Health Informatics Project Management

Centre for Health Informatics

Prof John Chelsom

Runs: 3 hours
Tutor: Prof John Chelsom
Mode of attendance: Classroom

Learning Objectives

- This session describes the best ways to manage a health informatics project
- Specific learning objectives are to:

1. Identify the key requirements for project management
2. See how Agile development methodologies work
3. Combine methodologies for the best management outcomes
4. Learn techniques for project estimating
5. Understand how and why informatics project go wrong

Health Informatics Project Management

- Managing Health Informatics Projects
- Using Agile Development Techniques
- Project Management Methodologies
- Project Estimating and Scheduling
- Why Do Informatics Projects Go Wrong?
- References and Further Reading

Information Sources

- Sources are listed in the references at the end of these slides

Some definitions and descriptions have been taken from quoted resources.

Retrieved October 2010.

Where consensus on definitions or descriptions Is required, these have been taken from Wikipedia.

Retrieved October 2010.

Managing Health Informatics Projects

Project Management

Project management is the discipline of planning, organizing and managing resources to bring about the successful completion of specific project goals and objectives.

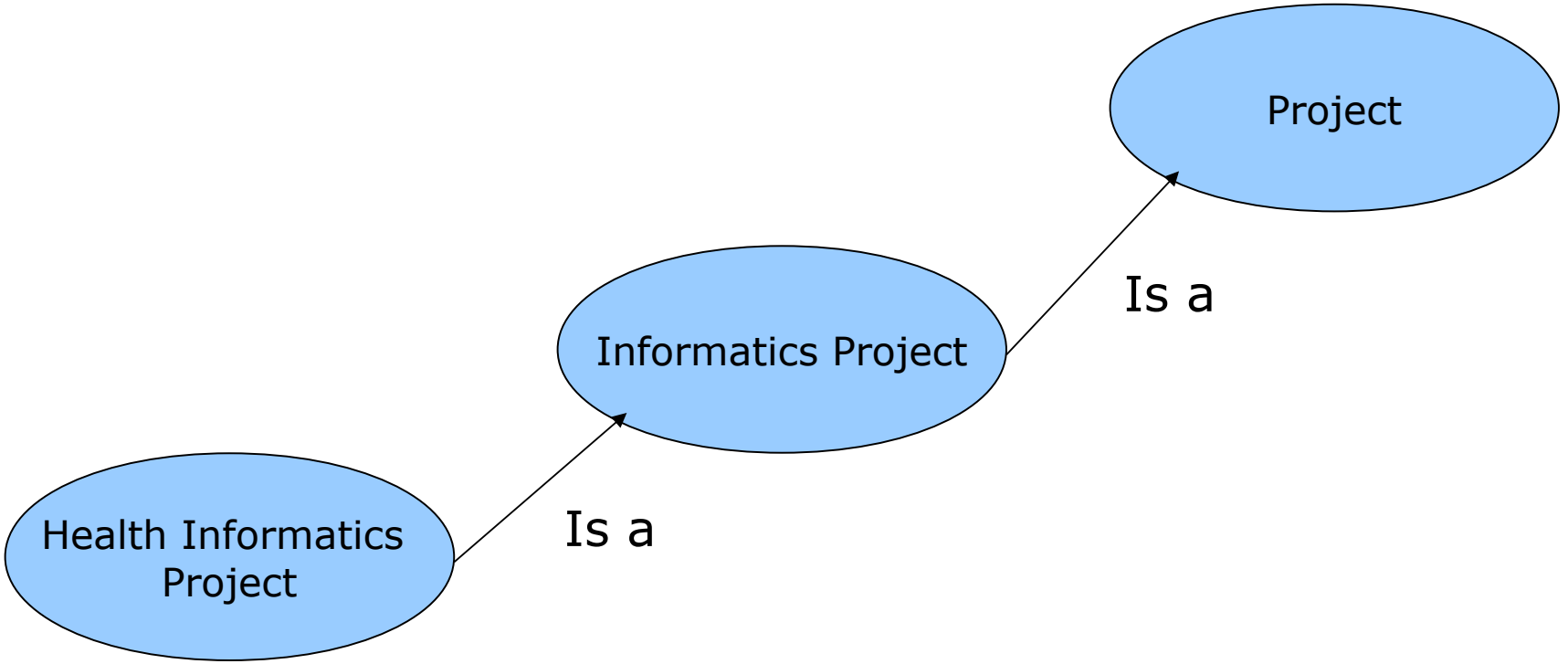
http://en.wikipedia.org/wiki/Project_management

- The four main elements of management are generally stated as being:
  - Planning
  - Leading
  - Organising
  - Controlling

- Three additional elements are often added (and are relevant to projects)
  - Resourcing
  - Coordinating
  - Motivating

Health Informatics Projects

- A Health Informatics project can be characterised by features of its conceptual classification



Characteristics of a Project

Some of the key characteristics of any project are the need to

- Justify the business case
  - Through analysis of the costs and benefits
- Clearly set the project objectives and scope
- Set up the project and its interfaces to the wider organisation
- Plan and control project activities throughout its lifecycle
- Deliver the end products to the required scope
- Verify that the end products meet the stakeholders' expectations
- Close the project down
  - Whether successful or not, the project needs to end

Characteristics of an Informatics Project

For an Informatics Project, special attention should be paid to other characteristics which increase the chances of success

- Manage software development using a proven methodology
  - Systematic approaches, refined through experience
- Fulfil and manage stakeholder expectations
  - Covering both functional and non-functional requirements
- Manage change
  - Accommodating change is generally good, but it needs to be well controlled
- Understand and manage risk
  - Every project carries risks and threats to its success which need to be acknowledged and managed

The Project Management Triangle

- Key stakeholders should understand the *Project Triangle*

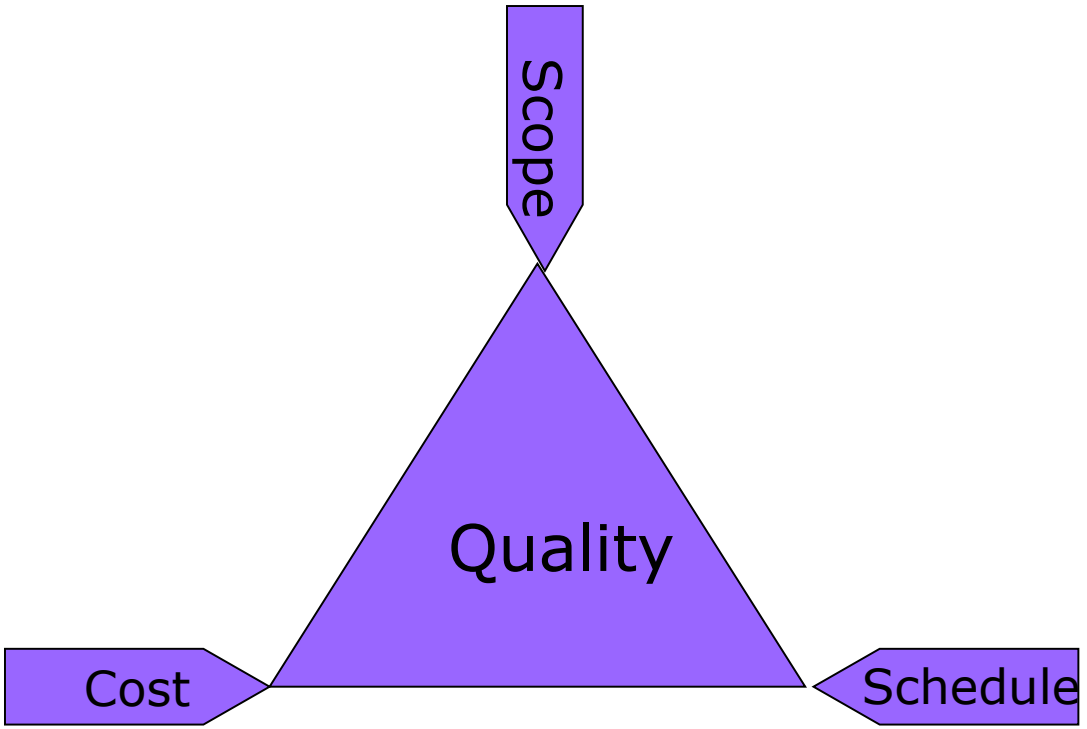
The Project Management Triangle is a model of the constraints of project management. It is often used to illustrate that project management success is measured by the project team's ability to manage the project, so that the expected results are produced while managing time and cost.

http://en.wikipedia.org/wiki/Project_management_triangle

The most usual variables in the Triangle are Cost, Schedule and Scope.

