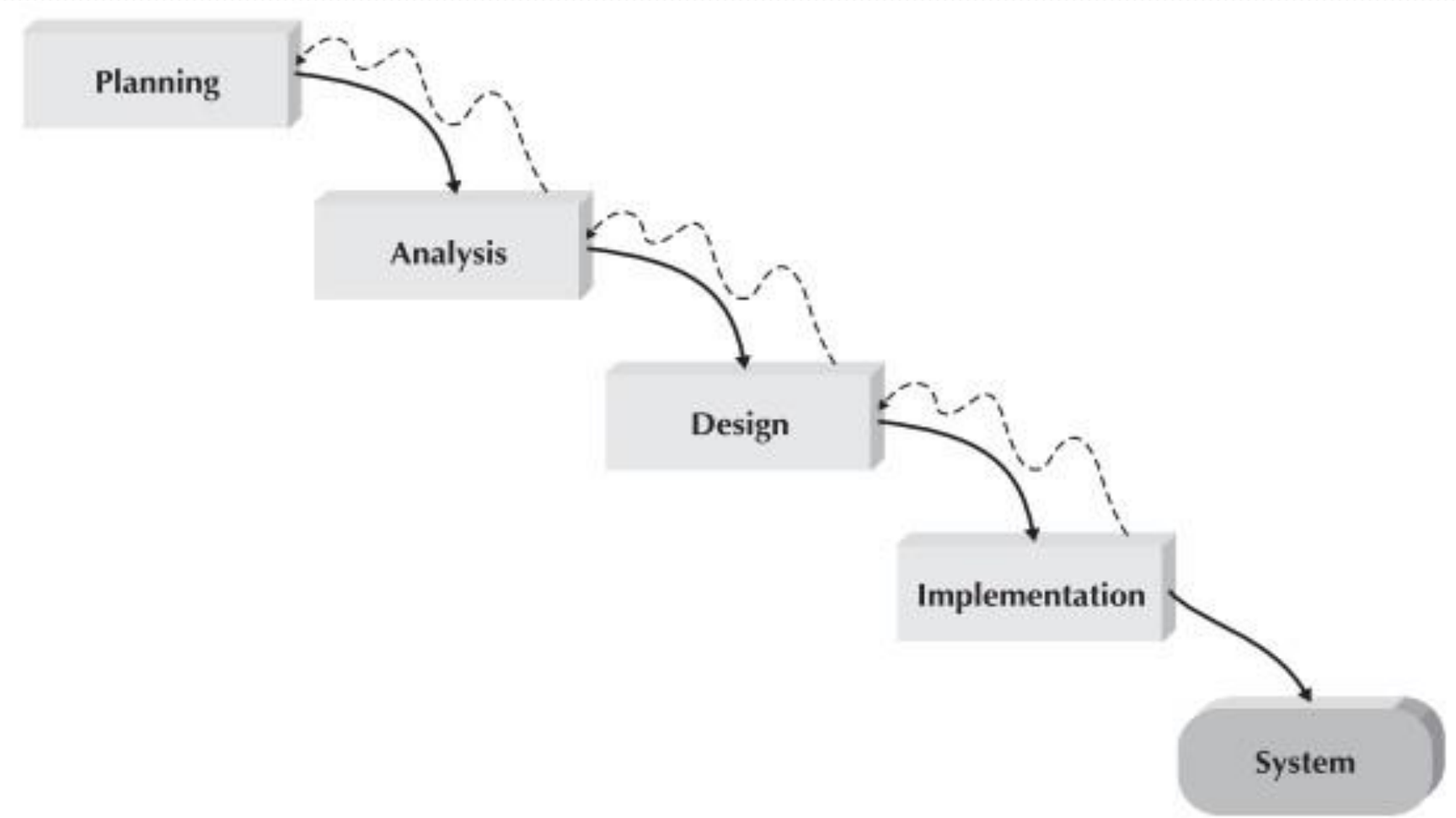
Software engineering has long moved on from this model …

Waterfall Development

- Proceeds sequentially from one step to the next.
- Lots of documentation, approved at each step.
- Can't go backwards.
- Advantages:
  - Requirements are identified long before programming starts.
  - Requirements do not change..
- Disadvantages:
  - If requirements are missed, expensive post-implementation changes required.
  - Does not cater for changes in the environment.



