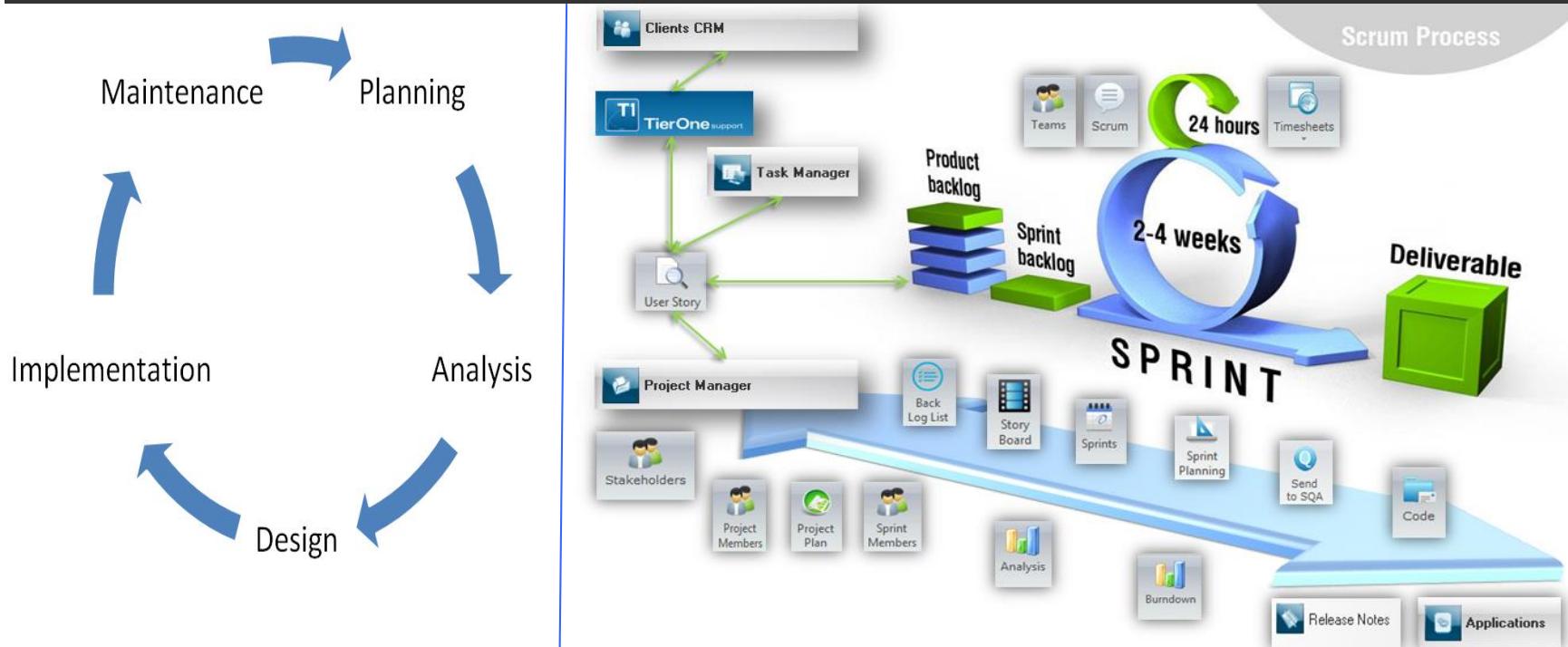




FIT2001 – Systems Development

Seminar 1: The nature of Systems Development

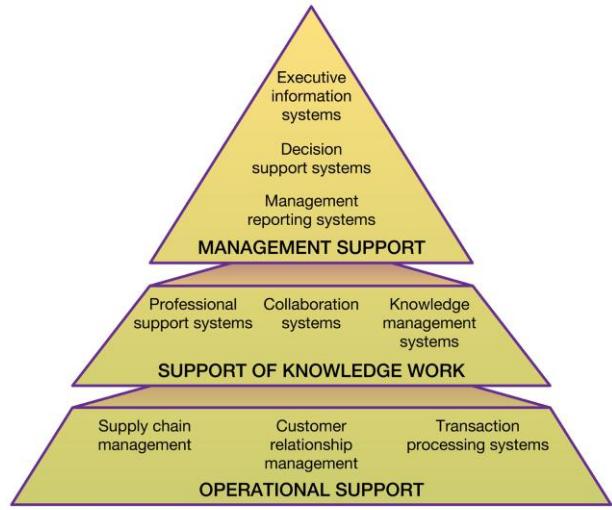
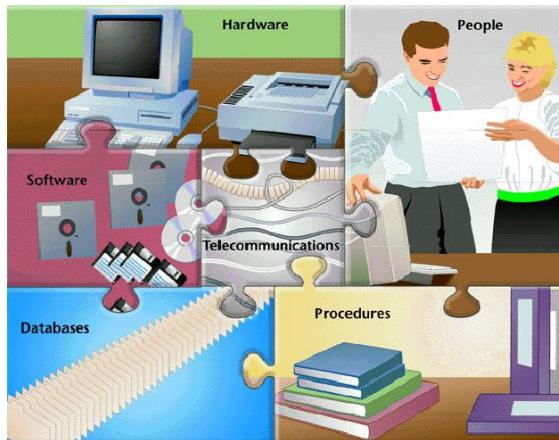


At the end of this seminar you will:

- Understand what information systems are
- Understand the key phases of the Systems Development Life Cycle, and effort distribution
- Have knowledge of the key roles and skills required of system developers

What are information systems?

- An integrated set of components for collecting, storing, and processing data and for delivering information
- Almost every organization relies on information systems to carry out and manage their operations, interact with their customers and suppliers, and compete in the marketplace.



- The main components of an information system are - people, procedures, hardware and software, databases, data warehouses, telecommunications



Think about an example
of an Information
System

Why is it an Information
System?

Is it a good Information
System?
Why? Why not?

Assessing an information systems

- Accuracy and reliability
- Accessibility
- Ease of use
- Flexibility
- Security
- Usefulness
- Timeliness
- Completeness

Customer Relationship Management (CRM) system

How CRM fits in with your Online Strategy



Copyright Sankhya Consultants 2013



The Design Phase involves:

So how do you develop these Information Systems?

- Very simply ...

using the process of SYSTEMS DEVELOPMENT

the process of creating and maintaining information systems

What do you think the steps would be?



Spend a minute thinking about the steps you
would need to follow to
BUILD A HOUSE

1. Initiation: Feasibility - Can it be done?



Can you afford to build what you want?

Are there any time constraints? Is the expertise available?

Need to do a 'quick and dirty' analysis of the requirements



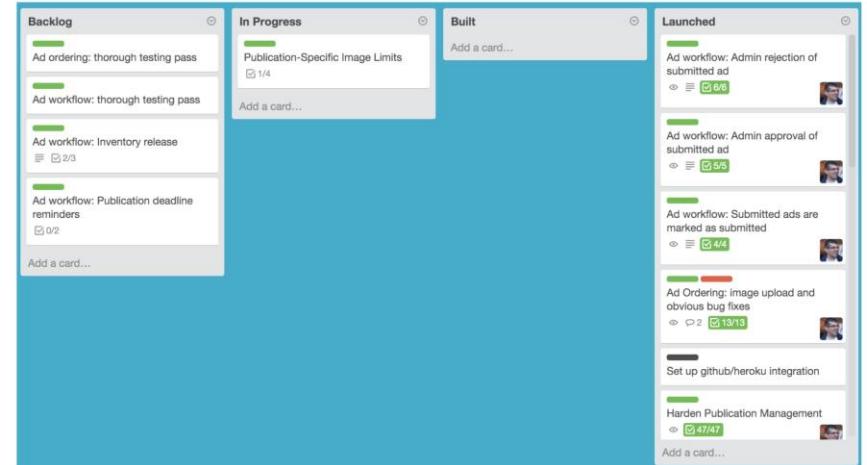
- Are you willing to compromise?
- What do you really want - mandatory vs. optional
- If there is a good chance that you can get what you want ON TIME and WITHIN BUDGET then you can go ahead

Planning your development project

- You now have the SCOPE of your project Will it stay fixed ?
- How do you manage **scope creep** ?



Just popping out to buy a bottle of milk



- Project Planning – must be done throughout the project (FIT2002)

2. Analysis – What do you want?

- Does the client know what they want?
 - determines how you go about the process
 - it is vital that you demonstrate to the client that you understand their requirements

Client requirements: 4 bedroom house with 2 toilets and a garage

Which house does the client want?



These houses together with a 1000+ other houses would meet the brief?

Build or Buy ?

- Do you have to build or can you buy a house that is exactly what you want ... you may just have to make a few modifications

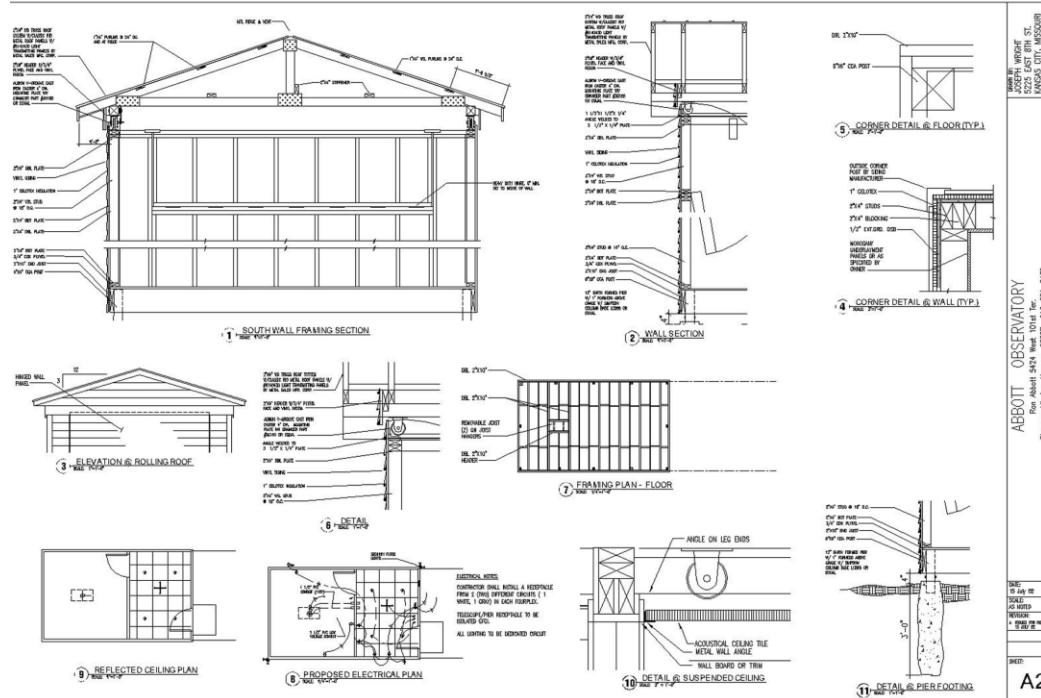


Analysis – Build or Buy pros and cons

BUILD?	BUY?
Pros	Pros
Business processes unique / complex	May be cheaper
Security / Competition	Know what you are getting
Cons	May improve business processes
Build in current problems	Implement quickly
May be expensive to build / maintain	Regularly upgraded, documents provided
Time consuming	Cons
Is expertise available in-house	Customisation / Integration may be expensive and time consuming
Don't know what you will get finally	On-going maintenance can be costly
	Vendor may go out of business

3. Design - How are you going to do it?

- Detailed plans for the build – shows integration of various components
 - Plans for carpenters, electricians, plumbers, plasterers, etc.



4. Implement

Build/Develop – Construct, Test that it is working

- Good analysis and design is essential for a good build
 - together with building expertise and thorough testing



.... however, just building expertise is NOT enough

Deploy - Is it ready? Can I move in now?

- Does it meet:
 - Government requirements
 - Sustainability requirements
 - CLIENT requirements



Are your clients happy ?



Very costly exercise if the requirements are not met



5. Support – Maintain it, Extend it

- Can it be easily maintained and fixed?
- Can it be added to it easily?

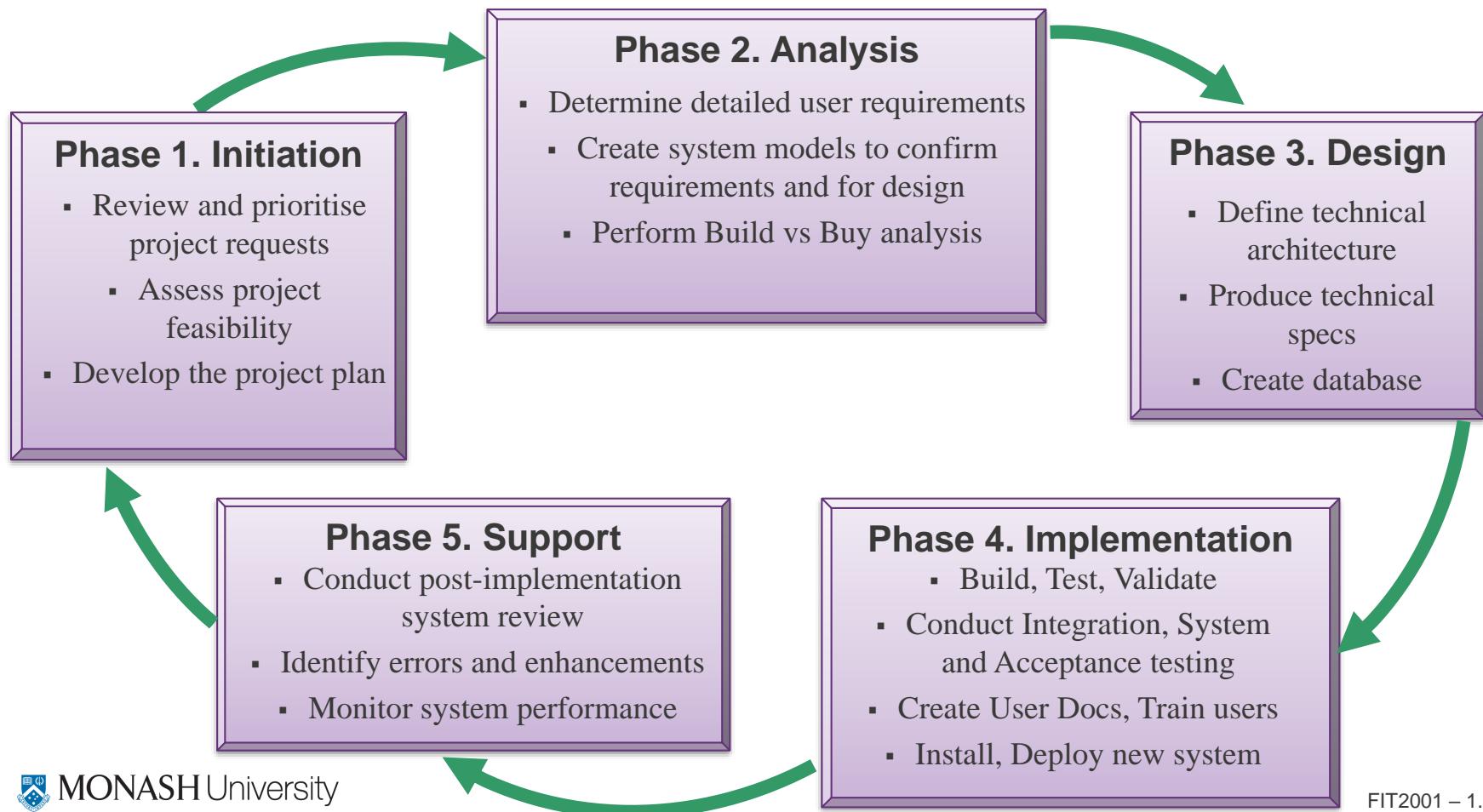


.... and finally it gets to the point where it is time to rebuild – new products, techniques, new rules, new expectations



Systems Development Life Cycle

SDLC - Phases



SDLC Case Study

ON THE SPOT COURIER SERVICES

Bill Wiley – start up, same day courier service

Initially just received delivery requests via texts on his mobile, but then customers started asking if he had a website where they could place orders

As the business grew, Bill hired another person to help with the deliveries. He could no longer use his van as the ‘warehouse’, he now needed a central warehouse where he could organise and distribute packages for delivery, and if it grew further someone at the warehouse to co-ordinate the arrival and distribution of the packages

What do we need to do to develop a system for Bill Wiley?

Effort distribution

- Distribution of efforts varies based on
 - Type of project
 - Size of project
 - together with building expertise and thorough testing
- From research conducted in 2015 the approximate median effort for the different phases:

Phase	Effort
Planning and Analysis	15%
Design	15%
Coding	35%
Testing & Implementation	25%

Can anyone do Systems development?

- Expertise and specialist knowledge required

Some of the wide range range of roles available:

Managerial – Project Manager, Team Leader

Functional – Systems Analyst, Business Analyst, Tester, Documenter, User Experience (UX) Designer

Technical – Systems Designer, Database Administrator, Solutions Architect, Developer /Programmer, Tester, User Interface (UI) Developer, Security

Other roles in Quality Assurance (QA), Documentation, Training and Deployment



System developers – Critical skills for every role

- **Understanding business** - awareness and sensitivity to the business processes and needs that require technology in the first place
- **Broad and up-to-date understanding of technology** – can be invaluable in creating the ‘best’ solutions for the organisation
- **Multiple Perspectives** - The ability to understand that there are multiple perspectives to solving a problems is required to find the best solution
- **People/Soft Skills** - the ability to interact with other people and to be a part of a team
- **Continuous Learning** – essential in a high-change industry, like IT

Job advertisement - 1

Some of your key accountabilities will also include:

Stakeholder Management and Communication: Identify stakeholders and see to the maintenance of collaborative working relations

Requirements Analysis and Lifecycle Management: Ensure the project meets the needs of the customer and business and is aligned to the overall strategy

Best practice and Delivery Methodology compliance: Understand and adhere to enterprise methodologies and processes such as the Westpac's Project Execution Framework (PEFm), Requirements Development Framework (RDF), Agile Execution Framework (AEF).

Risk Management: Ensure project related risk (both project and operations) is within desired levels and that risks are managed according to agreed Westpac's risk frameworks (Refer Operational Risk in Projects (ORiP))

Reporting: Regular status reporting as it relates to the project

Community of Practice - Business Analysis: Contribute to the Westpac Group's Community of Practice e.g. meeting attendance, knowledge up skilling/sharing

Job advertisement - 2

Ideally, you'll have previous experience in business process improvement/re-engineering, data analysis, system functional and non-function analysis. You'll be a true people person with the ability to positively influence and negotiate with various stakeholders. Getting the job done is what you do best, and you'll be a key driver in ensuring that obligations are met by holding yourself and others accountable for behaviours and outcomes.

In addition, you will have:

A strong understanding of technology and business systems strategically and operationally

Relevant business and/or technology tertiary qualifications

The ability to analyse situations or issues, by considering all options prior to recommending and implementing a solution

Working with project managers and other business analysts, you will support the team across a broad range of daily operational activities.

Job advertisement - 3

Document and analyse ‘as-is’ processes/procedures and collaborate with key stakeholders to develop appropriate ‘to-be’ processes/procedures

Assist technical teams (both internal and vendor based) to translate business requirements into functional and system requirements using a range of analysis models and tools such as workshops, workflow mapping, data modelling, document analysis and use cases.

Facilitate discussions and/or regular workshops to gather and confirm requirements as well as showcase solutions and potential options to ensure a common understanding.

Capture system requirements through analysis of business requirements, user stories and acceptance criteria.

Create user acceptance testing (UAT) documentation, and project documentation

Opportunity to act as Scrum master to ensure the solution team is on track for the planned sprints, by producing various reports for the PM.

Participate and contribute to continuous improvement ideas.

Essential reading:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016)
Systems Analysis and Design in a Changing World, 7th
Edition, Cengage Learning, Chapter 1
- See additional resources on Moodle

Workshop Preparation

**... forming a team – prepare your pitch –
upto 30 seconds**

**Thanks for watching
See you next week**



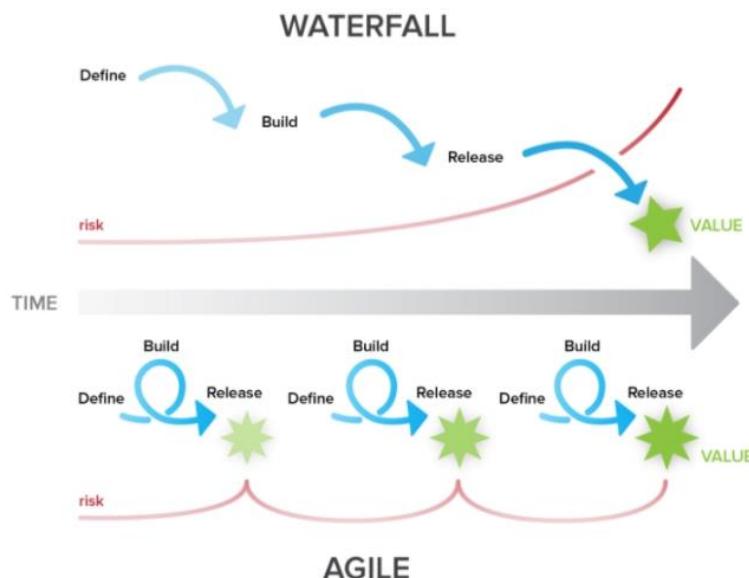
FIT2001 – Systems Development

Seminar 2: System Development Approaches

Agile Software Development

Stakeholder management

Chris Gonsalvez



Our road map:

- What are Information Systems?
- How do we develop them?
Systems Development (SDLC) – key phases
- Some System Development roles and skills

- Traditional vs. Agile approaches to developing systems
- A Focus on Agile development
- Stakeholder management

At the end of this seminar you will:

- Be aware of the different approaches to developing information systems
- Understand Agile software development - the Agile manifesto, the 12 Agile principles and key concepts
- Be able to identify and understand different kinds of stakeholders and their contributions to requirements definition

Information systems

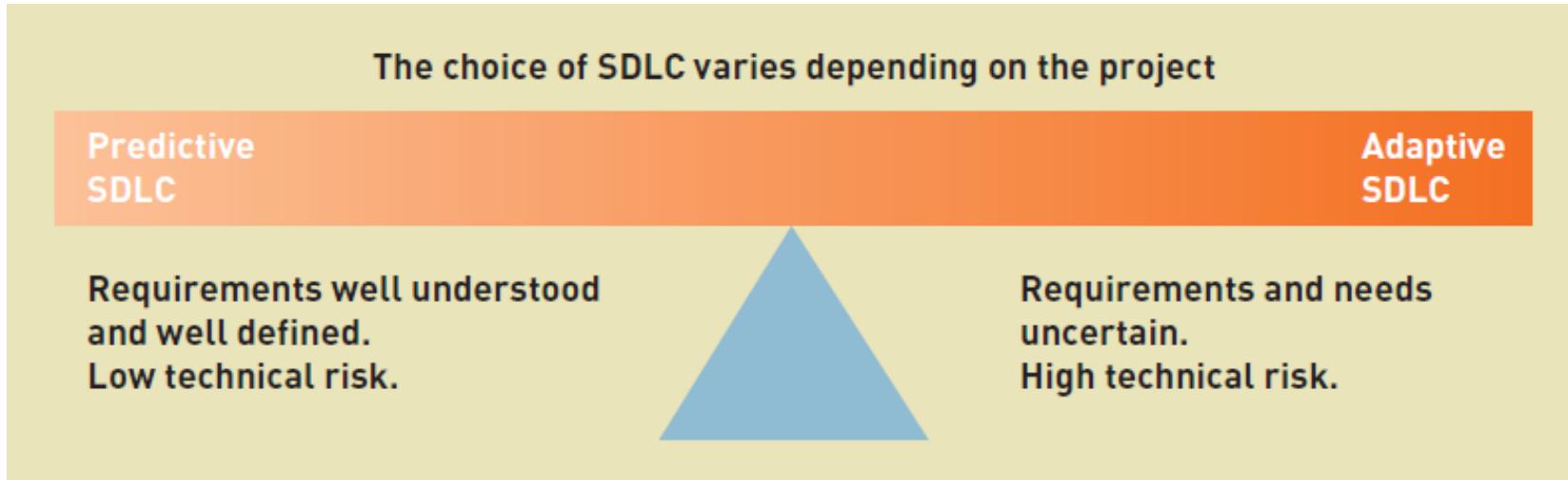
“A system which assembles, stores, processes and delivers information relevant to an organisation (or to society) in such a way that the information is accessible and useful to those who wish to use it, including managers, staff, clients and citizens.”

Buckingham et al. (1987)

in Avison & Fitzgerald 2006, p. 23

We now have to consider how to develop them

Development approaches



Systems Analysis and Design in a Changing World, 6th Edition – Figure 8.1, p228

Most projects fall somewhere on this continuum

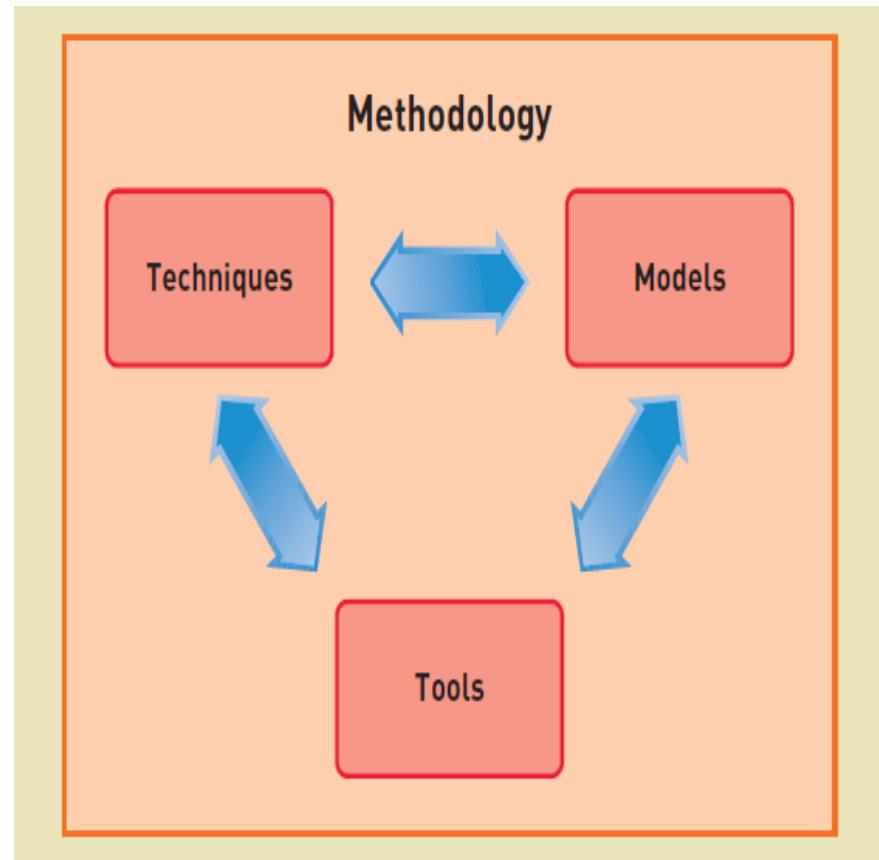
Frameworks / Methodologies definitions

Frameworks

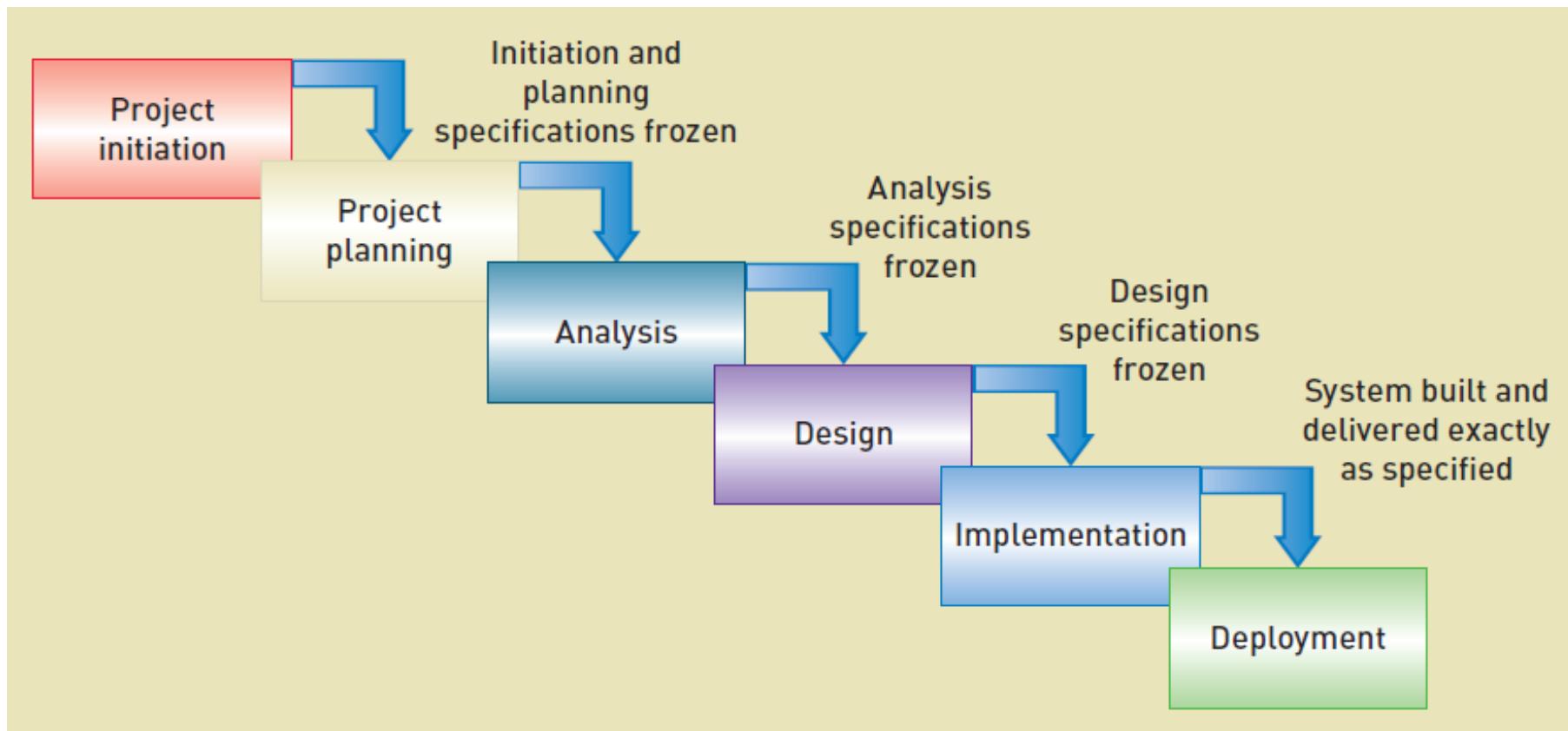
- Provide structure and direction on a preferred way to do something – guidance while being flexible

Methodologies

- A set of principles, tools and practices – conventions that an organisation / team agree to follow to achieve a particular goal.



Traditional Predictive thinking Structured Waterfall framework



Systems Analysis and Design in a Changing World, 6th Edition - Figure 8-3, p229

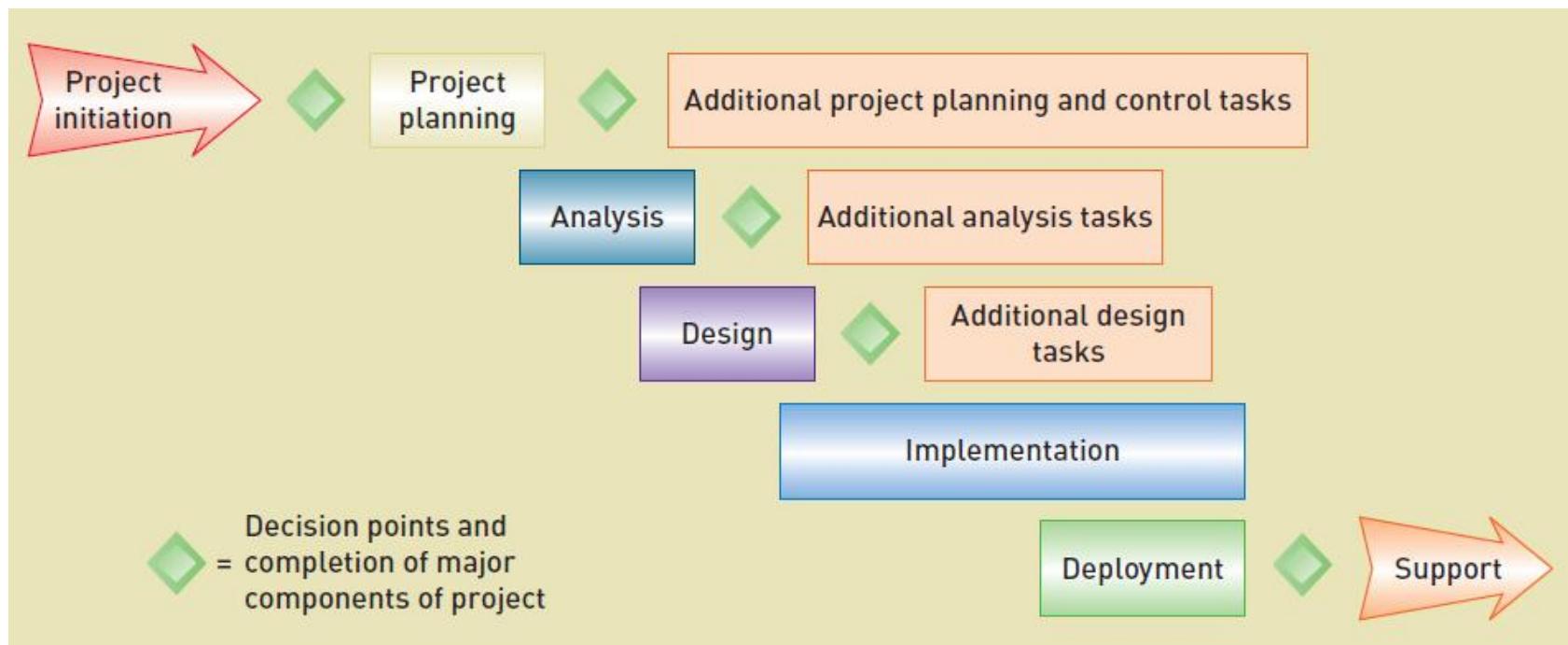
Waterfall framework

- Dominant since development methodology from early 70s to 90s
- Sequential stages – no overlap or iteration
- Strong emphasis on planning and specifications development
- Works well for clearly defined projects - requires thorough planning, extensive project documentation and tight control over the development process.
- Issues - tends to be slow, costly and inflexible.
 - Inability to adjust the product to the evolving market requirements often results in a huge waste of resources and the eventual project failure

Rarely developed this way anymore

Development often moved towards an Adaptive framework

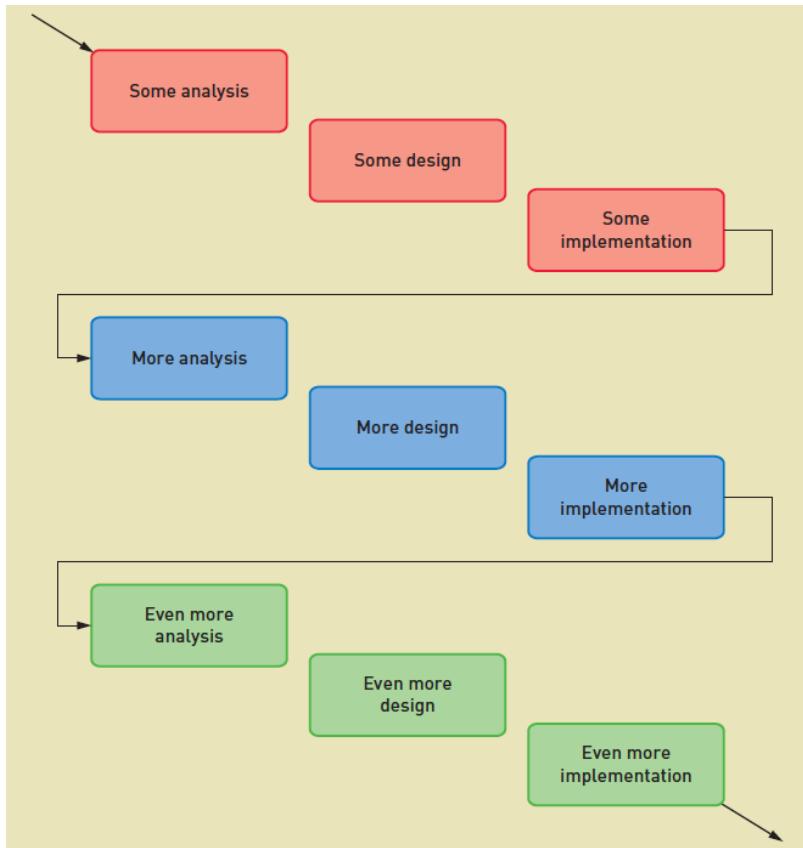
- More flexibility, but still assumes predictive planning and sequential phases



Systems Analysis and Design in a Changing World, 6th Edition – Figure 8.4, p230

Adaptive thinking

Agile Iterative frameworks



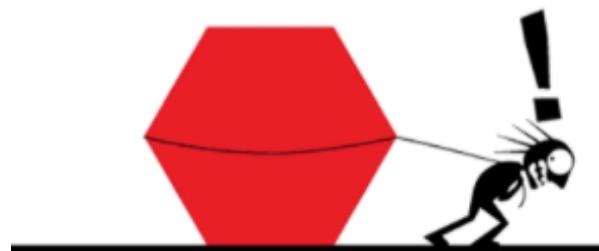
Iterative development

An approach to system development in which the system is “grown” piece by piece through multiple iterations

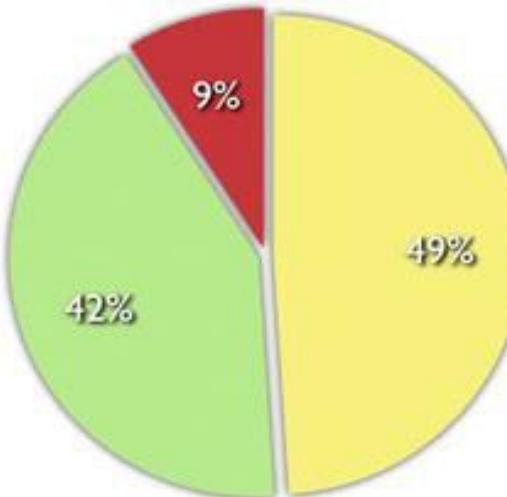
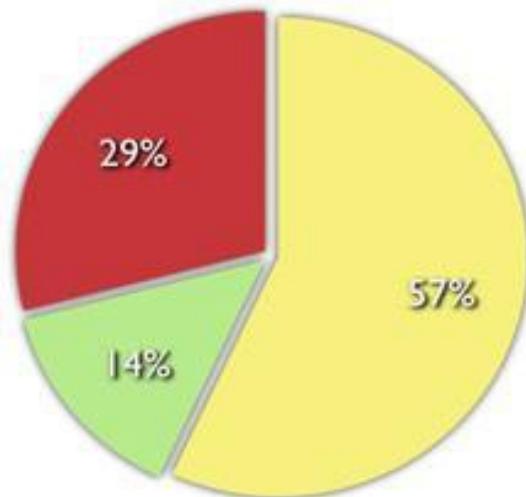
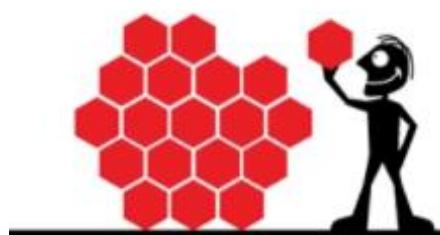
Systems Analysis and Design in a Changing World, 6th Edition – Figure 8.6, p231

Agile vs. Waterfall

THE WATERFALL PROCESS



THE AGILE PROCESS



- Successful
- Challenged
- Failed

Source: The CHAOS Manifesto, The Standish Group, 2012.

