Process, Teams and Comms

Don’t run around in a panic

Reusing this material

Quick Overview

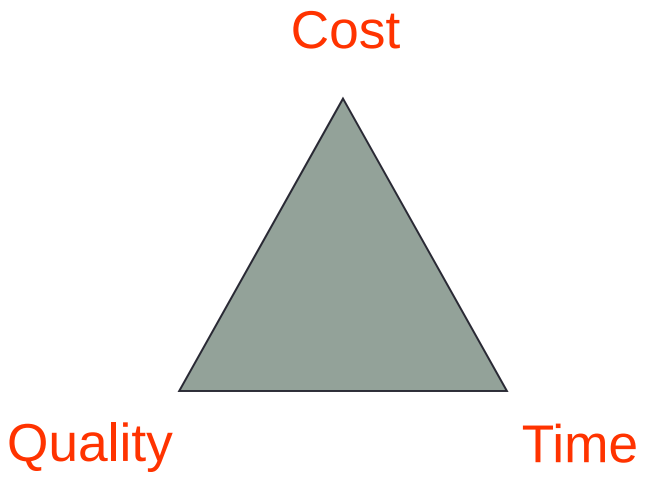
- Projects
  - What
  - Structure
- Teams
  - What
  - Structure
- Communications
  - Who, When, How

Project Definition

- A Project definition:
  - A Project is a set of actions involving a variety of human and structural resources with a specified set of goals to be achieved in a specified time period for a specified cost
- Three antagonistic attributes:
  - Quality
  - Time
  - Cost
- Let’s look at these antagonistic attributes in a software development project

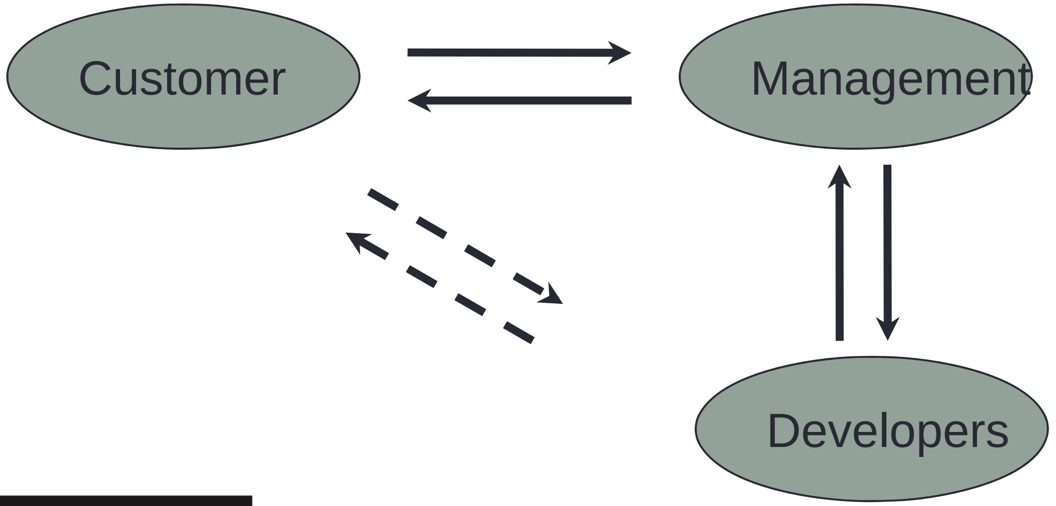
Project Definition *(cont)*

- Balance is the key
  - If you fix the Cost, reducing Time affects the Quality
  - If you fix the Quality, reducing Time increases the Cost
  - If you fix the Time, reducing Cost hits the Quality



Information Flow in a Project

- Three categories of involved parties
  - Customer
  - Manager
  - Developer
- This is the usual route of communication



Project Joys

- Projects have alarming properties:
  - Projects go over budget
  - Projects overrun
  - Projects go belly-up
  - Deliverables are buggy or incomplete
- There are two reasons for it:
  - They are complicated business by nature
  - They are surrounded by myths
- Despite all that the challenge of solving software puzzles generally makes for satisfying work

Software Development Issues

- Software development is inherently complex and difficult
  - You may not believe that but it is true

- Working in groups of X:
  - Write down some issues which could affect software development projects
  - Now try and categorise them in the following:
    - Technical
    - Customer
    - Developer
    - Management