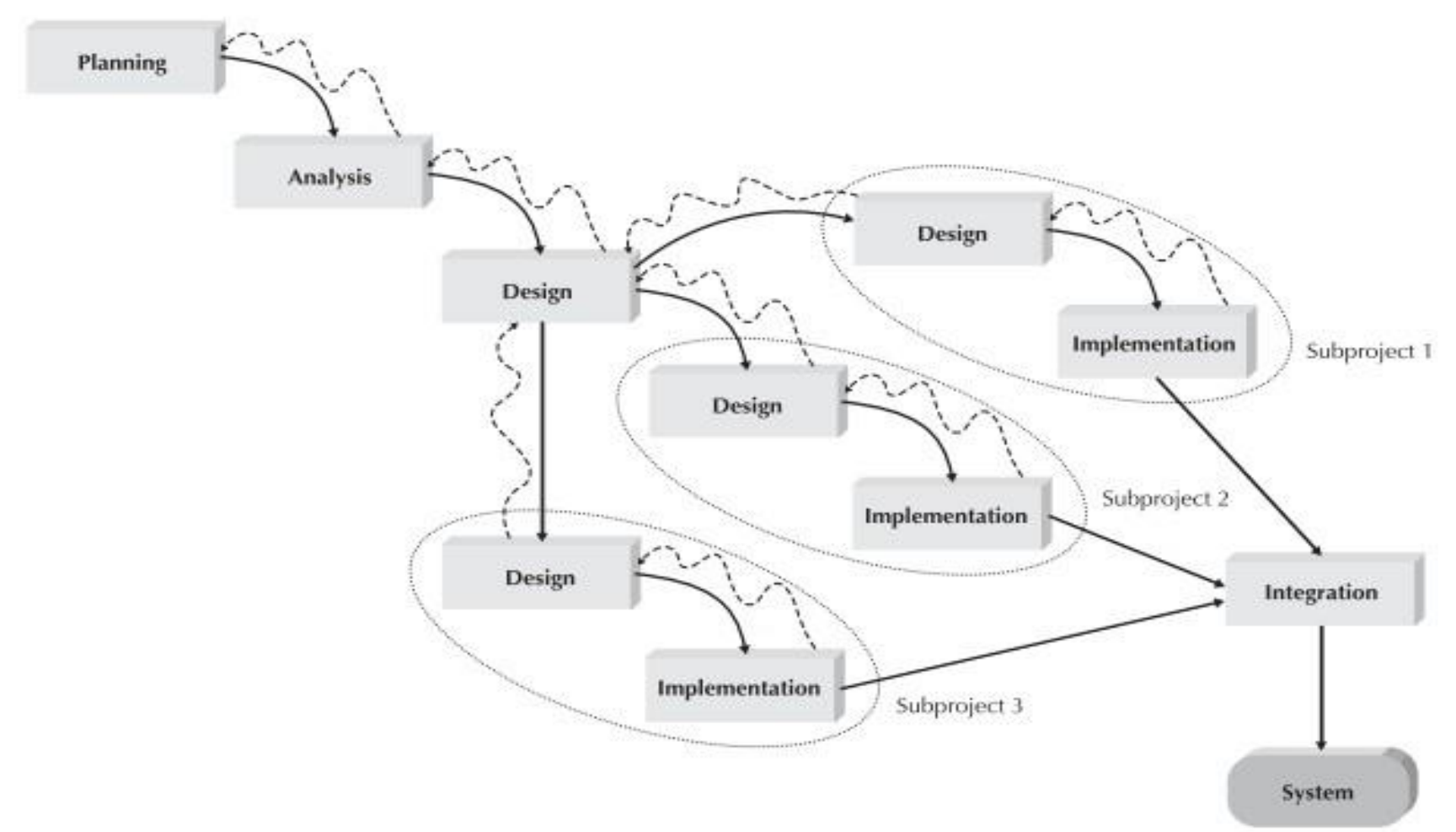
Waterfall Development



Parallel Development

- After a general high level design, sub-projects are created.
- Sub-projects designed and implemented in parallel.
- Sub-projects integrated at the end.
- Advantages:
  - Reduces time for total development, so less change in environment.
- Disadvantages:
  - Sub-projects may be interdependent, so integration is difficult.