Results of projects conducted by The Standish Group from 2002 to 2010

Agile vs. Waterfall Metrics

Metric	Waterfall	Agile
Planning scale	Long-term	Short-term
Distance between client and developer	Long	Short
Time from specification to implementation	Long	Short
Time to discover issues	Long	Short
Ability to meet deadlines	Poor	Good
Ability to respond quickly to change	Low	High

AGILE SYSTEMS DEVELOPMENT

Agile History

- Incremental approaches started as far back as the late 1950s – building software for IBM
- Mid 90s – issues with developing software, started mixing old and new ideas, focus on close collaboration with users, frequent delivery of business value – frameworks such as SCRUM, Extreme Programming started to appear
- 2001 – 17 software developers caught up to find commonalities in developing software – they disagreed about a lot, what they agreed upon became The Agile Manifesto - a set of value statements that form the foundation for Agile software development

What is Agile?

- Agile frameworks take an iterative approach to software development - project consists of small iterations
- Each iteration is a miniature project with a well defined scope
- At the end of each sprint, a potentially shippable product increment is delivered.
- Every iteration sees new features added to the product, which results in the gradual project growth.
- With the features being validated early and regularly, the chances not delivering what the clients wants reduces significantly.

Agile Manifesto - Values

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:



12 Agile Principles



Satisfy
The Customer



Welcome Changing
Requirements



Deliver Working
Software Frequently



Collaborate
Daily



Motivated
Individuals



Face-to-face
Conversation



Measure Of Progress
Through Working Product



Promote Sustainable
Development



Continuous Attention To
Technical Excellence



Simplicity
Is Essential



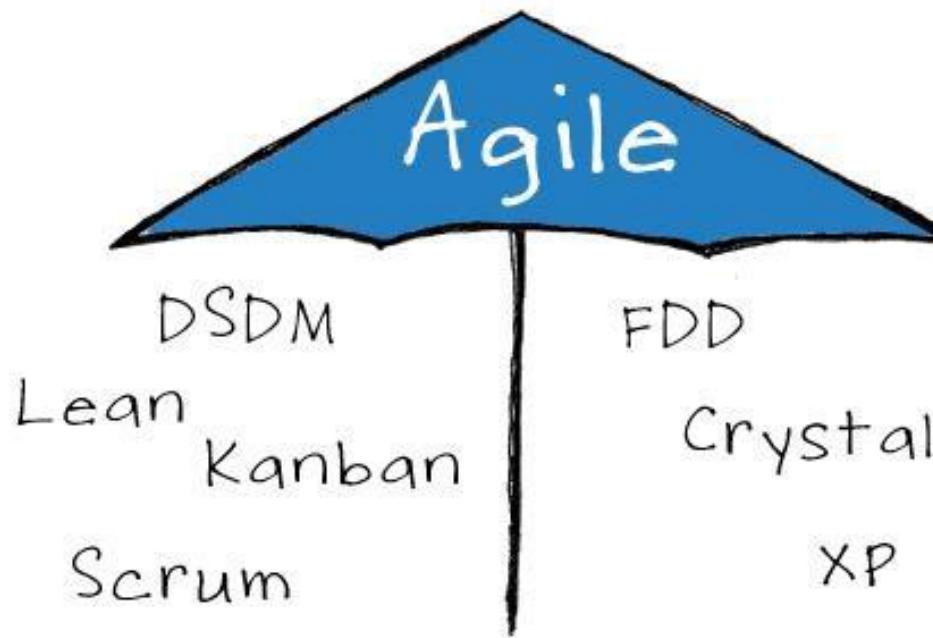
Self-organizing
Teams



Regularity Reflect On
Continuously Improving



Agile frameworks



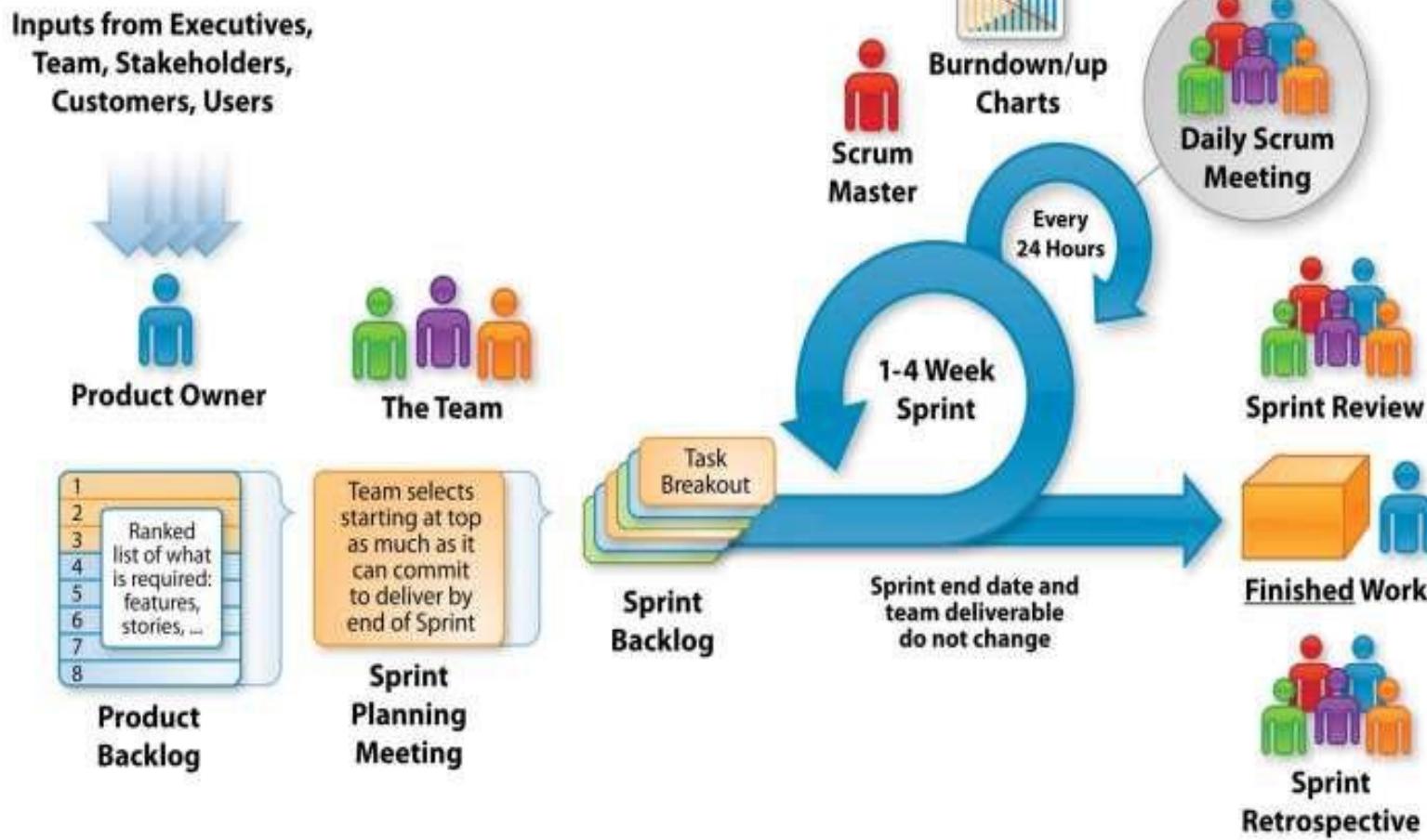
http://en.wikipedia.org/wiki/Agile_software_development#Agile_methods

Agile Framework example - SCRUM

- A framework based on agile principles
- Based on continuous improvement in product and process
- Delivers software (value) frequently
- A scrum project is a series of iterations called Sprints – typically 2-4 weeks long, based on an inspect and adapt cycle
- Produces outputs iteratively and incrementally, thus reducing risk and enhancing visibility

<https://www.agilelearninglabs.com/resources/scrum-introduction/>

SCRUM Framework



SCRUM Roles



Product owner

Client's representative, defines and prioritises product features, accept or reject work items

Scrum Master

Coach for scrum team, applying agile principles, ensures team's productivity, builds a successful team

Development Team

5-9 members in a self-organizing, high performance, cross-functional team (Developer, Tester, BA)



SCRUM Artifacts



Product Backlog



Sprint Backlog

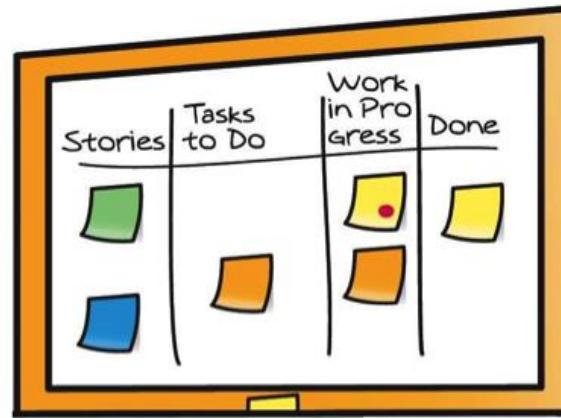


Product Increment

Burndown Chart



Kanban board

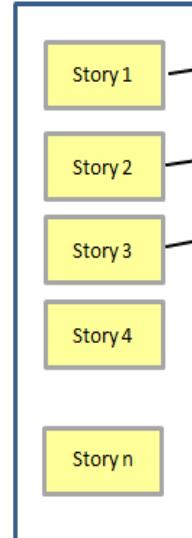


Product Backlog & Sprint Backlog

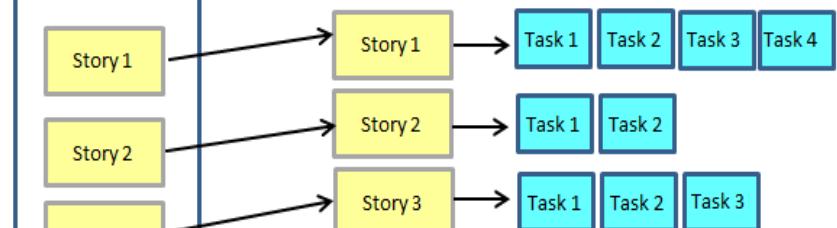
Product Backlog

- The single source of requirements
- Cumulative list of desired deliverable for the project – every feature, enhancement, bug fix, documentation requirement, every bit of work required by the team
- Prioritised to maximise value

Product Backlog



Sprint Backlog

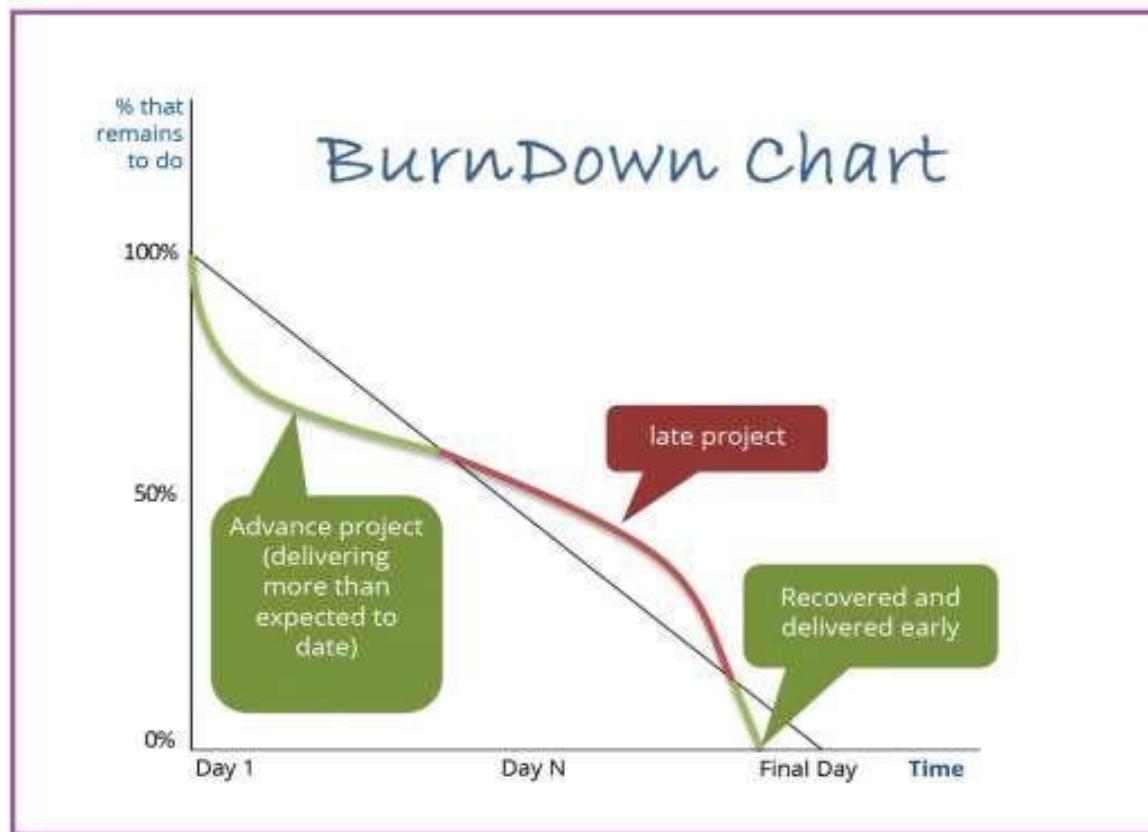


Sprint Backlog

- A list of tasks the team must complete to deliver an increment of functional software at the end of each Sprint.
- Once decided Team owns the Sprint Backlog – only they can decide on scope change

Sprint BurnDown Charts

- Shows the total estimated work remaining for the entire forecasted sprint backlog against time



Task Board (Kanban)

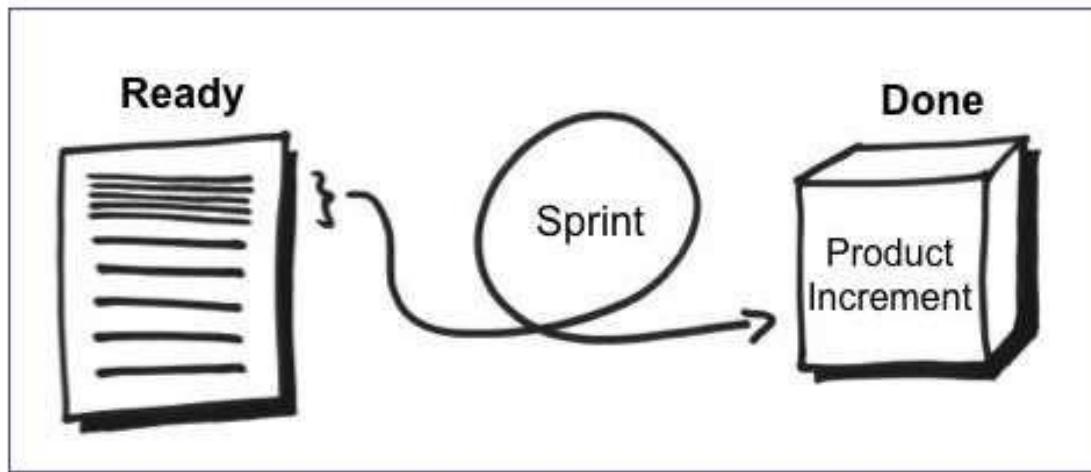
- Allows visibility, transparency across the project
- Displays the live status of team work and focus
- Most have – Backlog, To-do, In Progress (Doing) and Done status.



- Boards can be physical or digital
- They are often physical providing strong motivation for the team

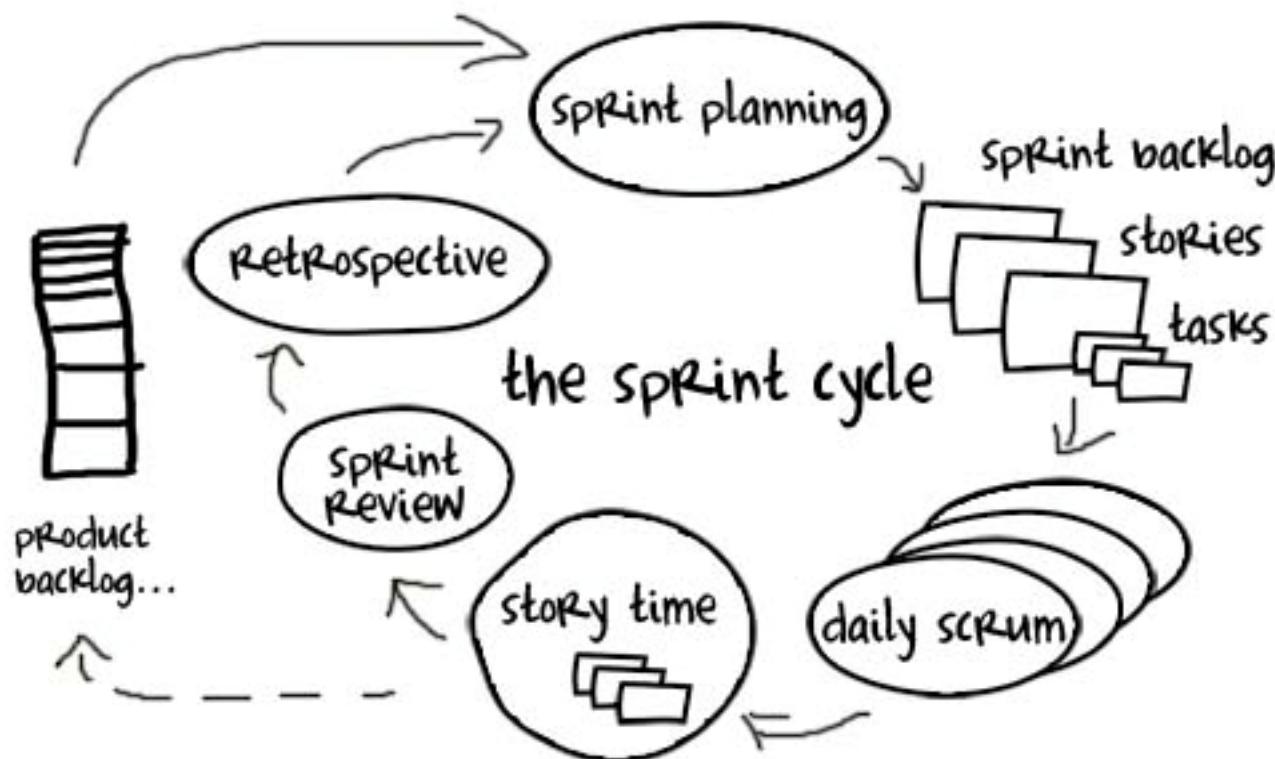
Product Increment

- A Product Increment is the end product for each sprint. It must:
 - Be of high enough quality to be given to users
 - Meet the Scrum team's current definition of DONE
 - Be acceptable to the product owner.



SCRUM Activities – A sprint cycle

The Sprint is a timebox of 2-4 weeks during which the team produces a potentially shippable Product Increment.



Sprint Activities

Start of the Sprint - Sprint Planning

Determine which items from the product backlog they will work on during the Sprint.

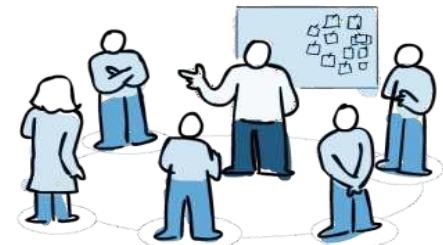
The **end result the Sprint Backlog** – defines the scope of the sprint

1. Discussion with product owner – WHAT will we do
2. Team does the detailed plan – HOW will we do it

During the Sprint - Daily Stand Up

Short (usually limited to 15 minutes) discussion where the team coordinates their activities for the following day. The only focus of the Daily Stand Up:

1. What I did since last daily scrum meeting
2. What I am planning to work on today
3. Impediments (Issues/blockers) if any?



Sprint Activities - At the end of the Sprint

Sprint Review

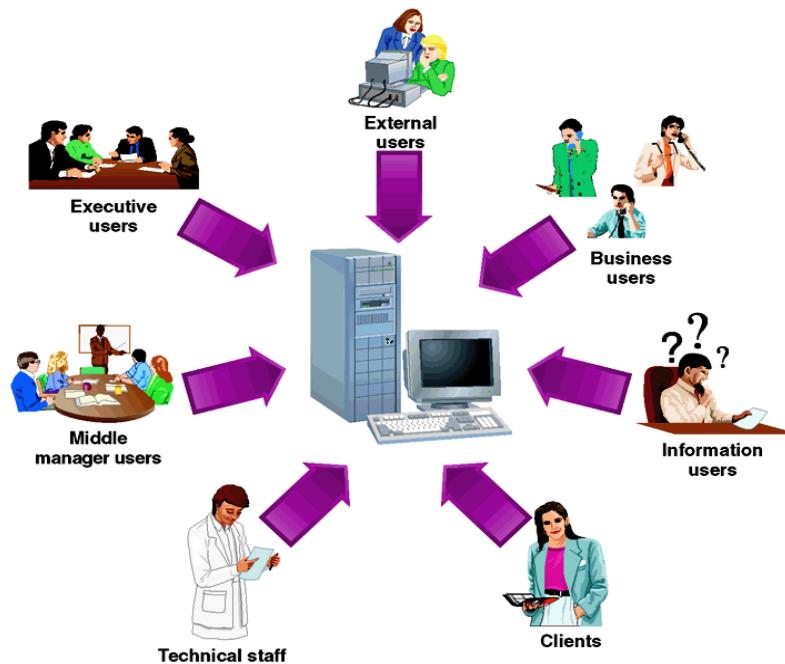
- The entire team does a review to get ‘Product Increment’ feedback from the Stakeholders
- Feedback goes into the ‘Product Backlog’ for future consideration.
- Not intended to provide a status report



Sprint Retrospective

- The team (including product owner) reflect upon how things went during the previous sprint
 - What went well
 - What could be improved
- They identify adjustments they can make moving forward

STAKEHOLDER MANAGEMENT

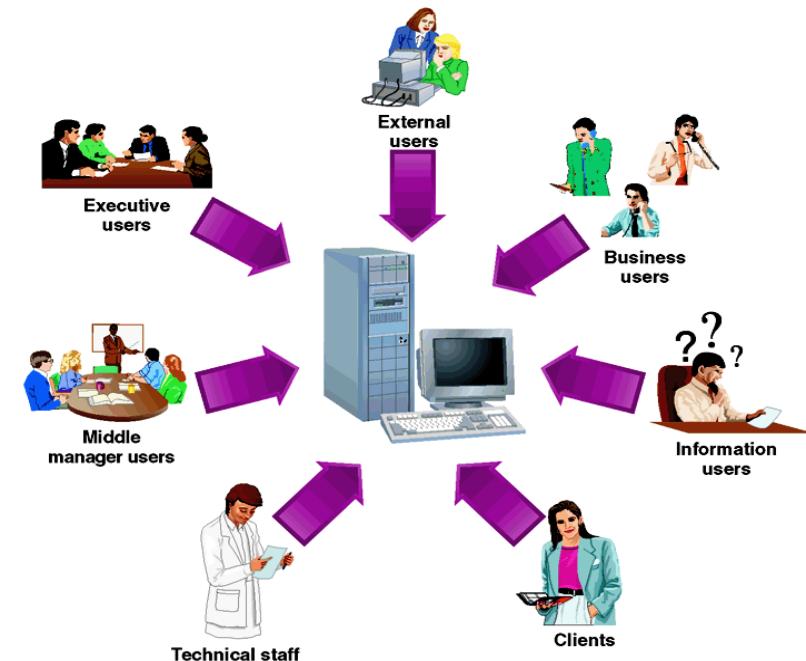


Who do you get these requirements from?

.... a range of stakeholders

People interested in the success (sometimes failure) of a system

It might include senior executives, project organisation roles, client organisation roles, system developers, IT operations, customers, etc.



Identify stakeholders

Need to identify the correct people:



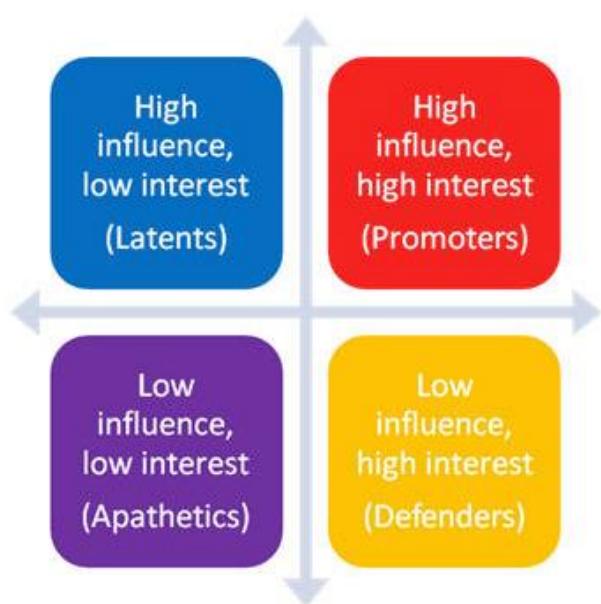
Identify stakeholders

Need to find out:

- Who gains and who loses from this development?
- Who controls change management of processes?
- Who will make the decisions?
- Who procures IT systems and who decides what to buy?
- Who controls resources?
- Who has specialist skills the project needs?
- Who has influence?
- *Project owner (facilitates progress) &*
Project sponsor (sells change to users) – both very important
 - Can have a major impact on project success

Prioritise and understand your stakeholders

- Someone's position on the grid shows you the actions you have to take with them
- Influence = Power



- You need to understand how they feel about the project
- Determines how you engage / communicate with them

Can block or advance your project

Managing stakeholder expectations?

- Systems fail if they don't meet expectations

Requirement:

Create a means to transport a single individual from home to place of work.

Management Interpretation



I T Interpretation



User Interpretation



Your role ...



... to get the disparate views

... create a
shared vision

Workshop Preparation

- Watch Seminar 2
- Watch the Industry Seminar
- Watch the Agile video in the resources

**Thanks for watching
See you next week**

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 10

Agile basics:

Agile Alliance - <https://www.agilealliance.org/agile-essentials/>
<http://www.agilelearninglabs.com/resources/scrum-introduction/>

Agile Video - <https://www.youtube.com/watch?v=OJfIDE6OaSc>



Information Technology

FIT2001 – Systems Development

Seminar 3: Investigating System Requirements
Information Gathering Techniques

Chris Gonsalvez

QUESTION

Very often

Often

Sometimes



Our road map:

- How do we find out what the user wants?
- Who do we need to talk to?
- Data gathering methods

- What are Information Systems?
- How do we develop them?
Systems Development (SDLC) – key phases
- Some System Development roles and skills
- Development approaches – focus on Agile



At the end of this seminar you will:

- Understand the requirements gathering process
- Explain the difference between functional and non-functional requirements
- Understand how to investigate and validate the requirements using a range of techniques, and be able to determine when each is best applied

What is requirements gathering?

- Investigating requirements using a range of techniques



Developing a deep understanding of the business domain

- Defining what the solution needs to be to meet the requirements

What is requirements gathering?



Bringing 'fresh eyes'
to the problem

Identifying opportunities for
improving business processes –
Business Process Re-engineering
(BPR)





*Requirements must be
verified by the client*



What do you need to gather?

Requirement categories	FURPS + categories	Example requirements
Functional	Functions	Business rules and processes
Nonfunctional	Usability Reliability Performance Security + Design constraints Implementation Interface Physical Support	User interface, ease of use Failure rate, recovery methods Response time, throughput Access controls, encryption Hardware and support software Development tools, protocols Data interchange formats Size, weight, power consumption Installation and updates

Systems Analysis and Design in a Changing World, 7th Edition – Figure p45-47



Investigating Requirements Case Study

ON THE SPOT COURIER SERVICES

Bill Wiley – start up, same day courier service

Initially just received delivery requests via texts on his mobile, but then customers started asking if he had a website where they could place orders

As the business grew, Bill hired another person to help with the deliveries. He could no longer use his van as the ‘warehouse’, he now needed a central warehouse where he could organise and distribute packages for delivery, and if it grew further someone at the warehouse to co-ordinate the arrival and distribution of the packages

What information do you need to gather for On the Spot Courier Services

We now know:

Why we want to gather requirements?

What we need to investigate?

Who we need to gather these requirements from?

. . . our next step is work out:

How to find out what the user wants?

What approaches do we use to gather User Stories?

(the Agile way of documenting requirements – more next week)

INVESTIGATING SYSTEM REQUIREMENTS

Can be quite challenging ...



... Users often find it difficult to articulate exactly what they want



Common fact finding techniques:

1. Interview users/stakeholders
2. Use questionnaires to gather information
3. Observe business processes
4. Review existing reports, forms, and procedure descriptions
5. Research vendor/competitor solutions
6. Prototyping *
7. Story-writing workshops *

* *To be discussed next week*

1. Interviewing



- An effective way to understand business functions and rules
- But time-consuming, resource intensive ... multiple sessions

Preparing for a Successful Interview

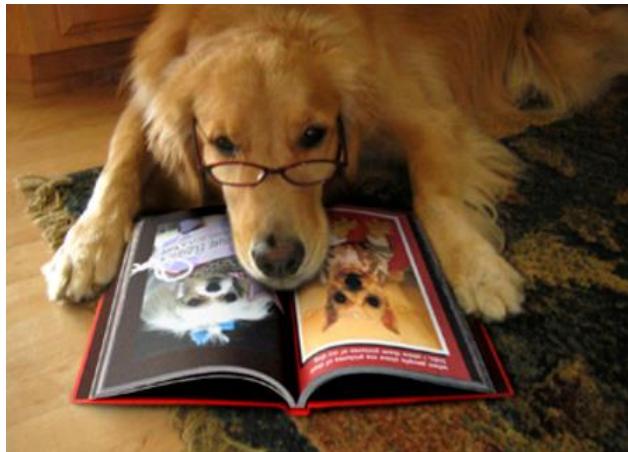
Set clear objectives

What types of
information do you
expect to get?



Preparing for a Successful Interview

Select appropriate stakeholders to interview



Do your research about the participant

Preparing for a Successful Interview

Determine the type of interview



One-on-one interview

Group interview



Preparing for a Successful Interview

Consider company documentation

Forms

This form is dated 1 April 2008 and is valid until 31 March 2009

Zurich Wealth Protection Application Form

Before completing, or signing, this application form please read the Zurich Wealth Protection Product Disclosure Statement (PDS). The PDS must be provided to you with this Application Form. It will help you to understand the product and decide if it is appropriate to your needs.

Please use black pen, BLOCK LETTERS and ticks (✓) where applicable. DO NOT USE HIGHLIGHTERS.

1.

Use this Application Form to apply for the products offered in the Zurich Wealth Protection PDS or to increase an existing policy.

What are you using this application for?

- To apply for new products
- To increase an existing policy -> provide policy number

2.

Generally, you can insure more than one person under the one policy. If there are more than two life insureds, you must fill out additional Application Forms.

Do you want to insure two people under this policy?

- Yes = fill in "Life Insured T" and "Life Insured 2"
- No = fill in "Life Insured T" only

3.

Who is applying to be insured under this policy?