Technical Issues

- There are problems regarding the tools we use to develop software
  - Prevalent languages are not entirely suitable for the tasks at hand
  - Support tools are either expensive or of limited use (or both)
    - Low level tools for high complexity tasks
    - Higher level tools are still being developed and have problems of their own
  - Software projects grow larger and larger
    - A project can be 50000 lines of code and still be considered small

Customer Issues

- Mainly lack of understanding
- The requirements gap problem
  - Customers are usually not experts in software development
  - Developers are usually not experts in the field that the project addresses
  - Somehow knowledge of the software’s requirements must get from customer to developer
  - Targets can (and do) change

Customer Issues *(cont)*

- The requirements gap makes software development hard enough
- Customer-related commercial pressures complicate things further
  - How can you choose a deadline when you don't know what is required of the software
  - Same goes for budget
  - This is often not understood by the customer

Developer Issues

- What is software development?
- We become programmers first, developers second
  - Our first programs are almost certainly written without knowledge of software development
- Many people have no formal training before becoming professional software developers
- Training is not geared towards development techniques

Developer Issues *(cont)*

- The training shortfall
  - Areas that training addresses strongly
    - Programming
  - Areas that training addresses to a certain extent
    - Design techniques
    - Documentation
    - Testing
  - Areas that are rarely addressed by training
    - Software development scheduling
    - Customer interaction
    - Software development administration
    - Etc...

Management Issues

- Resource Management
  - Having the right staff available at the right time
- Mythical Project
- Mythical Staff
- ….

The Great Development Myths

- A collection of misconceptions and assumptions which occur all too often in software development
- At the heart of this is the concept of the Mythical Developer
- This supports the concept of the Mythical Project

The Mythical Developer

- The Mythical Developer’s general attributes
  - Perfect memory
  - No morale problems
  - Fully multitasking - insignificant overheads
- The Mythical Developer’s software design attributes
  - Doesn’t need much of a design as they have done this sort of thing before
  - Will have the design come to them whilst coding
  - Will write a design which should not need any major revisions after the first version

The Mythical Developer *(cont)*

- Scheduling attributes
  - Great at estimating length of tasks
  - Will always finish a task on time
  - Can make up time later on
  - Will get twice as much done by working twice the hours
- Tracking and Resource Management attributes
  - Knows exactly where they are in a project
  - Never overwrites project files accidentally
  - Knows exactly what changes they have made to the code
  - Can turn a prototype code version into a good finished product
  - Is slowed down by all this reporting and paper work
  - Should be left to do *their own* thing

The Real Developer

- Of course, we know that we were not talking about a real person
- The profile of a Real Developer looks more like this...

The Real Developer *(cont)*

- The Real Developer's attributes
  - Has imperfect memory and variable morale
  - Is affected by multitasking and overworking
  - Will always need a well thought design
    - …and will always need to update it as work progresses
  - Will always fail in the initial estimation
  - May overwrite files
  - Will be more effective in the long run if project tracking and documentation is applied
  - Will need to recode less if coordinated with management and team
- They are only human

Mythical vs. Real

- Mythical Developers are (unsurprisingly) hard to find
- Yet all too often software projects are run as if staffed entirely by Mythical Developers
- Why is this?
  - Management misconceptions
  - Developer misconceptions

Management Misconceptions

- Management have to make sure that the clients are happy with the progress of the project
- Cutting out work that does not produce code looks good at the start of the project
- Developers are expected to “pull through” making use of their mythical properties
- Management struggle to appreciate the ability of their developers

Developer Misconceptions

- When optimistic, real developers can show signs of being mythical developers
  - Start of a project is the usual time for this
- Sometimes can show signs of being very human
  - Middle/end of a project is the usual time for this
- A challenging task can pull them off track

Addressing the Issues

- How can we solve the difficulties?
  - We can do something about technical issues
    - Tools technology advances
    - Management realise the need for appropriate tools for the job
  - We can do something about some developer issues
    - More project-oriented training for people entering development