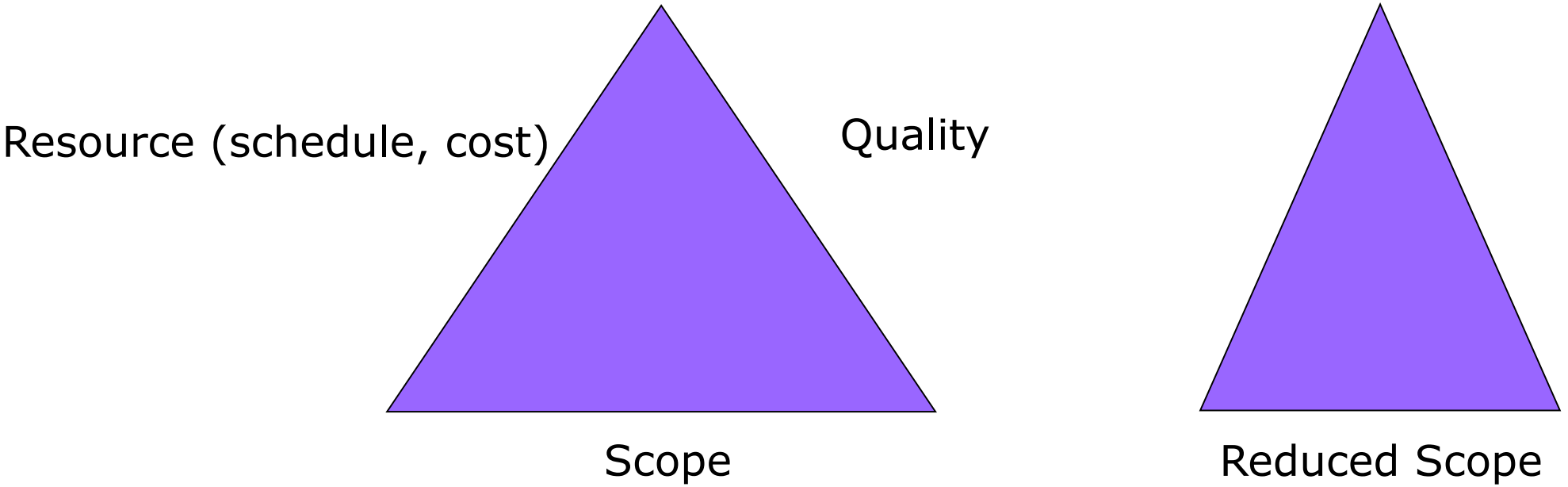
These variables can be seen as competing, in the sense that a change in the value of one will force change in the other two.

The area of the Triangle is sometimes denoted as the fourth variable – Quality.



Our Project Triangle

- Time and cost are closely related – they depend on the accuracy of the project estimates and the amount of resource deployed.
- So we like to combine them into a single variable (Resource) with the other sides of the triangle being Quality and Scope.



Usually only two of these three variables can be fixed – the third will need to be delivered under target (unless you are the lucky person who runs an IT project strictly to plan)

The best variable to flex to bring in a successful project is the Scope

Stakeholder Expectations and Requirements

A requirement is a singular documented need of what a particular product or service should be or do. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user.

http://en.wikipedia.org/wiki/Requirements

- Failing to properly manage stakeholder expectations and requirements is one of the key reasons that many projects fail
- There are three main types of requirement
  - Business - what must be delivered to provide value to the business
  - Process – methods or constraints in how the deliverables should be produced
  - System – requirements for the product (system) delivered
- System Requirements can be classified as
  - Functional – what the system does
  - Non-functional – how well the system performs in terms of usability, availability, reliability, supportability, testability, maintainability
- When requirements are elicited from stakeholders, they tend to provide both requirements and descriptions of system features

Software Development Methodology

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system

http://en.wikipedia.org/wiki/Software_development_methodology

- It has been widely recognised since the 1960's that software development is generally best practised by following a structured methodology that has been proven to achieve successful end products
- The two main characteristics of a Software Development Methodology are
  - A software development *Philosophy* which defines the overall approach to the development process
  - Tools, models and methods, to assist in the development process
    - This is actually the definition of *Architecture*, in its most general sense
- The tendency over forty years has been to move towards less structured, more collaborative software development methodologies
- It is possible (and is the approach we recommend here) to separate the software development methodology from the Project Management methodology

Managing Change

- When we speak of changes in a project we generally mean changes to
  - The overall business justification (business case)
  - Requirements as expressed by stakeholders
  - The environment in which the project and/or end products operate
- One of the worst things you can do in a project is to pretend that change won't happen and therefore have no plan for managing it when it does
- Equally as bad is creating a management regime where change is not allowed
- Successful projects acknowledge that change will happen, even encourage it, and have a robust project framework that accommodates that change

Understanding and Managing Risk

Risk can be defined as the threat or probability that an action or event, will adversely or beneficially affect an organisation's ability to achieve its objectives.

http://en.wikipedia.org/wiki/Risk

- One of the biggest risks to the success of a project is failure to recognise and manage risk itself
- Often the project stakeholders are in risk denial and choose to ignore risks which are quite evident
- Various systematic methods have been devised to manage risk in Informatics projects
- One of the key responsibilities of the Project Manager is to ensure robust risk management and communication with stakeholders regarding risks

Characteristics of a Health Informatics Project

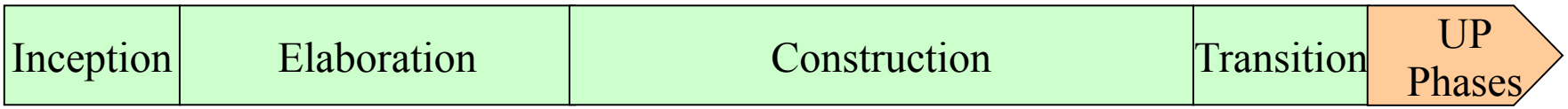
- Health Informatics projects have some specific characteristics which require careful management
  - Clinical safety
  - Patient confidentiality
  - Data Migration
  - Continuous live environment of healthcare
  - A unique set of stakeholders

Unified Process – Phases

The Unified Process is divided into four phases

- Inception
  - Specifies the vision of the end-product and its business case. Defines the scope of the project.
- Elaboration
  - Planning of activities and resources, eliciting requirements, specifying the system features and designing the architecture, eliminating the highest identified risks
- Construction
  - Entire system is developed and tested, with user documentation
- Transition
  - Move the completed product to the user community, training them in its use and signing off acceptance

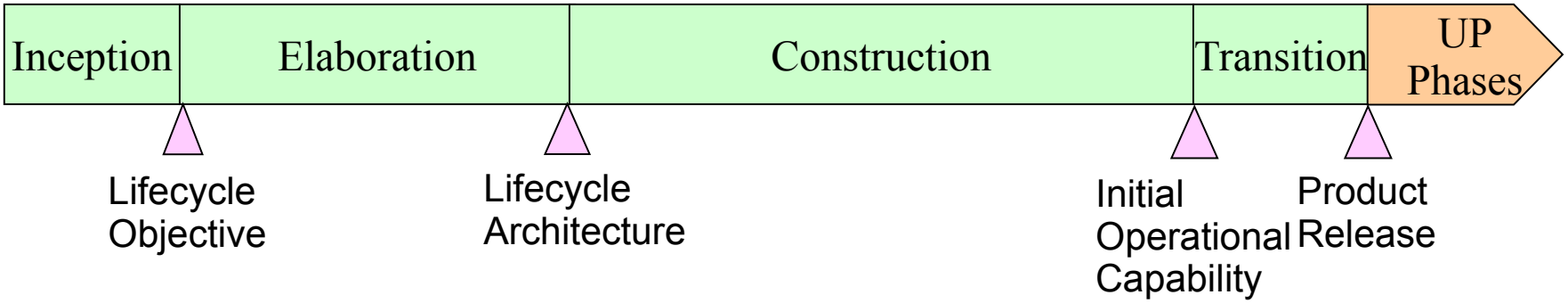
- Within each phase there may be several iterations



Unified Process – Milestones

Major milestones mark the boundaries between each phase

- Lifecycle Objective
  - Defined scope, estimated costs, understood requirements and risks
- Lifecycle Architecture
  - Detailed requirements, stable architecture, major risks resolved
- Initial Operational Capability
  - Functional system, full scope, release candidate
- Product Release
  - System accepted, objectives satisfied, ready for live operation



Disciplines

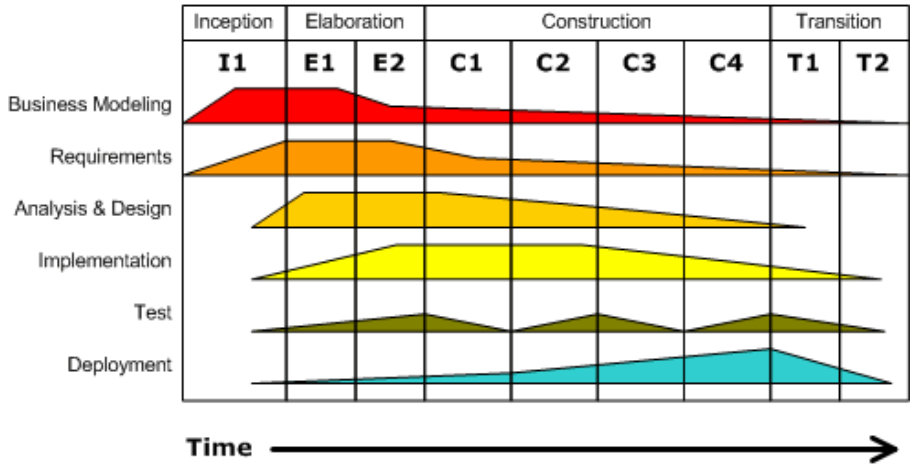
Disciplines are used as a way to group activities, according to the primary skill sets required to complete them.