Life insured 1

- Mr Mrs Ms Miss Other

last name

given names

- male female

date of birth

/ /

residential address

state

postcode

country of residency

postal address

state

postcode

work phone number ()

home phone number ()

mobile number ()

email

fax number ()

If you are only insuring one person go to section 4 on the following page →

Life Insured 2

- Mr Mrs Ms Miss Other

last name

given names

- male female

date of birth

/ /

residential address

state

postcode

country of residency

postal address

state

postcode

work phone number ()

home phone number ()

mobile number ()

email

fax number ()

Continue this application form on the next page ↗

Zurich Australia Limited ABN 52 000 018 191 | AFSL 221020

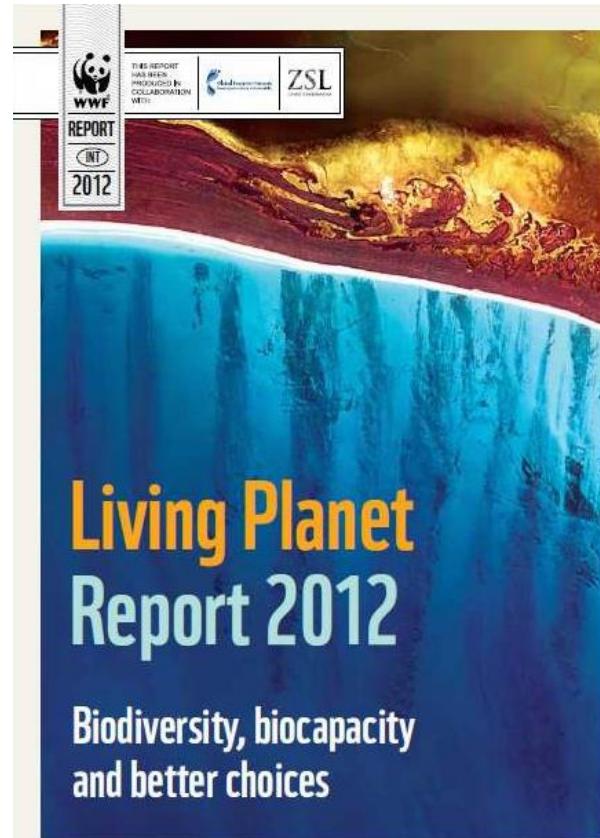
ACAS-001167-2008-04093344

Zurich Australian Superannuation Pty Limited | ABN 78 000 001 013 | AFSL 220198 | RIS Licence Number 08993216

1 Blue Street North Sydney 2000

page 1 of 28

Reports



Preparing for a Successful Interview

Develop
an
Agenda

Discussion and Interview Agenda	
Setting	
Objective of Interview	<i>Determine processing rules for sales commission rates</i>
Date, Time, and Location	<i>April 21, 2012, at 9:00 a.m. in William McDougal's office</i>
User Participants (names and titles/positions)	<i>William McDougal, vice president of marketing and sales, and several of his staff</i>
Project Team Participants	<i>Mary Ellen Green and Jim Williams</i>
Interview/Discussion	
1. Who is eligible for sales commissions?	
2. What is the basis for commissions? What rates are paid?	
3. How is commission for returns handled?	
4. Are there special incentives? Contests? Programs based on time?	
5. Is there a variable scale for commissions? Are there quotas?	
6. What are the exceptions?	
Follow-Up	
Important decisions or answers to questions	
See attached write-up on commission policies	
Open items not resolved with assignments for solution	
See Item numbers 2 and 3 on open items list	
Date and time of next meeting or follow-up session	
April 28, 2012, at 9:00 a.m.	

Systems Analysis and Design in a Changing World, 7th Edition, Figure 2.8, p53

Preparing for a Successful Interview

Avoid long interviews ...



Hard to absorb
large amounts of
information at one
time

Participants often
time poor

Several shorter interviews better .. can clarify information

Preparing for a Successful Interview - Summary

- Set objectives
 - Clear focus, What types of information do you expect to get
 - Select appropriate stakeholders to interview
 - Do your research about the participant
 - Determine the type of interview
 - One on one vs. Group
 - Consider outside information – forms, reports, etc.
 - Will make the interview time more productive
 - Develop an agenda
 - Documents objectives, nothing forgotten, logical progression
- ☞ Avoid long interviews because
- Stakeholders often time poor, Hard to absorb large amounts of information at one time
- ☞ Several shorter interviews better because:
- Can ask further probing questions to seek clarifications
 - Can verify requirements collected from previous iterations of interviews

Preparing for a Successful Interview - **Logistics**

- Planning – Interviewees have been sent:
 - Location and time
 - Objectives and list of questions
- Finalise interview arrangements - send reminders
- Arrive early
- Ensure that the room is prepared for conducting interviews
- Decide on a documentation method
 - Take notes, Recording, Video taped – always get permission

Tips and Tricks for a successful interview

- Interview Skills - Youtube video: Reporter Katie Couric
- Lead the conversation with the information that you already know
- Consider the interviewee's knowledge and role
- Use phrases and words that are easy to understand
- Ask lots of open-ended questions
 - Useful for identifying new ideas
 - Help analysts to identify a large number of business processes quickly
- Use closed questions to collect specific facts (e.g. how many forms a day a sales executive process?)
- Avoid biased or loaded questions
- Paraphrase important findings throughout the interview to make sure you have accurate understanding
- Ask for the opportunity to follow up after the interview to fill in any gaps you may have
- Checklist for conducting an interview

Checklist for Conducting an Interview

Before

- ❑ Establish the objective for the interview.
- ❑ Determine correct user(s) to be involved.
- ❑ Determine project team members to participate.
- ❑ Build a list of questions and issues to be discussed.
- ❑ Review related documents and materials.
- ❑ Set the time and location.
- ❑ Inform all participants of objective, time, and locations.

During

- ❑ Arrive on time.
- ❑ Look for exception and error conditions.
- ❑ Probe for details.
- ❑ Take thorough notes.
- ❑ Identify and document unanswered items or open questions.

After

- ❑ Review notes for accuracy, completeness, and understanding.
- ❑ Transfer information to appropriate models and documents.
- ❑ Identify areas needing further clarification.
- ❑ Thank the participants.
- ❑ Follow up on open and unanswered questions.

Systems Analysis and Design in a Changing World, 7th Edition, Figure 2.7, p52

How to Get the Information You Want

- Review current business problems

Beware: Excessive attention given to current system may result in suggesting a new system that only automates the current system

Determine the high level steps of the process.

Talk about activities and responsibilities for every role

Ask leading questions about each part of the process

Identify exception and error-handling

Ask questions to identify new system requirements and identify business opportunities

- Sample of question themes

Question themes in interviews

Theme	Questions to users
What are the business operations and processes?	What do you do?
How should those operations be performed?	How do you do it? What steps do you follow? How could they be done differently?
What information is needed to perform those operations?	What information do you use? What inputs do you use? What outputs do you produce?

Systems Analysis and Design in a Changing World, 7th Edition – Figure 2.6, p50

Interview – Follow up

- All documentation created after the interview should be reviewed by the participants for accuracy as soon as possible after the interview
- Follow up interviews are required to explain and verify the models with the interview participants, and ask further questions

You will have unresolved issues



They should be tracked and resolved

Sample ‘Open items’ list

A sample open-items list

.... unresolved issues need answers

OUTSTANDING ISSUES CONTROL TABLE						
ID	Issue Title	Date Identified	Target End Date	Responsible Project Person	User Contact	Comments
1	Partial Shipments	6-12-2005	7-15-2005	Jim Williams	Jason Nadold	Ship partials or wait for full shipment?
2	Return and Commissions	7-01-2005	9-01-2005	Jim Williams	William McDougal	Are commissions recouped on returns?
3	Extra Commissions	7-01-2005	8-01-2005	Mary Ellen Green	William McDougal	How to handle commissions on special promotions?

Systems Analysis and Design in a Changing World, 7th Edition, Figure 2.9, p55

2. Questionnaires

- Suited to gathering limited and specific information from a large number of stakeholders
- Good when the people are widely dispersed
- Can give a preliminary insight into business
- Not well suited for gathering detailed information
- Open-ended questions encourage discussion and elaboration, but stakeholders will often not complete them
- Must be written effectively – clear, flows well, respondents questions anticipated
- Sample Questionnaire



RMO Questionnaire

This questionnaire is being sent to all telephone-order sales personnel. As you know, RMO is developing a new customer support system for order taking and customer service.

The purpose of this questionnaire is to obtain preliminary information to assist in defining the requirements for the new system. Follow-up discussions will be held to permit everybody to elaborate on the system requirements.

Part I. Answer these questions based on a typical four-hour shift.

1. How many phone calls do you receive? _____
2. How many phone calls are necessary to place an order for a product? _____
3. How many phone calls are for information about RMO products, that is, questions only? _____
4. Estimate how many times during a shift customers request items that are out of stock. _____
5. Of those out-of-stock requests, what percentage of the time does the customer desire to put the item on back order? _____ %
6. How many times does a customer try to order from an expired catalog? _____
7. How many times does a customer cancel an order in the middle of the conversation? _____
8. How many times does an order get denied due to bad credit? _____

Part II. Circle the appropriate number on the scale from 1 to 7 based on how strongly you agree or disagree with the statement.

Question	Strongly Agree						Strongly Disagree
It would help me do my job better to have longer descriptions of products available while talking to a customer.	1	2	3	4	5	6	7
It would help me do my job better if I had the past purchase history of the customer available.	1	2	3	4	5	6	7
I could provide better service to the customer if I had information about accessories that were appropriate for the items ordered.	1	2	3	4	5	6	7
The computer response time is slow and causes difficulties in responding to customer requests.	1	2	3	4	5	6	7

Part III. Please enter your opinions and comments.

Please briefly identify the problems with the current system that you would like to see resolved in a new system.

3. Review existing reports, forms, and procedure descriptions

- Existing business documents and procedure descriptions within organization
 - Obtain preliminary understanding of processes
 - Identify business rules, discrepancies, and redundancies
 - Be cautious of outdated material
 - Can help guide interviews

Sample form

Sample order form

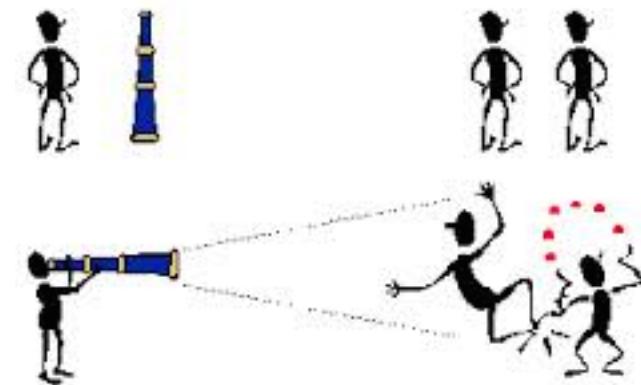
 <p>Ridgeline Mountain Outfitters</p>	<h1 style="text-align: center;">Ridgeline Mountain Outfitters—Customer Order Form</h1>													
<p>Name and address of person placing order. (Please verify your mailing address and make correction below.)</p> <p>Order Date <input style="width: 100px;" type="text"/> / <input style="width: 100px;" type="text"/></p>					<p style="margin: 0;">Gift Order or Ship To: (Use only if different from address at left.)</p>									
					<p>Name <input style="width: 100px;" type="text"/></p> <p>Address <input style="width: 300px;" type="text"/> Apt. No <input style="width: 100px;" type="text"/></p> <p>City <input style="width: 200px;" type="text"/> State <input style="width: 100px;" type="text"/> Zip <input style="width: 100px;" type="text"/></p> <p><input type="checkbox"/> Gift <input type="checkbox"/> Address for this Shipment Only <input type="checkbox"/> Permanent Change of Address</p>									
<p>Name <input style="width: 100px;" type="text"/></p> <p>Address <input style="width: 300px;" type="text"/> Apt. No <input style="width: 100px;" type="text"/></p> <p>City <input style="width: 200px;" type="text"/> State <input style="width: 100px;" type="text"/> Zip <input style="width: 100px;" type="text"/></p> <p>Phone: Day (<input style="width: 50px;" type="text"/>) <input style="width: 50px;" type="text"/> Evening (<input style="width: 50px;" type="text"/>) <input style="width: 50px;" type="text"/></p>					<p>Gift Card Message <input style="width: 100px;" type="text"/></p> <p>Delivery Phone (<input style="width: 50px;" type="text"/>) <input style="width: 50px;" type="text"/></p>									
Item No.		Description			Style	Color	Size	Sleeve Length	Qty	Monogram	Style	Price Each	Total	
MERCHANDISE TOTAL <input style="width: 100px;" type="text"/>														
Method of Payment														
Check/Money Order <input type="checkbox"/> Gift Certificate(s) <input type="checkbox"/> AMOUNT ENCLOSED \$ <input style="width: 100px;" type="text"/>														
American Express <input type="checkbox"/> MasterCard <input type="checkbox"/> VISA <input type="checkbox"/> Other <input type="checkbox"/>														
Account Number <input style="width: 200px;" type="text"/> MO YR <input style="width: 100px;" type="text"/>														
Expiration Date <input style="width: 100px;" type="text"/>														
Any additional freight charges <input style="width: 100px;" type="text"/>														
International Shipping (see shipping information on back) <input style="width: 100px;" type="text"/>														
Signature <input style="width: 100px;" type="text"/>														

Systems Analysis and Design in a Changing World, 7th Edition, Figure 2.11, p56

4. Observe business processes

- Varies from office walkthrough to performing actual tasks
- Not necessary to observe all processes at same level of detail
- May make users nervous, so use common sense

Hawthorne effect: also referred to as the **observer effect** refers to a phenomenon whereby workers improve or modify an aspect of their behaviour, or stop working in response to the fact that they are being watched



5. Research vendor/competitor solutions

- Many problems have been solved by other companies – have a look around for good ideas
- Positive contributions of vendor solutions
 - Frequently provide new ideas
 - May be state of the art
 - Cheaper and less risky
- Danger
 - May purchase solution before understanding problem

aim for when gathering requirements

Validating the requirements

- Meet with users regularly to get feedback on your understanding of the system
- You must confirm that your understanding of the requirements is correct
- You are aiming for requirements that are:

Complete – all functions identified

Unambiguous – nothing vague or fuzzy

Sufficient – level of detail okay

Testable – can check if working as intended

Consistent – no conflicts among requirements

No-one said it was easy ...



How the customer explained it



How the Project Leader understood it



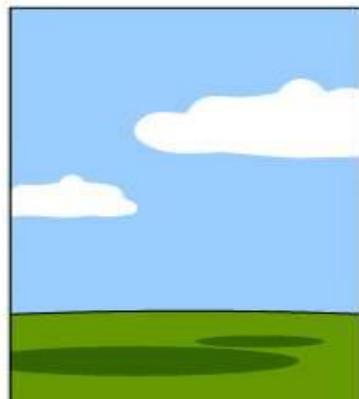
How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



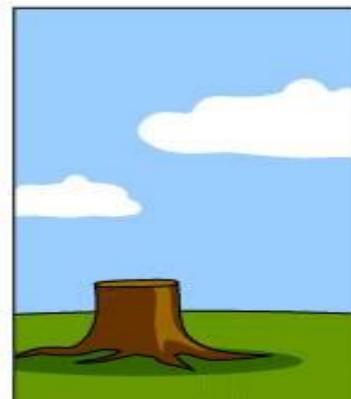
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

You now know the basics of:

- The role of requirements gathering in System development
- A range of techniques to help you investigate requirements

Workshop Preparation

- Watch Seminar 3
- Review any resources

**Thanks for watching
See you next week**

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 2

Resources:

- Interview Skills: Katie Couric video— not about requirements gathering but still very useful

<https://www.youtube.com/watch?v=4eOynrl2eTM>
- Questionnaires: very detailed (you are not required to know this level of detail), nonetheless a valuable resource

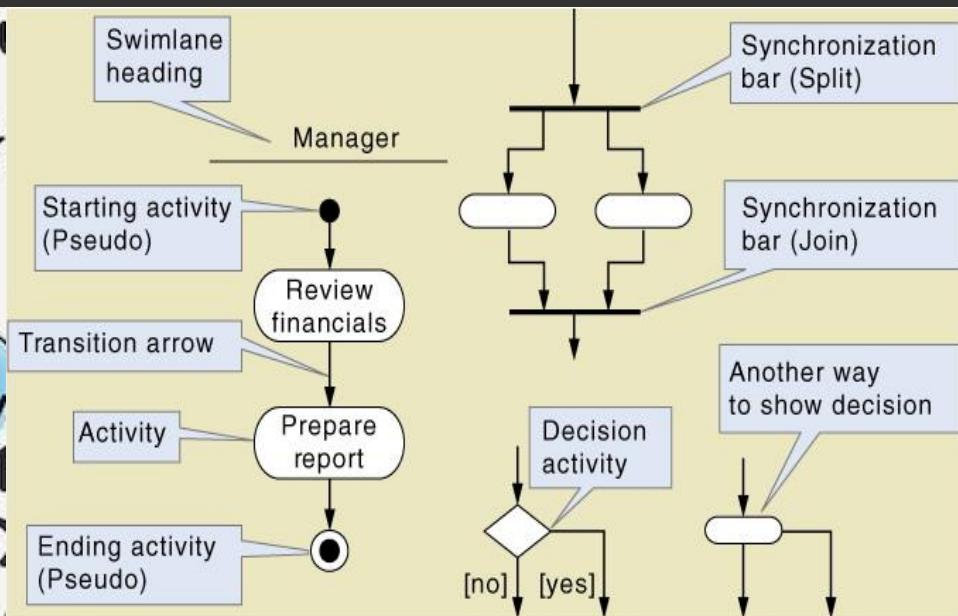
<https://www.youtube.com/watch?v=rSwVZJT9j1c>



FIT2001 – Systems Development

Seminar 4: Investigating and documenting system requirements
User Stories, Activity Diagrams

Chris Gonsalvez



Our road map:

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Development Approaches – Focus on Agile
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering techniques

- Why modelling?
- Investigating and documenting requirements
 - User Stories
 - Activity Diagrams

At the end of this topic you will:

- Explain the value of modelling in systems development;
- Be able to document system requirements using user stories;
- Understand the process for developing suitable user stories for a given scenario;
- Be able to document workflows with Activity diagrams.

Models - Why do we use them?

- The term ‘modelling’ has several interpretations

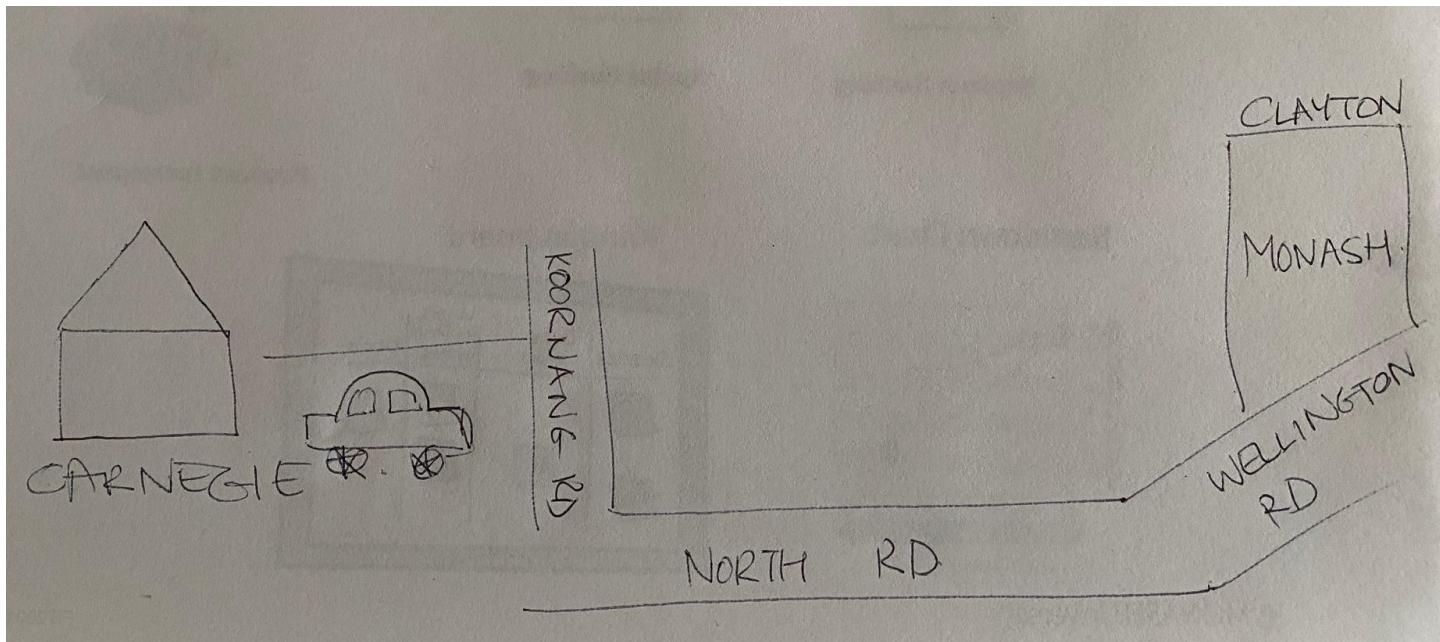


- In its simplest form, the model serves as an abstraction - an approximate representation of the real item that is being built
- It is a simplified picture of complex reality

Why model ?



- Spend a few seconds drawing how you would normally get to Uni from your home

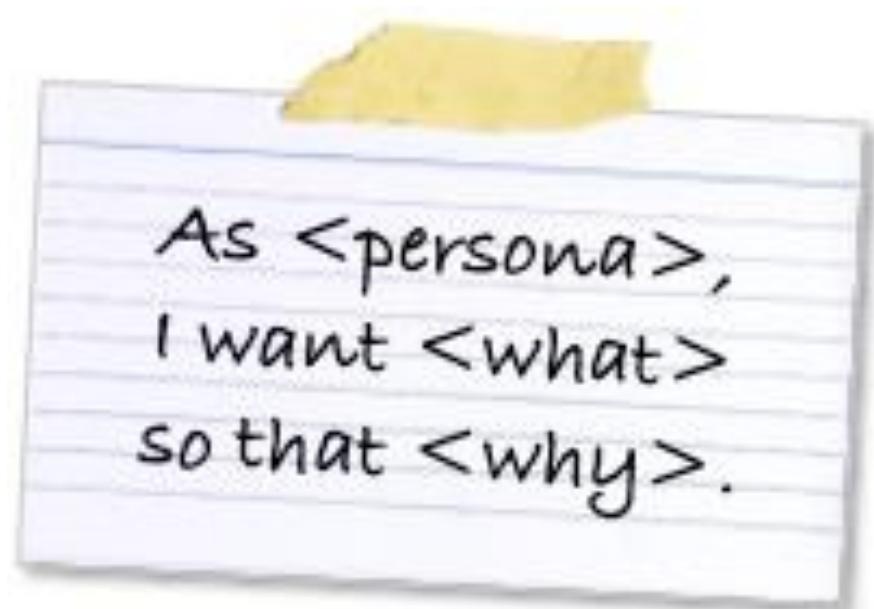


Reasons for modeling in systems analysis

- Reducing complexity of systems to be built by abstraction
- Communication with other development team members
- Communication with stakeholders/users
- Learning from the modelling process
- Documenting all the details of requirements for future maintenance/enhancement - *represents some key aspects of the system being built*
- Some examples: *Activity diagrams, Class diagrams, Use Case diagrams, Sequence diagrams*

Gathering requirements in Agile User Stories – What are they?

- Short, simple description of a product feature – told from the perspective of the user who wants that feature
- They go into the Product Backlog



Examples of User Stories:

- As a student, I want to find my grades messaged to me so that I know my results at the earliest possible time.
- As a book buyer, I want to read reviews of a selected book so that I can make a decision to buy it or not.
- As a parent, I want to see my child's attendance online so that I can take appropriate action if they are not attending

This is high level – they also have acceptance criteria

Why User stories?

- Encourages user communication and collaboration and real-time feedback
- Focus on end user value
- Planning is simplified – if it's too big and you can't estimate, make it smaller
- Avoids locking in design detail too early – focuses on the WHAT - leaves the technical aspects to the developers, testers, etc.
- Users do not need to be trained to understand User stories
- Never out of date ... just in time
- Eliminates weighty documentation – create what you need to deliver the story

Why User stories?

- Easier to communicate with users – Example:

Where does your mind go when I say:

- Steel body
- 4 wheels
- Tyre mounted to each wheel

The focus is on System attributes



User stories focus on User's goals

As a lazy man I want to
mow my lawn quickly
and easy

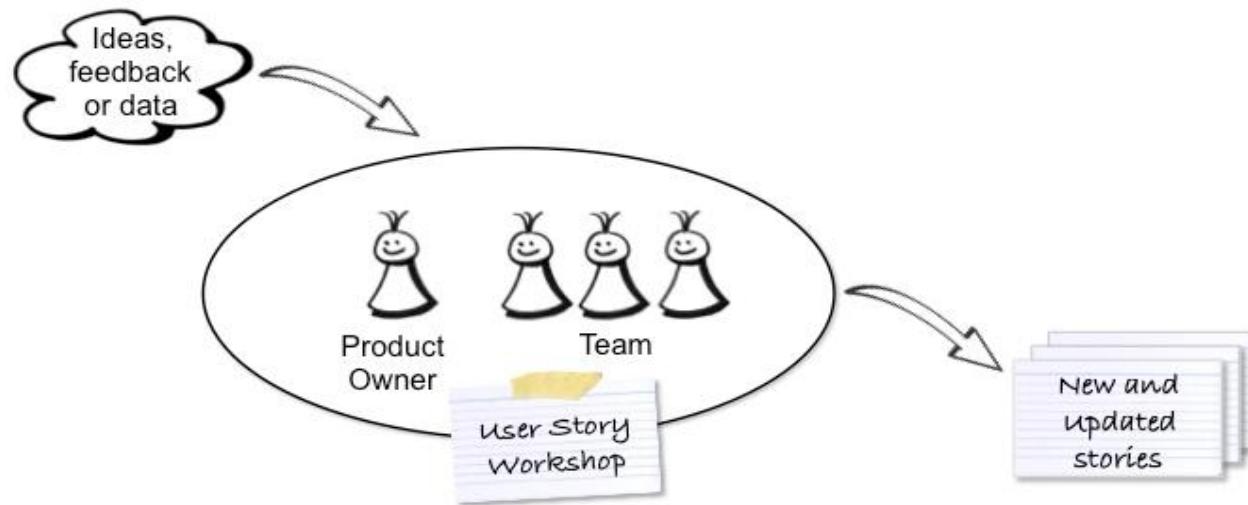
As a lazy man I want to
feel comfortable while
mowing my lawn



Writing User Stories - The process

7. Story Writing Workshops (Seminar 3)

- An good way to have a shared vision for the product
- Encourage the entire team to participate in the process - includes Product Owner, Development Team, and can include Users and Stakeholders ... Scrum Master usually facilitates



Running Story Writing Workshops

Possible steps

1. Focus the meeting on a single objective
2. Map the big picture – brainstorm, list of things users do, cluster, remove duplicates
3. Add user stories
4. Visualise the relationship between stories - Story mapping
5. Map out releases

- Sticky notes, markers, whiteboard - very visual
- Not too long, very interactive

How do you write User Stories

... The 3 Cs by Ron Jefferies

CARD

Traditionally written on index cards – short, concise
Annotated with notes, models, prototypes

CONVERSATION

Story details come out of conversations with the Product Owner (User)

CONFIRMATION

What is going to make the User accept the story as completed?

How do you write User Stories? ... Use a template as a starting point

- 1 Define your **end user**
- 2 Specify what **they want**
- 3 Describe **the benefit**
- 4 Add **acceptance criteria**

Copyright © 2010 Knowledge Train Limited

... Don't get hung up on the words and the format - the key is to understand what the user wants

“As a <user role>, I <want/need/can/etc> <goal> so that <reason>.”



How do you write User Stories? ... Identify the user type

- Avoid the generic role "User"
- Focus on the personas who interact with the system or who realize some value or benefit from the system
 - Who is this user? What motivates them? Who is an example of such a user?

Rita
the Reseller



Rhonda
the Receptionist



Amy
the Assistant



Esteban
the Executive



Chuck
the Call Center



How do you write User Stories? ... Ask questions

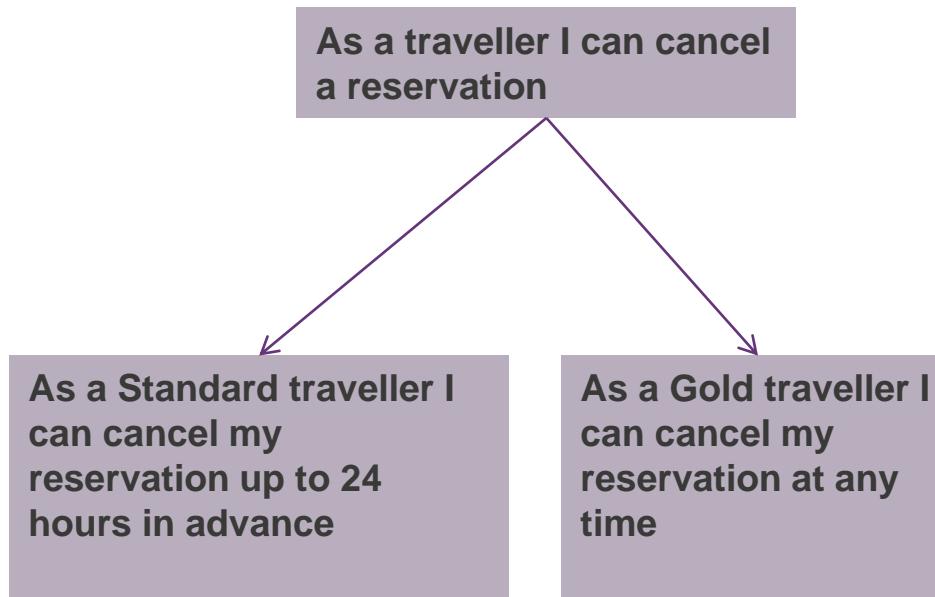
- What else might users of the system have done?
- What could go wrong, and what would the user have to do to recover?
- What might other types of users do to reach their goals?
- Wouldn't it be cool if... ?

How do you write User Stories? ... get the size right

- Should be small enough for the development team to create in a short time period.
- Should be big enough to represent business value in its own right -- it might build on something that has been done before
- Should be able to be delivered in its own right -- you might not want to do so but if you needed to you could.
- User stories that are too big, are harder to understand, estimate, and implement successfully.

How do you write User Stories? ... Story too large?

- EPIC - a LARGE story that is split into smaller user stories that share a strategic objective
- Product Backlog items tend to start as epics that are broken into smaller user stories during Sprint planning



- An epic will typically be delivered over several sprints.

How do you write User Stories? ... adding the Acceptance criteria

As a credit card holder, I want
to view my statement balance,
so that I can pay the balance
due

- Display statement balance upon authentication
- Display Total Balance
- Show “Payment Due Date” and “Minimum Payment Due”
- Display Error message if service not responding/ timeout

Acceptance criteria - a set of predefined requirements that must be met in order to mark a user story complete.

- Reduces ambiguity
- Prevents miscommunication
- Clearly defines what you need to test to meet the user's requirements

Example 1:

As a purchaser on the website

I want the ability to pay with a credit card

So that I can confirm my purchase immediately

Acceptance criteria:

- Accept Visa, Mastercard, Amex
- Accept valid credit cards – valid cc#, valid expiry date, valid cvv no.
- Generate purchase payment confirmed message
- Generate purchase payment failed message

Example 2:

As an online customer

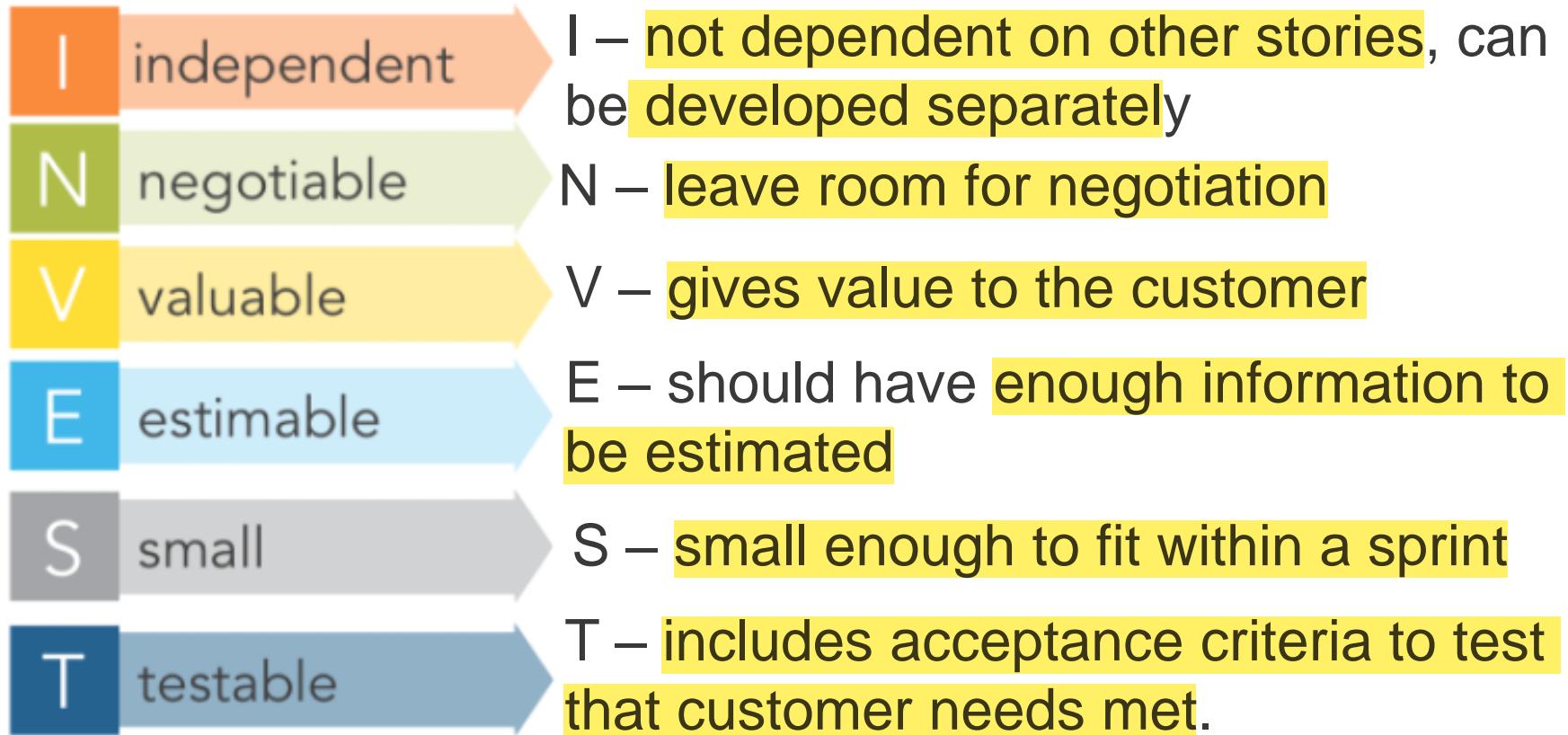
I want to be able to register online

So that I can start shopping online

Acceptance criteria:

- Customer must be able to submit registration form online
- The email used must not be a previously used email
- Customer must complete all mandatory fields on form
- Customer must receive a confirmation email with link after valid submission
- Registration must be confirmed after they click the link

How good is your User Story? **INVEST**



Ref: Bill Wake – Extreme Programming Explored

User Story - Common mistakes

- Too formal / too much detail
 - could result in skipping the conversation
- Technical tasks impersonating user stories
 - Does this represent what the user wants?
- Skipping the conversation
 - Risk moving in the wrong direction, overlooking specific customer needs

READ: Writing a great user story: <https://help.rallydev.com/writing-great-user-story>

Writing User Stories



- If Monash was developing a new Web Enrolment system – write 1 User Story for the ‘Student’ user and 1 User Story for the ‘Lecturer’ user

- Did you follow the template
- Did you remember the conditions of satisfaction (the acceptance criteria)

“As a <user role>, I <want/need/can/etc> <goal> so that <reason>.”