Addressing the Issues *(cont)*

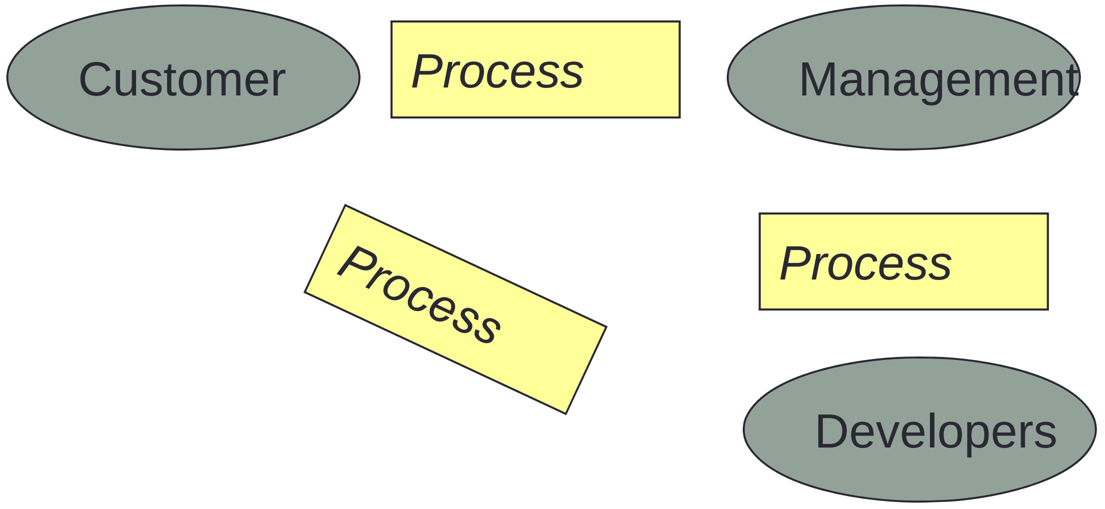
- Not all difficulties can be solved
  - We can do little about customer issues
    - The process of getting our requirements from the customer is always going to be involved
    - The customer will always want to know when the software will be finished-*nothing else*
  - Software development is by nature complex
    - So perhaps we should organise the effort better
- These issues will not go away, but there are techniques with which we can reduce the risks they represent

What Is Process?

- A description for “Process”:
  - “ Process is a wide category of activities whose application to a software project can substantially increase the chance of project success”
  - Process should not be SCARY!!!! or BUREAUCRATIC!!!!
- How can increase the chance of success?
  - By encouraging project partners (participants) to conduct meaningful communication
  - By allowing project partners (participants) to understand the state of the project at any given time
  - By providing project partners (participants) with methods to ensure that the project does not deviate uncontrollably from its ideal state
  - By dismissing myths

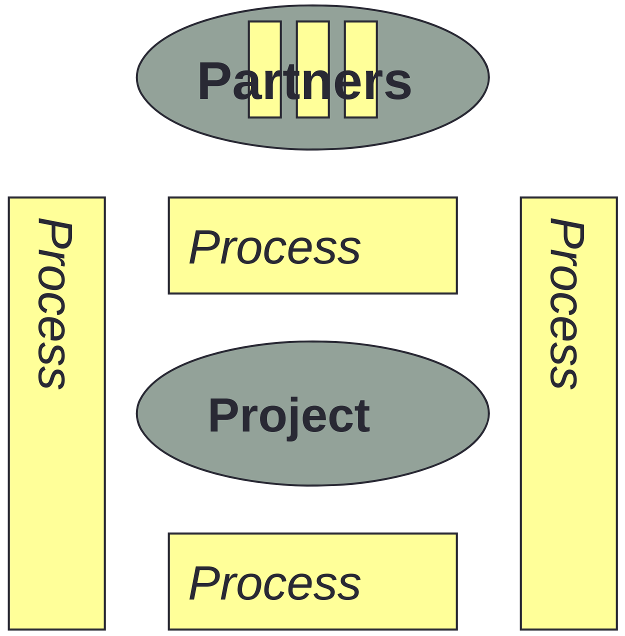
Where Is Process?

- Process sits between communicating parties and standardises their interaction



Where Is Process? *(cont)*

- Process also sits between project partners and the project itself guiding its development



What is in the Process?

- In groups of X, try and come up with what Process may cover in a project in terms of activity or task.

What Makes Up Process?

- Process is a collective term covering the following tasks:
  - Scheduling
  - Application design
  - Change control
  - Revision planning
  - Project group communication
  - Risk management
  - Role assignment
  - and more…

Process Misconceptions

- Be aware of what Process is not…
  - Process is not a radical new method for developing
  - Process is not a rigid and inflexible system
  - Process is not design
  - SCARY!!!!
  - BUREAUCRATIC!!!!