Parallel Development



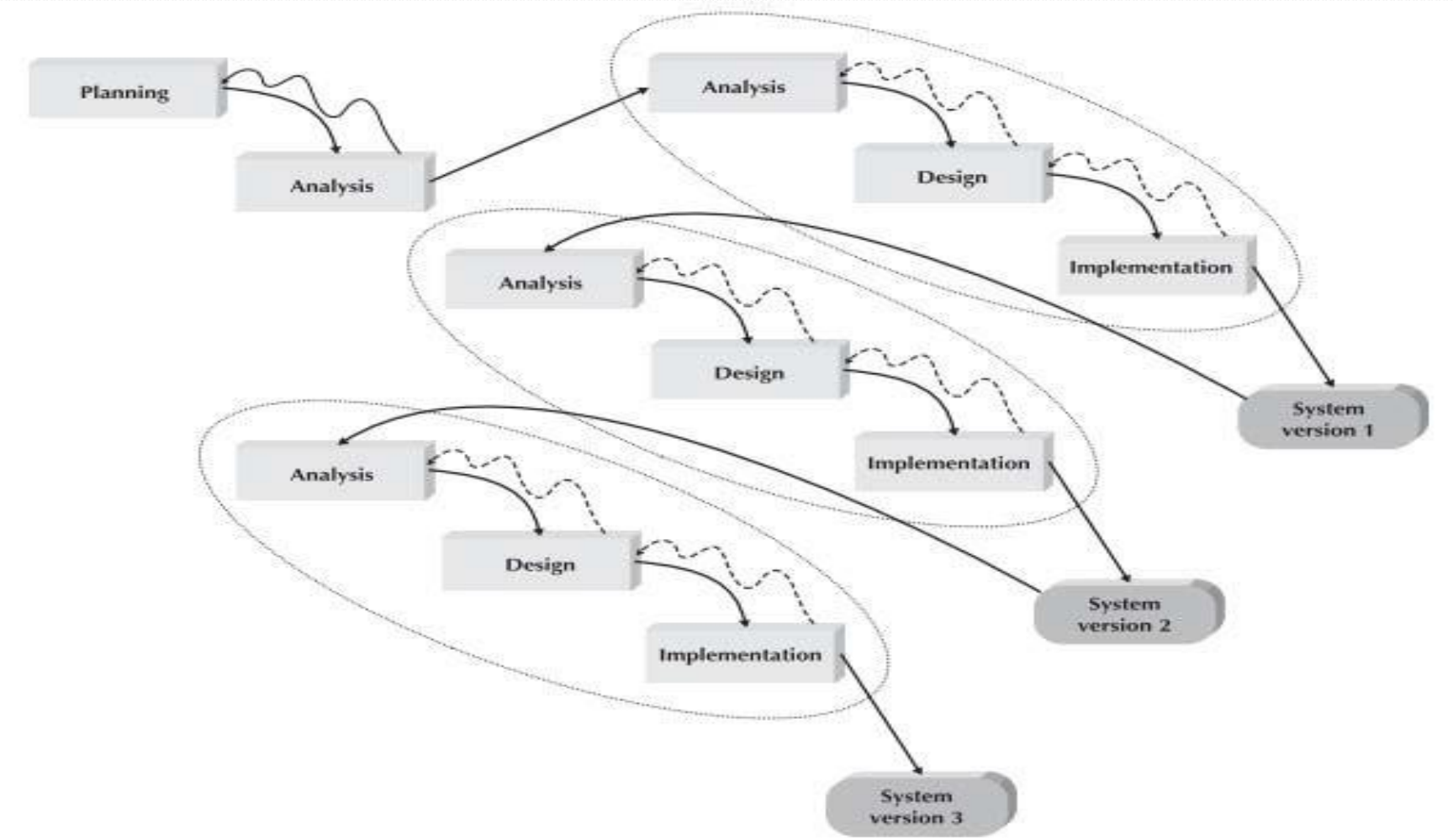
Rapid Application Development

- Started to be used in 1990s
- Try to get some parts of the system developed quickly and out to users for feedback.
- Advocate the use of Computer Aided Software Engineering (CASE) tools, Joint Application Design (JAD) sessions, visual programming tools, code generators.
- Can result in 'scope creep' – requirements change and increase as users work with the system.

Phased Development

- Begins with a high level design of the system concept.
- Defines a series of versions, with ever increasing functionality.
- For each version, there is analysis, design, implementation, release to users.
- Continues until system complete or obsolete.
- Advantages:
  - Gets something in the hands of the users quickly, so early value to organisation.
  - Can also identify problems and changed requirements early.
- Disadvantages:
  - Users are working with an incomplete system.
  - User expectation management.