Story Mapping

- Helps arrange user stories into a useful model for understanding the functionality of a system
- Helps identifying holes and omissions in your backlog, and effectively plan holistic releases that deliver value to the business
- Walk through the activities and ask questions such as:
 - What will the user most likely want to do next?
 - What mistakes could a user make here?
 - What could confuse a user at this point?
 - What additional information could a user need?
- *If multiple personas ask questions for each persona*

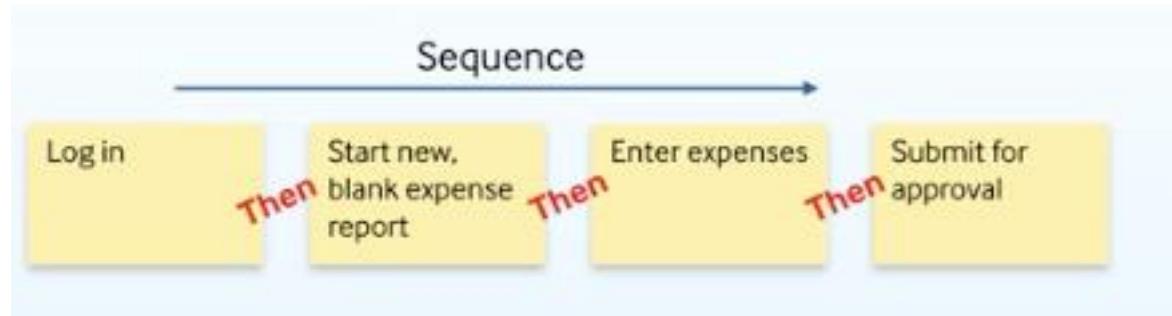
Story Mapping

- Example – Develop software to allow company employees to submit their expense reports and be reimbursed

Ref: Michael Cohn – Better User Stories

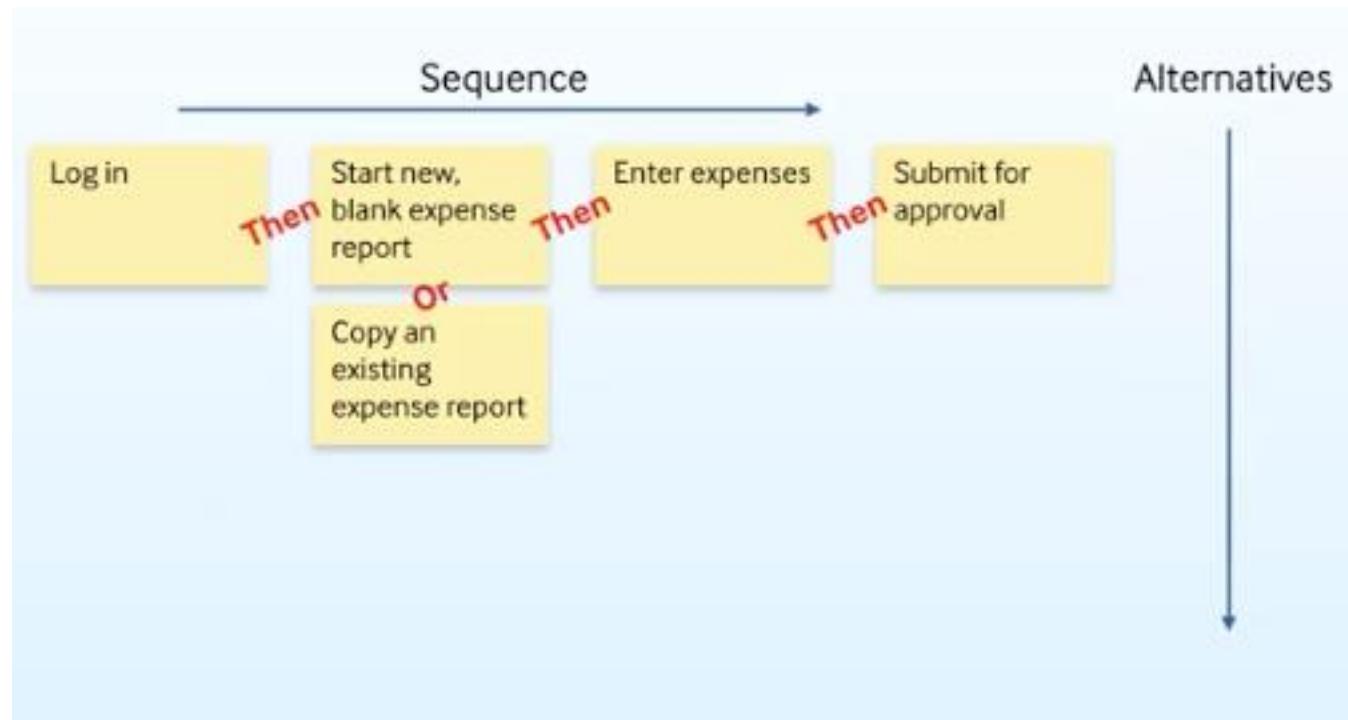
- Two dimensions
 - Sequence
 - Alternatives

1. We start with the sequence of activities in the process – ask Questions



Story Mapping

2. We then look at possible alternatives for activities



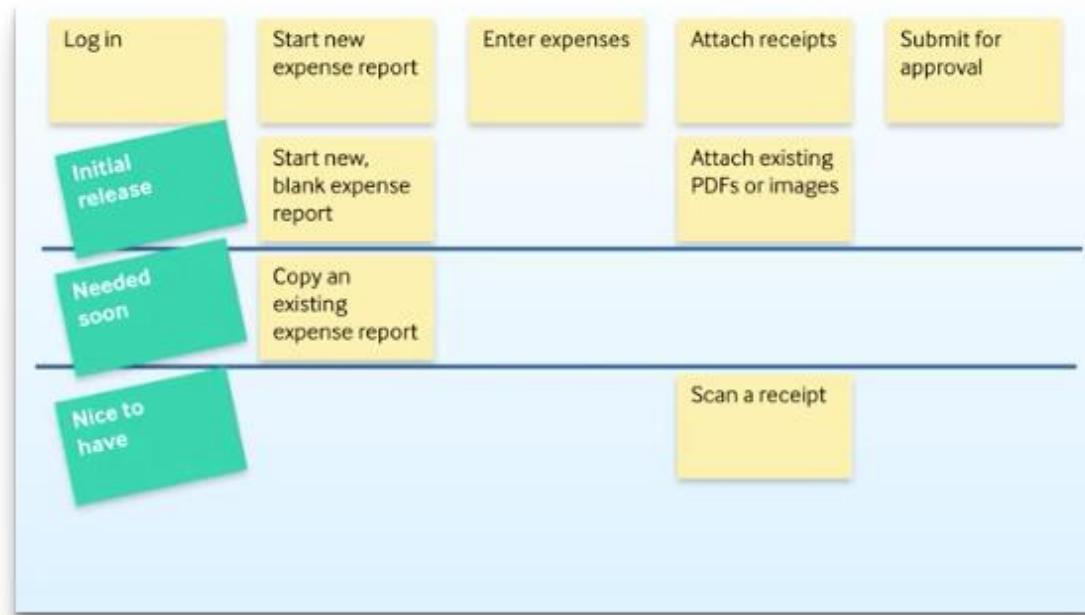
Story Mapping

3. More alternatives identified, a forgotten process added – the higher the value the higher up the story is placed on the alternative axis.



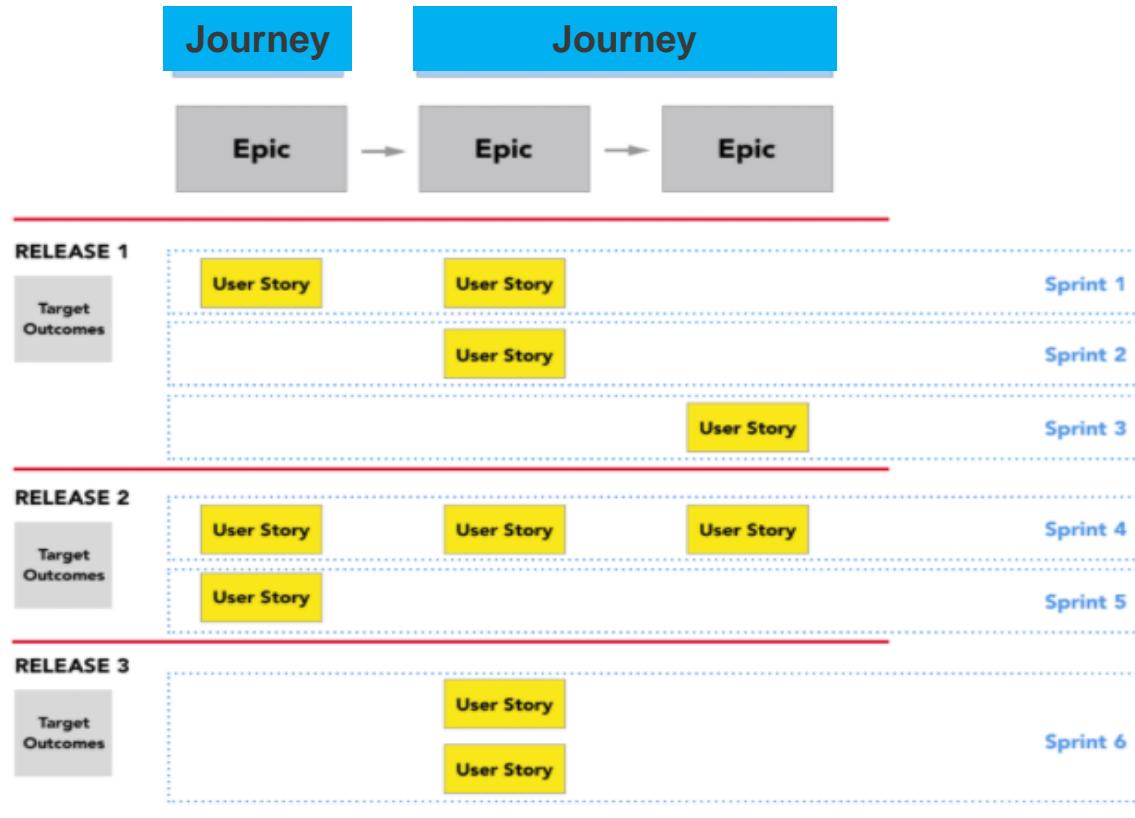
Release planning

- A Release is a set of functionality that makes sense to deliver to users at the same time – can include stories from different Epics
- Create horizontal swim lanes for each release – move stories up or down to form collections that would be the most important to build



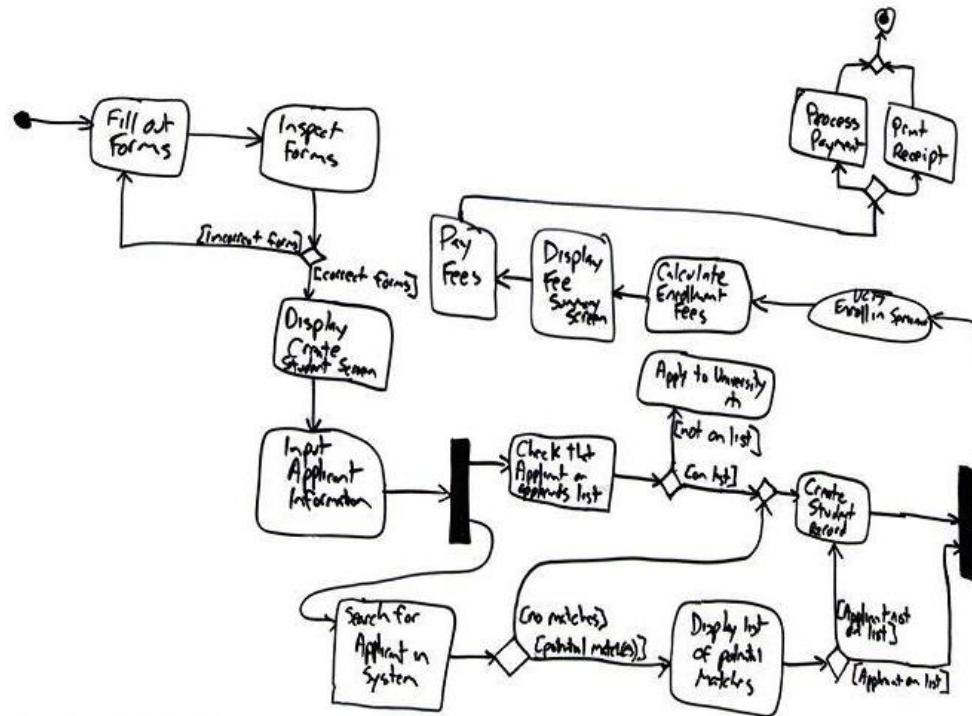
Release planning

- A Release can include multiple sprints
- Release 1 – 4 user stories make sense to be given to users



Activity diagrams

- “Activity diagrams are a technique to describe procedural logic, business processes, and work flows” - M. Fowler
- They add meaning/detail to our user stories (if required)

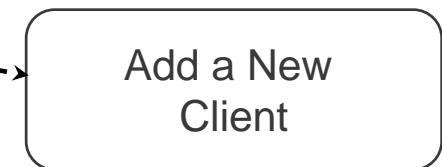


Copyright 2005 Scott W. Ambler

Activity diagram symbols.1

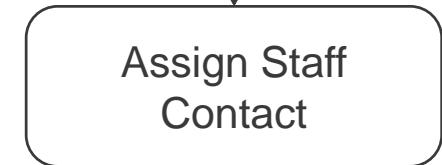
- **Activity / Action**

- rectangle with rounded corners
 - meaningful name

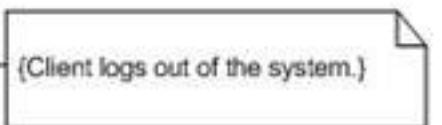


- **Control flows**

- arrows with open arrowheads



- **Notes**



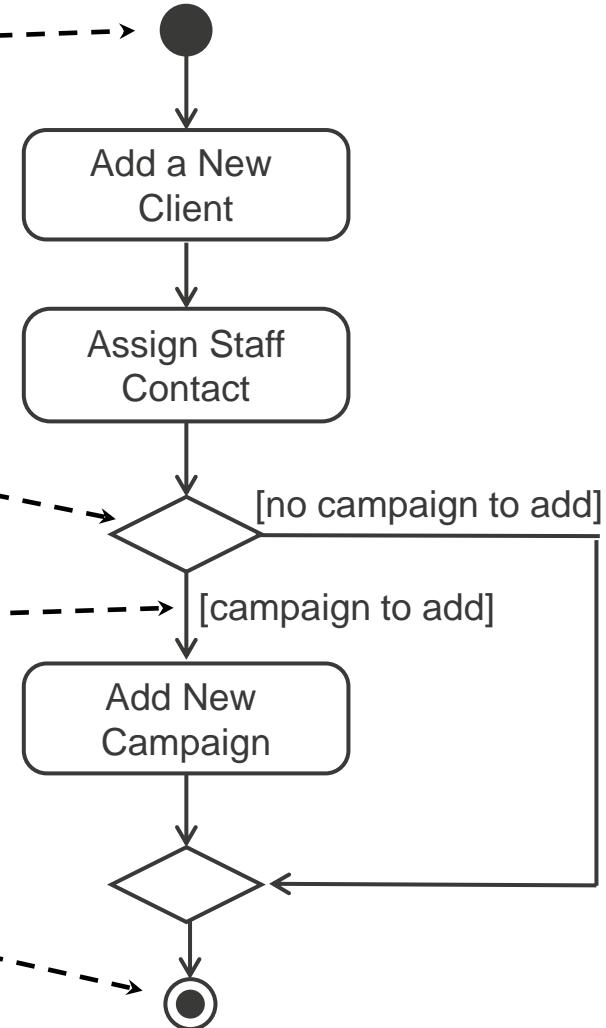
Activity diagram symbols.2

- **Initial node**
 - black circle

- **Decision nodes**
(and merge nodes)
 - Diamond

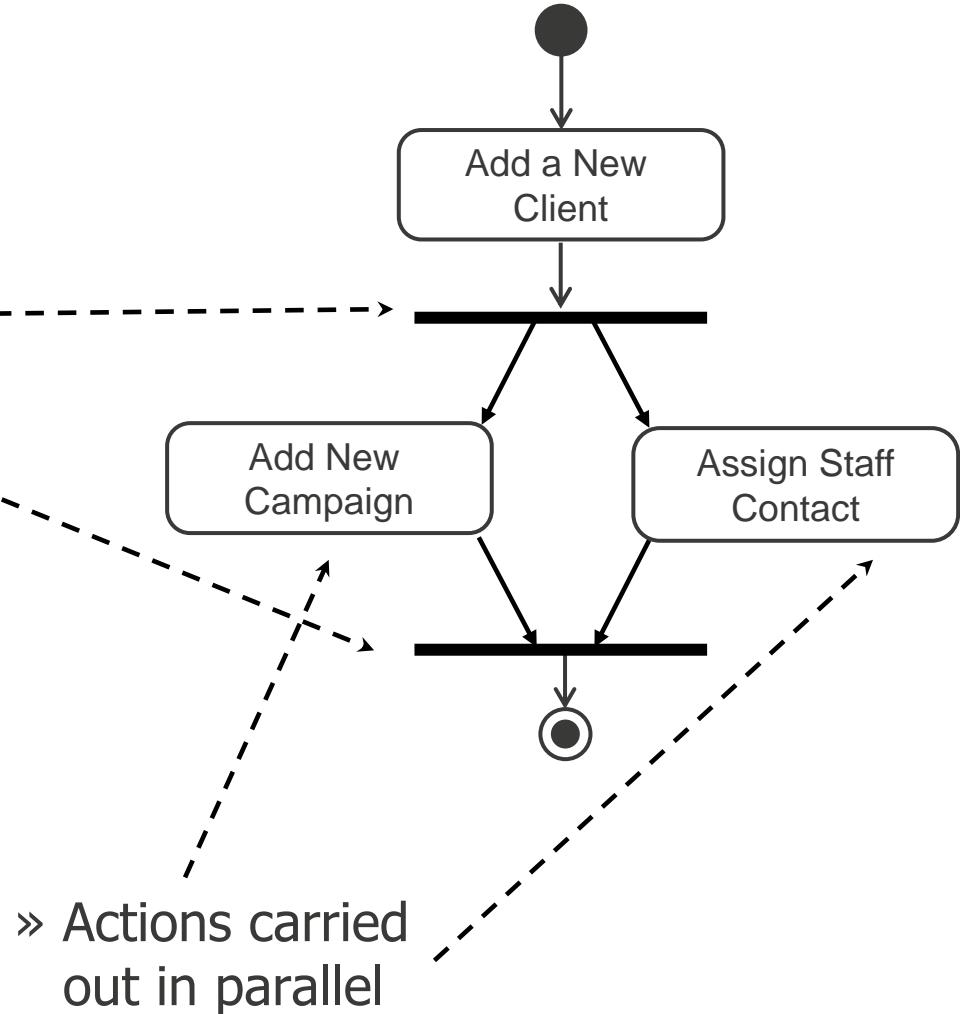
- **Guard conditions**
 - in **square brackets**

- **Final node**
 - black circle in white circle



Activity diagram symbols.3

- Split/Fork nodes
and Join nodes
 - thick bar



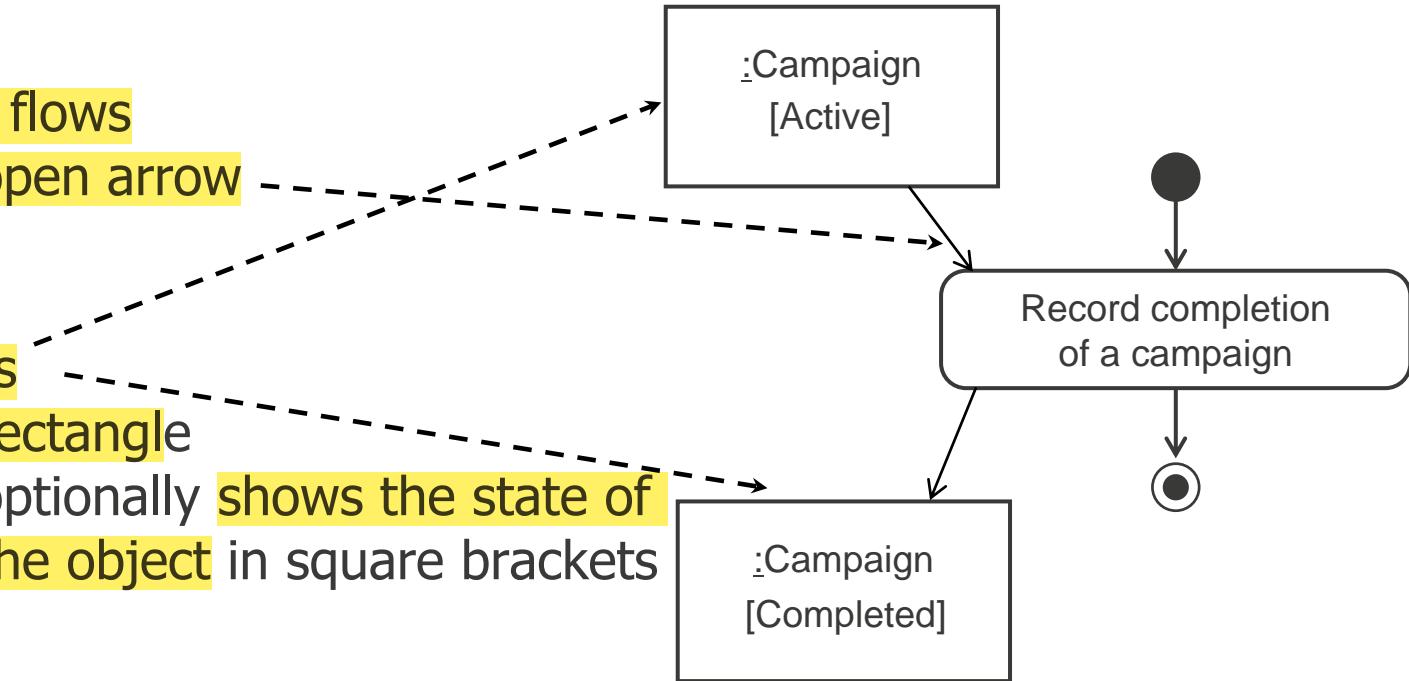
Activity diagram symbols.4

- **Object flows**

- open arrow

- **Objects**

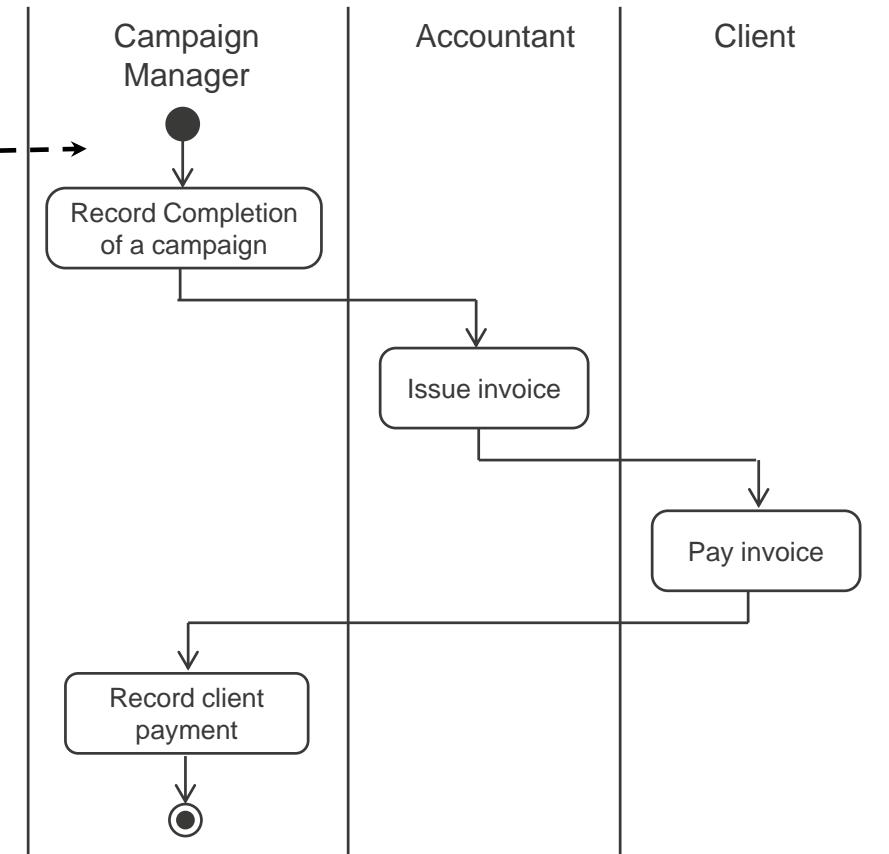
- rectangle
 - optionally shows the state of the object in square brackets



Activity diagram symbols.5

- **Activity Partitions (Swimlanes)**

- vertical columns
 - labelled with the person, organisation, department or system responsible for the activities in that column

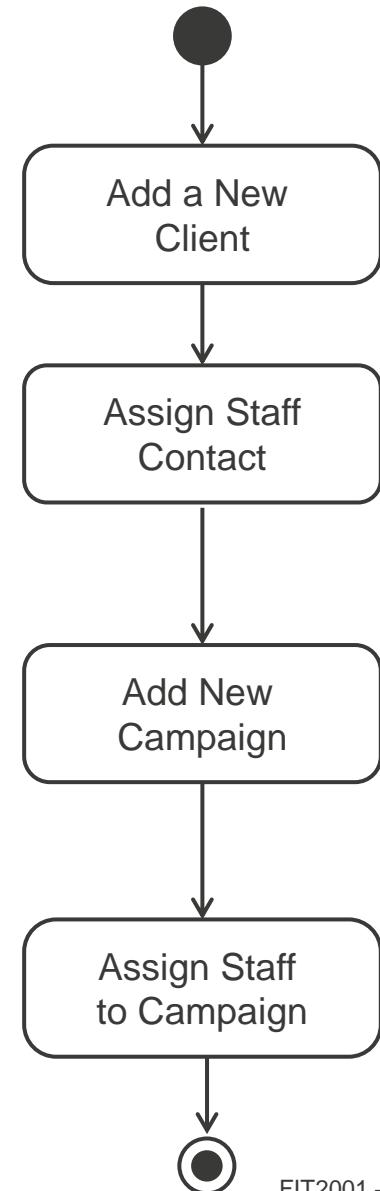


Scenario for Activity diagram discussion

- *When a new client has been accepted by our marketing firm, we assign them a staff contact to look after them. If they have requested a marketing campaign, we add a new campaign. We then assign staff resources to the campaign as required.*

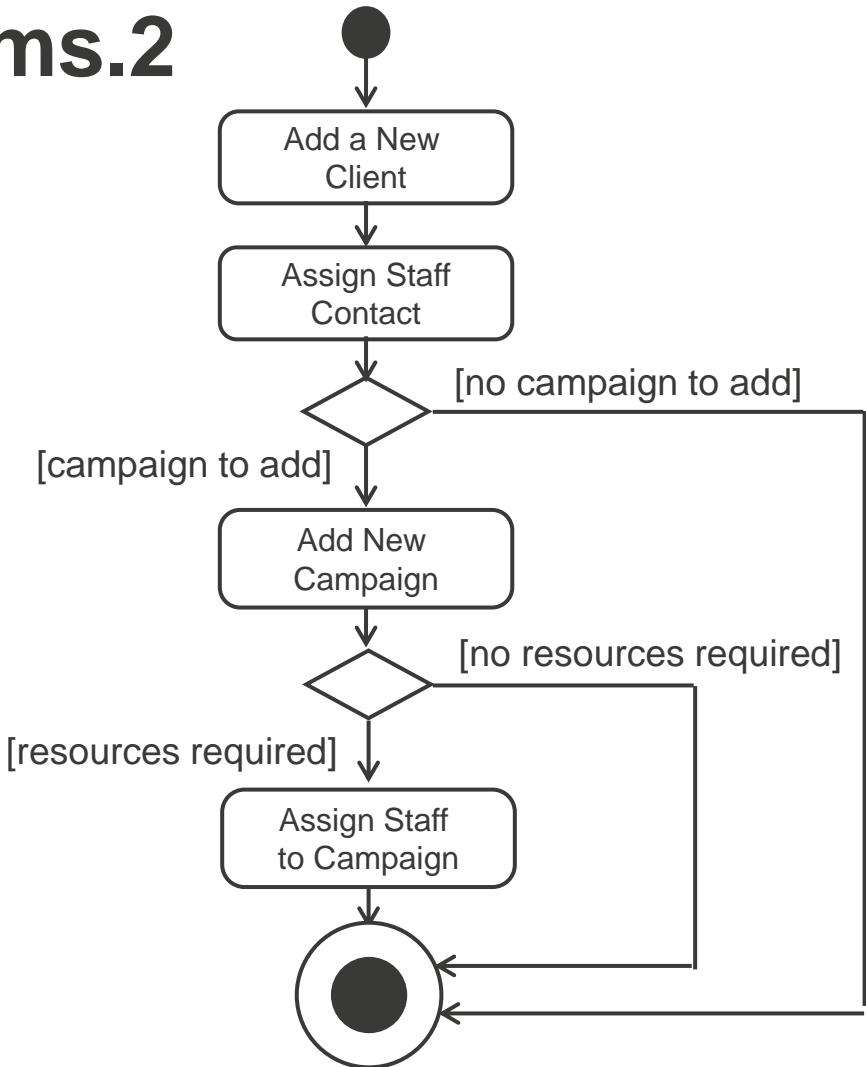
Drawing Activity diagrams.1

- Identify actions
 - What happens when a new client is added to the system
 - Add a New Client
 - Assign Staff Contact
 - Add New Campaign
 - Assign Staff to Campaign
- Organise the actions in sequential order with flows



Drawing Activity diagrams.2

- Identify any alternative flows and the conditions on them
 - sometimes there is a new campaign to add for a new client, sometimes not
 - sometimes they will want to assign staff to the campaign, sometimes not
- Add decision and merge nodes, flows and guard conditions to the diagram

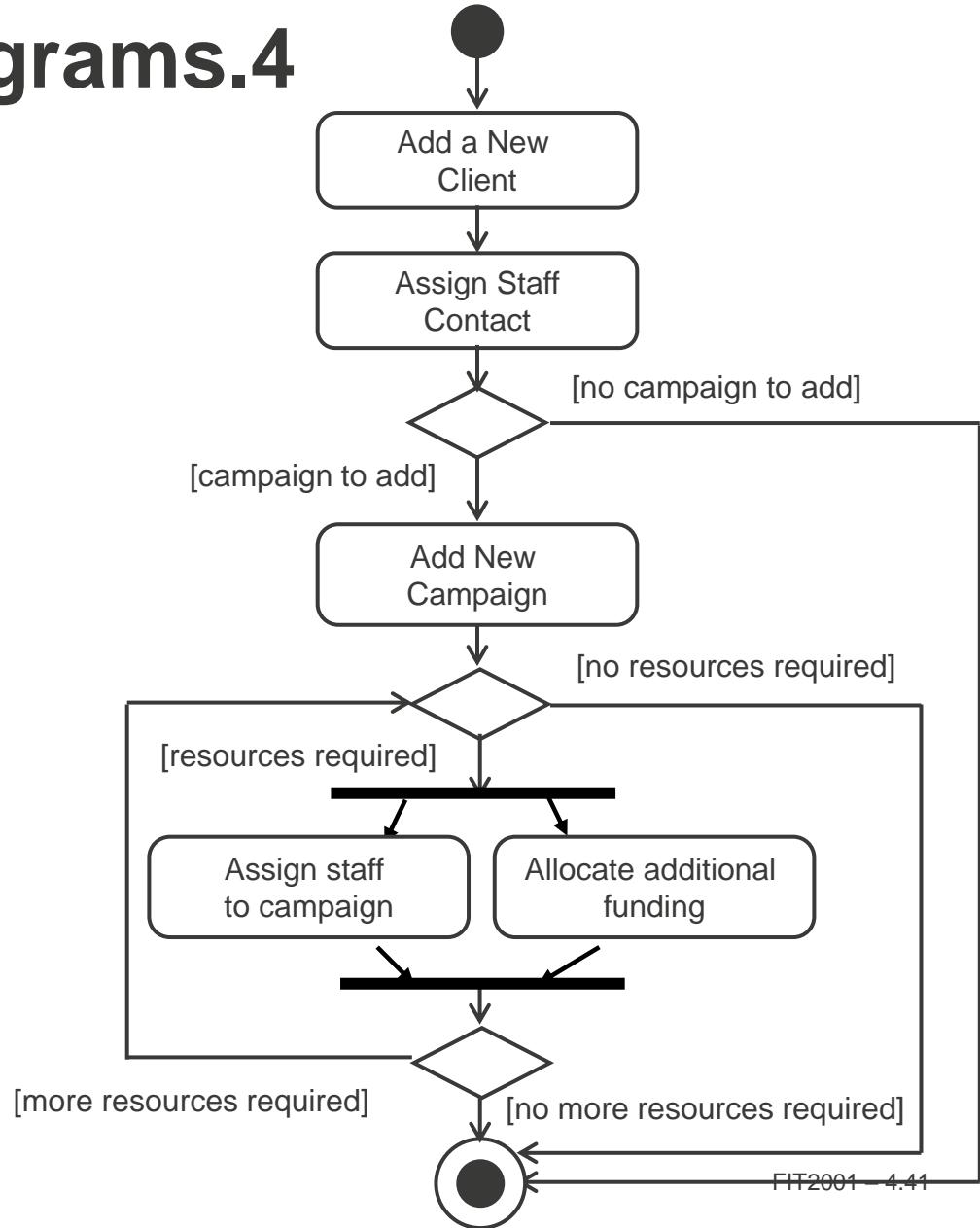


Drawing Activity diagrams.3

- Identify any actions that are carried out in parallel
 - there are none in this scenario, but what if we said that if resources are required, we assign staff and allocate funds
- Add fork and join nodes and flows to the diagram

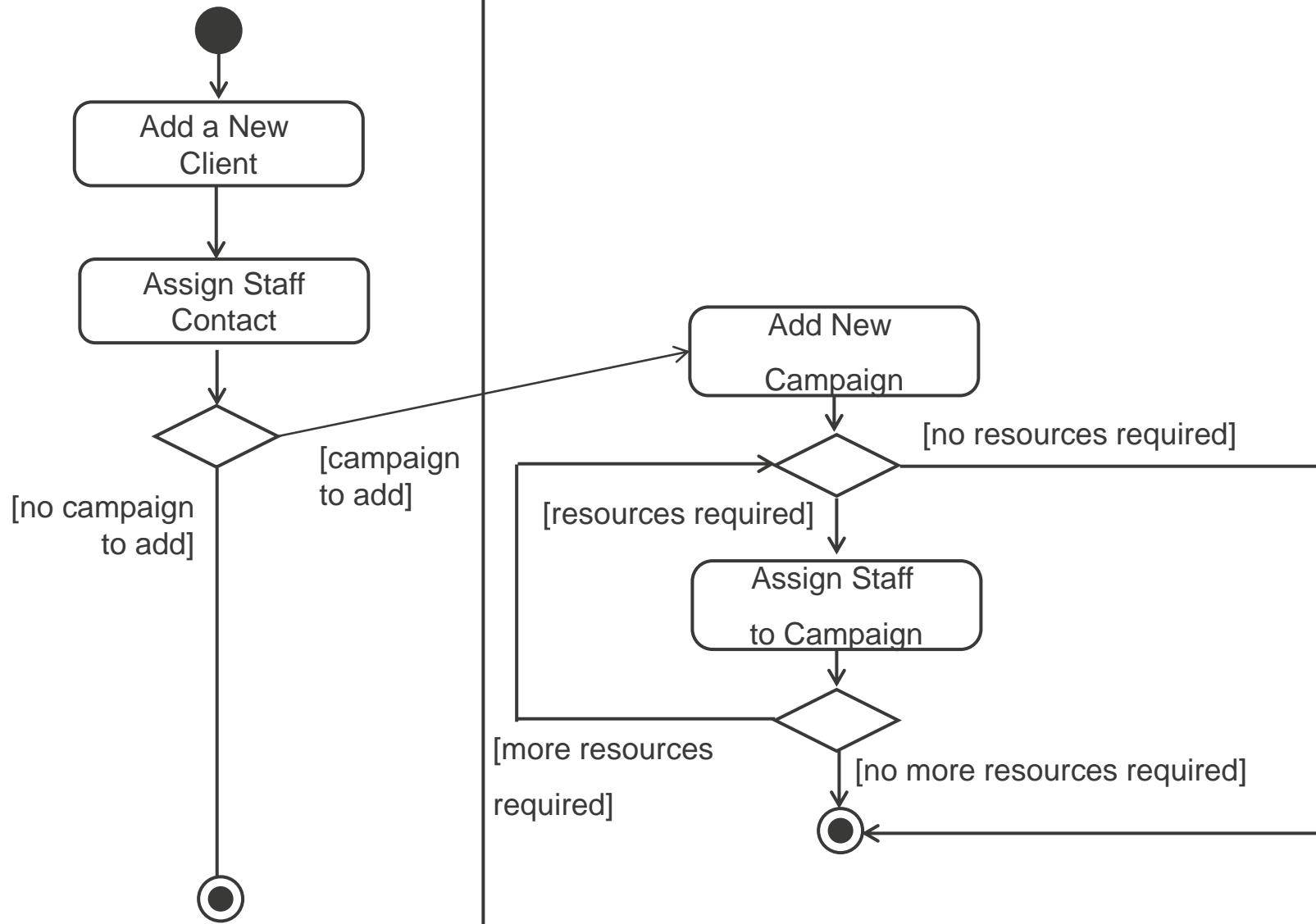
Drawing Activity diagrams.4

- Identify any processes that are repeated
 - they will want to assign staff to the campaign until there are no more staff required
- Add decision and merge nodes, flows and guard conditions to the diagram