The core technical disciplines are

- Business Modelling
- Requirements
- Analysis and Design
- Implementation
- Test
- Deployment

There are three additional supporting disciplines

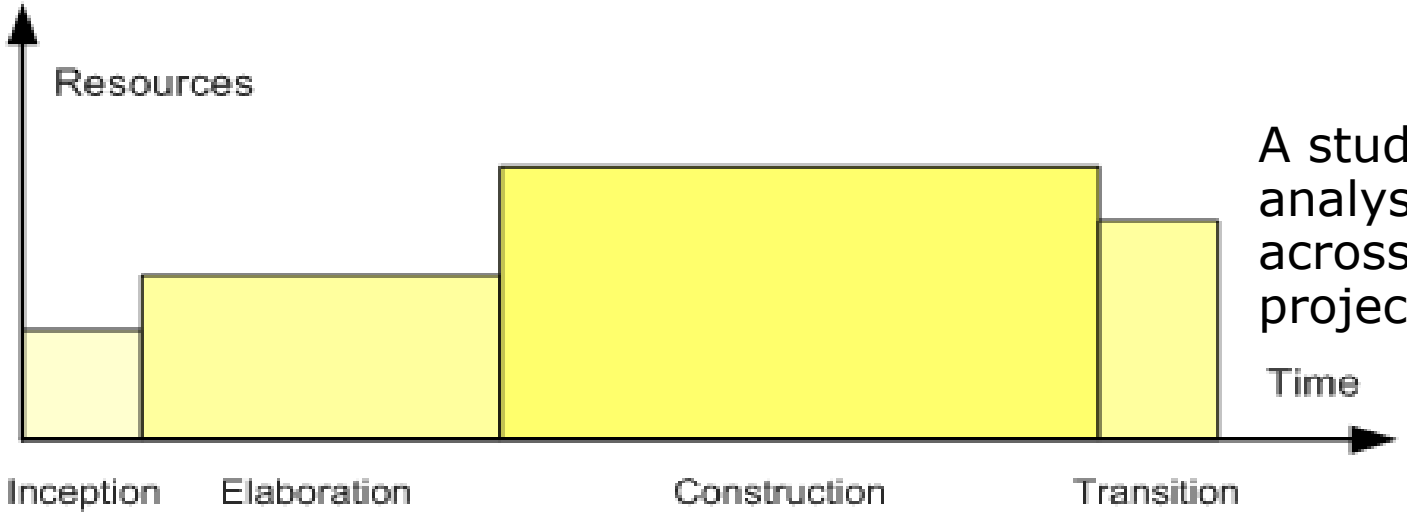
- Project Management
- Configuration / Change Management
- Environment



Each discipline has some involvement throughout the whole development lifecycle.

Knowledge Transfer which should be responsible for documentation and training deliverables falls under the Deployment discipline.

Unified Process – Metrics



A study published in 2002[7] analysed the effort and duration across a sample of eight UP projects.

Based on these benchmarks, a sensible schedule for a six month UP project is

Inception	2 weeks
Elaboration	6 weeks
Construction	3 months
Transition	1 month

Phase	Schedule	Effort
Inception	10%	5%
Elaboration	30%	20%
Construction	50%	65%
Transition	10%	10%

Using Agile Development Techniques

Lean Management

- The Agile software development process that we will learn about embraces the more general concept of *Lean Management*

> Lean, is a production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination.
>
> http://en.wikipedia.org/wiki/Lean_management

- The application of Lean concepts to software development was first made by Mary and Tom Poppendieck who cited seven main objectives
  - Eliminate waste
  - Build in quality
  - Create knowledge
  - Defer commitment
  - Deliver fast
  - Respect people
  - Optimise the whole
    - don't optimise individual tasks at the expense of the wider project

Lean Software Development

- In their book on Lean software development , Hibbs, Jewett and Sullivan describe six main practices to be followed
  - Source code control and automated builds
  - Automated testing
  - Continuous integration
  - Reuse more, use less code
  - Short iterations
  - Customer participation

- These practices are vital components of the Agile software development methodology we recommend
  - They fit well within the overall framework of the Unified Process
  - They are key to reducing risk and running successful projects

Agile Software Development – Manifesto

The Agile Manifesto is a statement of the principles that underpin agile software development. It was drafted [in] February 2001, [when] representatives of various new methodologies [..] met to discuss the need for lighter alternatives to the traditional heavyweight methodologies.

http://en.wikipedia.org/wiki/Agile_manifesto

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.[3]

Agile is Lean, But..

- Agile software development processes are certainly lean, but Agile is not a direct derivative of Lean Management

Similarities between Lean and Agile

Close involvement of the customer to deliver to actual (not perceived) requirements

Focus on delivering to the quality specified by the customer

Aim to maximise the efficiency of the development process

Encourage changes to requirements during the development process

Differences between Lean and Agile

Lean takes an overall view of the business; Agile methodologies just cover software development

Lean encompasses principles and practices, but does not have any formal methodologies; Agile includes formal methodologies, such as Scrum

The primary focus of Lean is the elimination of waste; the focus of Agile is to produce demonstrable software as quickly as possible, developing iteratively.

Scrum Methodology

One of the charms of the Rugby Union game is the infinite variety of its possible tactics. Whatever tactics a team aims to adopt, the first essential is a strong and skilful pack of forwards capable of winning initial possession from set pieces [scrums]. For, with the ball in its hands, a team is in a position to dictate tactics which will make the best use of its own particular talents...[4]

Quoted from The Oxford Companion to World Sports and Games ed John Arlott, OUP 1975.

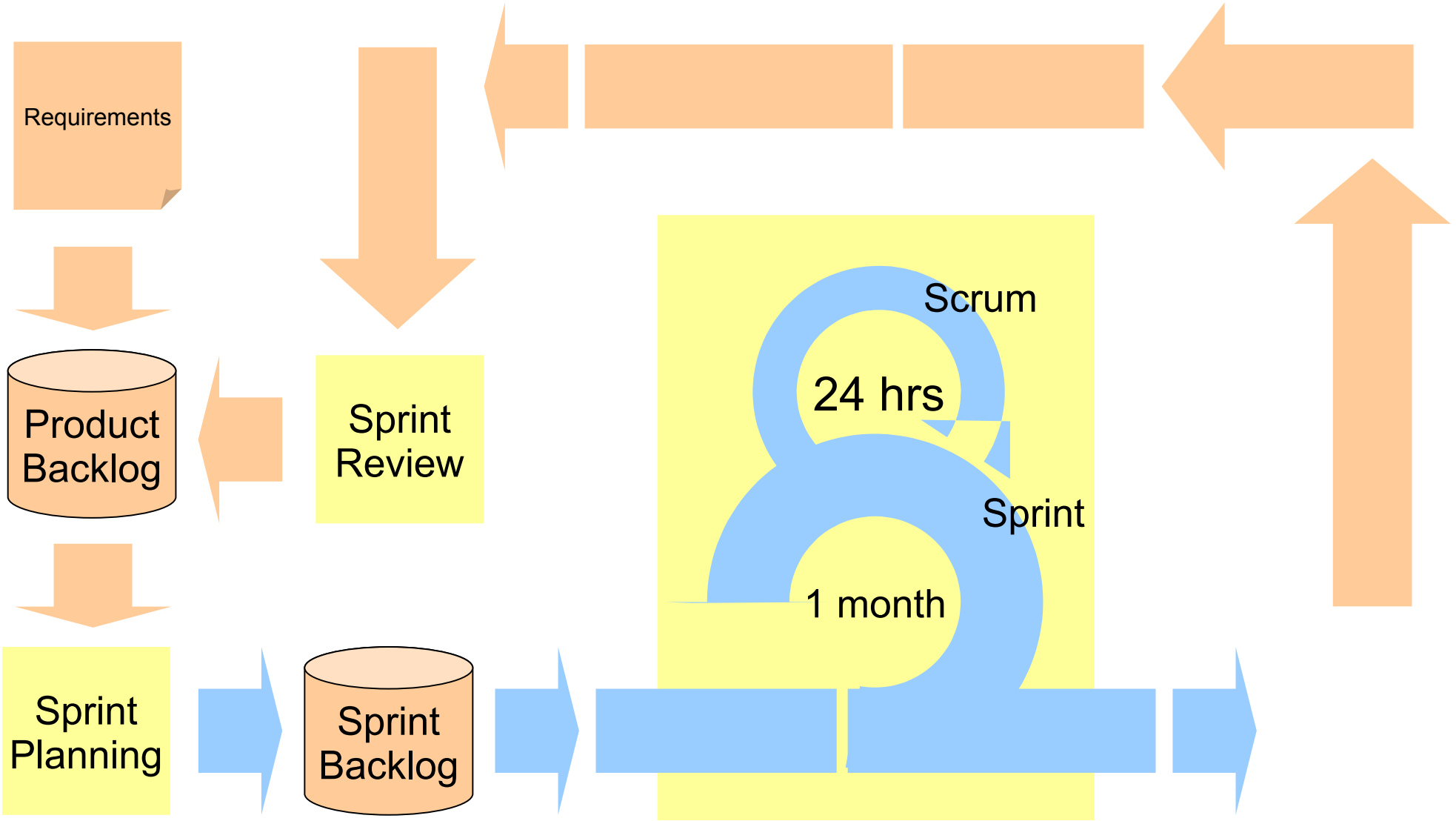
- Just as the concepts of Lean Management started in Japan, so the conception of the agile Scrum methodology came from Japan, as first mentioned by Takeuchi and Nonaka in 1986[4].
- The methodology was practised and refined by early adopters such as Ken Schwaber and Mike Beedle in the 1990's, who were key players in the formulation of the Agile Manifesto.
- This century has seen more widespread awareness and adoption of the Scrum methodology, promoted by organisations such as the Scrum Alliance[7].
- One organisation which tried Scrum in its early days (mid 1990's) was the US healthcare IT provider IDX.