Process Is Not a Radical Method

- Process is not a new paradigm for software development
  - It is merely the implementation of sensible practices to keep a project on track and manage the risks of software development

- All projects have some sort of Process
  - If a project is not considered to have Process, it actually just has a very poor Process

Not a Rigid, Inflexible Process

- When talking about Process, many developers think of RIPs (Rigid, Inefficient Processes)
  - Process can be made rigid and inflexible, but it is not necessarily so

Process Is Not a Silver Bullet

- Process will not automatically save your project
  - Bad Process will probably not help your project
- Process must be included and practised in an intelligent manner to be of use
  - From the start of the project as well
- But done in this way, it will almost certainly help

Process Benefits

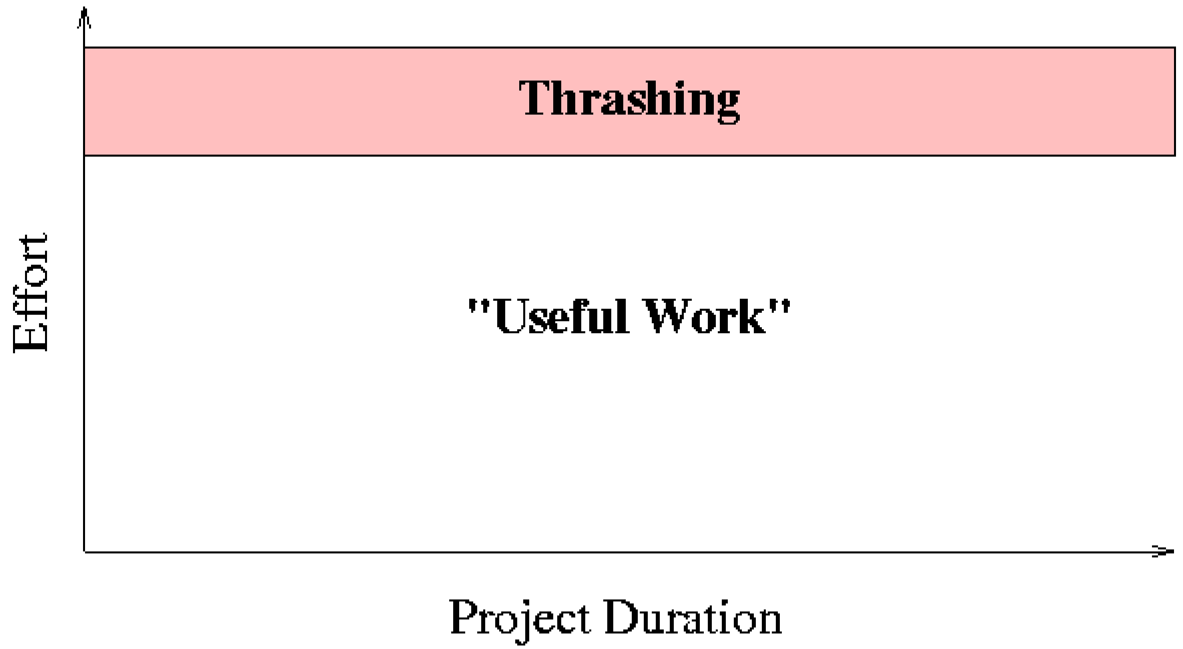
- Process is not a bad thing
  - Because it keeps the project within its targets
    - Makes sure there are some, for a start!
  - Because it keeps management and developers in contact
  - Because it keeps customers happy
    - Gives them some insight
    - Helps to provide the expected results
- **So why do we keep avoiding it?**

The “No-Process” Project

- Success is anticipated from mythical properties
- Important concepts
  - Let the developers do their thing
  - Process = non-coding effort = thrashing
  - Management wants code as proof of progress
  - Management wants to know percentages done