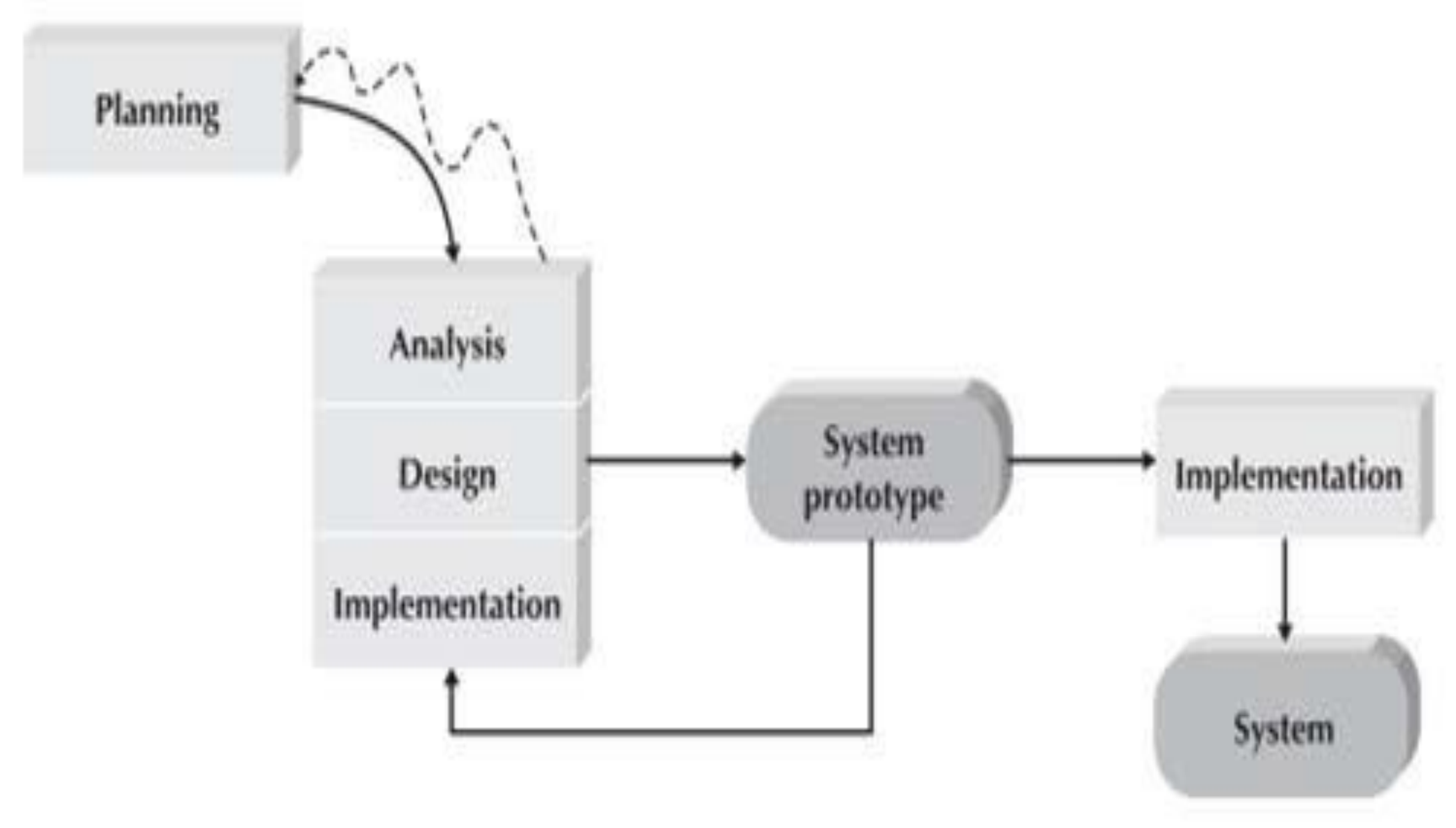
Phased Development



Prototyping

- Iterative.
- Very preliminary design used to create first prototype.
- Gradually refined.
- Analysis, design and implementation performed concurrently.
- All iterative.
- Advantages:
  - Early user involvement and feedback.
- Disadvantages:
  - Such rapid change that methodological design and analysis is difficult.
  - Can be very inefficient.

Prototyping



Throwaway Prototyping

- Adopts a more conventional waterfall process overall, but prototypes are built along the way, then discarded.
- Prototypes address specific issues.
- Each prototype has its own analysis, design, implementation cycle, and is used to refine the design.

Agile Development

- Most recent.
- More focussed around design philosophies than specific details of methodology.
- Follow the 12 principles:
  - Software is delivered early and continuously
  - Changing requirements are embraced
  - Customers and developers work together
  - Face to face communication is key
  - Focus on technical excellence and good design
  - Regular reflection.

Agile manifesto, 2001 (http://agilemanifesto.org)

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile development principles (1)

Individuals and interactions over processes and tools

- People forming the development team are most important
- Relationships and communications are very important (among developers and wider with management, users, …)

Working software over comprehensive documentation

- Refocusing on the actual aim of producing (high quality) software
- Only write documentation that is really necessary and actively use it
- Code (carefully selected) parts of the system as early as possible
- Test completed parts (with the users) as part of the development process

Agile development principles (2)

Customer collaboration over contract negotiation

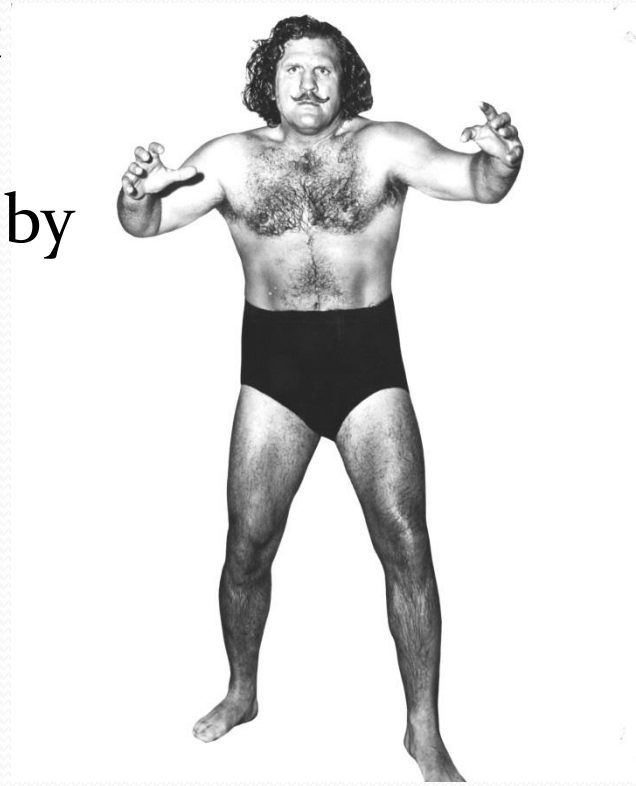
- Have close (daily) contact with the customer/user
- Help everyone understand the other's point-of-view/needs/pressures

Responding to change over following a plan

- Acknowledge that it is near impossible to specify everything correctly in advance
- Build change and flexibility into the process

Drawbacks of Agile Development

- Require co-location of development team.
- The process if not highly managed (by definition), so can turn into 'programmers gone wild', hacking together solutions.
- Little documentation.
- Suitable for large, mission critical systems?



Extreme Programming

- Communication, simplicity, feedback, courage.
- Feedback to users continuously.
- Embrace change.
- Quality first.
- Testing is continuous.
- Evolutionary, incremental.
- Heavy on-site user input.
- Relies on a small, close-knit team.