Principles of Scrum

Scrum is made up of three roles, three ceremonies, and three artefacts.[8]

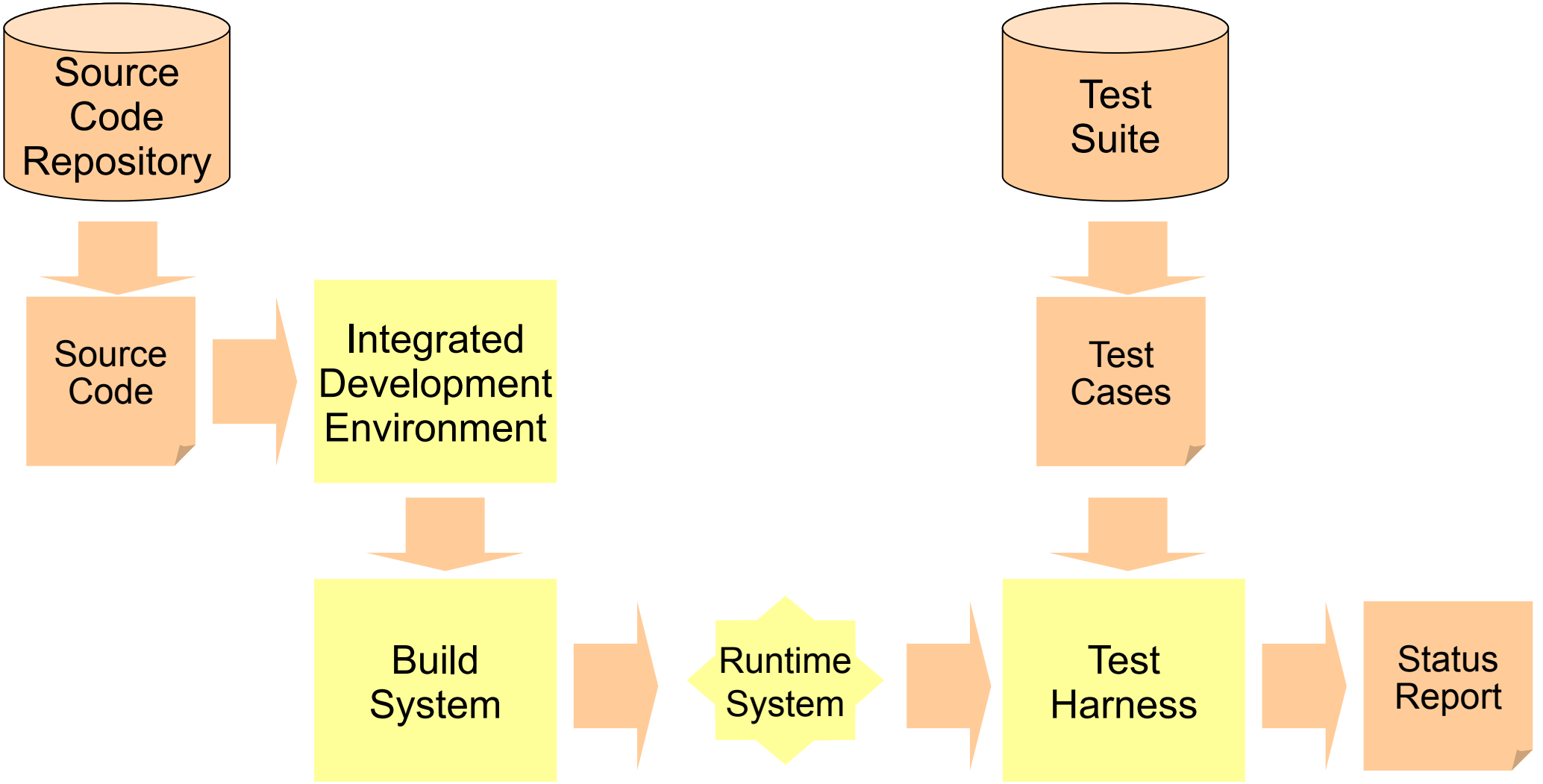
- The three key roles
  - Product Owner, Scrum Master, Team
- Three key ceremonies
  - Sprint Planning, Daily Scrum, Sprint Review
- Three key artefacts
  - Product Backlog, Sprint Backlog, Burndown Chart

Key Components of Scrum



Continuous Build and Test

- An important prerequisite to adopting Scrum is to run an Integrated Development Environment with continuous build and automated test



Roles and Responsibilities

A pig and a chicken are walking down a road. The chicken looks at the pig and says, *"Hey, why don't we open a restaurant?"* The pig looks back at the chicken and says, *"Good idea, what do you want to call it?"* The chicken thinks about it and says, *"Why don't we call it 'Ham and Eggs'?" "I don't think so,"* says the pig, *"I'd be committed, but you'd only be involved."*

http://en.wikipedia.org/wiki/Scrum_(development)

- The Pigs
  - *Product Owner* represents the customer and owns the Use Cases and Product Backlog (of system features)
  - *Scrum Master* acts as the buffer between the Team and other roles
  - *Development Team* is the self-organising unit responsible for delivery
- The Chickens
  - *Users* are the end users of the product
  - *Other stakeholders* to whom the product will deliver value
  - *Managers* (in the development organisation) have wider responsibility for organising the realisation of the product and its value to stakeholders

The Scrum Master

- The Scrum Master is not the leader of the Team, but is a member who acts as the interface between the Team and the wider project.
- The responsibilities of the Scrum Master include
  - Liaising with the Product Owner
  - Forming Scrum Teams
  - Planning the Sprint
  - Organising the Daily Scrum
  - Manage the Sprint Backlog
  - Measure and report progress

- For very small projects, the roles of Scrum Master and Project Manager may be combined
- Sometimes the role may also be combined with that of Product Owner

Project Management Methodologies

Project Management Methodologies

- The Unified Process and Scrum are software development methodologies
- They include some insights on management of the overall project (especially the UP) but this is not their focus
- There are three widely adopted methodologies that are dedicated to project management
  - PMBOK
  - CMMI
  - PRINCE2

PRINCE2

PRINCE2 is a generic, tailorable, simple to follow project management method. It covers how to organise, manage and control your projects. It is aimed at enabling you to successfully deliver the right products, on time and within budget.

http://www.ogc.gov.uk/methods_prince_2.asp

- Originally a vendor-owned methodology for managing IT projects
- Bought by UK government and renamed as PRINCE
  - Projects in Controlled Environments
- Made into a generic project management methodology in 1996
  - PRINCE2
- Latest version released in 2009
- Owned by the Office of Government Commerce (OGC)
  - Freely available for anyone to use

Prince2 Overview

- The five key features of PRINCE2 listed by the OGC are[1]:
  - business justification
  - defined organisation structure for the project team
  - product-based planning approach
  - dividing projects into controllable stages
  - flexibility to apply at an appropriate level of detail

- The overall methodology consists of
  - A process model
    - with six processes
  - A component model
    - with eight components
  - A commitment to product-led design

Project Governance

- One of the key strengths of PRINCE2 is its approach to Project Governance

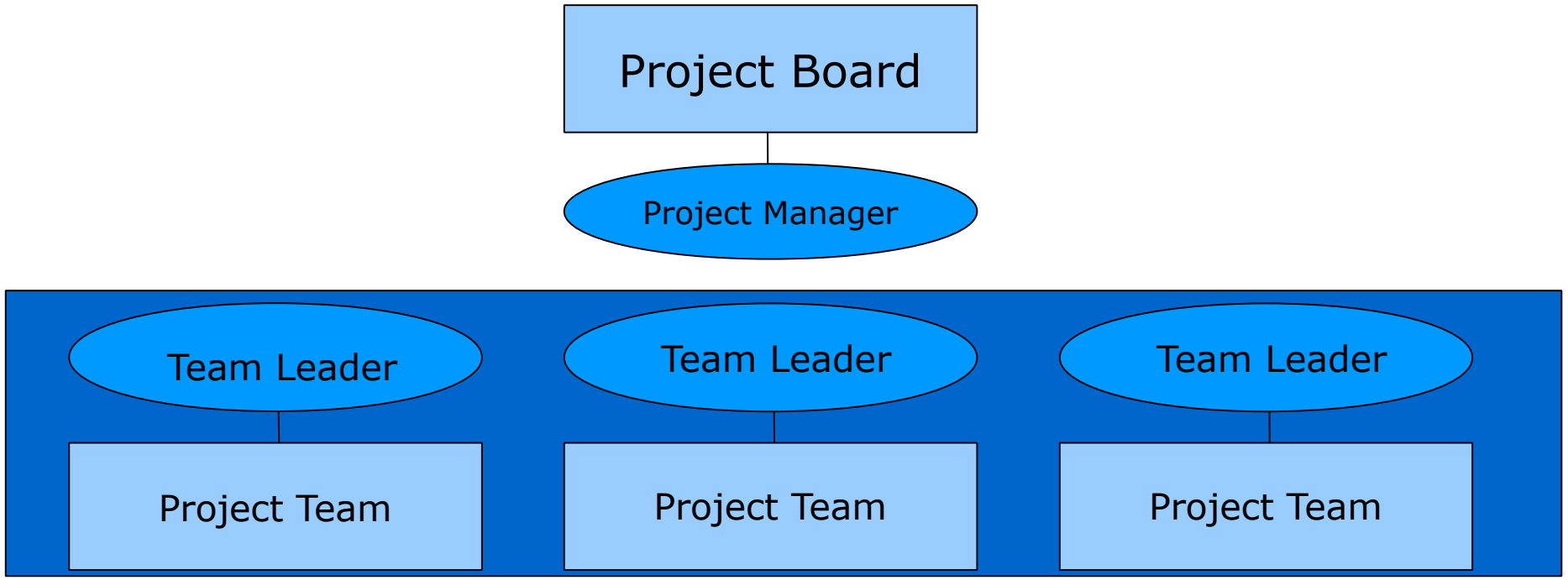
Governance relates to consistent management, cohesive policies, processes and decision-rights for a given area of responsibility.