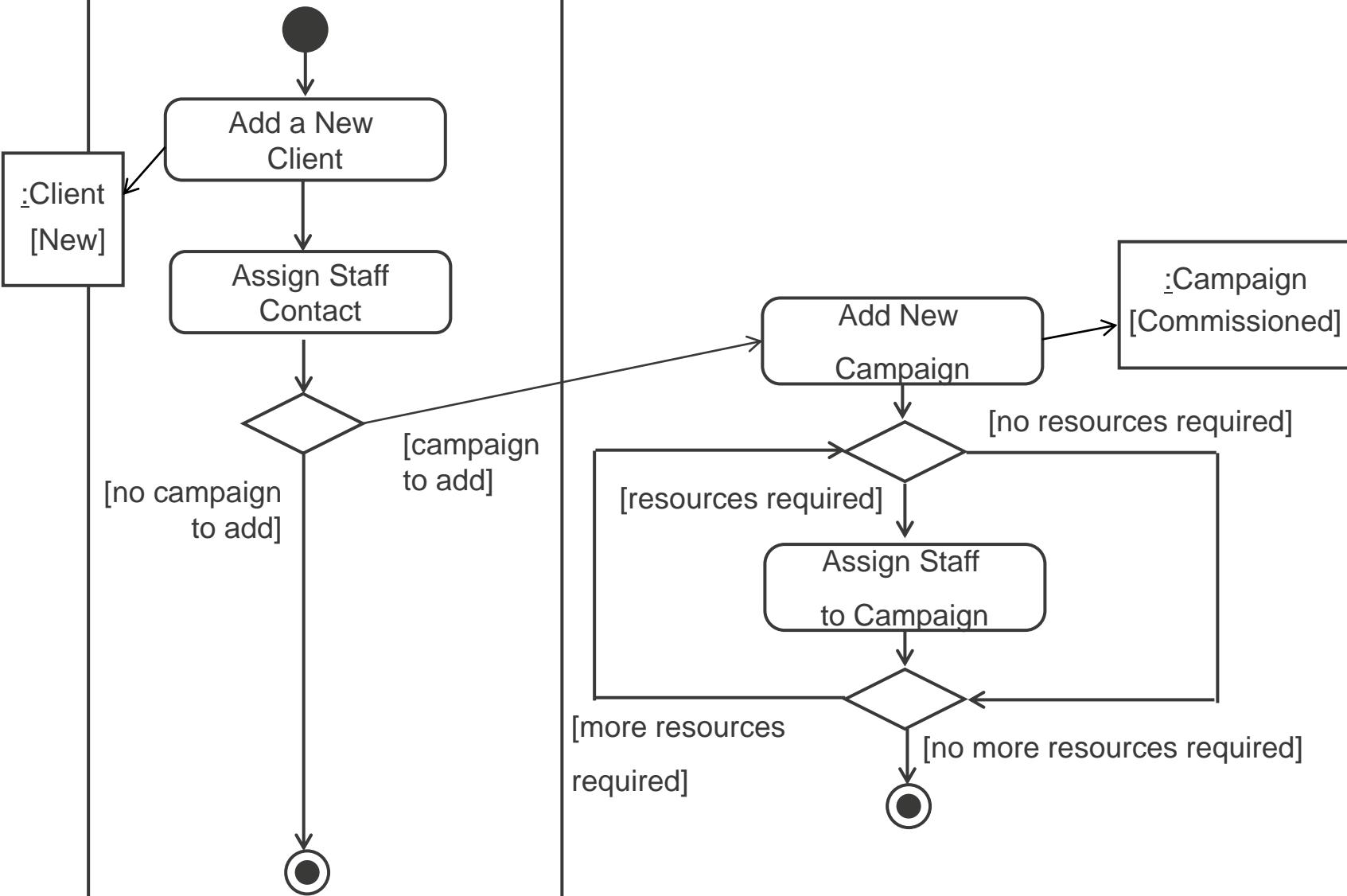
Drawing Activity diagrams.5

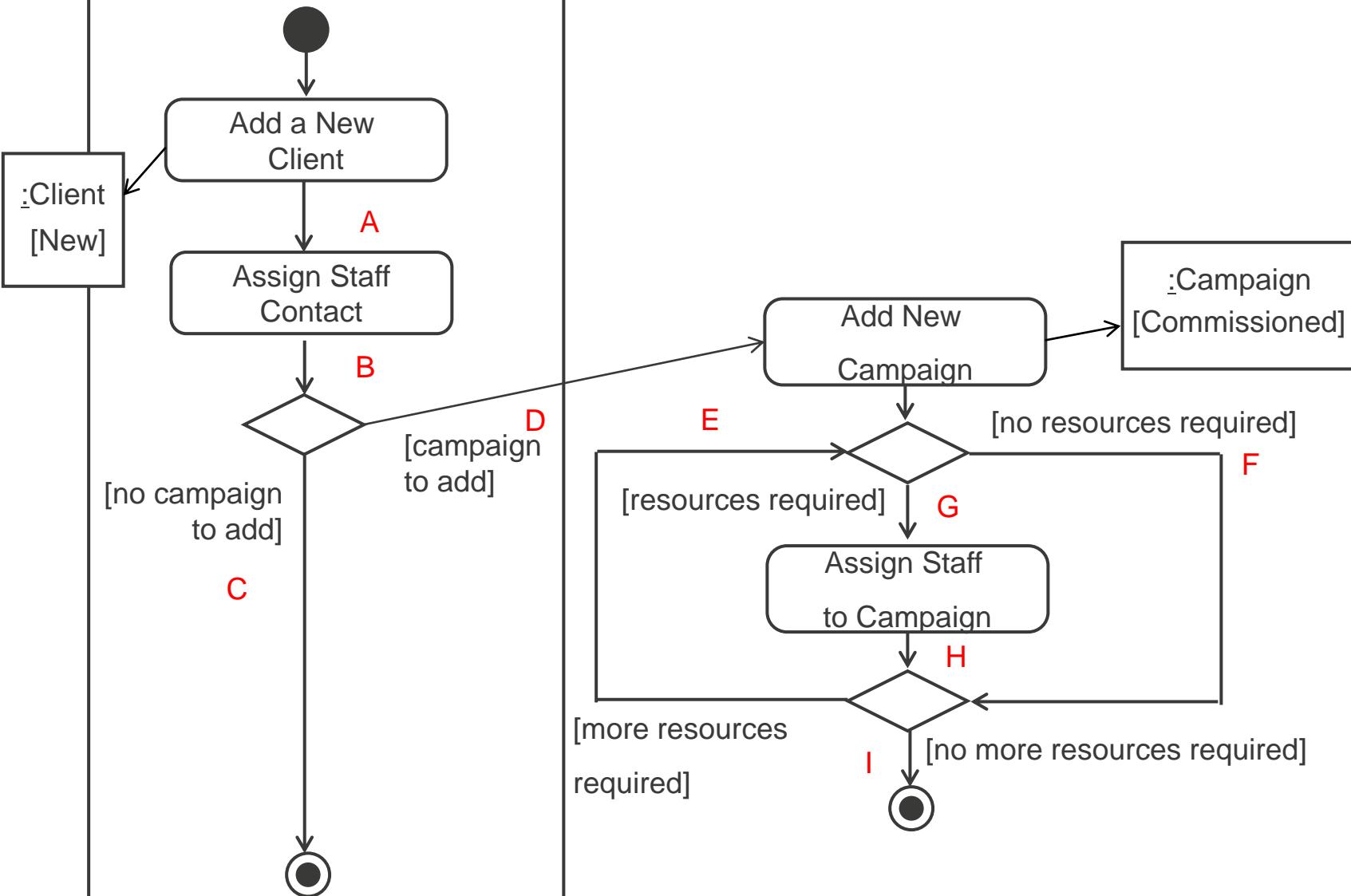
- Are all the activities carried out by the same person, organisation or department?
- If not, then add swimlanes to show the responsibilities
- Name the swimlanes
- Show each activity in the appropriate swimlane



Drawing Activity diagrams.6

- Are there any object flows and objects to show?
 - these can be documents that are created or updated in a business activity diagram
 - these can be object instances that change state in an operation or a use case
- Add the object flows and objects





ON THE SPOT COURIER SERVICES

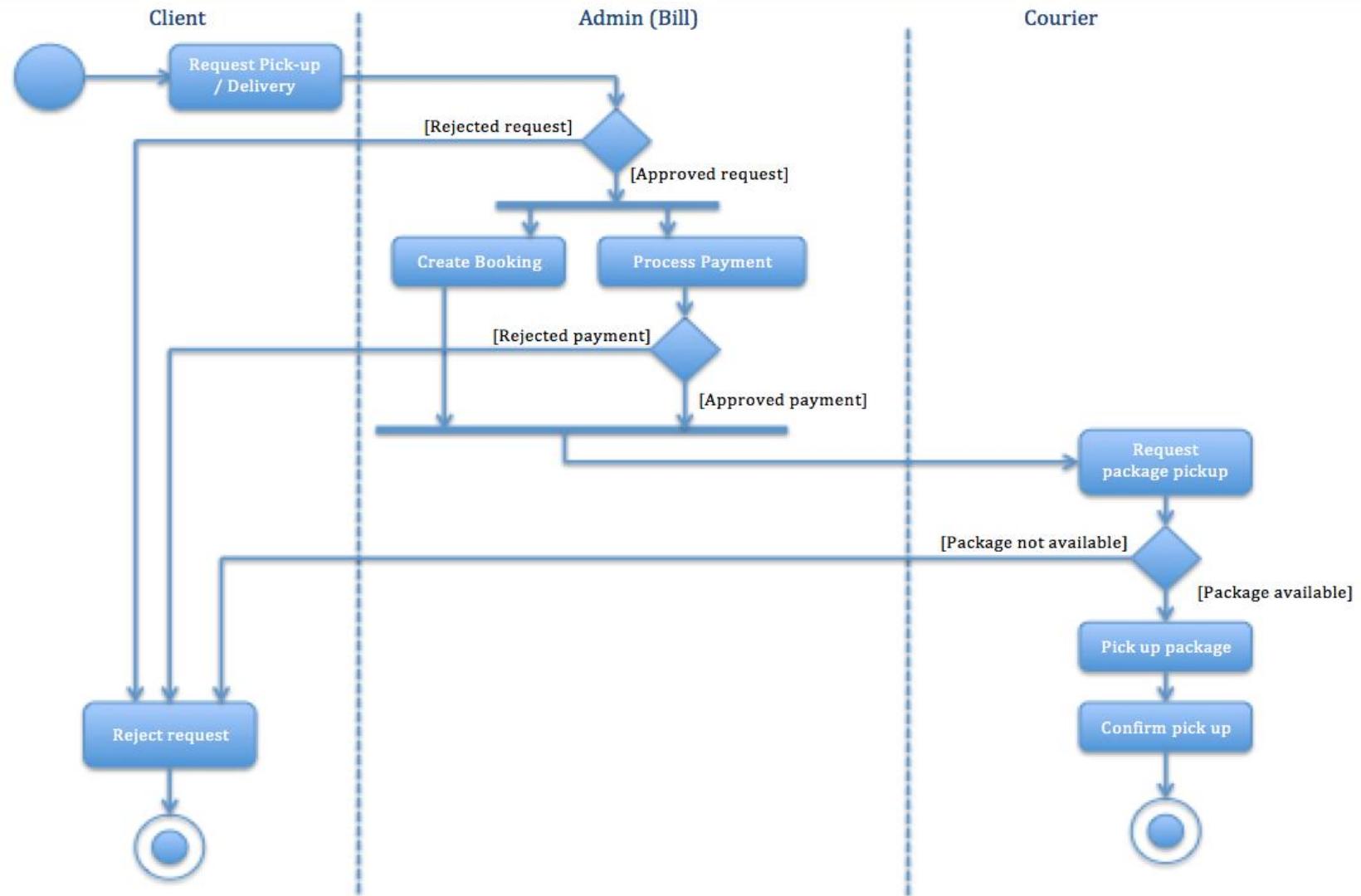
Bill Wiley – Pick-up and Delivery function

When Bill Wiley receives a request for pickup, he checks to see if the pick up is valid, and rejects it if it isn't valid. If it is valid, he enters the pickup information on a form and processes the payment. If the payment is approved, he contacts his courier staff with the pick up information, otherwise he rejects the request. When the courier picks up the package they confirm pickup. If the package is not available for pickup, the request is rejected.

Draw an activity diagram for the Pick-up and Delivery function for On the Spot Courier Services



Pick-up and Delivery function: Activity Diagram

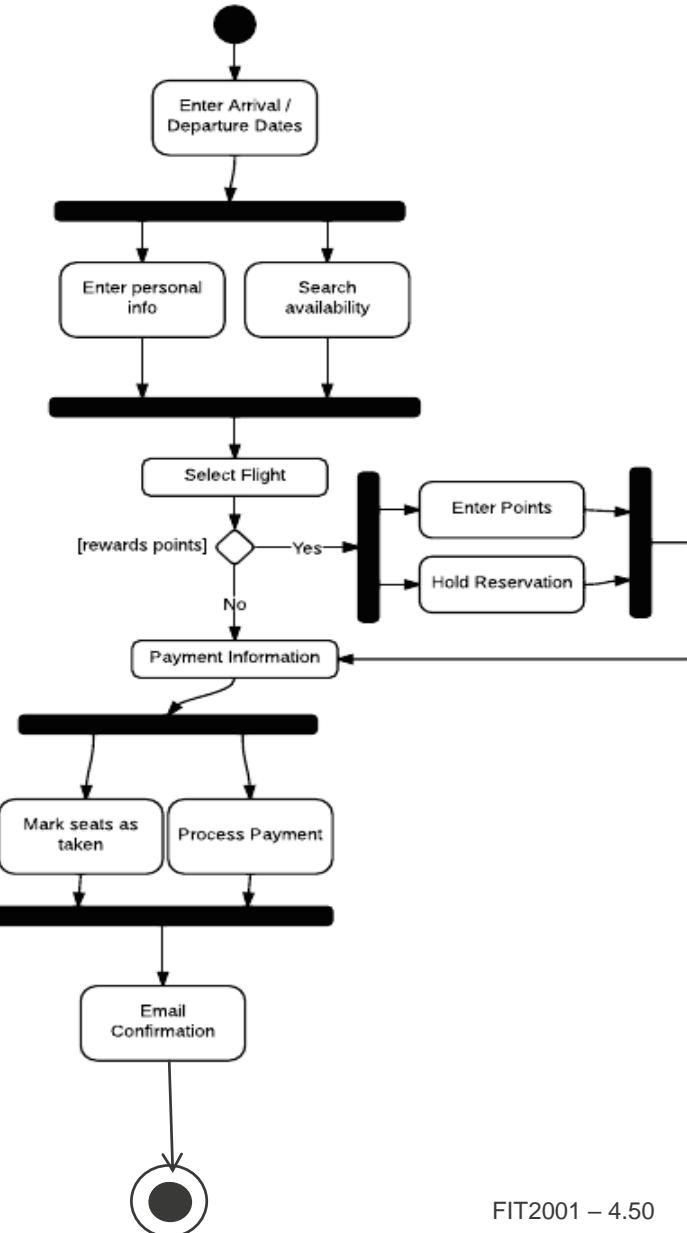


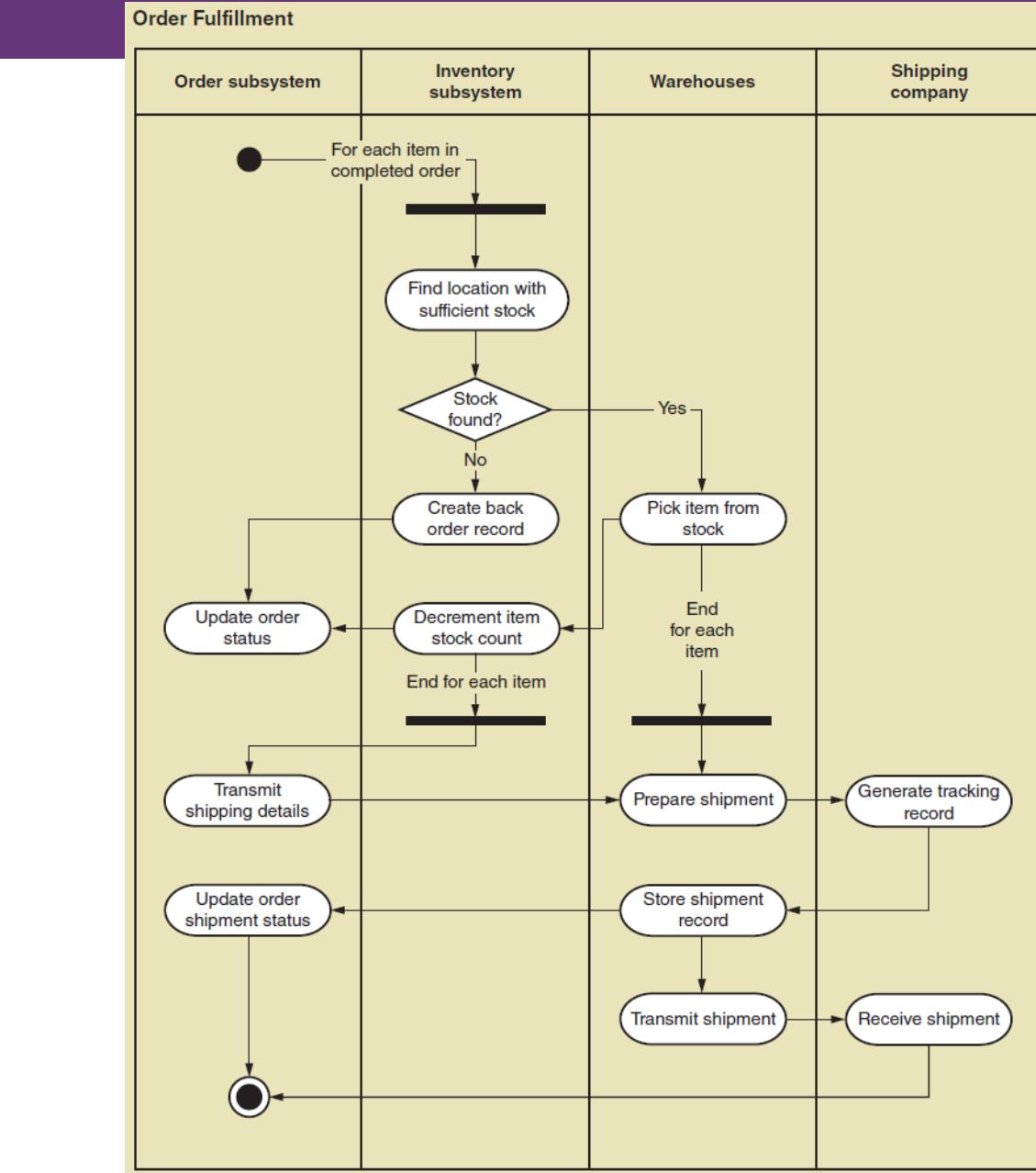


Additional examples to help you ...

Example: Airline Reservation

This example shows the process of reserving a flight. First, you enter the dates. Once you submit your desired flight plan, you can enter your personal information and at the same time the system could be searching availability. The system flow then joins back into one and you can select the specific flight on the dates you want to fly. This activity diagram gives you two different paths dependent on whether you are using reward points. After entering payment information, the system performs two processes at the same time and then sends out a confirmation email.





Workshop Preparation

- Watch Seminar 4
- Review any resources
- Assignment 1 Interview preparation

- Meet with your team regularly to prepare to complete all the requirements for Assignment 1
- Be prepared for your first interview with your client.

**Thanks for watching
See you next week**

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapters 2 (pp. 58-63) & Chapter 3 (pp. 71-73)

Resources:

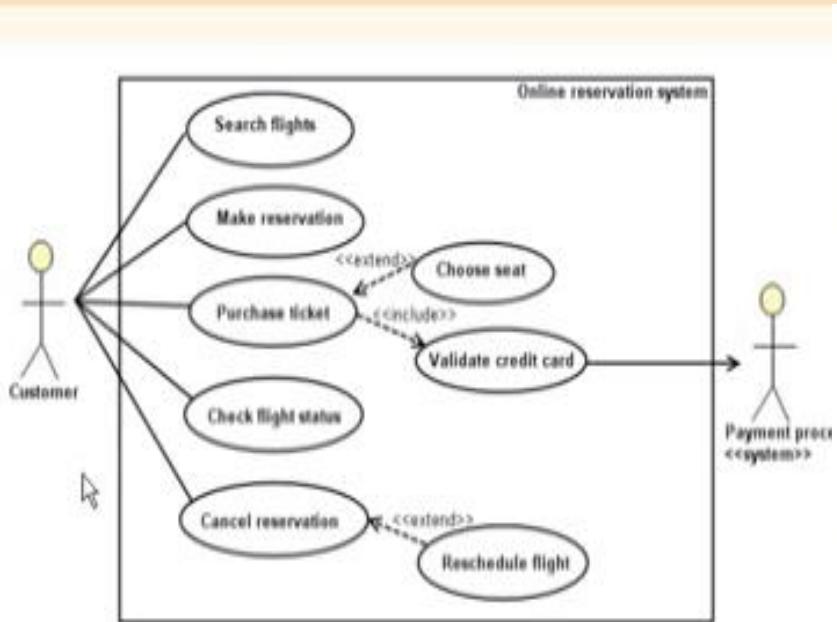
- User Stories
 - Videos - <http://vimeo.com/43601248>
<http://www.agilenutshell.com/episodes/2-userstories>
 - Article - <https://help.rallydev.com/writing-great-user-story>



FIT2001 – Systems Development

Seminar 5: Investigating and documenting system requirements
Use Case Diagrams & Use Case Descriptions

Chris Gonsalvez



Use Case Template	'Create Order' Use Case
Use Case Name	Create order
Use Case Description	Create order is the ability to request the purchase of a product
Actor	Order Creator
Pre-conditions	<ul style="list-style-type: none">Order Creator has been identified
Basic Flow	<ol style="list-style-type: none">Order Creator selects 'order product' actionSystem requests customer/product identification informationOrder Creator provides customer/product identification informationSystem requests mailing informationOrder Creator provides mailing informationSystem verifies mailing informationSystem requests order be submittedOrder Creator submits orderSystem submits product order for processingSystem confirms product order
Post-conditions	<ul style="list-style-type: none">Product order has been createdProduct is not in stockProduct has been discontinuedA customer's initial order is over \$200
Alternate Flows	

Our road map:

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering techniques
- Documenting requirements

- Documenting requirements
 - Use case diagrams
 - Use case descriptions

At the end of this topic you will:

- Understand the purpose of Use Cases
- Be able to identify Use Cases
- Be able to draw Use Case diagrams and write Use Case descriptions

Use Cases

What are Use Cases?

- Invented by Ivar Jacobson in the late 1960s ... started being used more frequently in late 1980s, not used as much in Agile
- A more traditional way of capturing functional requirements of a system - includes business rules and descriptions of system behavior
- A Use Case is a **description of all the ways an end-user wants to "use" a system**. Use Cases capture all the possible ways the user and system can interact that result in the user achieving the goal. They also capture all the things that can go wrong along the way that prevent the user from achieving the goal.

Use cases VS. User stories

User Stories	Use Cases
Focussed on the result and benefit of system functions	More granular, describes how the system will act
Minimal documentation	Detailed documentation
Small increments for getting feedback	Most done up front



- Hybrid model - User Story as an important launching off point for the more detailed Use Case
- In Industry Use Cases used more often when projects have more stringent documentation requirements (e.g. government projects).

How to identify use cases

- User goal technique
 - Most commonly used in industry
 - User goals for interacting with the new system are identified
 - Simple and effective
- Event decomposition technique
 - Business events are identified

User goal technique.1

The following steps help identify use cases:

1. Identify all the users
2. Classify them
 - Functional roles – Sales, Marketing
 - Organisational roles – Executive, Management, Operational

User goal technique.2

Steps to identify use cases (cont.):

3. Interview each type of user to create a preliminary list of use cases

- Find out a list of specific goals they will have when using the new system
- Express them in VERB-NOUN format (e.g. Add customer)
- Interview and ask them to describe the tasks the system can help them with
- Probe further to refine the tasks into specific user goals, “I need to Ship items, Track a shipment, Create a return”

4. Look for duplicates ... resolve inconsistencies

5. Look for users with same needs

6. Review with users

User goal technique: Example

User	User goal and resulting use case
Potential customer	Search for item Fill shopping cart View product rating and comments
Marketing manager	Add/update product information Add/update promotion Produce sales history report
Shipping personnel	Ship items Track shipment Create item return

- Speak to potential customers through focus groups
- Interview users from shipping and marketing departments

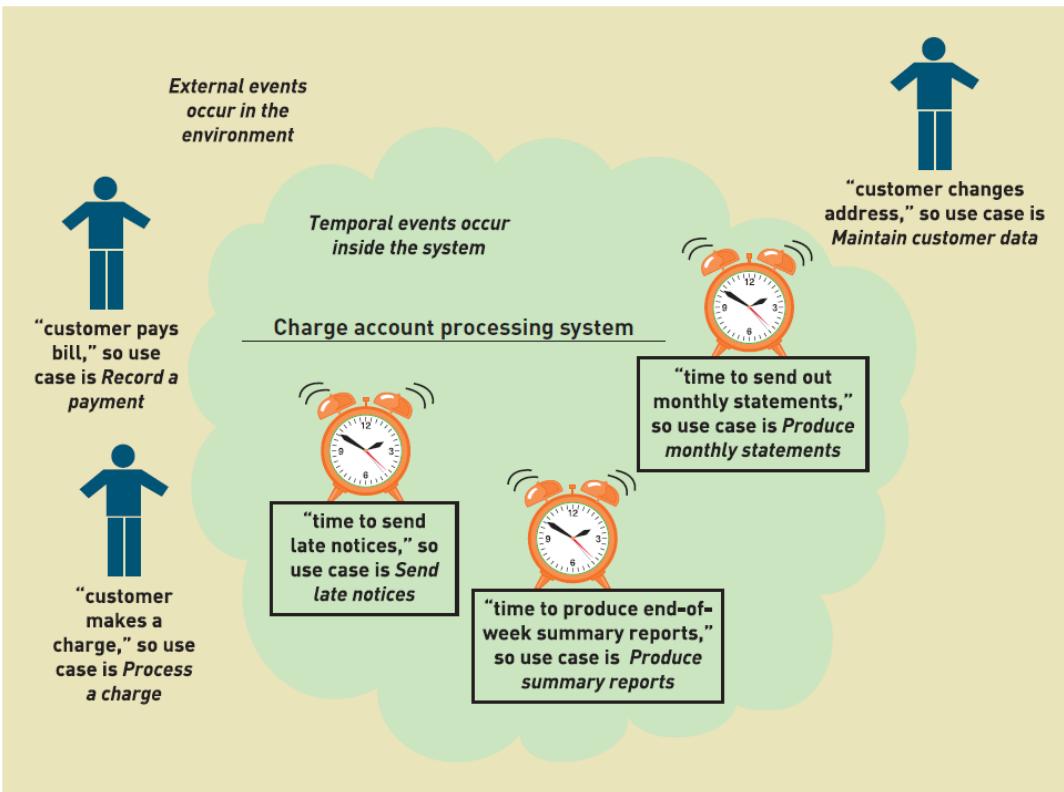
Identify Use Cases

- Event Decomposition Technique

- More comprehensive and complete technique
- Identify the events that occur to which the system must respond.
 - Event— something that occurs at a specific time and place, can be described, and should be remembered by the system
- For each event, name a use case (verb-noun) that describes what the system does when the event occurs

Types of Events – External Events

An event that occurs outside the system,
usually initiated by an external agent or actor



Checklist:

Wants something resulting in a transaction - Customer buys a product

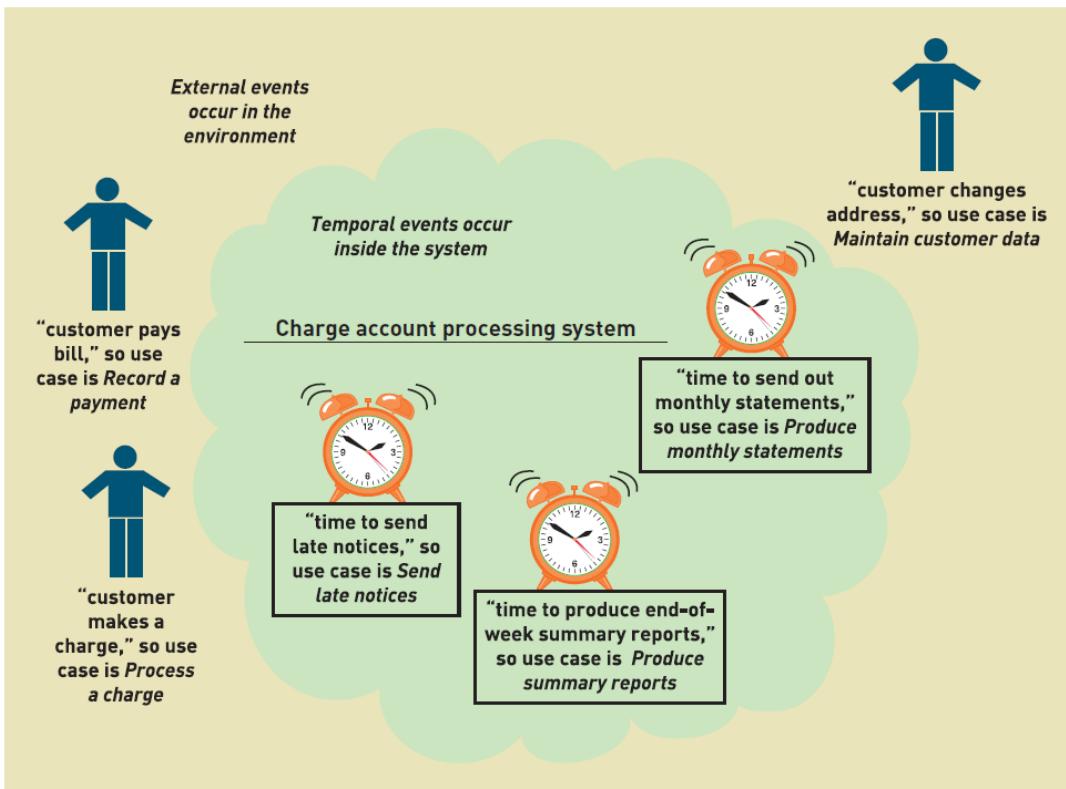
Wants some information - Customer wants to know product details

External data changed and needs to be updated - Customer has new address and phone

Management wants some information - Sales manager wants update on production plans

Types of Events – Temporal Events

An event that occurs as a result of reaching a point in time



Checklist:

Internal outputs needed at points in time

- Management reports (summary or exception)
- Operational reports (detailed transactions)
- Internal statements and documents (including payroll)

External outputs needed at points of time

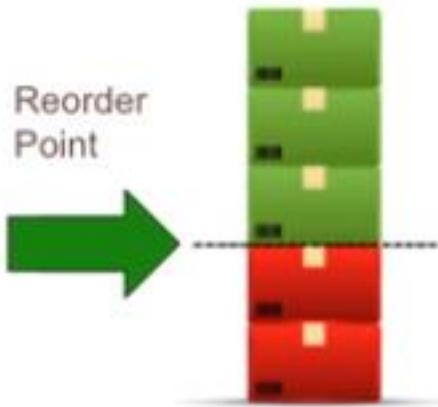
- Statements, status reports, bills, reminders

Types of Events – State Events

- an event that occurs when something happens inside the system that triggers some process

Eg. reorder point is reached for inventory item

Reorder Point (ROP)



Find the actual event that affects the system



Customer thinks about getting a new shirt



Customer drives to the mall



Customer tries on a shirt at Sears



Customer goes to Walmart



Customer tries on a shirt at Walmart



Customer buys a shirt
(the event that directly affects the system!)

Tracing a sequence of transactions resulting in many events



Customer requests a catalog



Customer wants to check item availability



Customer places an order



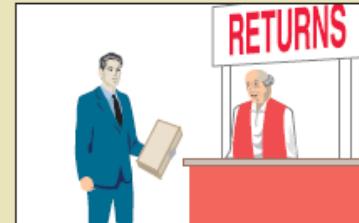
Customer changes or cancels an order



Customer wants to check order status



Customer updates account information

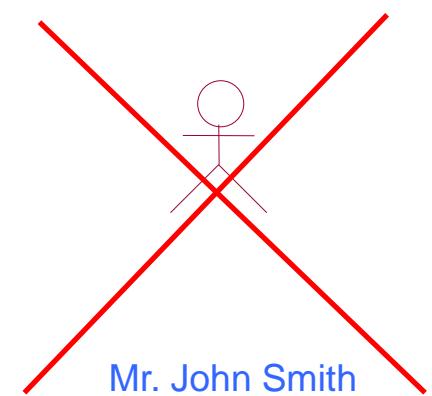
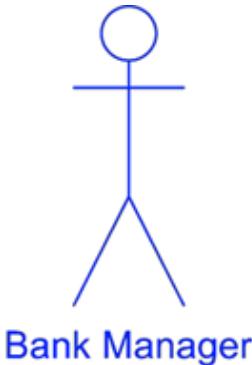


Customer returns the item

Use Case Diagram Elements

ACTOR

- A person or another software entity (such as a system timer) that initiates the functionality provided by a Use Case
- A coherent set of roles that users of Use Cases play when interacting with Use Cases
 - Roles not users or people
 - User may have more than one role
- Primary vs. Secondary (next slide)



Use Case Diagram Elements

Primary vs. Secondary ACTOR

- **Primary Actors:** The Actor(s) using the system to achieve a goal. The Use Case documents the interactions between the system and the actors to achieve the goal of the primary actor.
- **Secondary Actors:** Actors that the system needs assistance from to achieve the primary actor's goal.
- Example: A bank loan officer wants to review a loan application from a customer, and part of the process involves a real-time credit rating check.
- Use Case Name: Review Loan Application
- Primary Actor: Loan Officer
- Secondary Actors: Credit Rating System

Alistair Cockburn's book "Writing Effective Use Cases"

Use Case Diagram Elements

USE CASE

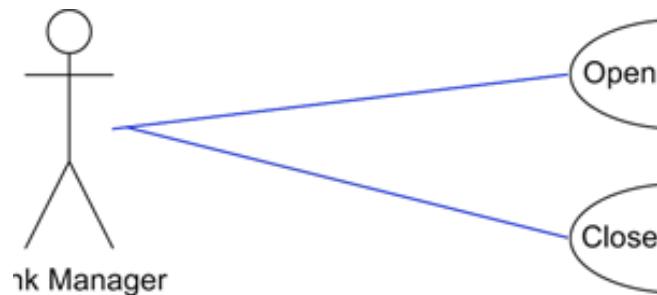
- A Use Case is an interaction between the system and a person or another system to achieve a result
- A required “bit” of functionality
- It yields an observable result of value to an actor
- Typically named with a verb then a noun eg. View Timetable



Use Case Diagram Elements

ASSOCIATION

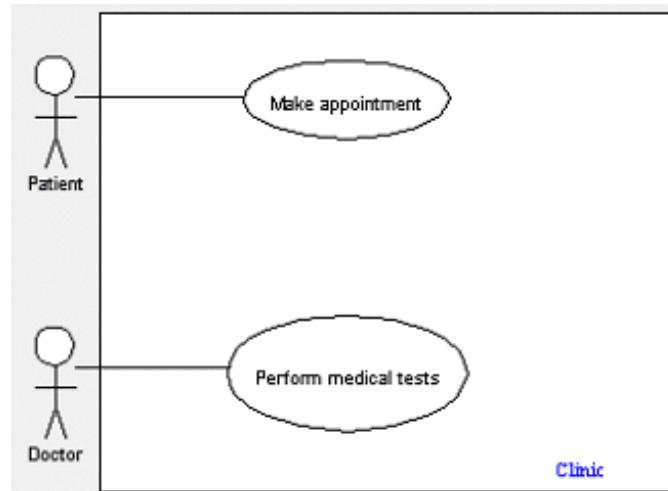
- The **Association** line indicates that a particular Actor makes use of the functionality provided by a particular **Use Case**.