The “No-Process” Project

- The mythical “No-Process” Project in action

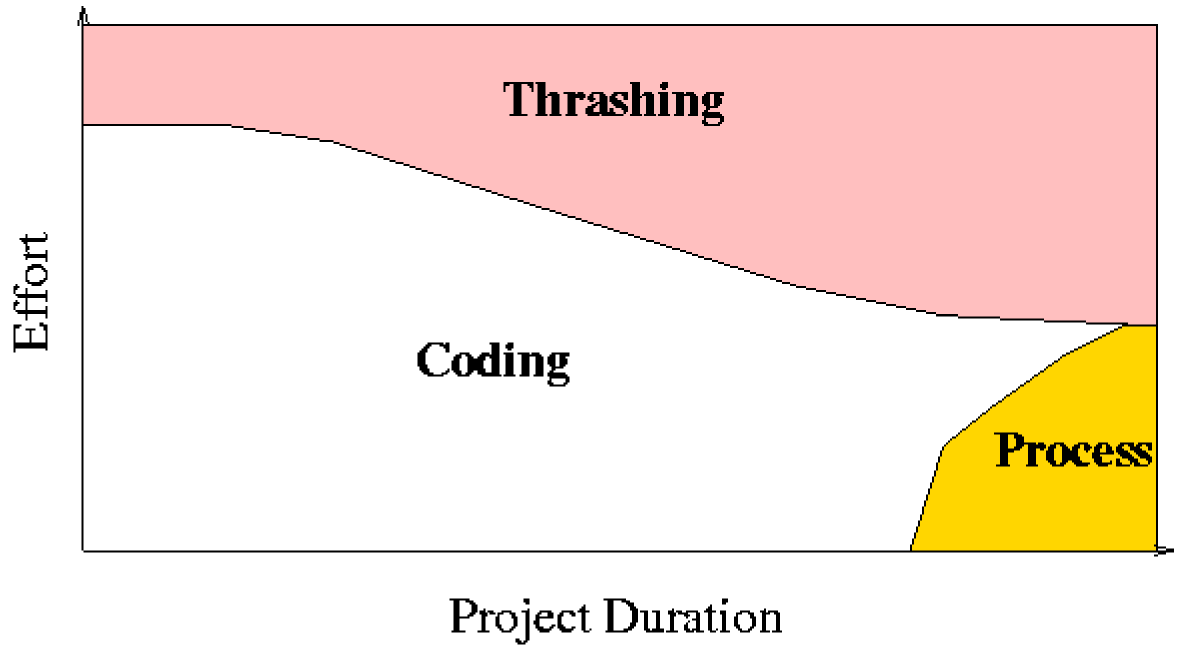


The "No-Process" Project *(cont)*

- The real "No-Process" Project
- Important concepts
  - Development starts quickly
  - As its complexity increases, so does the thrashing
  - Partly finished code is considered finished to appease management
  - Percentages rarely mean anything
  - Process brought in, but by now is too late to help
  - If the project is lucky, it will finish before thrashing takes over and no useful work is done any more
  - If unlucky, abandon project

The “No-Process” Project *(cont)*

- The real “No-Process” Project in action



The “Process-Based” Project

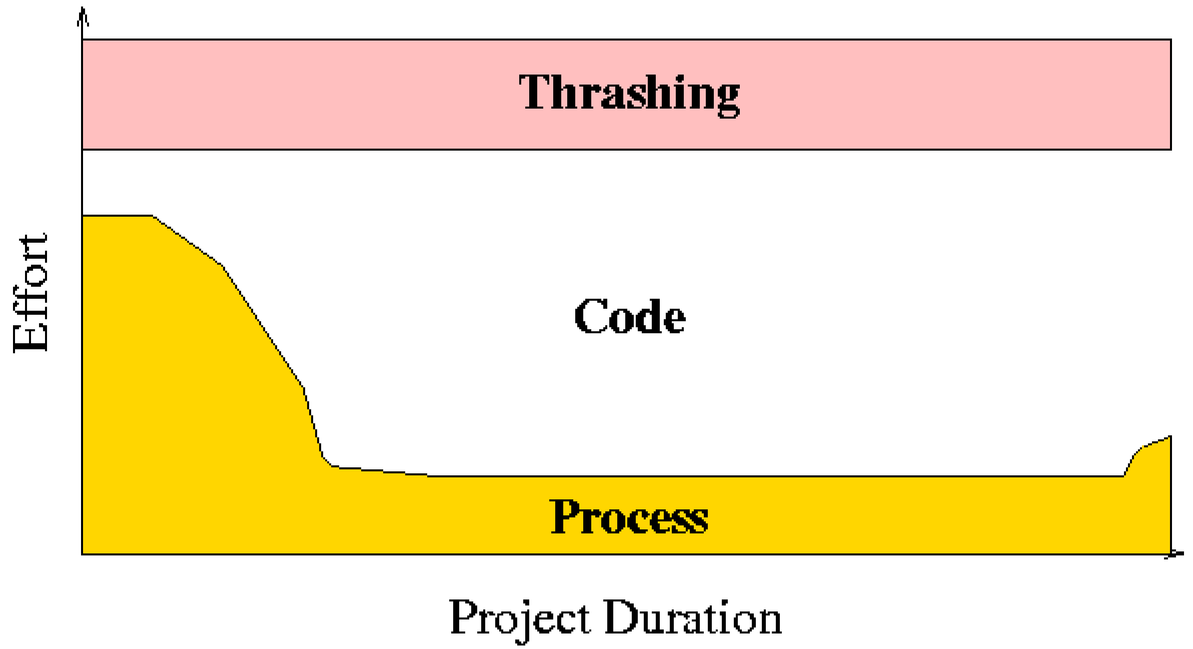
- The mythical “Process-Based” Project
- Important concepts
  - Spends too much time mired in bureaucracy
  - Not enough time utilised for useful work
  - Process is the anathema of the Mythical Developer
    - “… and we have got so many of them around”
- Please note: the myths are pessimistic