http://en.wikipedia.org/wiki/Governance

- The key elements of project governance are[2]
  - Defining the relationships between all stakeholders
  - Describing how project information should flow between stakeholders
  - Ensuring appropriate review of project issues
  - Ensuring adequate approvals are obtained at each stage of the project

Project Organisation

- PRINCE2 provides a clear structure for the organisation of a project and the allocation of roles and responsibilities
- A basic overall project organisation chart is recommended by PRINCE2
- The Project Manager plays a key role as the interface between the project teams and the Project Board



The Project Board

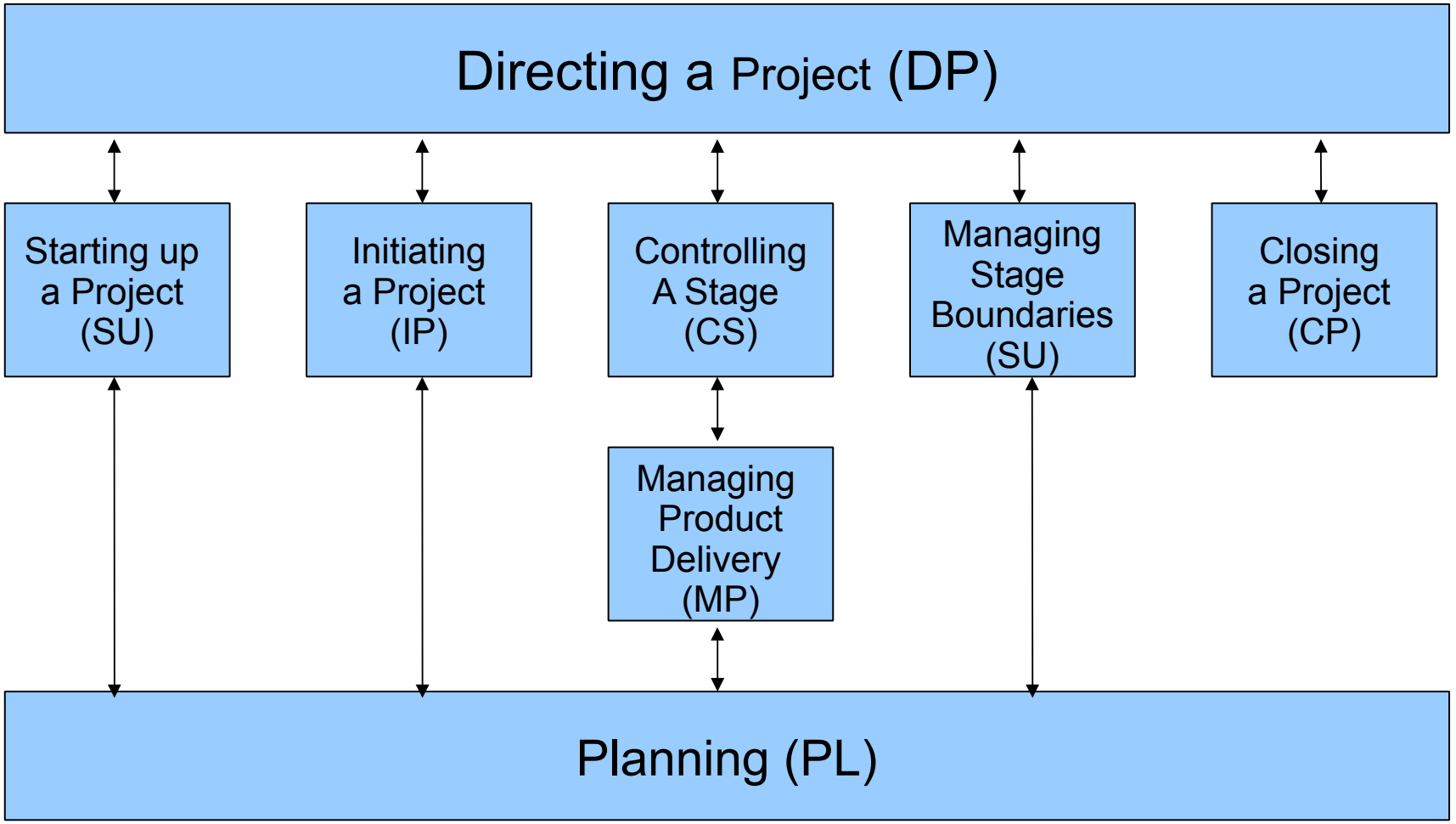
- The Project Manager reports to the Project Board which assumes *ownership* of the project
- The Project Board represents the three key types of stakeholder
  - Business
    - Realise the benefits of the end products and carry the cost of the project
  - User
    - Interact with the end products once the project is completed
  - Supplier
    - Is responsible for delivering the end products and managing the project

Members of the Project Board

- Three key roles should be present on the Project Board
  - Project Executive
    - 'Owns' the project and is responsible for the Business Case
    - Chairs the Project Board and is the ultimate decision-maker
  - Senior User
  - Senior Supplier

PRINCE2 Process Model

- The PRINCE2 Process Model defines a sequence of activities for the project which is managed in Stages



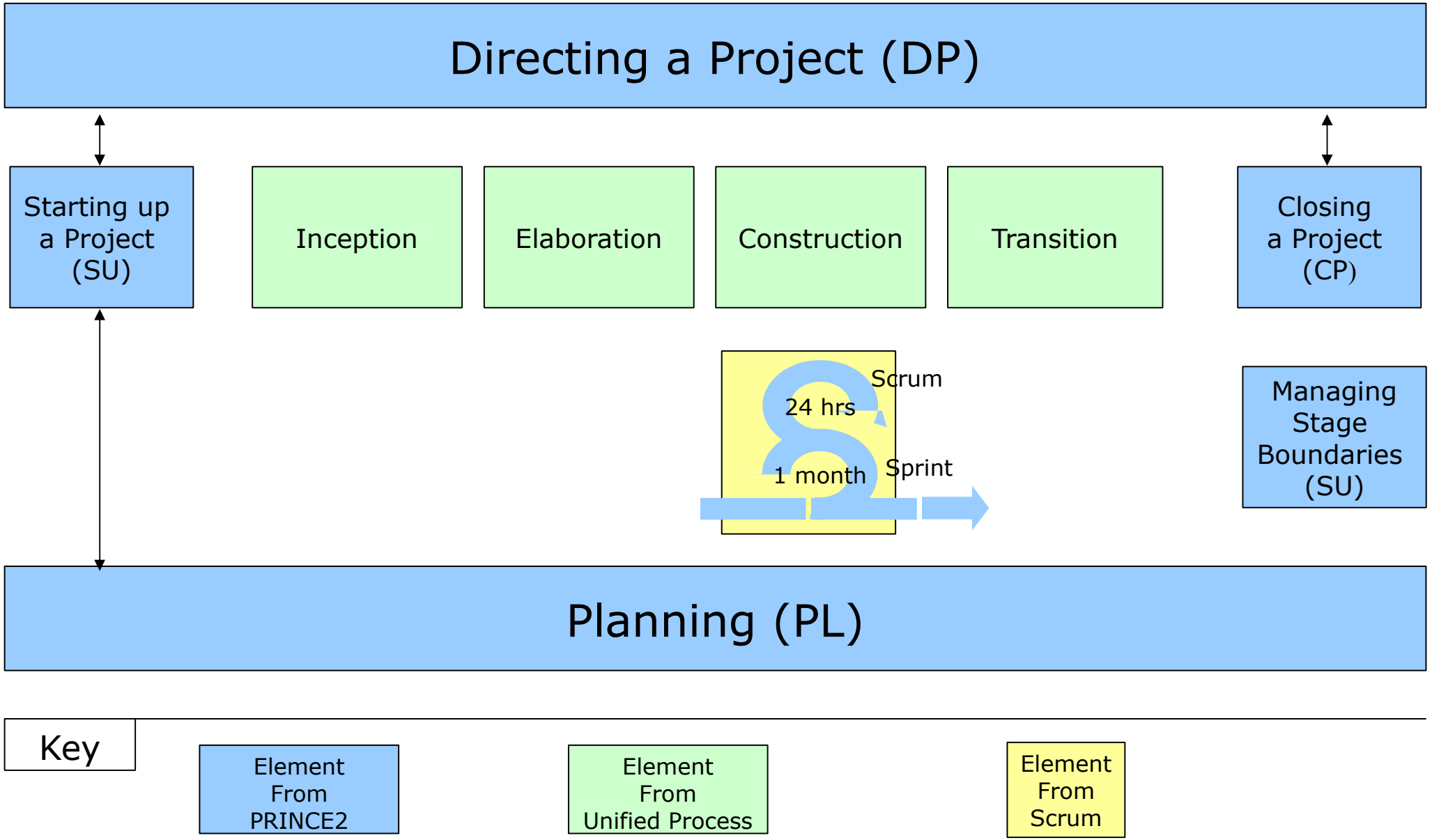
Project Management Methodology

- There is clear evidence that suggests the best methodology for managing a Health Informatics Project
  - Many projects fail
  - The causes of failure are known and preventable
  - Good use of a project management methodology helps to address these issues
  - Health Informatics Projects can be managed successfully using a methodology that combines the best of
    - PRINCE2
    - Unified Process
    - Scrum