Scrum

- Made up of 30 day sprints.
- Each sprint delivers a system to the user.
- No requirements changes during a sprint.
- Scrum meeting each day:
  - Previous day's accomplishments.
  - Plan for today.
  - Any obstacles.
- Again, may not be suitable for large projects.

Which Methodology to Use?

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Excellent	Good	Good
That Are Reliable	Good	Good	Good	Poor	Excellent	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

What kind of person is involved in Systems Analysis?

- Agents of change
  - Identify ways to improve the organization
  - Motivate & train others
- Ability to see both technical and user sides of a problem.
- Big picture plus detail.

Skills needed for Systems Analysis

- Skills needed:
  - Technical: must understand the technology
  - Business: must know the business processes
  - Analytical: must be able to solve problems
  - Communications: technical & non-technical audiences
  - Interpersonal: leadership & management
  - Ethics: deal fairly and protect confidential information

Jobs involving Systems Analysis

- Business Analyst
  - Focuses on the business issues
- Systems Analyst
  - Focuses on the IS issues
- Infrastructure Analyst
  - Focuses on the technical issues
- Change Management Analyst
  - Focuses on the people and management issues
- Project Manager
  - Ensures that the project is completed on time and within budget
- Systems Architect
- Data Modeller
- User Interface/Interaction Designer

IT Project Management

- Project Management is the process of planning and controlling system development within a specified time at a minimum cost with the right functionality
- A project is a set of activities with a specified beginning and end point meant to create a system that brings value to the business
- Project Managers monitor and control all tasks and roles that need to be coordinated.
- Analysis and Modelling are key to the success of IT projects.

Spectacular IT project failures

- UK prison IT system:
  - http://www.zdnet.com/article/uk-prison-it-massive-and-spectacular-failure/
- See reading from Nelson, 2007 on Stream.

Why are failures so common?

- Underestimating time and effort.
- Unreasonable demands/expectations.
- Use of unknown/new technologies.
- Too much focus on technology:
  - Driven by technology rather than business value.
  - Failing to take human, organisational, cultural aspects into account.

Project Identification

- A business need is identified – management, business unit, IT department, external (e.g. politicians, senior management).
- Could result from some problem (losing market share etc.).
- Speculative project identification.
- The business need should drive the high level requirements.

The System Request

- A document that describes the reasons for and the value added from building a new system
- Contains 5 elements:
  - Project sponsor: the primary point of contact for the project
  - Business need: the reason prompting the project
  - Business requirements: what the system will do
  - Business value: how will the organization benefit from the project
  - Special issues: Anything else that should be considered

What is a Business Case?

- Commonly (and loosely) used term to describe a document that justifies why a system should be built/changed/upgraded.
- Can range from a couple of paragraphs to a lengthy document.
- If you want to spend time or money on a project, you will need to make the case to management.

Feasibility Analysis

- Is this project feasible?
  - What are the risks?
  - Can these risks be overcome?
- Major components:
  - Technical feasibility (Can we build it?)
  - Economic feasibility (Should we build it?)
  - Organizational feasibility (Will they use it?)

Technical Feasibility

- Identify risks in the following areas:
  - The functional area: Are analysts familiar with this portion of the business?
  - The technology: Less familiarity generates more risk
  - Project size: Large projects have more risk
  - Compatibility: Difficult integration increases the risk

Economic Feasibility (Cost-Benefit Analysis)

- Identify the costs and the benefits
- Assign values to the costs and benefits
- Determine the cash flow
- Determine the value using one or more methods:
  - Net present value (NPV)
  - Return on investment (ROI)
  - Break-even point