The “Process-Based” Project *(cont)*

- The real “Process-Based” Project
- Important concepts
  - Process is integral to the project
  - Throughout the project, all work done is managed through Process
  - Helps keep thrashing to a tolerable level
  - Less chance of project failure

The “Process-Based” Project *(cont)*

- The real “Process-Based” Project in action



Projects Summarised

- The mythical project is everyone’s dream
  - Produces timely projects, at a low cost and with great features
- Only one problem
  - Mythical developers are thin on the ground
  - Most projects must make do with real developers
  - A mythical project with real developers ends up looking like a real project

Projects Summarised *(cont)*

- The Real Project summarised
  - The real project can be a nightmare
  - Projects can end up running over schedule, over budget
  - Projects can end up being way below desirable quality
  - Projects can even fail to finish
- How can we turn this around?
  - “Process” is the way to concert, monitor and standardise software development in your project
- Using Process does not guarantee success
  - It does make it far more likely though

Project Teams

- Task
    - Form teams of X – with people you either don’t work with, haven’t worked with or don’t know
    - Make sure you know who each other is
    - You will be working together for the rest of the course
- Task
    - What are the roles on a project team in software development?
    - Quick list

Teams & Role Assignment

- Traditional software development has a number of established roles
  - Manager
  - Developer
  - Tester
  - End User Liaison
- These may or may not be precisely defined

Role Assignment *(cont)*

- Other roles may be completely overlooked
  - Risk Officer
  - Change Officer
  - Quality Assurer
- These roles are created by the use of good Process within a project
  - But we still need to make sure these tasks are carried out properly.

Role Assignment *(cont)*

- Why are such roles usually overlooked
  - They are simply not thought of (poor Process)
  - The project is too small to justify team members assigned to perform them
  - They are considered peripheral to the development of the application
  - They are deemed to be “obvious” and are expected to just “happen” during the course of development
- But failure to manage risks and quality are two of the biggest reasons why software projects fail

Role Assignment Execution

- Each role has a definite and established set of duties
- The act of assigning roles is like giving team members different hats to wear
    - Team members swap hats when necessary and perform the tasks associated with it
    - Assigning project time to carrying out the role makes team members change their hats to carry out the role’s duties

Roles

- A list of established roles
  - Project manager
  - Product manager
  - Architect
  - User interface designer
  - End user liaison
  - Developer
  - Quality assurance tester
  - Tool smith
  - Build coordinator
  - Risk officer
  - End user documentation specialist
  - Test client

Role Allocation

- How you assign roles depends on project size
  - Large projects
    - Each role may be carried out by a team of people
    - Many team members per role
  - Small projects
    - Each team member gets several roles to fulfil
    - Many roles per team member

Role Assignment Example

- A typical small project
- Recognised roles
  - Manager
  - Developer
  - Technical reviewer (maybe)
- These roles should have an established set of duties
- All other roles are implicit
  - Some carried out in an ad hoc fashion
  - Some not carried out at all