Selecting the Methodology

- Combined methodology aims to build on the strengths of each, based on the available evidence that
  - PRINCE2 is strong on project governance and overall (generic) project management process
  - The Unified Process is a good overall software development methodology
  - Agile/Scrum is best for iterative development that fits within the iterative approach of UP and the Stages of PRINCE2
- Evidence from the English National Programme for IT shows that
  - PRINCE2 was required for overall control of a UK public sector IT project
  - The Unified Process was a good model of (iterative) software development for products and solutions within NpfIT projects
  - Introduction of Agile/Scrum yielded improved results and delivered better working systems

Combined Methodology for Health Informatics



Project Estimating and Scheduling

How to do Project Estimating

In his excellent book on *IT Project Estimation*[1], Paul Coombs describes twelve rules of estimation

1. Your estimate will be wrong
2. You can always provide an estimate
3. Every estimate must have a contingency allowance
4. It is harder to make the list of items to be estimated than it is to estimate each item
5. The quality of your estimate depends on your familiarity with the proposed project
6. The more bits its in, the longer it gets
7. Write down your assumptions
8. The contingency allowance is proportional to the risk
9. There is no method that works
10. The project duration in months must be greater than the average number on the team
11. Have someone else review your estimate
12. Hold a project post mortem

Project Estimating – Basic Technique

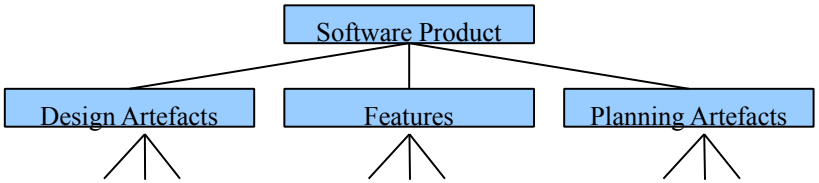
- The basic process of project estimating consists of four steps
    1. Break down the project into a set of elements (tasks)
    2. Estimate the effort required to complete each task
    3. Add all the estimates to get an overall figure
    4. Add a contingency

- There are various techniques for calculating the estimates, including
    - Three point analysis
    - Wideband Delphi
    - Function point analysis
    - COCOMO
- Here we will take a quick look at the first two of these
    - For information on the others see the book by Paul Coombes[1]

Work Breakdown Structure

A work breakdown structure (WBS) in project management and systems engineering, is a tool used to define and group a project's discrete work elements (or tasks) in a way that helps organize and define the total work scope of the project. http://en.wikipedia.org/wiki/Work_breakdown_structure

- Original concept came with the Program Evaluation and Review Technique (PERT) used in the defence sector
- Expanded to cover software products by the Project Management Institute and documented in the PMBOK



- Start with the overall end **product** and break it down into a hierarchy of **sub-products** (think always of products, not tasks)
- Each sub-product has a clearly identified owner in the project and a set of tasks that is required to produce it
- The WBS forms a stable basis for estimating (and re-estimating) the effort (and then resources, cost, schedule) for delivering the project

WBS Principles

- 100% Rule
  - The WBS must contain 100% of the sub-products created, including all design, test and planning artefacts
  - In the hierarchical breakdown, the sum of work in any child elements must equal 100% of the work in the parent
- Mutually exclusive elements
  - There can be no overlap in the products
- What to breakdown – products or activities?
  - The elements in the breakdown should be products, not activities
  - This fits with the product-led principle of PRINCE2
  - Which for Scrum equates to feature-driven estimates
- How far to breakdown?
  - The level of detail to decompose the products can be determined by the 'rules' about the activities that produce them

Three Point Analysis – Task Estimation

The three-point estimation technique is based on statistical methods, and in particular, the normal distribution.

http://en.wikipedia.org/wiki/Three-point_estimation

For each task, three estimates are made

a = the best-case estimate

m = the most likely estimate

b = the worst-case estimate

The estimate, E, and standard deviation SD are made assuming a triangular distribution of the data

$E = (a+4m+b)/6$

$SD = (b-a)/6$

Three Point Analysis - Project Estimating

To make an overall project estimate, based on task estimates

1. Break the project into a set of tasks (work breakdown structure)
2. Estimate the E value and SD for each task.
3. Calculate the E value for the total project work as

   $E(\text{Project Work}) = \Sigma\, E(\text{Task})$

1. Calculate the SD value for the total project work as

   $SD(\text{Project Work}) = \sqrt{\Sigma\, SD(\text{Task})^2}$

Confidence levels (i.e. can be derived from the estimates as follows

- Confidence level for E is approximately 50%
- Confidence level for E+SD is approximately 70%
- Confidence level for E+2×SD is approximately 95%
- Confidence level in E+3×SD is approximately 99.5%

Recommended to use the 95% confidence level (E+2×SD) for all project and task estimates.

Project Estimating – Wideband Delphi

The Wideband Delphi estimation method is a consensus-based estimation technique for estimating effort. It derives from the Delphi Method which was developed in the 1940s at the RAND Corporation as a forecasting tool.
http://en.wikipedia.org/wiki/Wideband_Delphi

- The technique is similar to Three Point Estimating but without a statistical basis
- The WBS is given to a team (ideally of three to five) and each individual produces their own estimate of each element
- The team then meets to review the estimates and reach consensus on the amount of effort and the assumptions
- The documented assumptions can be used to track risks associated with the final estimate

Project Scheduling

- Closely linked to estimating is the task of project scheduling
  - It has been estimated that only about 20% of projects are completed within the original schedule[3]
- To many, the task of scheduling a project comes down to drawing a Gantt chart using MS Project and trimming the activities to fit within a pre-determined timeframe which is deemed acceptable
- The process of project scheduling should occur once the WBS and estimates are complete
  - If the WBS doesn't fit in the available time, then decide which of the Project Triangle variables will flex
  - Usually this should be the project scope, but can at the start be the project resources
    - i.e. change the amount of effort, or extend the timescale

Project Scheduling – Gantt Charts

A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts have become a common technique for representing the phases and activities of a project work breakdown structure (WBS), so they can be understood by a wide audience.

http://en.wikipedia.org/wiki/Gantt

- For many people the Gantt chart is synonymous with MS Project
  - http://office.microsoft.com/project/
- A good cross-platform, open source Gantt tool is GanttProject

  http://www.ganttproject.biz/

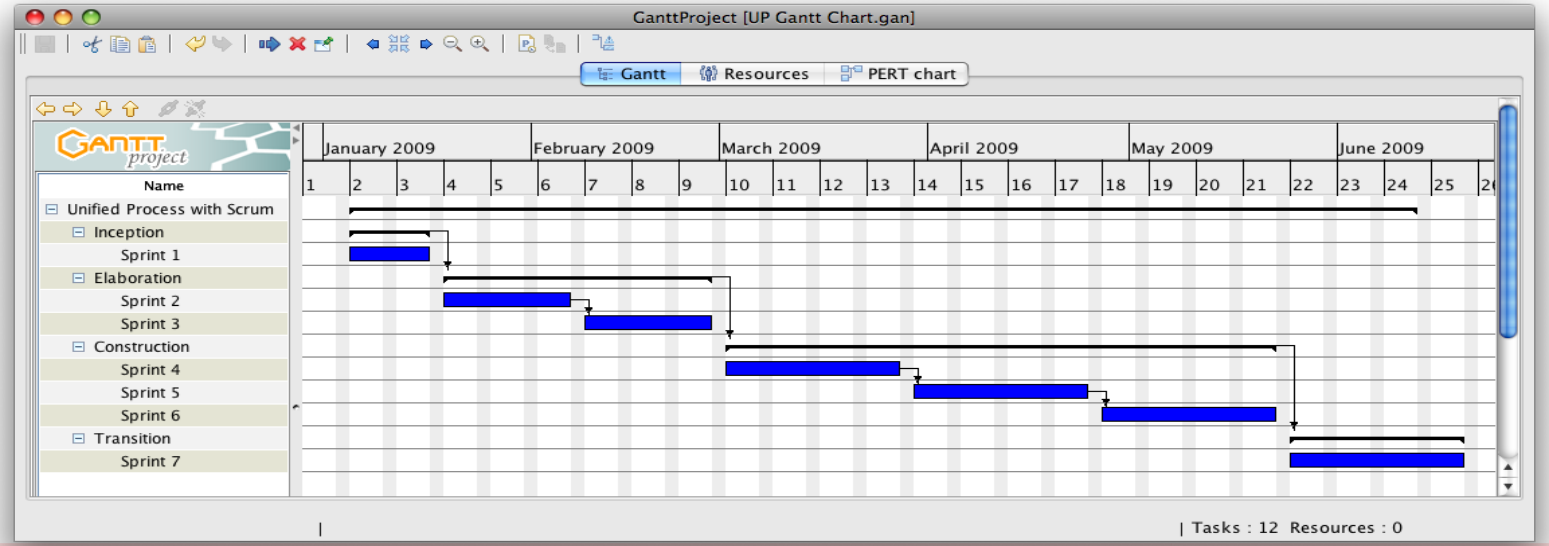
  Or try OpenProj

  http://sourceforge.net/projects/openproj/

- Ideally, a Gantt chart should be used as a visualisation and communication tool
- Used *after* creating a Work Breakdown Structure