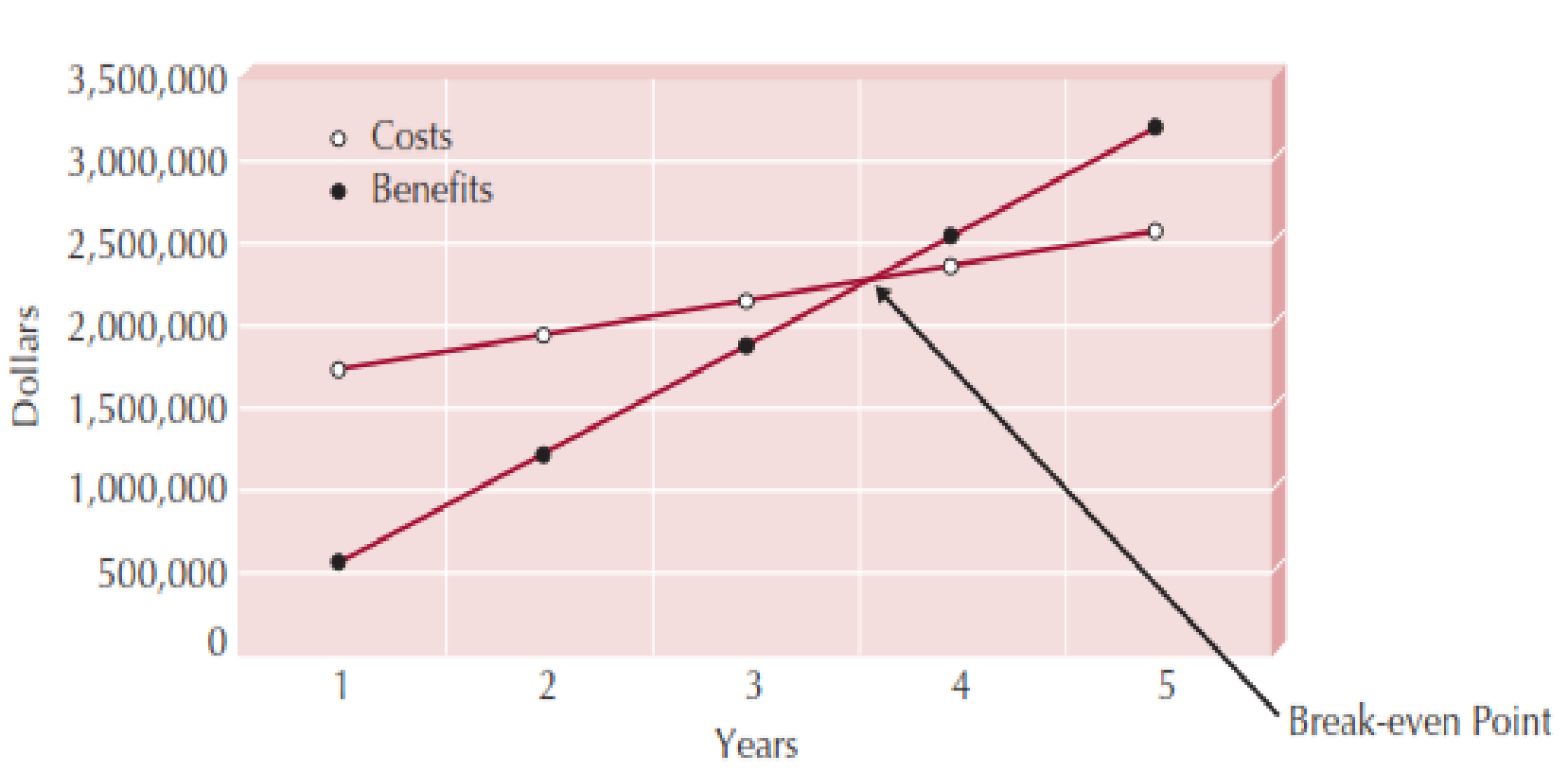
Formulas for Determining Value

Calculation	Definition	Formula
Present Value (PV)	The amount of an investment today compared to that same amount in the future, taking into account inflation and time.	$\frac{\text{Amount}}{(1 + \text{interest rate})^n}$ n = number of years in future
Net Present Value (NPV)	The present value of benefit less the present value of costs.	PV Benefits − PV Costs
Return on Investment (ROI)	The amount of revenues or cost savings results from a given investment.	$\frac{\text{Total benefits} - \text{Total costs}}{\text{Total costs}}$
Break-Even Point	The point in time at which the costs of the project equal the value it has delivered.	$\frac{\text{Yearly NPV*} - \text{Cumulative NPV}}{\text{Yearly NPV*}}$ *Use the Yearly NPV amount from the first year in which the project has a positive cash flow. Add the above amount to the year in which the project has a positive cash flow.

Example Cost-Benefit Analysis

	2008	2009	2010	2011	2012	Total
Increased sales	500,000	530,000	561,800	595,508	631,238	
Reduction in customer complaint calls	70,000	70,000	70,000	70,000	70,000	
Reduced inventory costs	68,000	68,000	68,000	68,000	68,000	
TOTAL BENEFITS:	638,000	668,000	699,800	733,508	769,238	
PV OF BENEFITS:	619,417	629,654	640,416	651,712	663,552	3,204,752
PV OF ALL BENEFITS:	619,417	1,249,072	1,889,488	2,541,200	3,204,752	
2 Servers @ $125,000	250,000	0	0	0	0	
Printer	100,000	0	0	0	0	
Software licenses	34,825	0	0	0	0	
Server software	10,945	0	0	0	0	
Development labor	1,236,525	0	0	0	0	
TOTAL DEVELOPMENT COSTS:	1,632,295	0	0	0	0	
Hardware	54,000	81,261	81,261	81,261	81,261	
Software	20,000	20,000	20,000	20,000	20,000	
Operational labor	111,788	116,260	120,910	125,746	130,776	
TOTAL OPERATIONAL COSTS:	185,788	217,521	222,171	227,007	232,037	
TOTAL COSTS:	1,818,083	217,521	222,171	227,007	232,037	
PV OF COSTS:	1,765,129	205,034	203,318	201,693	200,157	2,575,331
PV OF ALL COSTS:	1,765,129	1,970,163	2,173,481	2,375,174	2,575,331	
TOTAL PROJECT BENEFITS – COSTS:	(1,180,083)	450,479	477,629	506,501	537,201	
YEARLY NPV:	(1,145,712)	424,620	437,098	450,019	463,395	629,421
CUMULATIVE NPV:	(1,145,712)	(721,091)	(283,993)	166,026	629,421	
RETURN ON INVESTMENT:	24.44%	(629,421/2,575,331)				
BREAK-EVEN POINT:	3.63 years	[break-even occurs in year 4; (450,019 – 166,026)/450,019 = 0.63]				
INTANGIBLE BENEFITS:	This service is currently provided by competitors Improved customer satisfaction					