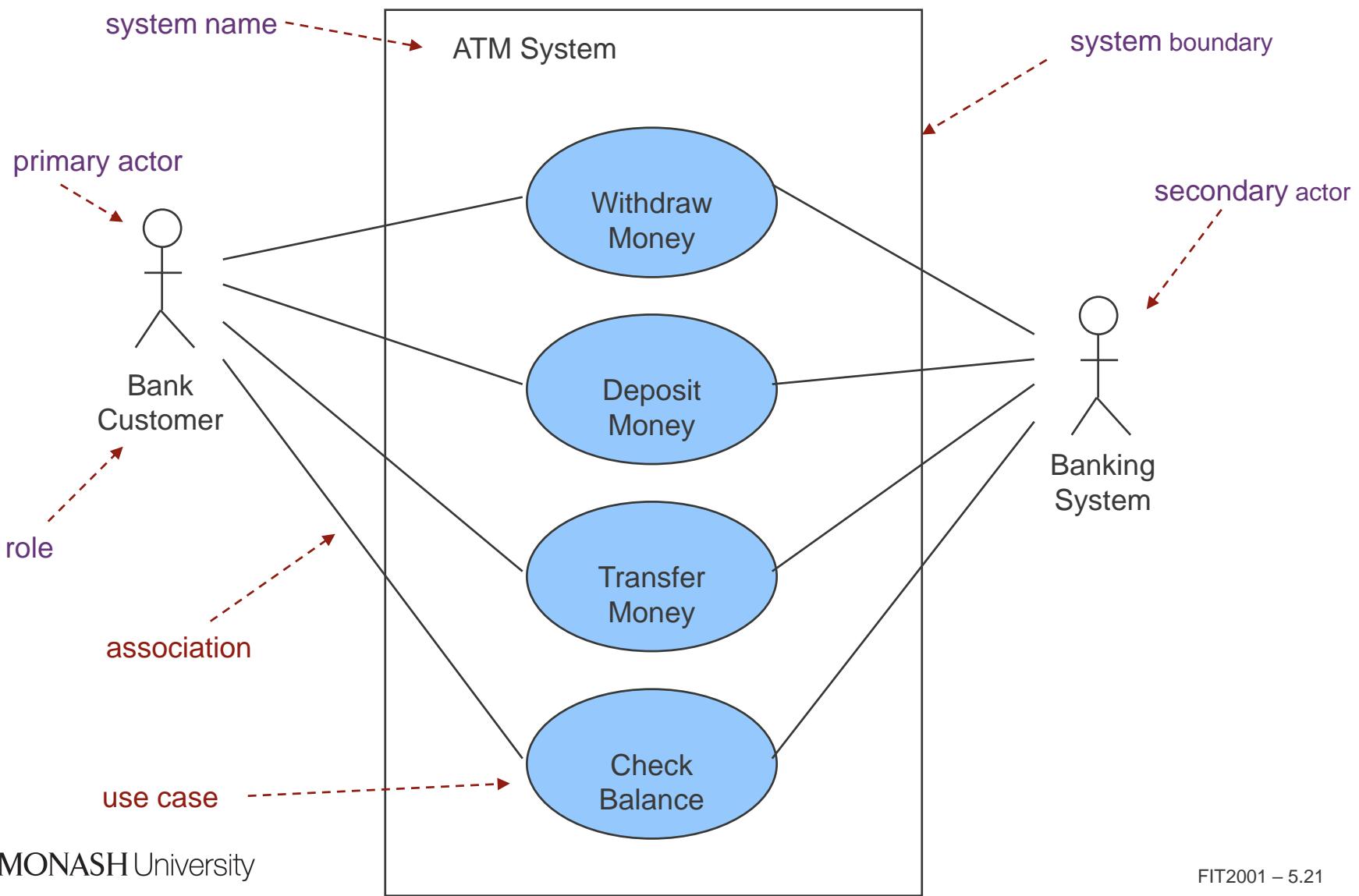
Use Case Diagram Elements

SYSTEM BOUNDARY

- Defines the scope of what the system will be.
- Shown as a rectangle spanning all the use cases.



Example



Use Case Diagram Elements

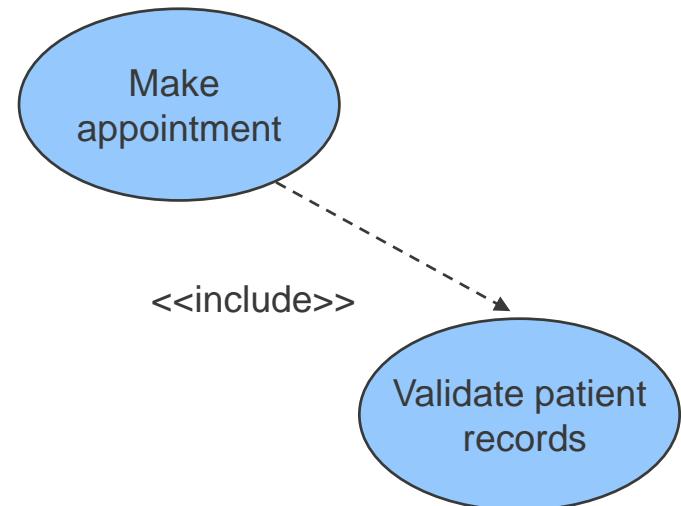
USE CASE RELATIONSHIPS

- A relationship between two use cases is basically a dependency between the two use cases.
- They can be:
 - **Include** - models encapsulated behaviors that can be inserted into a use case and possibly reused across multiple use cases.
 - **Extend** - models significant extensions and behaviors that can occur as additions to the use case model.
 - **Generalisation** models conceptual similarity between use cases.

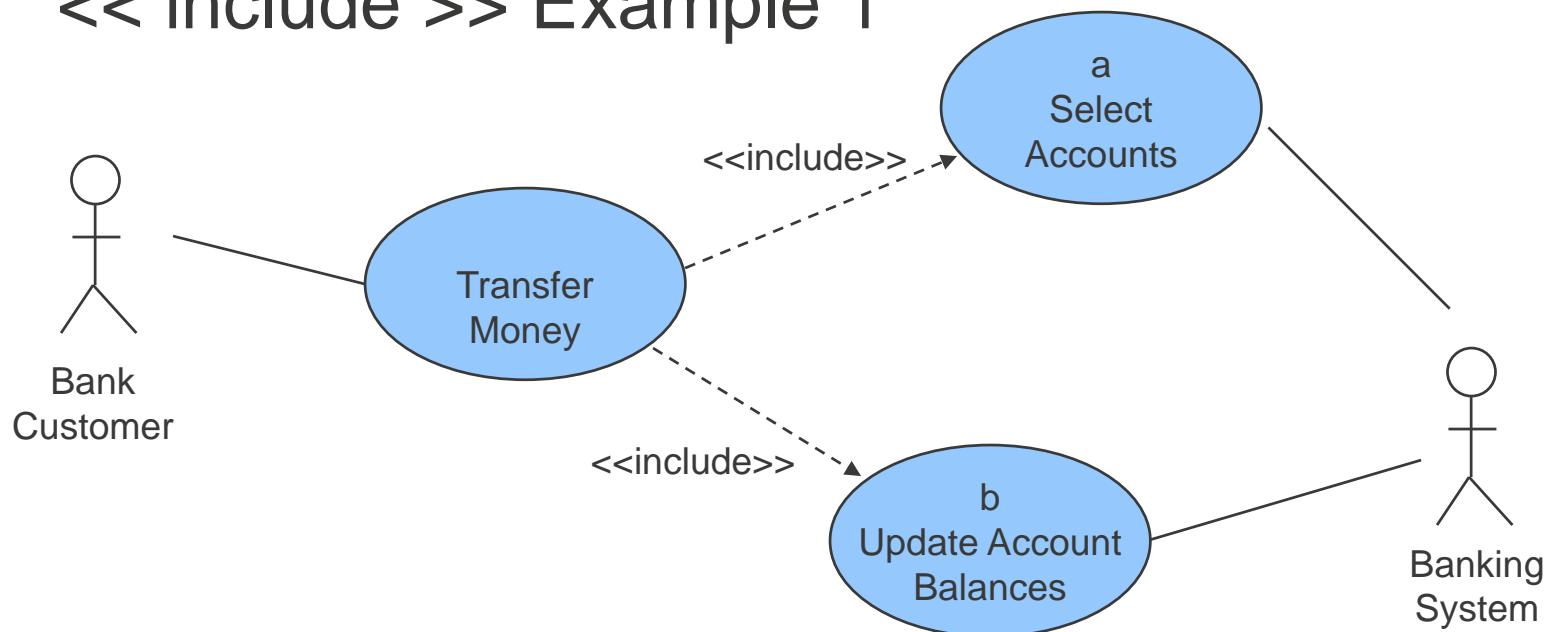
Use Case Diagram Elements

RELATIONSHIP << include >>

- Captures commonality among use cases
- Re-usability helps avoids repetition (saves time and money) – manages redundancy
- An include relationship is depicted with a directed arrow having a dotted shaft.
- Eg. The "Validate patient records" use case is contained within the "Make appointment" use case. Hence, whenever the "Make appointment" use case executes, the business steps defined in the "Validate patient records" use case are also executed.



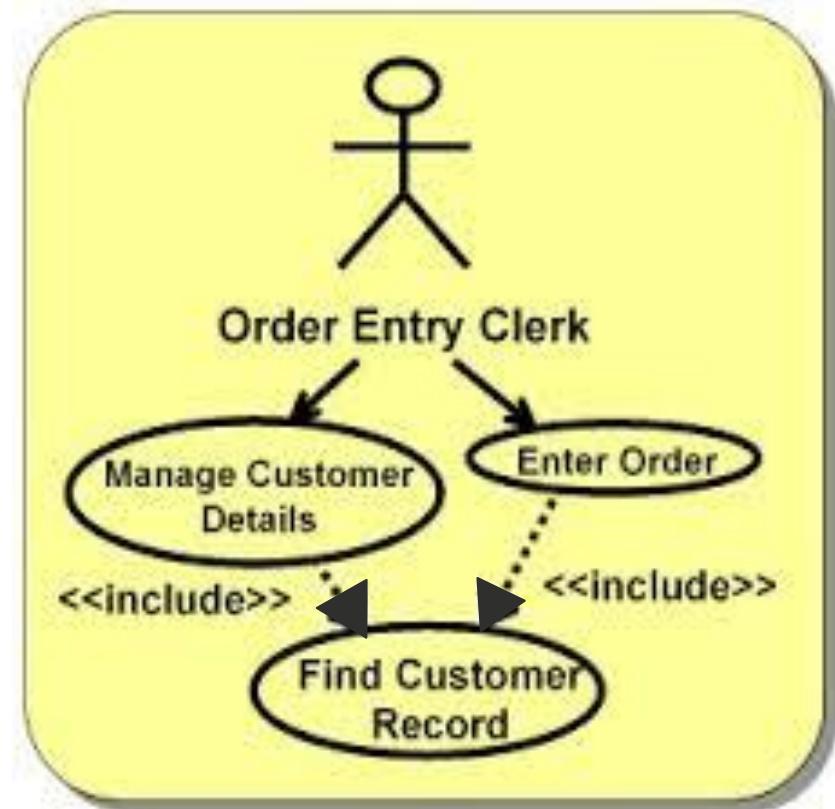
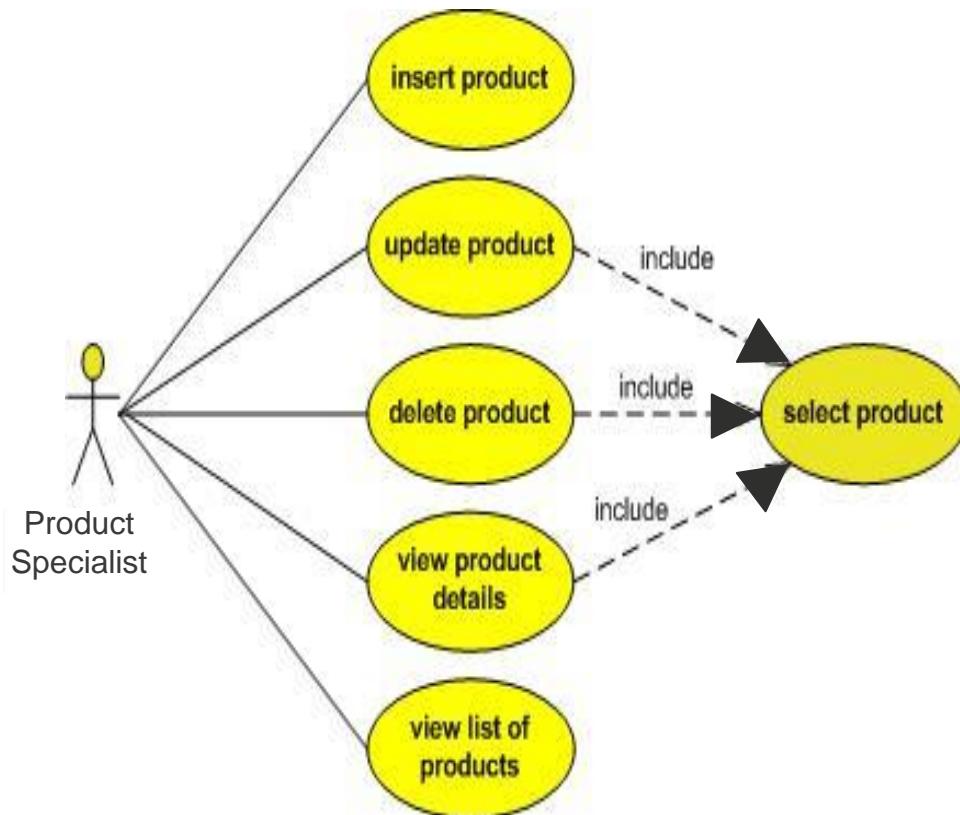
<< include >> Example 1



This is an **include dependency**.

It indicates that use case a and b are “included” in the base use case and will be invoked as part of ‘Transfer money’. They have been separated because of they are used by other use cases (reuse).

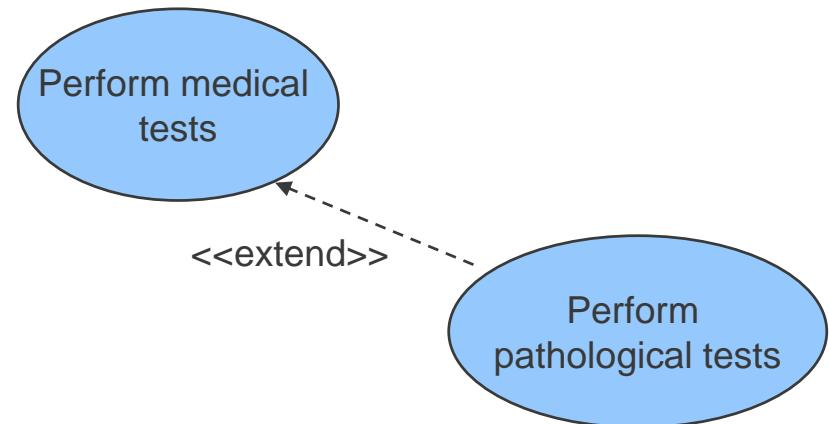
<< include >> More examples



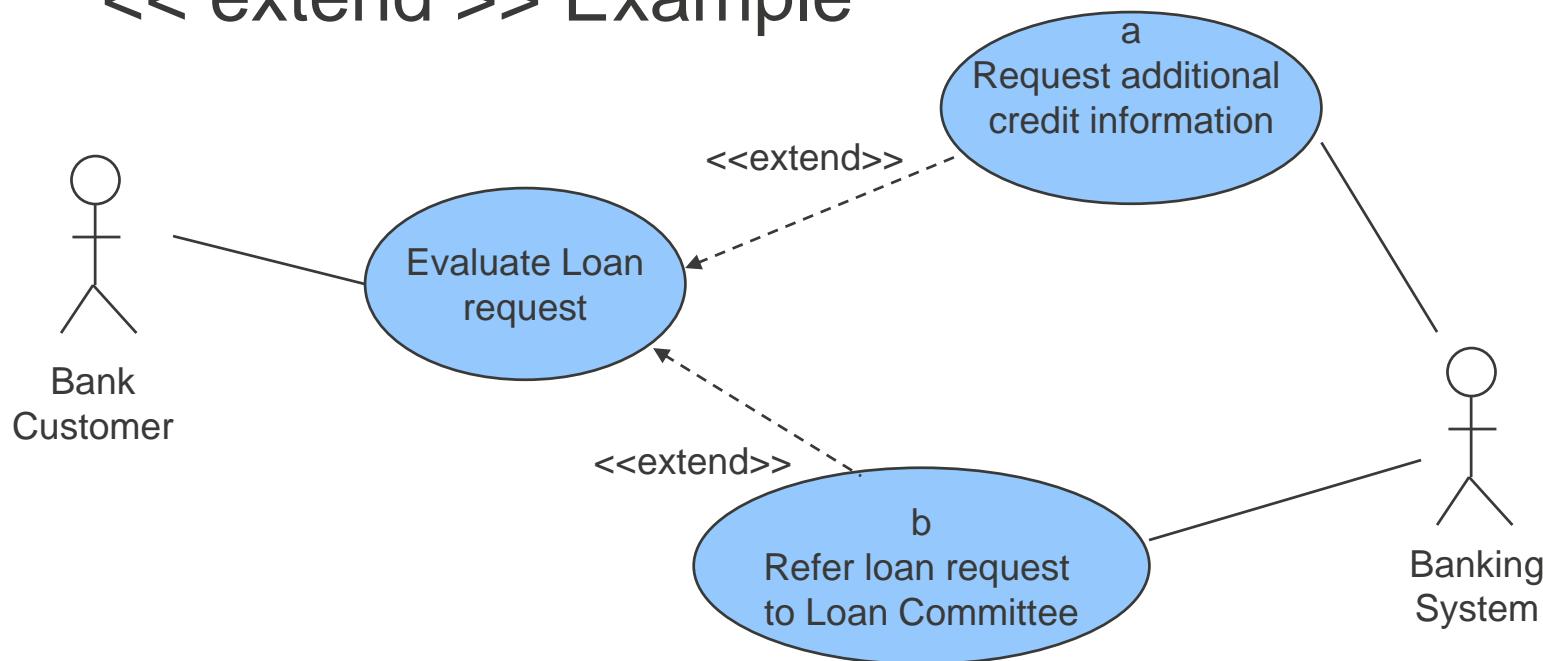
Use Case Diagram Elements

RELATIONSHIP << extend >>

- The child use case adds to the existing functionality and characteristics of the parent use case (optional/exception behaviour)
- The parent case can function without the child case.
- An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow.
- Eg. The "Perform pathological tests" (child) use case extends the functionality of the "Perform medical tests" (parent) use case. It is used only when required.



<< extend >> Example



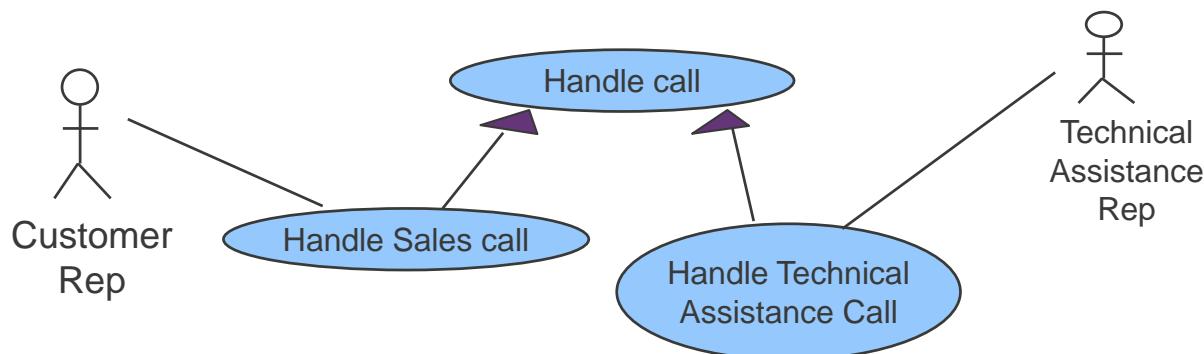
This is an **extend dependency**.

It indicates that use case a and b may or may not be invoked.

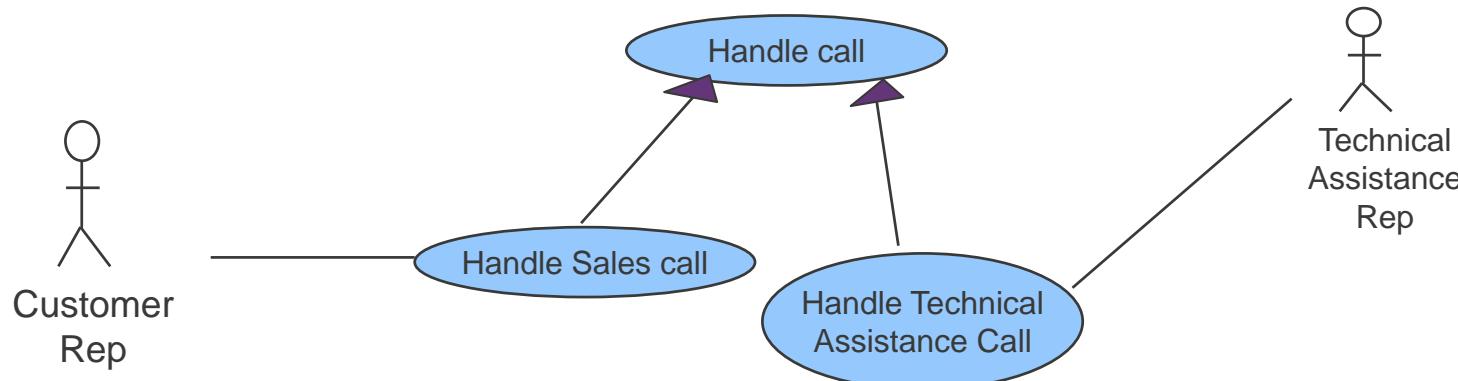
Use Case Diagram Elements

RELATIONSHIP – Generalisation

- If use cases have common behaviour, structure and similarities, their common parts can be factored out into a parent use case to optimise the model – the parent case is often abstract ... not ever directly called by an actor.
- A generalization relationship between parent and child use cases is one in which the child is a more specialised form of the parent use case. The child inherits the behaviors of its parents and adds new behaviors, and specializes in behaviors inherited from the parent.
- A generalised use cases is depicted drawing an open, solid arrow from the specialized (child) use cases to the general (parent) use case.



Generalisation Example 1



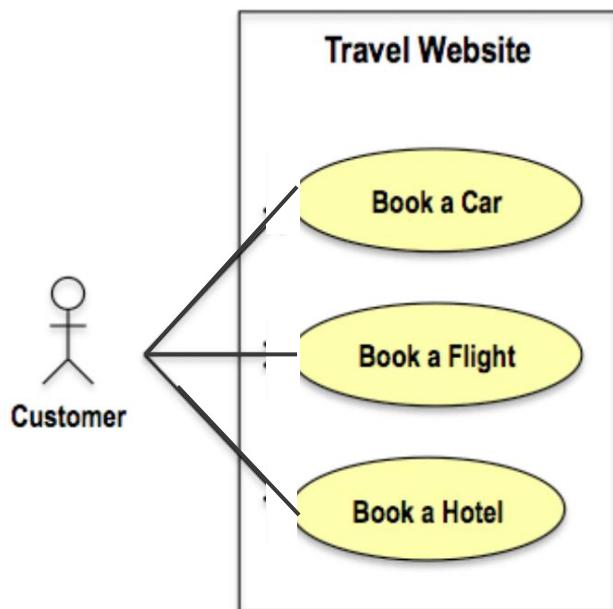
- The 'Handle Sale Call' or 'Handle Technical assistance call' substitutes 'Handle Call'. They inherit the behaviour of 'Handle Call' and add their own behaviour. 'Handle call' is an abstract use case.

'Handle call' – behaviour ²₉ inherited by child use cases

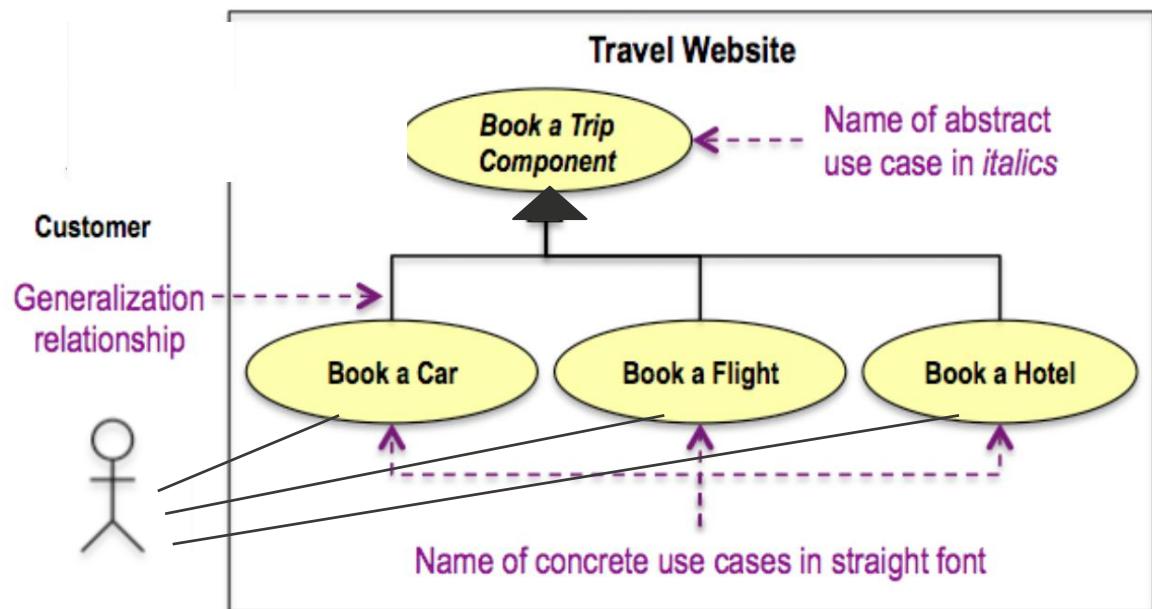
- 1.Transfer call to available representative
- 2.Mark representative as busy
- 3.Start record call
- 4.Stop record call
- 5.Log call details
- 6.Mark representative as free

Generalisation Example 2

Start: Model concrete use cases to represent different actor goals.



Optimize: Generalize the concrete use cases to a generalization use case to represent the core booking process. Generalize the associations to an association on the generalization use case, so the specialization use cases inherit it.



Note: From a Customer's point of view, nothing has changed. Any Customer still sees only the concrete (public) specialization use cases, not the abstract (private) generalization use case. This is about **model optimization** and nothing else.

Ref: <http://www.batimes.com/articles/generalization-and-use-case-models-part-2.html>

What is a Use Case Description?

“The use case description (narrative, specification) provides the details of the functionality that the system will support and describes how the actors will use the system in order to obtain a specific result of value.”

Use Case Template	'Create Order' Use Case
Use Case Name	Create order
Use Case Description	Create order is the ability to request the purchase of a product
Actor	Order Creator
Pre-conditions	<ul style="list-style-type: none">Order Creator has been identified
Basic Flow	<ol style="list-style-type: none">Order Creator selects 'order product' actionSystem requests customer/product identification informationOrder Creator provides customer/product identification informationSystem requests mailing informationOrder Creator provides mailing informationSystem verifies mailing informationSystem requests order be submittedOrder Creator submits orderSystem submits product order for processingSystem confirms product order
Post-conditions	<ul style="list-style-type: none">Product order has been created
Alternate Flows	<ul style="list-style-type: none">Product is not in stockProduct has been discontinuedA customer's initial order is over \$200

Use Case Description – Template *

Number	<i>Unique use case number</i>	
Name	<i>Brief noun-verb phrase</i>	
Summary	<i>Brief summary of use case major actions</i>	
Priority	<i>1-5 (1 = lowest priority, 5 = highest priority)</i>	
Preconditions	<i>What needs to be true before use case “executes”</i>	
Postconditions	<i>What will be true after the use case successfully “executes”</i>	
Primary Actor(s)	<i>Primary actor name(s)</i>	
Secondary Actor(s)	<i>Secondary actor name(s)</i>	
Trigger	<i>The action that causes this use case to begin</i>	
Main Scenario	Step	Action
	Step #	<i>This is the “main success scenario” or “happy path.”</i>
	...	<i>Description of steps in successful use case “execution”</i>
	...	<i>This should be in a “system-user-system, etc.” format.</i>
Extensions	Step	Branching Action
	Step #	<i>Alternative paths that the use case may take</i>
Open Issues	<i>Issue #</i>	<i>Issues regarding the use case that need resolution</i>

Use Case Description Example

Number	1
Name	Withdraw Money
Summary	User withdraws money from one of his/her accounts
Priority	5
Preconditions	User has logged into ATM
Postconditions	User has withdrawn money and received a receipt
Primary Actor(s)	Bank Customer
Secondary Actor(s)	Customer Accounts Database

Continued ...

Use Case Description - Example

Trigger	User has chosen to withdraw money	
Main Scenario	Step	Action
	1	System displays account types
	2	User chooses account type
	3	System asks for amount to withdraw
	4	User enters amount
	5	System debits user's account and dispenses money
	6	User removes money
	7	System prints and dispenses receipt
	8	User removes receipt
	9	System displays closing message and dispenses user's ATM card
	11	User removes card
	10	System displays welcome message
Extensions	Step	Branching Action
	5a	System notifies user that account funds are insufficient
	5b	System gives current account balance
	5c	System exits option
Open Issues	1	Should the system ask if the user wants to see the balance?



Writing Use Case Descriptions

Don't focus on perfection – be productive

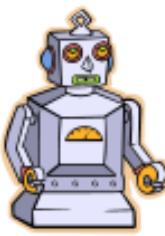
- Not about getting it right the first time
- An iterative process where you work and refine.

1. Identify the Actors

- Any "object" or person that has behavior associated with it. Users and Systems.



Humans



Machines



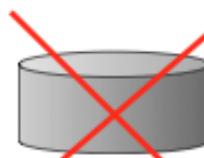
External systems



Sensors



Organizational Units

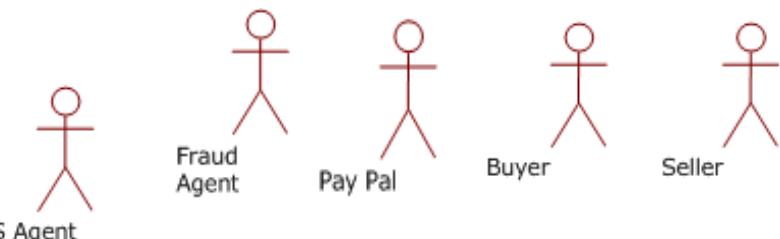


Database



Printer

- Eg. Who are the actors that interact with ebay



2. Identify the Goal

- From the high-level scenario
- By brainstorming
- By asking

“What does this Actor want to do?”

“What will the actor use the system for?”

“Will the actor create, store, change, remove or read information in the system?”



3. Define the Pre-conditions

- Something that must happen before the Use Case can start; something that must be in place before the Use Case can start
- Identify Pre-Conditions by asking:
“What must be in place for the Use Case to begin?”
“How do you know you need to do this?”
- Examples:
User account exists, User has enough money in their account, There is enough disk space

4. Define the Post-conditions

- The result, or successful outcome of the Use Case
- Identify the Post-Conditions by asking:
“What is the successful result of this process or Use Case?
- Examples:
Money was transferred to the user account,
User is logged in, The file is saved

5. Describe the ‘Main Flow’.1

- Primary Scenario / Happy Day Scenario
The simplest sequence – everything goes right
Starts with pre-conditions and ends with post conditions
- Describe the Main Flow by asking:
“What must happen to achieve the goal / outcome?”
“What does the actor need to do next?”
“What might happen next?”
“What do you need to do to get from the trigger to the outcome?”

5. Describe the ‘Main Flow’.

- Example:
 1. Admin enters Course name, code & description
 2. System validates Course code
 3. System adds course to the database and shows confirmation message
- Identify opportunities for reuse

5. Describe the ‘Main Flow’.

Guidelines for effective writing

- Only one side (actor or system) should do something in a single step
- Each step should lead to some progress
 - NOT ‘User clicks Return key’
- Use simple sentences
 - Actor asks for money
 - System asks for amount
 - Actor gives amount
 - System gives the money
 - NOT *Get the amount from the user and give him the money*

6. Describe the Alternate Flows.1

- Describes the variations/exceptions to the Main Flow
 - An exception or error flow to any line item in your basic flow
 - An additional flow, not necessarily error based, but a flow that COULD happen
- Discover alternate flows by asking:
“What might affect this Use Case?
“What could go wrong?”
Listen for “sometimes”, “maybe”, and “it depends”

6. Describe the Alternate Flows.2

- Examples:

- While a customer places an order, their credit card failed
- While a customer places an order, their user session times out
- While a customer uses an ATM machine, the machine runs out of receipts and needs to warn the customer

How do you know when you have identified all your Use Cases?

- When all actors are specified
- When every functional requirement has a use case which satisfies it
- The CRUD (Create, Read, Update, Delete) technique can help validate, refine or cross-check use cases

Use Cases and CRUD Technique

For each type of data (data entity or domain class), verify that a use case has been identified that creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.

Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

ON THE SPOT COURIER SERVICES

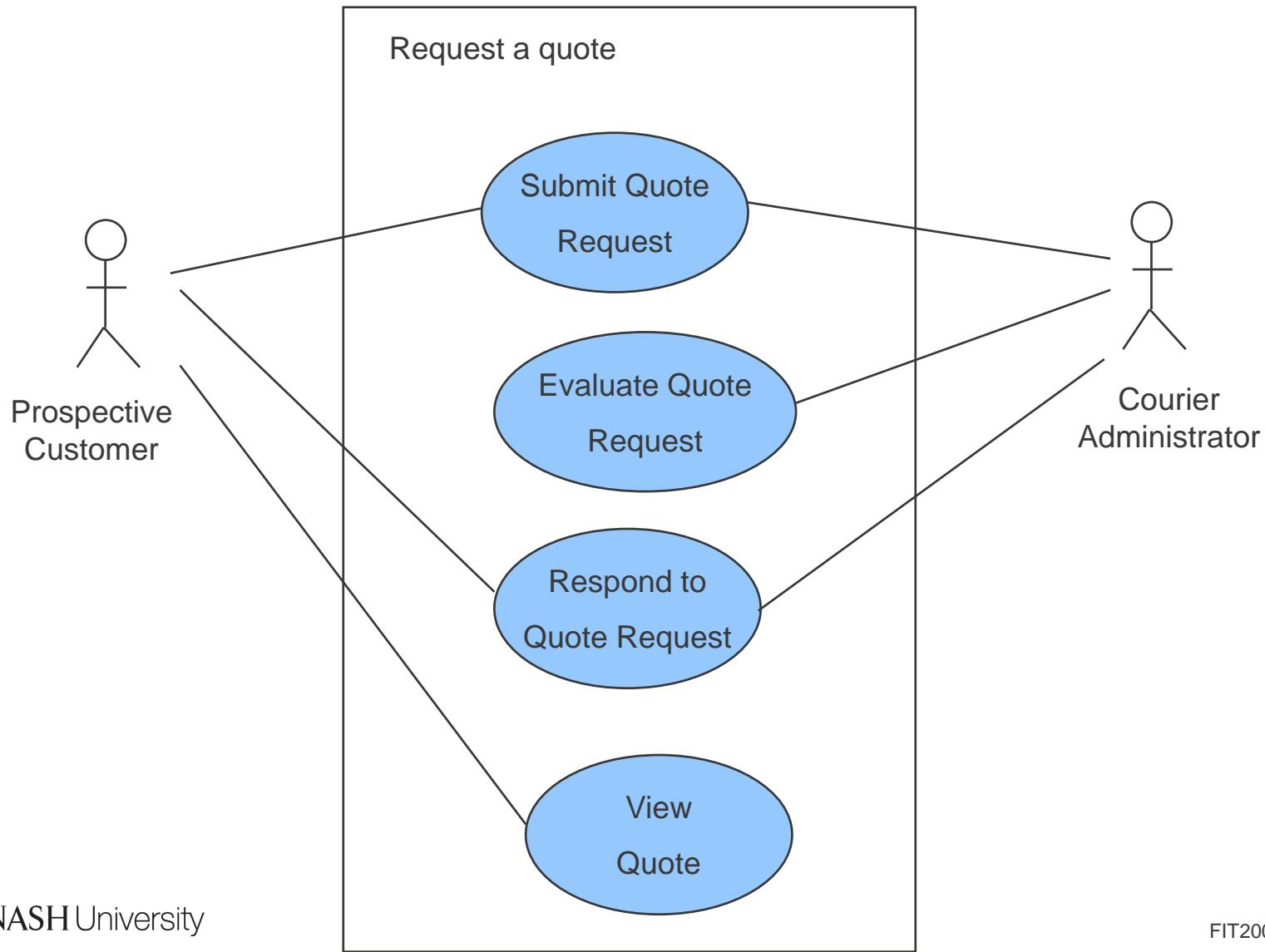
Request a Quote

We would like the customer to enter in the pick up and delivery address, the package details and the date and time of pickup and delivery. If the package does not meet our size requirements or we are unable to do the pick up or delivery on the specified dates, we let them know that we cannot provide a quote, otherwise we provide a quote.