Using Gantt Charts

- It is easy to misuse the Gantt chart and fall into the trap of thinking that the chart *is* the project plan
- No activity on the chart is achievable unless it has been
  - Estimated
  - Provided with sufficient time and resources to complete
- Ideally, the Gantt chart is a visual aid to presenting the project plan and progress towards goals
  - Not the tool that is used for estimating, resourcing or planning itself



Ensuring Patient Safety in Health Informatics Projects

DSCN 14/2009 and DSCN 18/2009

DSC Notice: 14/2009 V1.2 issued in August 2010 by the NHS Information Standards Board and entitled "Application of Patient Safety Risk Management to the Manufacture of Health Software"

To be compliant with the standard, the manufacturer must establish, document and maintain throughout the lifecycle an ongoing process for identifying clinical hazards associated with the health software product, estimating and evaluating the associated clinical risks, controlling these risks, and monitoring the effectiveness of the controls throughout the lifecycle. This process must include the following elements:

- context, requirements and scope identification;
- creation of clinical risk management plan;
- setting the requirements for and defining the competencies of personnel;
- clinical hazard identification;
- clinical risk analysis;
- clinical risk evaluation;
- clinical risk control;
- residual clinical risk acceptance;
- creation of clinical safety case report(s);
- post deployment monitoring;
- post-production maintenance of clinical risk management process.

Details of how clinical risk should be managed in health software are provided in the DSCN18/2009 "Health informatics — Guidance on the management of clinical risk relating to the deployment and use of health software Formerly ISO/TR 29322:2008(E)"

Clinical Risk Management Plan

In accordance with DSCN 18/2009, this plan includes at least the following:

The scope of the planned clinical risk management activities, including identifying and describing the health software system, what it is intended to do, how it will do it, the clinical context in which it will be used and the life-cycle phase(s) which the plan covers

Assignment of responsibilities and authorities

Requirements for review of clinical risk management activities

Identification of relevant risk management procedures and processes to be used

Criteria to be used in analysis and evaluation of the risks

Criteria for assessing clinical risk acceptability, based on the organization's policy for determining acceptable clinical risk and which are fundamental to the overall success of the risk management process

Verification activities

Activities in case of changes to the health software system (e.g. patches, updates, etc.)

Activities related to the collection and review of relevant post-deployment information and the feedback of that information both into the health organization's risk management processes and into the relevant manufacturers' processes, e.g. through health software systems, user groups or geographic communities or professional groups etc.

Clinical Safety Case Report

The template for a Clinical Safety Case Report contains the following sections (as per DSCN 18/2009):

1. Introduction and system description including how this safety case report relates to other interrelated systems and any wider governance arrangements.

2. Process management systems applied up to the relevant point in the life-cycle process.

3. Hazard identification and risk assessment process.

4. Applicable criteria.

5. Statement of residual risk.

6. Overall clinical safety justification.

7. References and supporting information.

Assessing Clinical Risk

Clinical risk is defined in DSCN 18/2009 as the combination of the likelihood of occurrence of harm and the severity of that harm.

Likelihood of occurrence of harm

Likelihood	Meaning
Very high	Certain or almost certain; highly likely to occur
High	Not certain but very possible; reasonably expected to occur in the majority of cases
Medium	Possible; not unlikely to occur
Low	Could occur but in the great majority of occasions will not
Very low	Negligible or nearly negligible possibility of occurring

Clinical Risk – Severity of Harm

Severity	Consequence	Patients affected
Catastrophic	Death	Multiple
	Permanent life-changing incapacity and any condition for which the prognosis is death or permanent life-changing incapacity; severe injury or severe incapacity from which recovery is not expected in the short term	Multiple
Major	Death	Single
	Permanent life-changing incapacity and any condition for which the prognosis is death or permanent life-changing incapacity;	Single
	Severe injury or severe incapacity from which recovery is expected in the short term	Multiple
	Severe psychological trauma	Multiple
Considerable	Severe injury or severe incapacity from which recovery is expected in the short term	Single
	Severe psychological trauma	Single
	Minor injury or injuries from which recovery is not expected in the short term	Multiple
	Significant psychological trauma	Multiple
Significant	Minor injury or injuries from which recovery is not expected in the short term.	Single
	Significant psychological trauma	Single
	Minor injury from which recovery is expected in the short term	Multiple
	Minor psychological upset; inconvenience	Multiple
Minor	Minor injury from which recovery is expected in the short term; minor psychological upset;	Multiple

Clinical Risk Acceptability Matrix

The acceptability of clinical risk is classified on a scale of 1 to 5, where the scale can be interpreted as follows:

1. Acceptable risk, no further action required
2. Acceptable but notable risk, should be monitored during ongoing use
3. Unacceptable but manageable risk - a risk control action should be taken to avoid or mitigate so that the risk can be brought into the acceptable range
4. Unacceptable and difficult to manage - should investigate a risk control action to avoid or mitigate, but any such action is likely to incur significant cost
5. Unacceptable, with little or no likelihood that any action can be taken to avoid or mitigate

Likelihood	Severity of Harm				
	Minor	Significant	Considerable	Major	Catastrophic
Very high	3	4	4	5	5
High	2	3	3	4	5
Medium	2	2	3	3	4
Low	1	2	2	3	4
Very low	1	1	2	2	3

Statement of Residual Risk

After all clinical risk control measures have been implemented and verified some risks will remain – these are the residual risks.

The organization will need to decide if the overall residual clinical risk posed by the health software system is acceptable. This is achieved using the criteria defined in the clinical risk management plan.

Policy For Acceptance of Residual Risk

One common risk acceptability policy is the "as low as reasonably practicable" (ALARP) approach outlined in DSCN18/2009, which states:

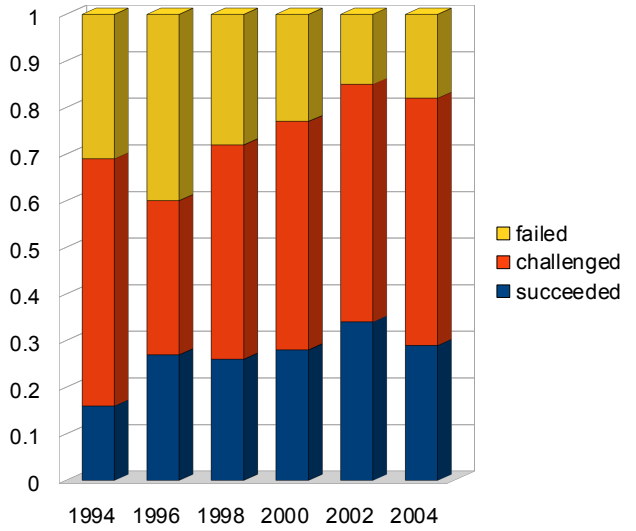
After a particular risk control option has been applied there are three possible results:

1. the residual risk exceeds the agreed criterion for risk acceptability;
2. the residual risk is acceptable because it is so small as to be insignificant;
3. the residual risk is between the two states above; for these risks, the residual risk is acceptable for the option that reduces the risk to the lowest practicable level, bearing in mind the benefits resulting from its acceptance and taking into account the (intolerable) costs involved in any further reduction.

Why do Informatics Projects go Wrong?

High Risk of Failure

- Evidence suggests that a large proportion of Informatics projects are unsuccessful
- Study on 214 EU projects between 1998 and 2005 shows that nearly a quarter (23.8%) failed completely[1]
- The classic survey on IT project failure was the 1995 CHAOS Report from the Standish Group[2]
  - This survey has consistently shown failure rates of 25%-30%
  - And around 50% of projects fail to meet the original objectives in some significant way



Sustained Level of Failure

- Since the first CHAOS report, work by the Standish Group showed improvements in success rates, but this changed in 2009
- Standish Group – 1995 CHAOS Report[2]
  - Quantitative survey results (for 1994)
    - 31% of software projects fail to be completed
    - 53% 'challenged' on time, budget, features
    - 16% deemed to be successful
    - On average, completed projects had 42% of the originally planned features
  - Results in the 2009 survey show[3]
    - 24% of software projects fail to be completed
    - 44% 'challenged' on time, budget, features
    - 32% deemed to be successful
    - Which is a downturn, following many years of reported improvement since the original study in 1994

Success Criteria

- Define success criteria during project inception
  - Refine and reset as the project progresses

- Define success for each stakeholder group
  - Business owners
  - System users
  - Service users
  - Developers

Examples of Success Criteria

Business owners

Satisfies users and service users

Provides value for money

Meets statutory requirements

Service users

Improves quality of care

Achieves better clinical outcome

Informs choices

System Users

Satisfies service users

Saves time

Reduces risk and errors

Developers

Satisfies the stakeholders

Profitable

Reference for further business

Why do things go wrong?