Example Break-Even Point



Organizational Feasibility

- Will the users accept the system?
- Is the project strategically aligned with the business?
- Conduct a stakeholder analysis
  - Project champion(s)
  - Organizational management
  - System users
  - Others

Project Selection

- Projects are approved, declined or delayed based on value added vs. risks
- Project portfolio management
  - Goals:
    - Maximize cost/benefit ratio
    - Maintain an optimal mix of projects based on:
      - Risk
      - Size, cost & length of time to complete
      - Purpose, scope & business value
  - Limited resources require trade-offs
- Selected projects enter the project management process

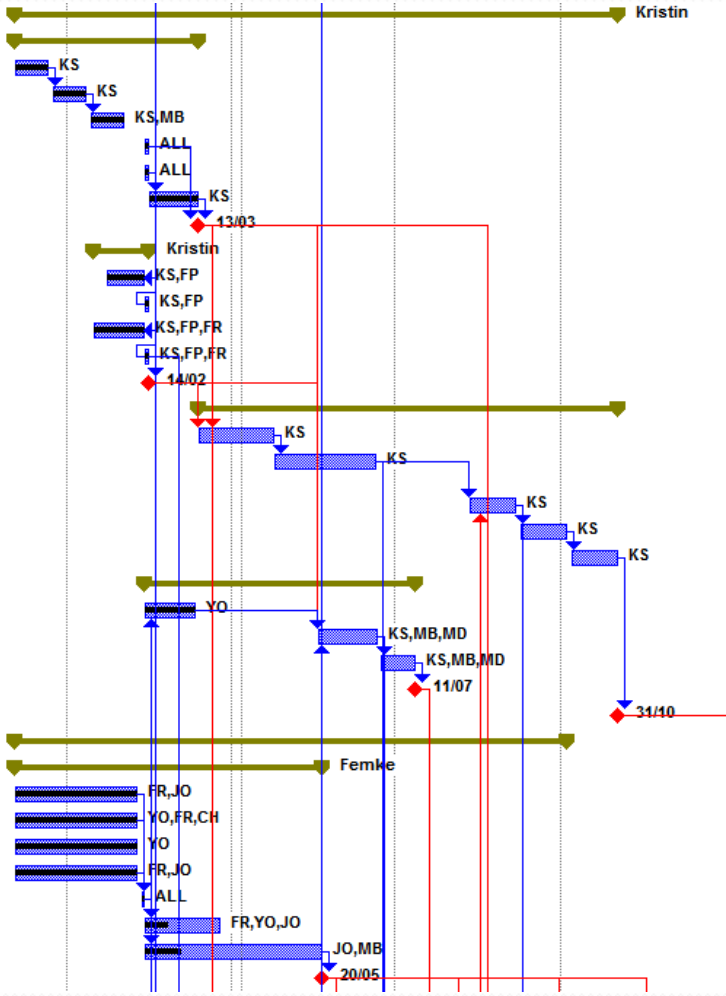
Project Management Tools

- Aids in creating workplans
- Identify all tasks, their sequence and estimate the time to complete each one
- Work breakdown structures (WBS): a hierarchy of tasks to identify:
  - Duration of each task
  - Current status of each task
  - Task dependencies (shows which tasks must be completed before others can begin)
- Gantt charts: horizontal bar chart that shows the WBS graphically
- Network diagrams: PERT and CPM

Gantt Charts

- Show tasks, duration, start and end and resources (e.g. staff) in graphical format.