Create a Use Cases for the Request a Quote function for On the Spot Courier Services



Use Case modelling - Example





Workshop Preparation

- Watch Seminar 5

**Thanks for watching
See you next week**

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 3 (pp. 73-88)

Use Case resources:

☞ Getting started with Use Case Modelling:

<http://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf>

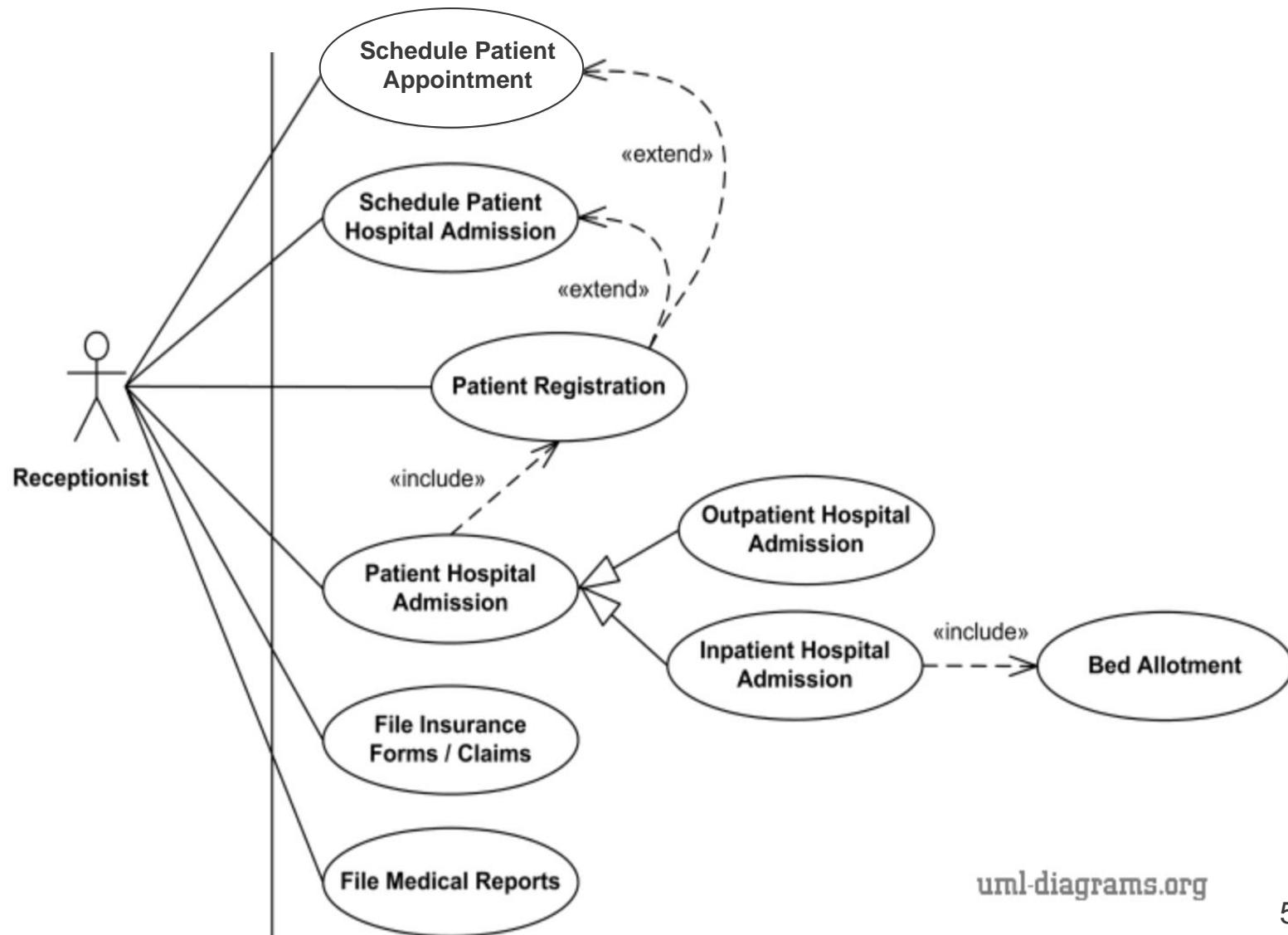


Additional example to help you ...

Use case Example description

- **Purpose:** *Describe major services (functionality) provided by a hospital's reception.*
- **Hospital Reception** subsystem or module supports some of the many job duties of hospital receptionist. Receptionist schedules patient's appointments and admission to the hospital, collects information from patient upon patient's arrival and/or by phone. For the patient that will stay in the hospital ("inpatient") she or he should have a bed allotted in a ward. Receptionists might also receive patient's payments, record them in a database and provide receipts, file insurance claims and medical reports.

Use case diagram example:



Use case Example description

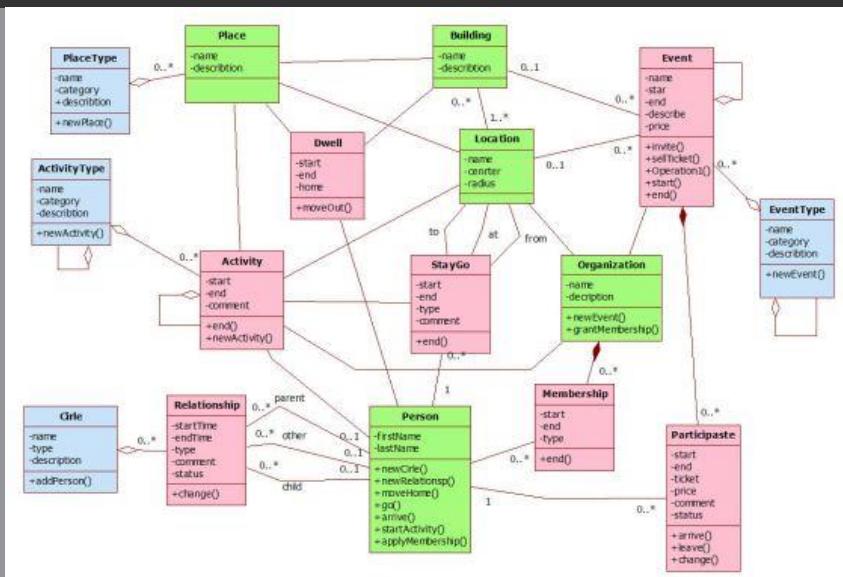
- **Purpose:** Assess an *insurance claim*
- A customer puts in an insurance claim request with the option to include 3 repair/replacement quotes. The insurance claims officer first checks to see if the customer has valid insurance. Once that is done the claim is assessed to ensure that it is valid, and the claim is either rejected or accepted, and the customer is informed of the assessment outcome. If the claim is accepted, and the customer has attached quotes, they are sent to the quotes assessment team to review. The assessment team selects the most suitable quote and informs the insurance claims officer of their selection. The insurance claims officer then contacts the selected quote company to start the repair/replacement process, and informs the customer that the process has started with the selected company. If the customer does not include quotes, the assessment team will select a suitable company to do the repairs/replacement.



FIT2001 – Systems Development

Seminar 6: Domain and Class Modelling

Chris Gonsalvez



Our road map

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Some Requirements gathering and documentation techniques
 - User stories, Use Cases, Activity diagrams

Requirements gathering techniques

- Domain and Class Modelling



At the end of this topic you will be able to:

- Understand the concepts of objects, classes, attributes and associations
- Identify classes of objects ('things') and associations from a problem statement
- Create a domain model class diagram (class diagram).

Seminar structure:

Domain modelling steps:

1. Identifying ‘things’ - classes
2. Identify simple relationship among ‘things’ – associations
3. Identify multiplicity of associations
4. Identify association classes - many to many associations
5. Identify characteristics of ‘things’- attributes
6. Identify complex relationships among ‘things’ – Generalisation / Specialisation
7. Identify complex relationships among ‘things’ – Whole-Part
8. Example
9. Evolution of UML Models as system is developed

Domain modelling

- A Domain class model is a *graphical representation of relationships between “**THINGS**”*
- These are the “**THINGS**” (tangible/intangible – people, events) in a system that analysts need to keep information about in the business problem domain
- In object oriented approaches generally, the domain model is a UML class diagram which represents ‘**THINGS**’ as a **class of objects**

Note: The Domain model example (shown in the seminar) has been removed since it is the Design Class Diagram. Please refer to slide 6.44 for the Domain model example.

‘Things’ and system requirements

- Analysts can understand system requirements by identifying **THINGS**
 - that people deal with when they do their work
 - about which information needs to be stored
- When analysts understand and model ‘**THINGS**’, combined with the **Create Read Update Delete – CRUD concept** – a lot of the system functionality is defined.
- Two techniques are available to identify **a list of things**
 - Brainstorming
 - List of nouns

Step 1. Identify classes – ‘Things’

- Find out “**THINGS**” which are important to your stakeholders and users
 1. Identify a user and a set of user stories (business functions)
 2. Brainstorm with the user to identify THINGS involved when carrying out the user story – THINGS about which information should be captured
 3. Ask questions to identify things:
 - Do the THINGS refer to any locations? (e.g. Warehouse)
 - Do the THINGS refer to any roles involved? (e.g. Customer)
 - Do the THINGS involve any events? (e.g. Shipment)
 - Do the THINGS refer to any items of interest (e.g. Package)

Find THINGS: List of nouns

Noun is a person, place or thing

- Identify all nouns about the system using
 - information you have gathered from interviews and workshops with the user
 - as current procedures you have observed
 - as current reports or forms
- Refine the list of THINGS, and record assumptions or issues to explore

EXAMPLE

Description:

- When customers put in a quote request, they first enter their email address. We then request the dates of pickup and delivery, the pick up and delivery addresses and, the details of all the packages. Once we know about packages, we then provide a quote

Customer

Quote
Request

Package

Quote

Identifying nouns as ‘things’ from a Use Case Description

Main Success Scenario (or Basic Flow)

1. Customer arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale**.
3. **Cashier** enters **item identifier**.
4. System records **sale line item** and presents **item description**, **price**, and running **total**. Price calculated from a set of price rules.
Cashier repeats steps 2-3 until indicates done.
5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment**.
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions**) and **Inventory** systems (to update inventory).
9. System presents **receipt**.
10. Customer leaves with receipt and goods (if any).

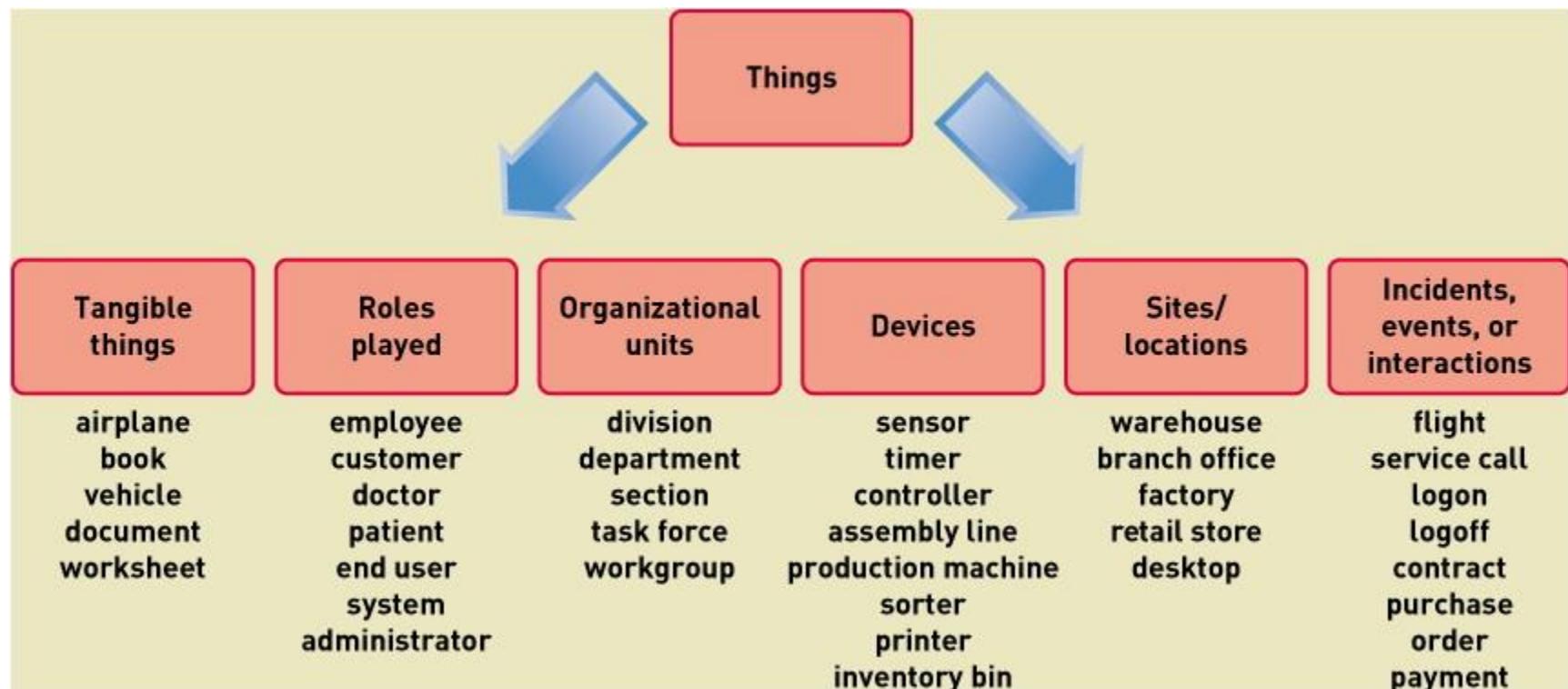
Extensions (or Alternative Flows):

7a. Paying by cash:

1. Cashier enters the cash **amount tendered**.
2. System presents the **balance due**, and releases the **cash drawer**.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.



Types of “THINGS” - Classes



Consider a range of categories in your problem space.1

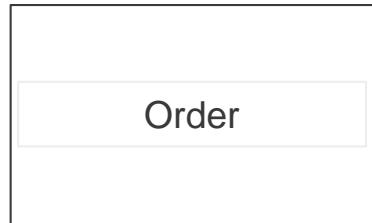
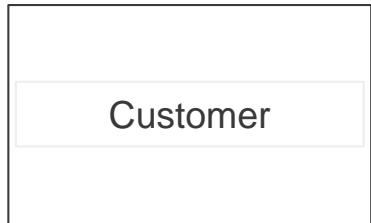
Conceptual Class Category	Examples
Physical or tangible objects	Register, Airplane
Specifications, Designs, Descriptions of things	Product Specification, Flight Description
Places	Store, Airport
Transactions	Sale, Payment, Reservation
Transaction line items	Sales Line item
Roles of people	Pilot, Student, Lecturer
Container of other things	Store, Airplane
Things in a container	Item, Passenger
Organizations	Sales Department, Airline
Events	Sale, Payment, Meeting, Crash, Landing

Consider a range of categories in your problem space.2

Conceptual Class Category	Examples
Processes	Selling a product, Booking a seat
Rules and policies	Refund policy, Cancellation policy
Catalogues	Product Catalogue, Parts Catalogue
Records of finance, work, contracts, legal matters	Receipt, Ledger, Employment Contract, Maintenance Log
Financial instruments and services	Line of credit
Manuals, documents, Reference papers, Books	Daily price change list, Repair manual

Step 1 – Example: Identifying the classes

- Draw the classes of objects ('things') that represent concepts from the problem domain – the system you are developing.
- Rectangles are used to represent a single domain class

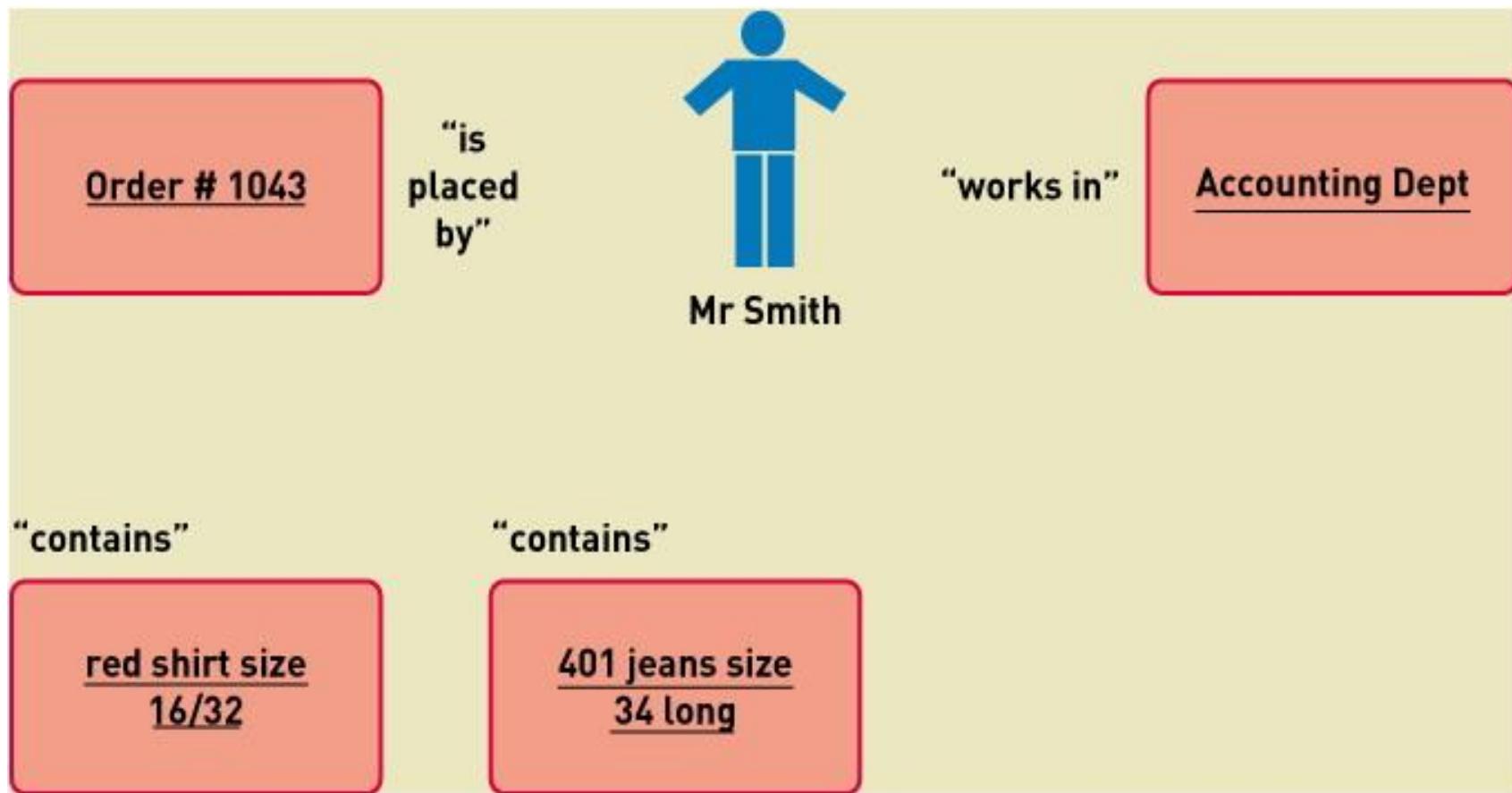


Step 2. Relationship among ‘things’ Finding associations

- A relationship between instances of ‘things’ that indicate some meaningful connection
.. referred to as a ‘relationship’ in ER modelling
- Associations occur in **two directions**
- Example: A customer places an order.
Likewise, an order is placed by a customer

3. Relationship among THINGS - Associations

Associations “naturally” occur between THINGS



Finding associations

- Start the addition of associations by using the Common Associations List
 - It contains common categories that are usually worth considering.

2. Associations – Finding associations

Common Associations List.1

Category	Examples
A is a physical part of B	Wing --- Airplane
A is a logical part of B	Sale Line Item --- Sale Flight Leg --- Flight Route
A is physically contained in/on B	Register --- Store Item --- Shelf Passenger --- Airplane
A is logically contained in B	Item Description --- Catalogue Flight --- Flight Schedule
A is a description for B	Item Description --- Item Flight Description --- Flight
A is a line item of a transaction or report B	Sale Line Item --- Sale Maintenance Job --- Maintenance Log
A is known, logged, captured, reported, recorded in B	Sale --- Register Reservation --- Flight Manifest

2. Associations – Finding associations

Common Associations List.2

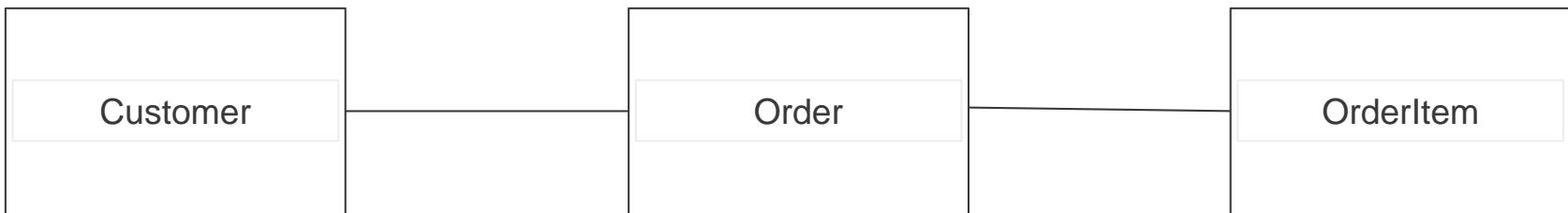
Category	Examples
A is a member of B	Cashier --- Store Pilot --- Airline
A is an organisational sub-unit of B	Department --- Store
A uses or manages B	Cashier --- Register Pilot --- Airline
A communicates with B	Customer --- Cashier Reservation Agent --- Passenger
A is related to a transaction B	Customer --- Payment Passenger --- Ticket
A is related to another transaction B	Payment --- Sale Reservation --- Cancellation
A is owned by B	Register --- Store Plane --- Airline
A is an event related to B	Sale --- Customer, Sale --- Store Departure --- Flight



2. Associations – Finding associations

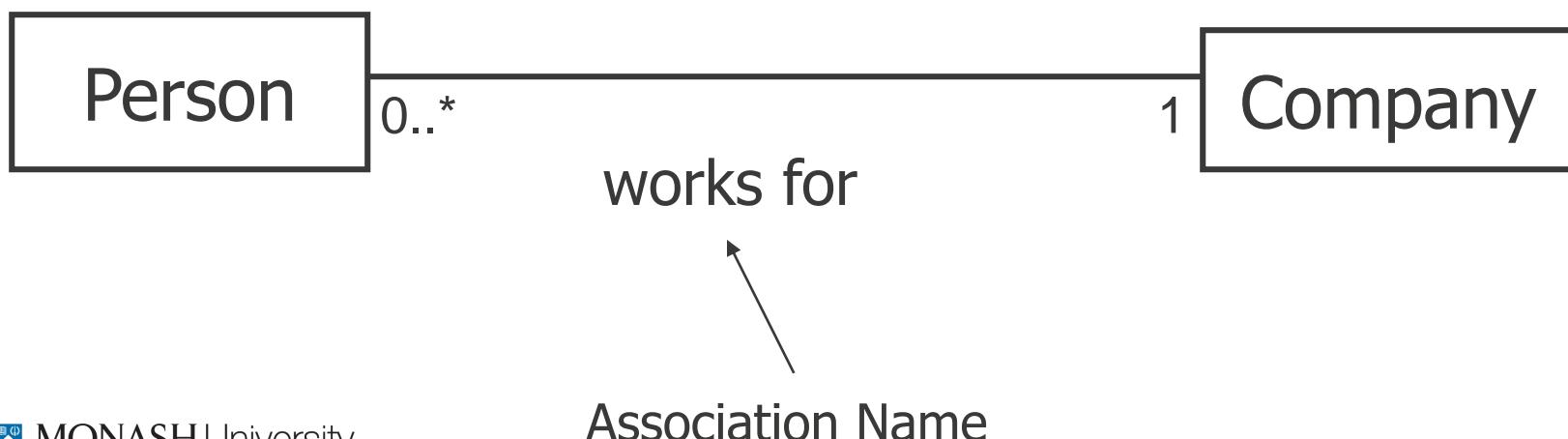
Step 2 – Example: Find associations among your classes

- A customer places orders
- An order has items



Step 3. Find the number of associations between classes - Multiplicity of Associations

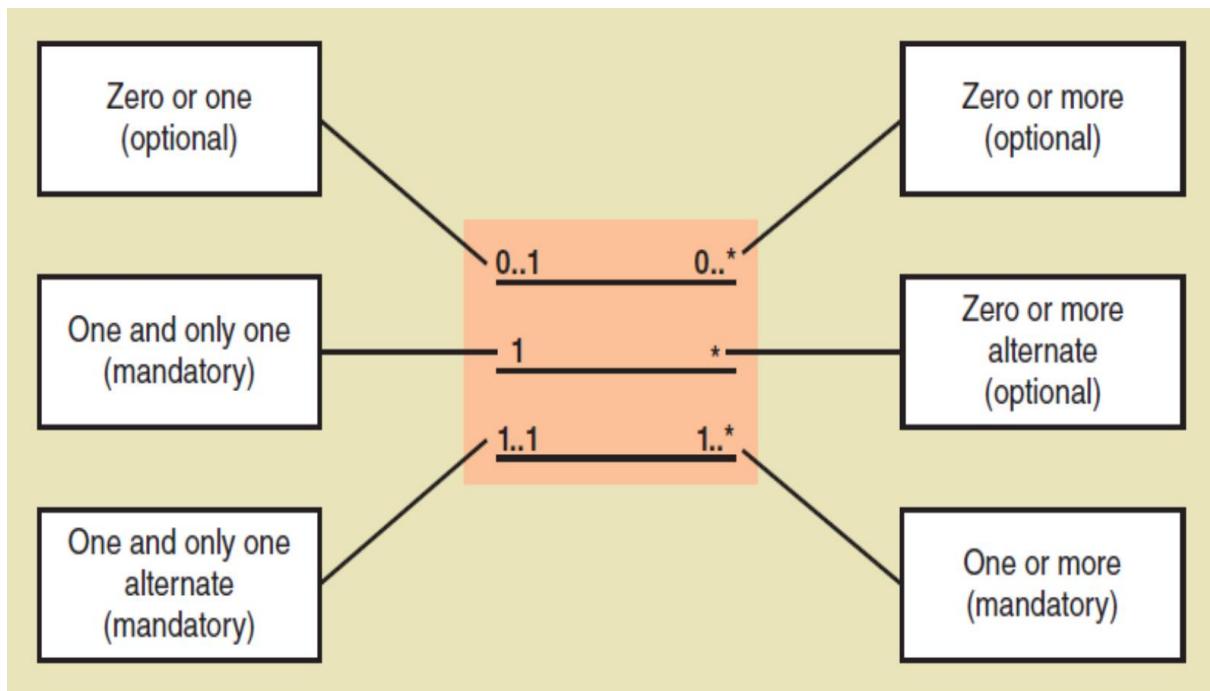
- Multiplicity – the term for the number of association between ‘things’ (classes) – established for each side of an association
 - .. *in ER modelling known as Cardinality*



Associations: Multiplicity notation

Minimum and Maximum multiplicity

- If minimum is zero, the association is optional
- If minimum is at least one, the association is mandatory



Multiplicity Indicator	Meaning
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

3. Multiplicity of associations

Example showing 3 classes

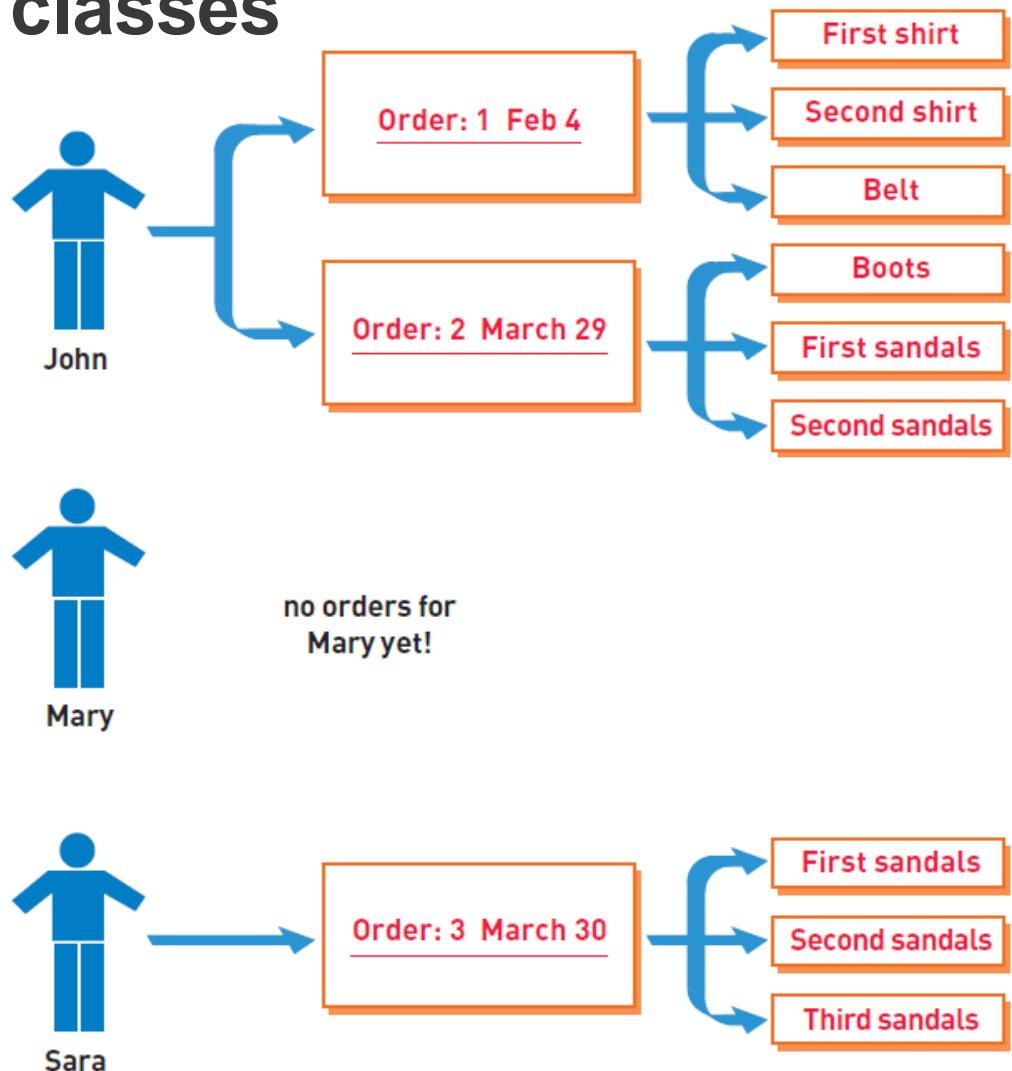


What are the classes?

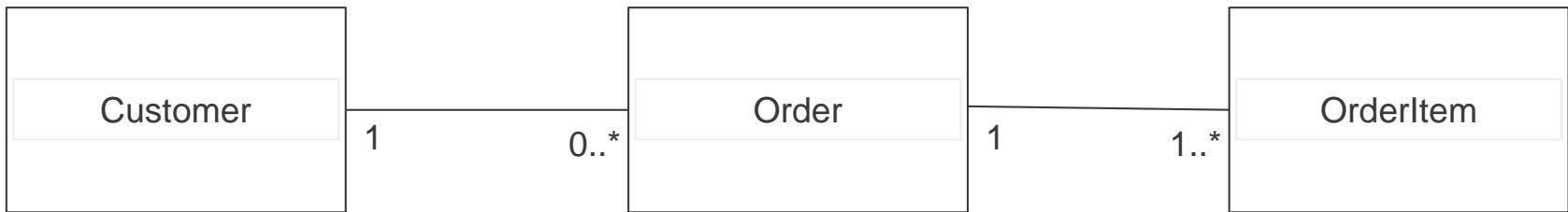
How many associations are there?

What are the minimum and maximum multiplicities in each direction?

What type of associations are they?



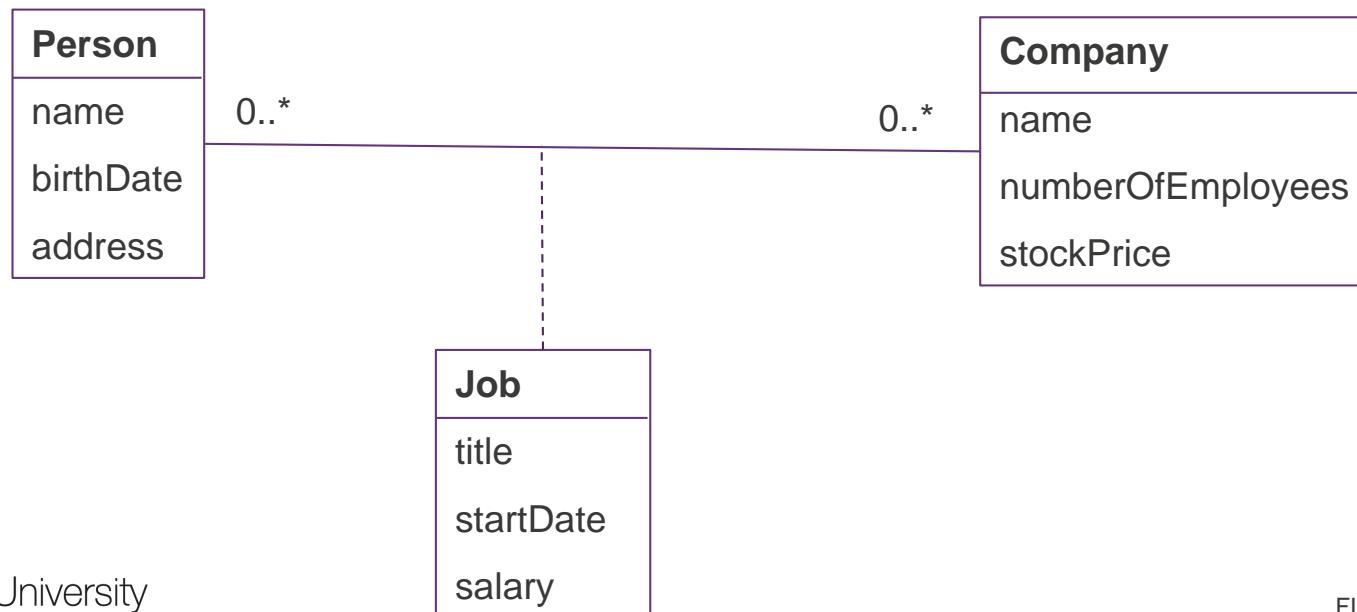
Step 3 – Example: Add multiplicity of associations to the class diagram



- A customer may have placed no orders, or they could have placed many orders
 - An order can only have been placed by a single customer
 - An order must consist of at least one item but could consist of many items
 - An ordered item can only belong to one order

4. Check if there are any Association Classes, and add them in.

- An association class is used to capture certain characteristics of an association between two classes.
- These characteristics do not belong to the classes being associated but instead belong to the relationship between the classes.