NAO/OGC list of commons causes of project failure[4]

1. Lack of clear link between the project and the organisation's key strategic priorities, including agreed measures of success.
2. Lack of clear senior management and Ministerial ownership and leadership.
3. Lack of effective engagement with stakeholders.
4. Lack of skills and proven approach to project management and risk management.
5. Too little attention to breaking development and implementation into manageable steps.
6. Evaluation of proposals driven by initial price rather than long-term value for money (especially securing delivery of business benefits).
7. Lack of understanding of and contact with the supply industry at senior levels in the organisation.
8. Lack of effective project team integration between clients, the supplier team and the supply chain.

Maximising Chances of Success

- Make sure the Project Executive (owner) understands this list
- Pay attention to all stakeholders
  - Educate and involve stakeholders in good implementation practice
- Establish acceptance and success criteria
- Decide what's going to give in the Project Triangle
  - resource, scope or quality
- Don't pretend there aren't risks
  - Measure and manage the risks
- Don't pretend that change doesn't happen
  - Allow changes, in a controlled way
- Use iterative, agile development processes
- If the project seems crazy, don't do it!

SWOT Analysis

- SWOT analysis is a useful way to identify risks and changes which might impact a project

STRENGTHS	WEAKNESSES
Internal attributes that are helpful to achieving the objective e.g. knowledge of team, buy-in of stakeholders	Internal attributes that are harmful to achieving the objective e.g. no support from CEO, first use of new technology,
OPPORTUNITIES	**THREATS**
External conditions that are helpful to achieving the objective e.g. prove new technology, the users urgently need this system	External conditions that are harmful to achieving the objective e.g. Government targets change, Service Users increase massively

- Strengths, Weaknesses, Opportunities and Threats are usually elicited in a 'brainstorming' type session with stakeholders.

Every Project Has a Risk of Failure

- A Risk is anything that can stand in the way of 'success' in the project and is currently unknown or uncertain
  - Success means meeting the original success or acceptance criteria
- Risks can be categorised as
  - Direct risks – can be controlled by the project
  - Indirect risk – outside the project control (threats)

Costs, Gain, Risk www.standishgroup.com

- Total Cost of Ownership (TCO)
  - Originating in 1980's, popularised by Gartner Group
  - Usually far more than the cost of acquisition
  - See www.tcotool.org
- Return on Investment
  - In financial terms, the amount of money made (saved) compared with the cost
- Risk
  - in quantitative terms, it's the probability of an undesired outcome
  - threats to the organization's success, such as physical, estimating, financial, or political
  - Trade off between the probability of an event (risk) materialising and its impact
    - i.e. some low probability risks can have a very high impact if the do materialise

Managing Risk

- Every project should run a Risk Register

- To manage risk effectively, the project must be proactive in deciding for each risk whether to
  - Avoid – change the project so that it is not affected
  - Transfer – move the impact outside the project
  - Accept – decide to live with it, and plan accordingly

- When accepting a risk the project should
  - Mitigate – take steps to reduce likelihood and/or impact
  - Plan contingency – what to do is the risk materialises

Don't be surprised when a risk with a high level of probability does actually happen – saying 'I told you so' isn't proper management

Risk Register

- A Risk Register should be structured along the lines shown below
- Regular review and update, with summary at Project Board meetings
- Immediate review/update if key risks materialise

- The risk register should consist of entries with (at least) the following elements:
  - Unique ID
  - Description
  - Likelihood (e.g. very high, high, medium, low, very low)
  - Impact (severity of harm, when patient safety)
  - Policy (e.g. avoid, transfer, accept = residual risk)
  - Mitigation plan

Example Risk Register

ID	Description	Likelihood	Impact	Policy	Mitigation
1	Drug information is not available when clinicians view prescription data	High	Severe	Avoid	Integrate drug diction using standard interfaces
2	Data transfer to external systems will not be acceptable	Medium	Significant	Accept	Use open standard format for data and render in XML so that the data can be easily transformed to other formats
3	Data migration will expose legacy data to unauthorised users	High	Major	Avoid	Ensure that all staff involved in data migration are properly trained and CRB checked
4	Clinical users like their existing system and may not accept the new one	Medium	Major	Accept	Involve users in system requirements definition and include representatives in each Sprint Review
5	Estimated budget is too low to complete the project	High	Major	Accept	Use agile Scrum Methodology to make sure deliverables are available within budget

Change Happens

- As the saying goes - “Change Happens”
  - So we need to be ready to deal with it
- Changes can be within the project
  - impacting scope, resources, quality
  - Most often the result of
    - changing requirements
    - wrong assumptions/estimates
    - unforeseen problems in design or implementation
- Changes can be external to the project
  - Changed business drivers/requirements
  - Changes to stakeholder groups, environments, etc

Managing Change

- Agile development processes, such as Scrum, are designed to cope with change
- Changes to scope, resources and quality can be accommodated
- Sprint review, sprint planning
- More formal change control mechanisms may be required
- e.g. if requirements or design specifications have been 'signed off' and contractually committed

References and Further Reading

References and Further Reading

Managing Health Informatics Projects

1. Royce, Winston (1970) *Managing the Development of Large Software Systems.* Proceedings of IEEE WESCON 26 (August): 1-9.
   Available at: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

2. Wikipedia. *Unified Process* http://en.wikipedia.org/wiki/Unified_process

3. Jacobson, Booch and Rumbaugh (1999) *The Unified Software Development Process.* ISBN 0201571692

4. Kruchten P (2003) *The Rational Unified Process – An Introduction.* Addison-Wesley. ISBN 032119770-4

5. OpenUP. http://epf.eclipse.org/wikis/openup/

6. Arlow J and Neustadt I (2005) *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design.* Addison-Wesley. ISBN 0321321278

7. D. J. Reifer (2002) *Let the Numbers Do the Talking.* CrossTalk, March 2002, pp. 4-8.

References and Further Reading

Using Agile Development Techniques

1. Poppendieck M and Poppendieck P (2003) *Lean Software Development: An Agile Toolkit for Software Development Managers.* Addison-Wesley. ISBN 0321150783
2. Hibbs C, Jewett S, Sullivan M. (2009) *The Art of Lean Software Development.* O'Reilly. ISBN 978-0-596-51731-1
3. Beck *et al* (2001) The Agile Manifesto for Software Development. http://agilemanifesto.org/
4. Takeuchi and Nonaka (1986). *The New New Product Development Game.* Harvard Business Review (January 1986) pp 137-146.
5. Agile Advice. *Working With Agile Methods (Scrum, XP, Lean)* http://www.agileadvice.com/
6. Schwaber K, Beedle M. (2002) *Agile Software development with Scrum.* Prentice Hall. ISBN 0-13-2077489-3
7. Scrum Alliance. www.scrumalliance.org
8. Scrum Alliance. *What is Scrum?* http://www.scrumalliance.org/pages/what_is_scrum
9. Wikipedia. *Scrum (development)* http://en.wikipedia.org/wiki/Scrum_(development)

References and Further Reading

Project Management Methodologies

1. OGC PRINCE2 homepage. http://www.ogc.gov.uk/methods_prince_2.asp
2. *Project Governance.* Wikipedia. http://en.wikipedia.org/wiki/Project_governance
3. PMI. *A Guide to the Project Management Body of Knowledge (PMBOK Guide) - 4th Edition.* ISBN: 978-1-93389051-7
4. OGC. *Managing Successful Projects with PRINCE2.* TSO. ISBN: 978-0-11331059-3
5. Graham N (2008). *PRINCE2 For Dummies.* John Wiley. ISBN 978-0-470-51919-6
6. Jay M. Siegelaub (2004) *How PRINCE2 Can Complement PMBOK and Your PMP.* PMI Global Congress Proceedings. http://www.prince2.com/prince2-download.asp

References and Further Reading

Project Estimating and Scheduling

1. Coombs, Paul (2003). *IT Project Estimation - A Practical Guide to the Costing of Software.* Cambridge University Press. ISBN 0-52153285-X
2. Wikipedia. Work Breakdown Structure. http://en.wikipedia.org/wiki/Work_breakdown_structure
3. PMI. *Practice Standard for Work Breakdown Structures.* ISBN 1933890134
4. Wikipedia. Three Point Estimation http://en.wikipedia.org/wiki/Three-point_estimation
5. Wikipedia. Wideband Delphi. http://en.wikipedia.org/wiki/Wideband_Delphi
6. Standish Group International (2009). CHAOS Summary 2009. www.standishgroup.com/newsroom/chaos_2009.php
7. Wikipedia. *Gantt Chart.* http://en.wikipedia.org/wiki/Gantt_chart
8. Open Office (2005). Project-Management with Gantt-Charts. http://documentation.openoffice.org/HOW_TO/

References and Further Reading

Why do Informatics Projects go Wrong?

1. McManus J & Wood-Harper T (1998). *A Study in Project Failure.* British Computer Society. www.bcs.org/server.php?show=ConWebDoc.19584
2. Standish Group International (1994). *The Chaos Report.* Available at www.net.educause.edu/ir/library/pdf/NCP08083B.pdf
3. Standish Group International (2009). *CHAOS Summary 2009.* www.standishgroup.com/newsroom/chaos_2009.php
4. Office of Government Commerce (2005). *Common Causes of Project Failure.* www.ogc.gov.uk/documents/cp0015.pdf