* **Define the KI Architecture - how the components fit together (WP2)**	**240 days**	**Mon 03/12/07**	**Fri 31/10/08**
Create high level architecture	**74 days**	**Mon 03/12/07**	**Thu 13/03/08**
Identify very high level architecture options for discussion	15 days	Mon 03/12/07	Fri 21/12/07
Create document describing options for discussion -> deliverable	15 days	Mon 24/12/07	Fri 11/01/08
Briefly review software capabilities to identify limitations and options (e.g. Knowledge Smarts, Gnizr)	15 days	Mon 14/01/08	Fri 01/02/08
Discuss architecture options	2 days	Wed 13/02/08	Thu 14/02/08
Select a very broad architecture	2 days	Wed 13/02/08	Thu 14/02/08
Create a document describing the selected architecture at a high level -> deliverable	20 days	Fri 15/02/08	Thu 13/03/08
Milestone: high level architecture complete	0 days	Thu 13/03/08	Thu 13/03/08
Select representation structures	**22 days**	**Wed 16/01/08**	**Thu 14/02/08**
Review semantic representation structures (ontologies, FTCs, other structures)	15 days	Wed 23/01/08	Wed 13/02/08
Select a semantic representation structure	2 days	Wed 13/02/08	Thu 14/02/08
Review ontology languages	20 days	Wed 16/01/08	Wed 13/02/08
Select an ontology language and version	2 days	Wed 13/02/08	Thu 14/02/08
Milestone: semantic structure and ontology languages selected	0 days	Thu 14/02/08	Thu 14/02/08
Design the information architecture for the registry (ebRIM)	**166 days**	**Fri 14/03/08**	**Fri 31/10/08**
Review available information models for the registry	30 days	Fri 14/03/08	Thu 24/04/08
Design the KI detailed information architecture and interfaces (describes in detail how the information will be structured in the KI) and how ontologies will be included	40 days	Fri 25/04/08	Thu 19/06/08
Design the KI detailed model using the scientific source, web service and knowledge ontologies and appli	20 days	Mon 11/08/08	Fri 05/09/08
Create a research report describing the detailed information architecture -> deliverable	20 days	Mon 08/09/08	Fri 03/10/08
Create a standards specification describing the information architecture work -> deliverable	20 days	Mon 06/10/08	Fri 31/10/08
Create detailed KI software architecture	**108 days**	**Wed 13/02/08**	**Fri 11/07/08**
Review literature on knowledge infrastructures	20 days	Wed 13/02/08	Tue 11/03/08
Convert and expand broad architecture into a more detailed software architecture	25 days	Mon 19/05/08	Fri 20/06/08
Create a paper describing possible architectures and the selected architecture -> deliverable	15 days	Mon 23/06/08	Fri 11/07/08
Milestone: detailed KI architecture complete	0 days	Fri 11/07/08	Fri 11/07/08
Milestone: architecture complete	0 days	Fri 31/10/08	Fri 31/10/08
Define, Design, Implement KI Knowledge Structures	**220 days**	**Mon 03/12/07**	**Fri 03/10/08**
Marine Instruments and Parameters Domain Ontologies (WP1)	**122 days**	**Mon 03/12/07**	**Tue 20/05/08**
Prepare domain ontology workshop	50 days	Mon 03/12/07	Fri 08/02/08
Review existing domain ontologies, taxonomies, FTCs, etc.	50 days	Mon 03/12/07	Fri 08/02/08
Create review paper of domain ontologies -> deliverable	50 days	Mon 03/12/07	Fri 08/02/08
Identify requirements and scope for domain ontologies	50 days	Mon 03/12/07	Fri 08/02/08
Capture domain knowledge from experts during workshop - deliverable	2 days	Mon 11/02/08	Tue 12/02/08
Write paper describing lessons learnt from the workshop -> deliverable	30 days	Wed 13/02/08	Tue 25/03/08
Complete marine instruments domain ontologies -> deliverable	70 days	Wed 13/02/08	Tue 20/05/08
Milestone: marine instruments domain ontologies created	0 days	Tue 20/05/08	Tue 20/05/08



	Activity	Year 1	Year 2	Year 3
1	Study contextual aspects that contribute to language use and interpretation in English			
2	Study use of spatial relations in Maori			
3	Study contextual aspects that contribute to language use and interpretation in Maori			
4	Compare English and Maori spatial language models and contextual aspects			
3	Develop of Spatial Context Ontology			
4	Develop of model for contextual language use			
5	Develop of reasoning method			
6	Evaluate models against test data and iteratively improve			

Scope Management

- Scope “creep”
  - Occurs after the project is underway
  - Results from adding new requirements to the project
  - Can have a deleterious effect on the schedule
- Techniques to manage the project scope:
  - Identify all requirements at the outset
  - Allow only those changes deemed absolutely necessary
  - Carefully examine the impact of suggested changes
  - Delay some changes for “future enhancements”
  - Time boxing

Environment & Infrastructure Management

- Environment—Choose the right set of tools
  - Use appropriate CASE tools to:
    - Increase productivity and centralize information (repository)
    - Utilize diagrams—more easily understood
  - Establish standards to reduce complexity
- Infrastructure—Document the project appropriately
  - Store deliverables & communications in a project binder
  - Use Unified Process standard documents
  - Don't put off documentation to the last minute

Types of Standards	Examples
Documentation standards	The date and project name should appear as a header on all documentation.
	All margins should be set to 1 inch.
	All deliverables should be added to the project binder and recorded in its table of contents.
Coding standards	All modules of code should include a header that lists the programmer, last date of update, and a short description of the purpose of the code.
	Indentation should be used to indicate loops, if-then-else statements, and case statements.
	On average, every program should include one line of comments for every five lines of code.
Procedural standards	Record actual task progress in the work plan every Monday morning by 10 AM.
	Report to project update meeting on Fridays at 3:30 PM.
	All changes to a requirements document must be approved by the project manager.
Specification requirement standards	Name of program to be created
	Description of the program's purpose
	Special calculations that need to be computed
	Business rules that must be incorporated into the program
	Pseudocode
	Due date
User interface design standards	Labels will appear in boldface text, left-justified, and followed by a colon.
	The tab order of the screen will move from top left to bottom right.
	Accelerator keys will be provided for all updatable fields.

Case Study: Australian Capital Territory SDMS

- Goal: to automate, record, enable land administration
- Two existing systems:
  - CAD system
  - Oracle based textual system
- Requirement: a single integrated system to support all business functions

Case Study: Australian Capital Territory SDMS

- Waterfall model.
- First analysed existing systems and processes.
- Designed new processes.
- Designed user interfaces and applications to support those processes.

Case Study: Australian Capital Territory SDMS – Lessons Learnt

- Massive scope creep.
- Moved to phased methodology.
- User involvement much too late in the process.
- Full set of specifications: requirements, functional, design.
- Design spec impractical.
- Very complex system technically and organisationally.

What Next?