The Manager’s Roles

- Most of a manager’s roles will have an executive nature
  - Project Manager
  - Product Manager
  - End User Liaison
- Often in small projects, managers are also technically minded
  - Architect

The Developer’s Roles

- As well as Development, one or more developers will have to take on some of the duties below
  - Architect
  - User Interface Designer
  - Quality Assurance tester
  - Tool smith
  - Build coordinator
  - End user documenter

The Technical Reviewer’s Roles

- The technical reviewer can act in many of the “devil’s advocate” roles
  - Quality assurer
  - Change officer
  - Risk officer
  - Test client
- Being removed from the daily Process of the project enables the technical reviewer to look at the project from a different angle

RACI Matrix

- Responsible, Accountable, Informed, Consulted
- Helps to confirm/clarify roles and responsibilities
- Helps to manage all the roles, responsibilities and tasks across departments, people and processes, …
- Particularly when a number of organisations are involved.
- Also known as RAM – Responsibility Assignment Matrix and other names, eg ARCI, Linear Responsibility Chart.
- A number of variations too
  - RASCI where S = Support
  - CAIRO = where O = Omitted or Out of the lop
  - DACI – Driver, Approver, Contributors, Informed
  - ….

RACI Matrix (cont).

- Responsible – the person who actually does the work
- Accountable – the person answerable to the completion of the work, The person who delegates the task to the responsible person.
- Consulted – the people asked their opinion
- Informed – the people kept up to date on progress, usually only one way communication

RACI Matrix Example

- Responsible, Accountable, Informed, Consulted

Tasks or Roles	Manager	Developer	Technical Reviewer
Project Management	A,R	C	I
Product Management	A,R	C	I
Architect	A,C	R	C,I
User Interface Design	A	R	C,I
Build Coordination	A	R	I
Risk Officer	A,C	C,I	R
Change Officer	A,C	C,I	R
…….			

A Practice Start

- Over the course you will be working on a prototype piece of software, you have to work as a team, analyse it, do some design and reworking, usability work, planning and estimation.

- Now – in your teams, decide on the main roles you think you would need (note in these exercises everyone should be able to contribute at all points)

- Draft a team structure – who has what hats?

Some Questions

- Why Communicate?
- Who Communicates?
- How and When?

Bad Communication

- What is it?
- Effects of Bad Communication
- And what happens is
  - Deadlines are missed
  - Time is wasted
  - Requirements are not satisfied
  - Deliverables are buggy or incomplete
  - Team gets frustrated
  - Users get disillusioned
  - Funders get nervous
- And our project
  - Goes over budget
  - Over-runs
  - Goes belly-up and down the tubes

An incomplete list of ways

- Meetings – Formal and informal
- Reports
- Emails
- Singular Phone call
- Conference call
- Voice mail
- Web Communications

Mode of Communication	Pros	Cons
Formal Meeting	Focused, interactive, group	Scheduling delays with stakeholders, discoverable information
Informal Meeting	Quick, interactive	Unpredictable, individual
Report	Detailed message, group	No immediate feedback
Email	Quick, focused, group	Unfocused response, discoverable information
Phone Call	Focused, interactive	Unpredictable connection, individual
Conference Call	Consistent message, group	Uncertain coverage
Voice Mail	Quick, focused	Limited content, individual
WebX	Focused, interactive, distributed group	Technology

Recording What Communication Works

- May seem a boring task
- Record what works
  - What has most participation
  - What moves the work forward
  - How the actions and knowledge are passed on
- Easy to do? – depends on what is being recorded

Recording Meetings

- Decide not to record the meeting at all
- Take written notes and minutes
- Record key points visibly, such as on newsprint or a chalkboard
- Tape--usually by audiotape, but occasionally by videotape as well

- Quick Task: In your teams, have a short meeting to discuss a topic – each of you try and record what was said in the meeting – then compare

Public vs Private

- Does everything have to be public?
- What is the difference to the project?
- Legal Reasons?
- Social Reasons?
- Effects on developers?

Tools

- Can technology help?
- Telephones
- Web Conferencing
- Blogs
- IM
- Microblogs
- Wikis
- Email
- ……

Communications

- Projects involve communication
  - Customers
  - Managers
  - Developers
  - Users
- Neglecting communication can doom projects
- Many and varied
  - Co-operative activities
  - Information communicated
  - Tools available
- Tools support but cannot replace process