**Step 5 – Identify the attributes for your classes.
This may change over time as you discover
more about the system.**

- Attribute – a specific piece of information about a THING
- Identifier/key
 - An attribute that uniquely identify a THING
 - E.g. student ID, invoice number, passport number
- Compound attribute
 - An attribute that contains a set of related attributes
 - E.g. first name, middle name and last name,
 - E.g., home phone, mobile phone, work phone of a customer

What would be the attributes of the Student ‘thing’?

Attributes and values

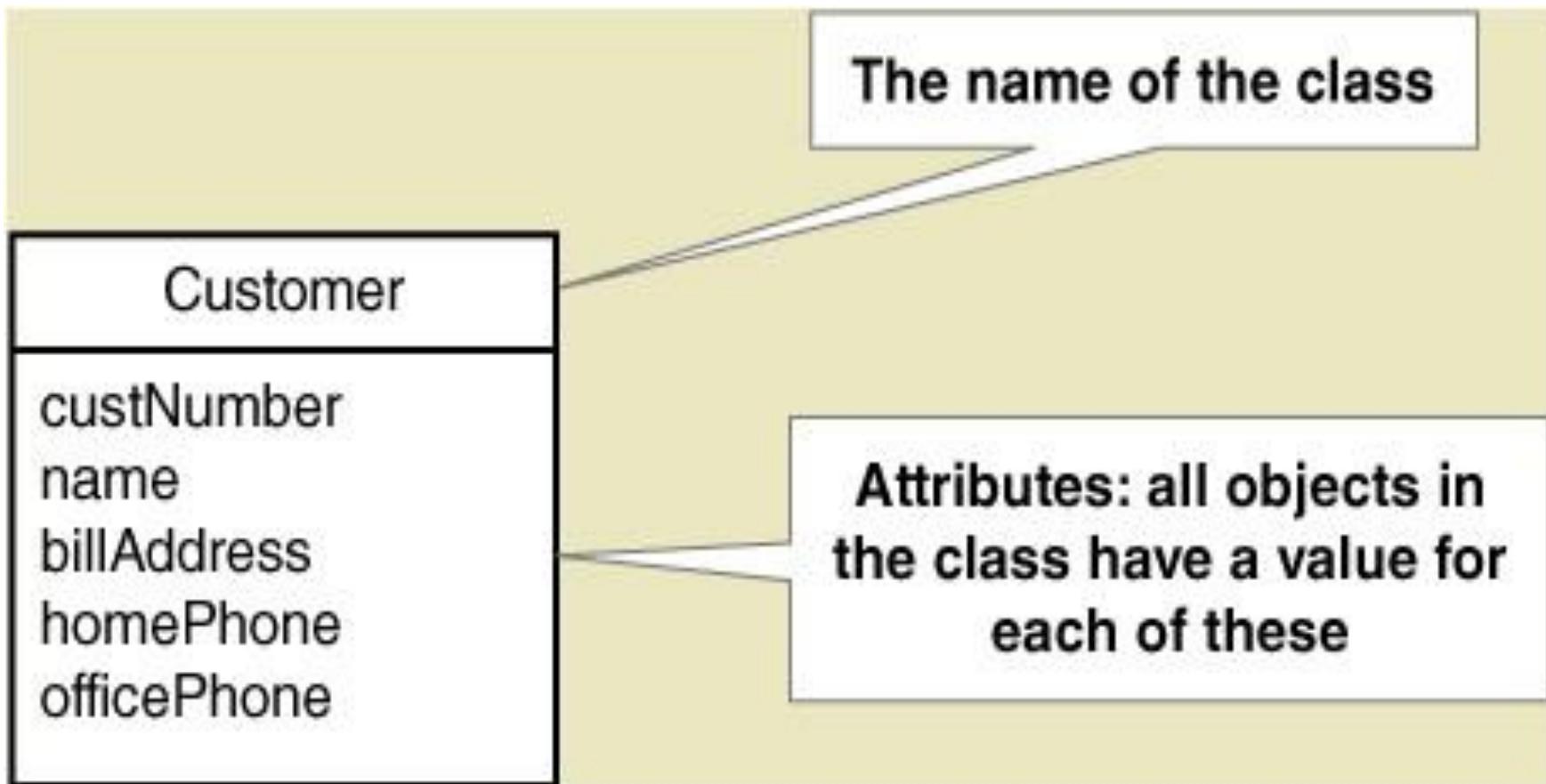
Examples of attributes of customer are listed in the 1st Column

Values of attributes for specific customers are shown in the 2nd column

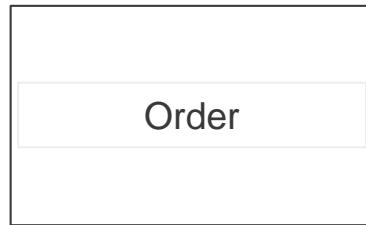
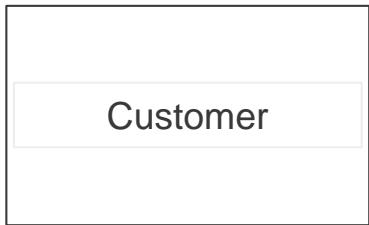
All customers have these attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First name	John	Mary	Bill
Last name	Smith	Jones	Casper
Home phone	555-9182	423-1298	874-1297
Work phone	555-3425	423-3419	874-8546

Adding the attributes to a class – Drawing convention

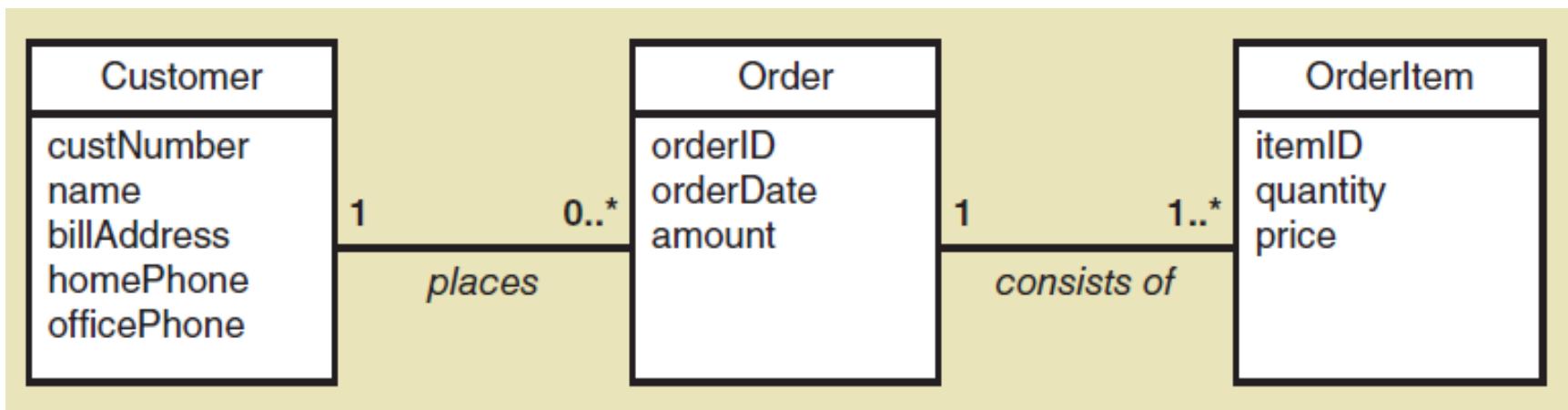
Each class has 2 sections:



What are the attributes for the classes?



Step 5 – Example: Add the attributes to the classes

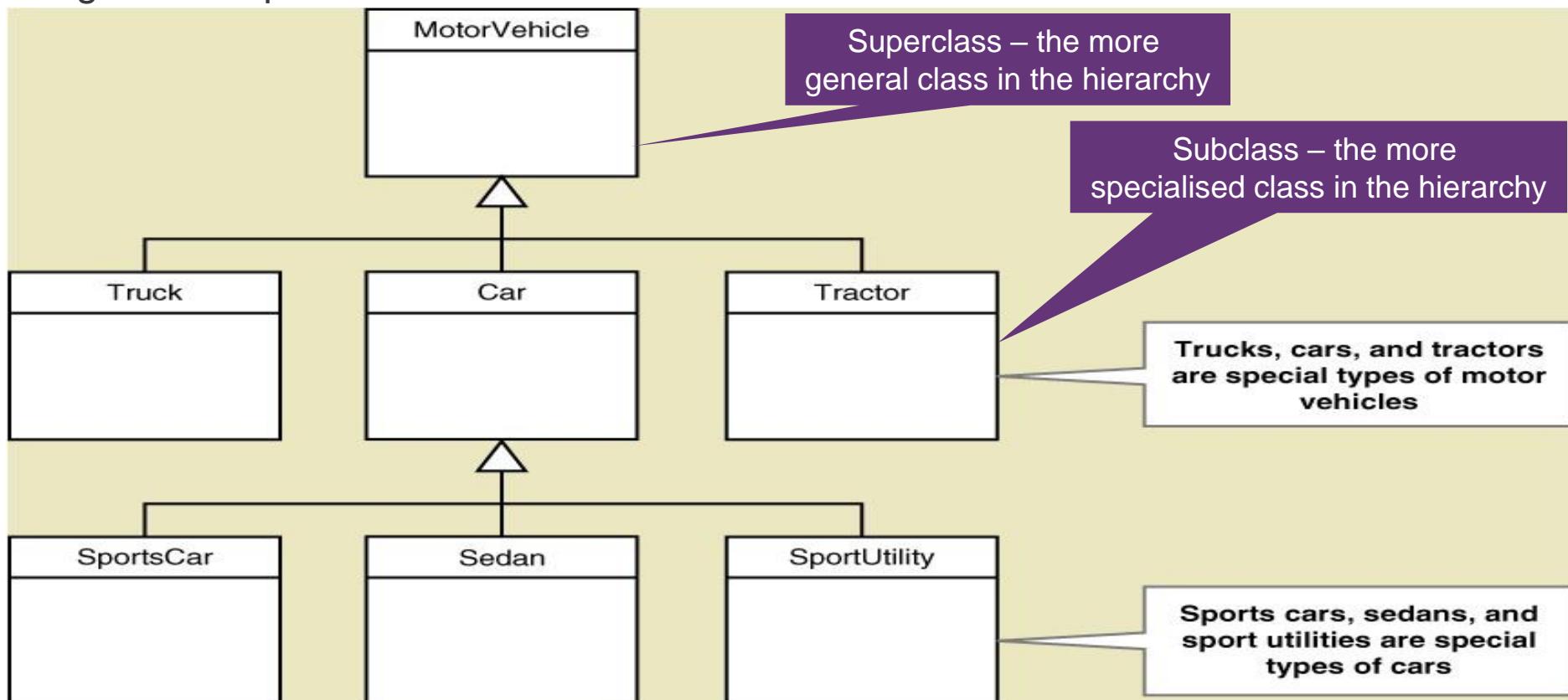


- These are some of the attributes – the attributes you capture depend on the requirements of the system you are building

6. Generalisation / Specialisation

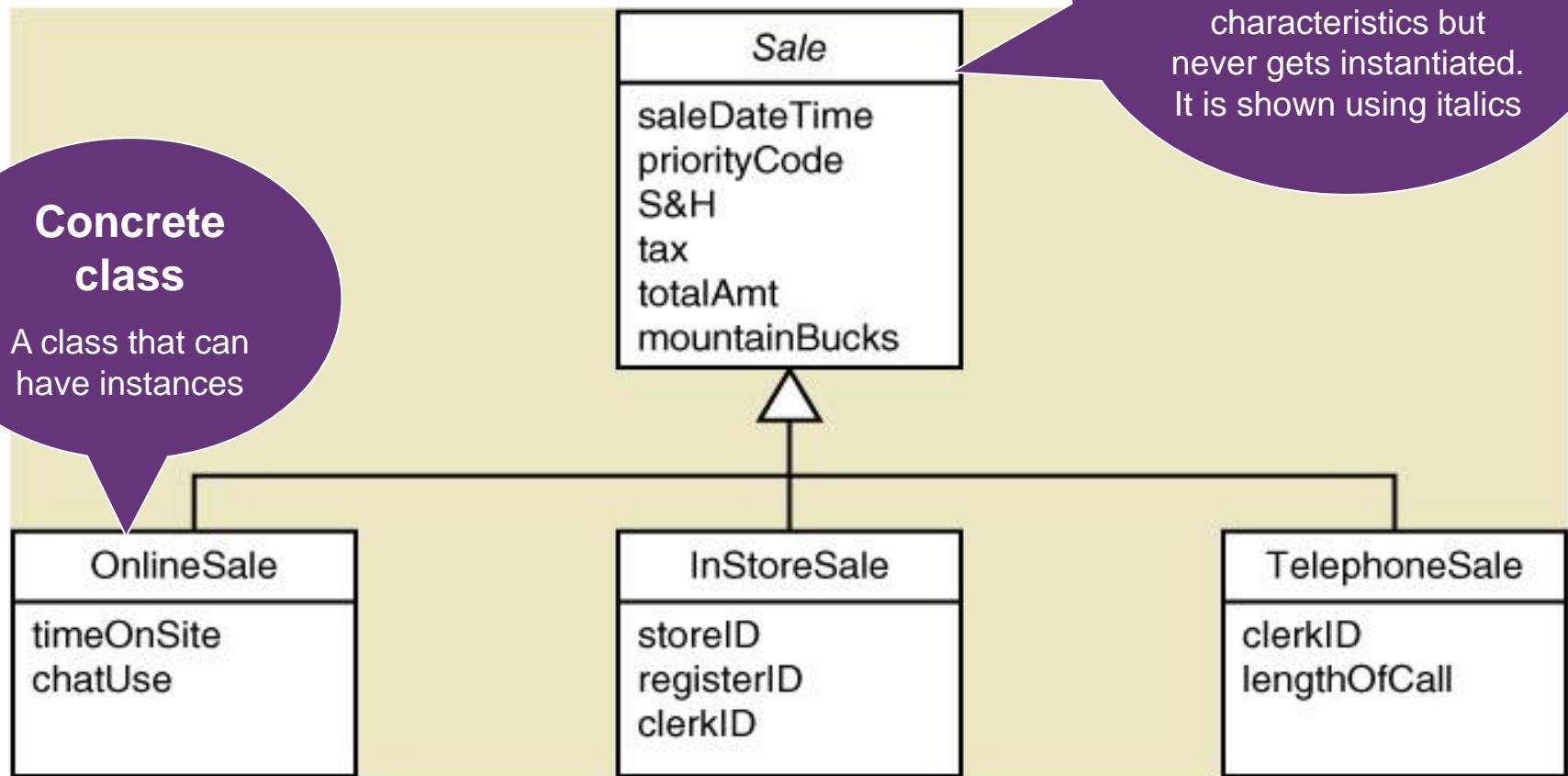
Step 6. Identify complex relationships: Generalisation/Specialisation

A hierarchical relationship where subordinate classes are special types of the superior classes. Often called an Inheritance Hierarchy ... an **'is a' relationship** where subclasses inherit characteristics of the more general superclass



6. Generalisation / Specialisation

Step 6 – Example 1:



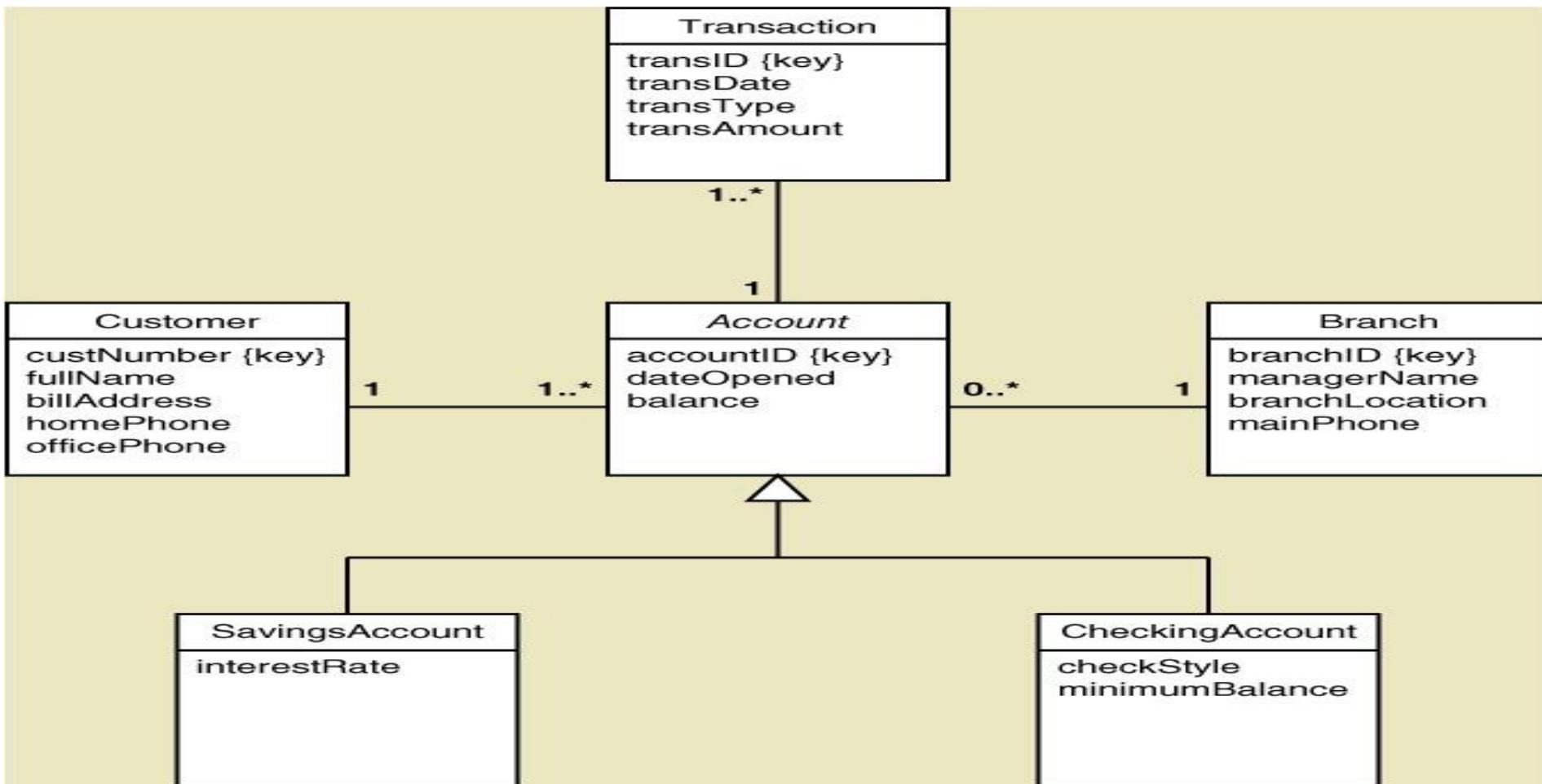
Abstract class

A class that allows subclasses to inherit characteristics but never gets instantiated. It is shown using italics

Concrete class

A class that can have instances

Step 6 – Example 2:

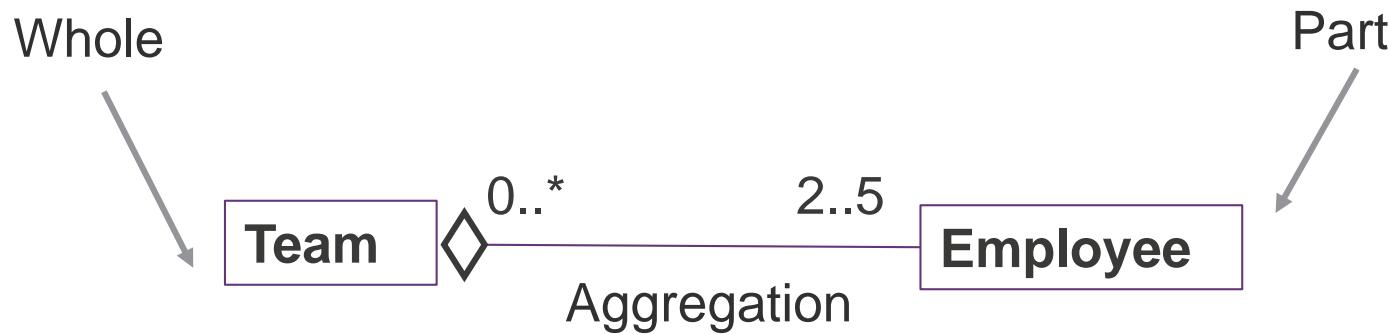


Step 7. Identify complex relationships: Whole-Part

- Whole - part relationship - a relationship between classes where one class is part of or a component portion of another class ... represents a 'has-a' relationship

Aggregation

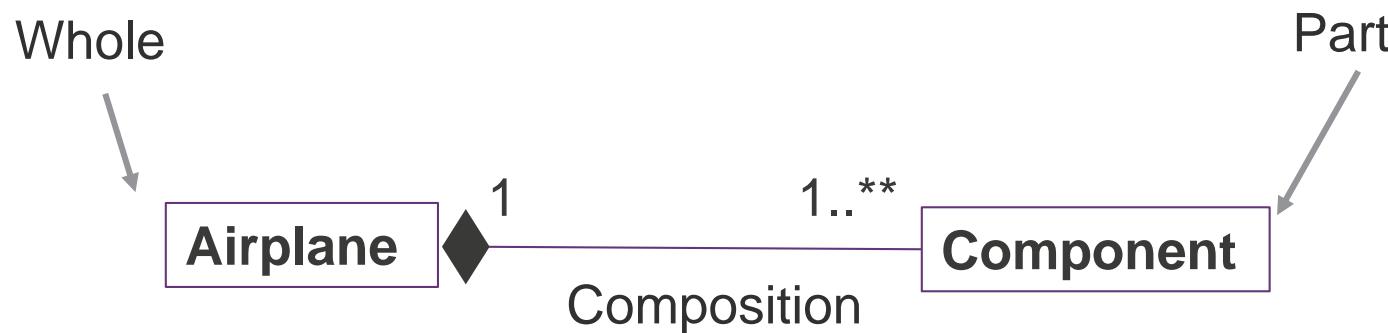
- A special form of association that models a whole-part relationship between an aggregate (the whole) and its parts
- The component part exists separately and can be removed and replaced
 - Team has employees
 - Company has divisions



- To depict an aggregate relationship, a unfilled diamond is used

Composition

- A form of aggregation with strong ownership, where there is a strong *lifecycle dependency* between instances of the container class and instances of the contained class(es): if the container is destroyed, normally every instance that it contains is destroyed as well.



- To depict a composition relationship, a filled diamond is used

EXAMPLE: On the Spot Courier Services

Request a Quote

When a customer puts in a quote request we would like the customer to enter their email address, the pick up and delivery address, the details of all the packages and the date of pickup and delivery. If the package does not meet our size requirements or we are unable to do the pick up or delivery on the specified dates, we do not provide a quote. Otherwise we provide a quote based on the package size (we have 3 sizes) and the distance which is charged per km travelled.

Identify the ‘things’, and relationships for On the Spot Courier Services ‘Request a Quote’ function

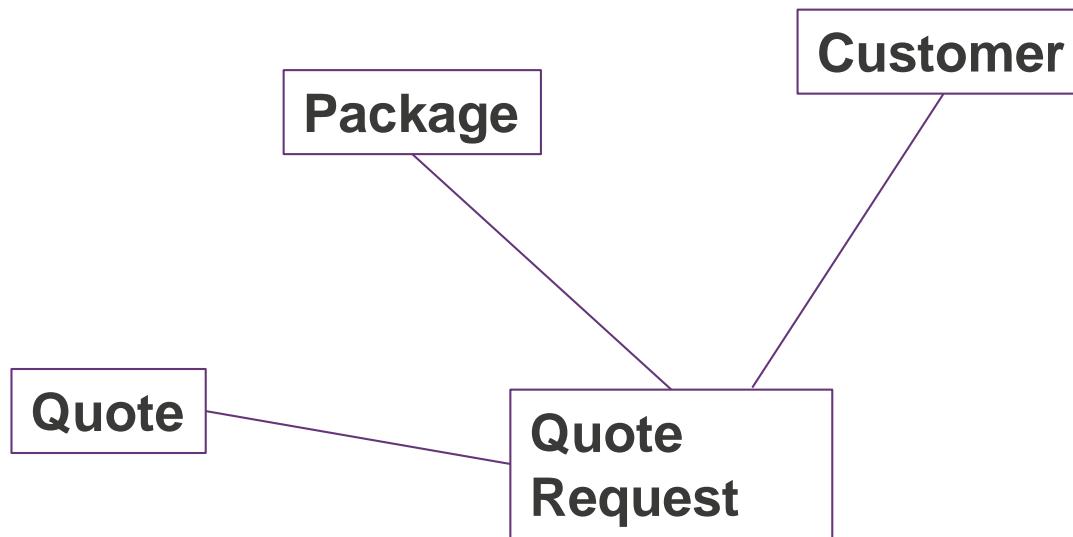


1. Identify the ‘things’ – classes

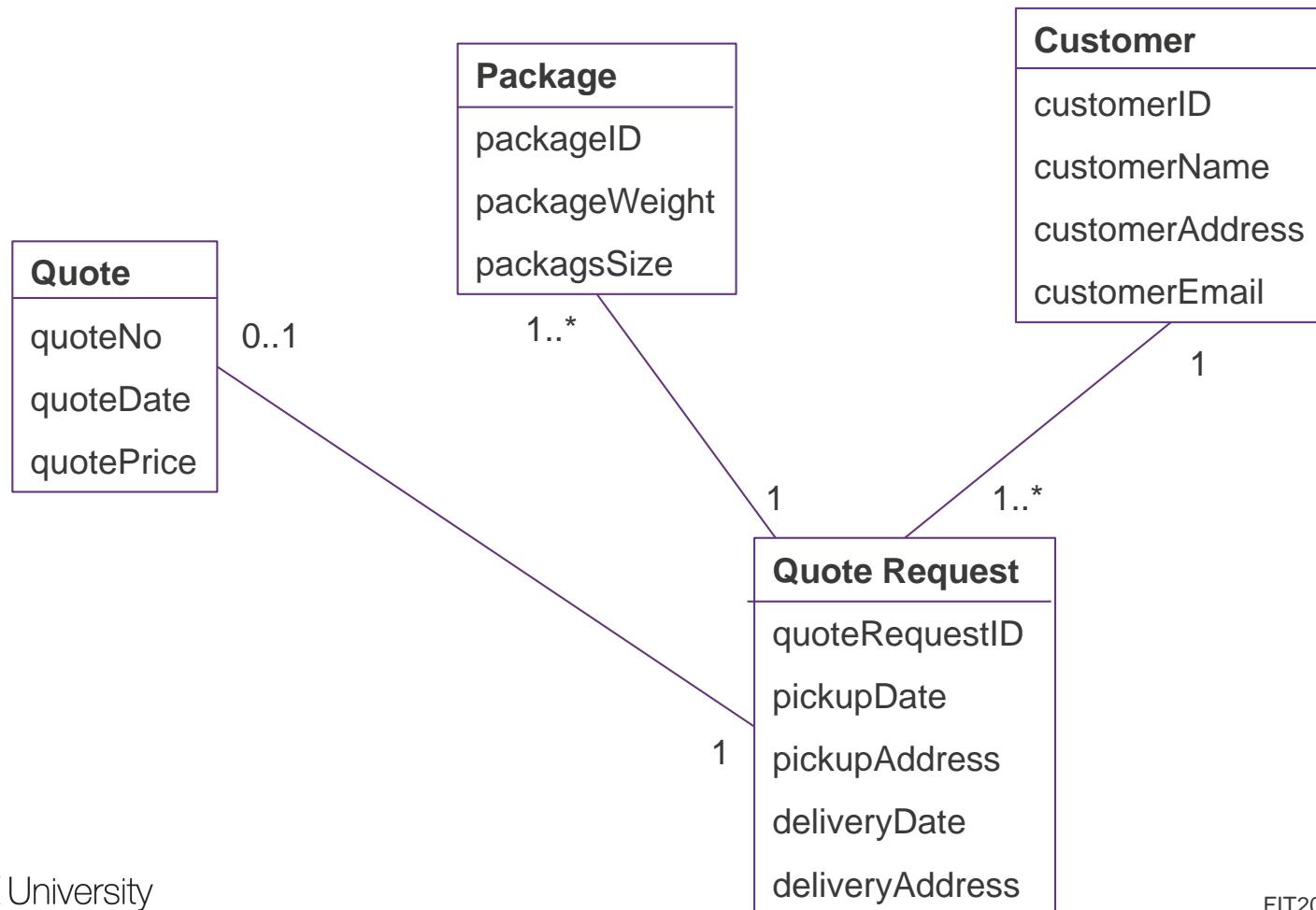
Request a Quote

When a customer puts in a quote request, they first enter their email address. If they are a new customer we then request their name and address, if they are a current customer we display these details. We then request the dates of pickup and delivery, the pick up and delivery addresses and, the details of all the packages. If the packages do not meet our size requirements we do not provide a quote. Otherwise we provide a quote based on the size of the package(s) (we have 3 sizes for pricing purposes) and the distance which is charged per km travelled.

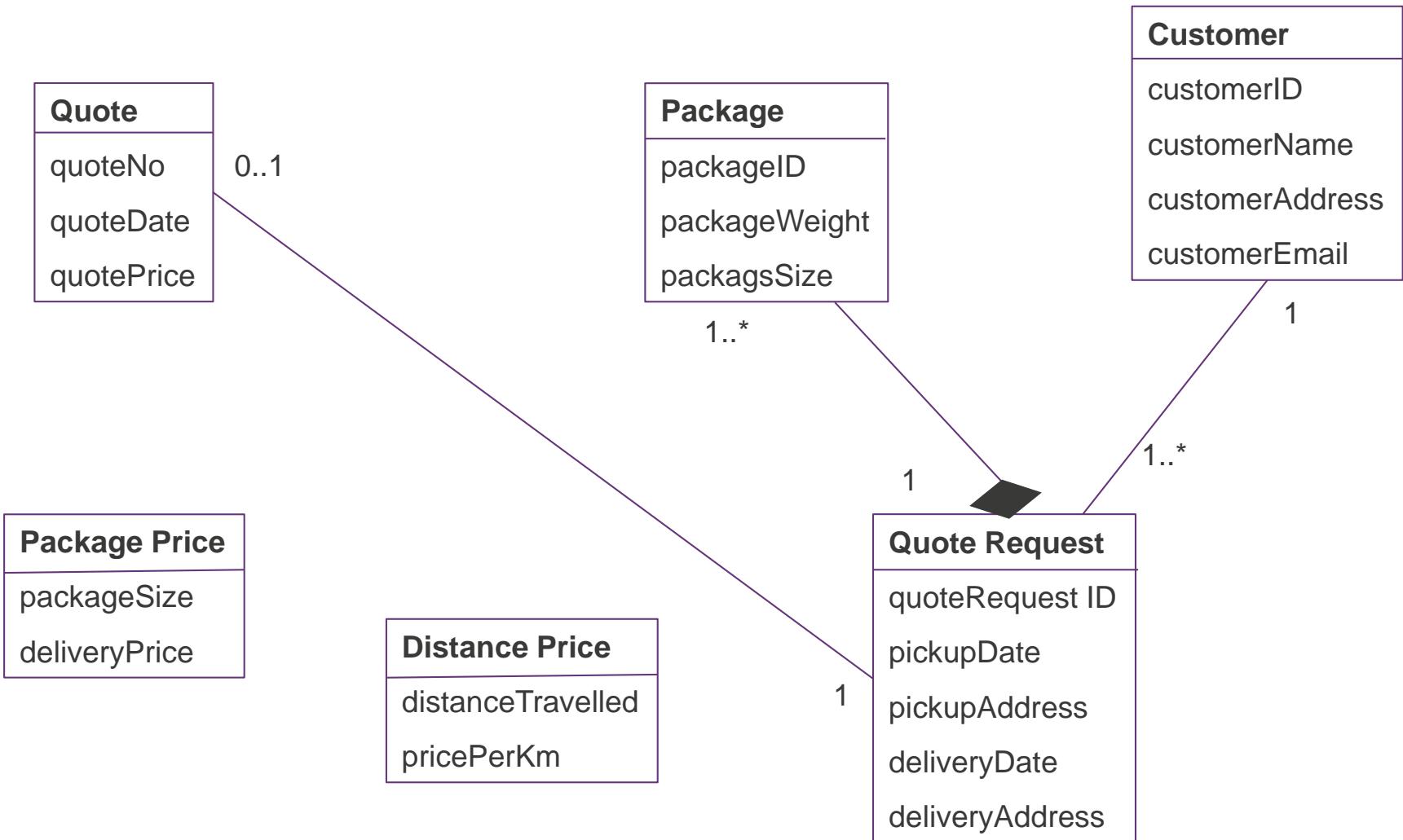
2. Draw initial domain model with classes and associations



3. Draw the Domain model class diagram (Class Diagram) – with attributes



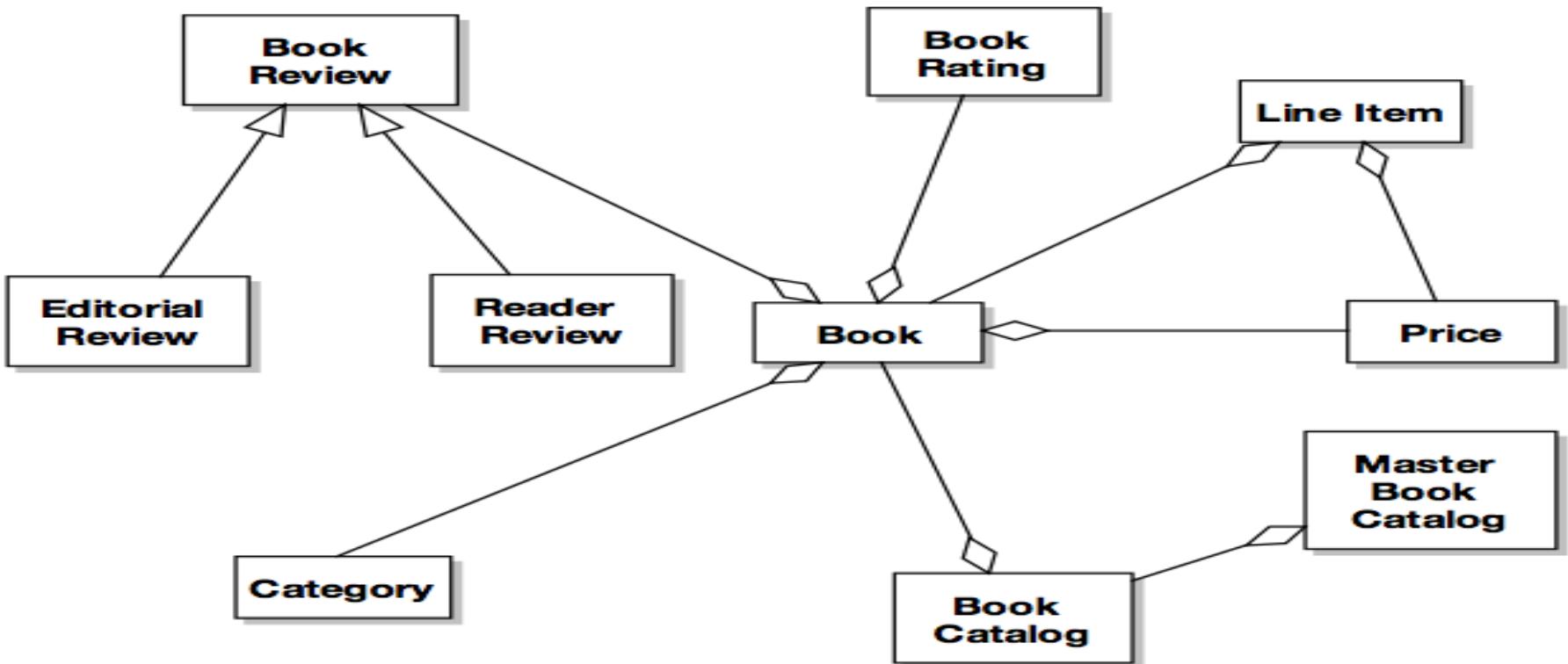
4. Add complex relationships (if they exist)



9. Evolution of UML class models as a system is developed

1. Initial domain model

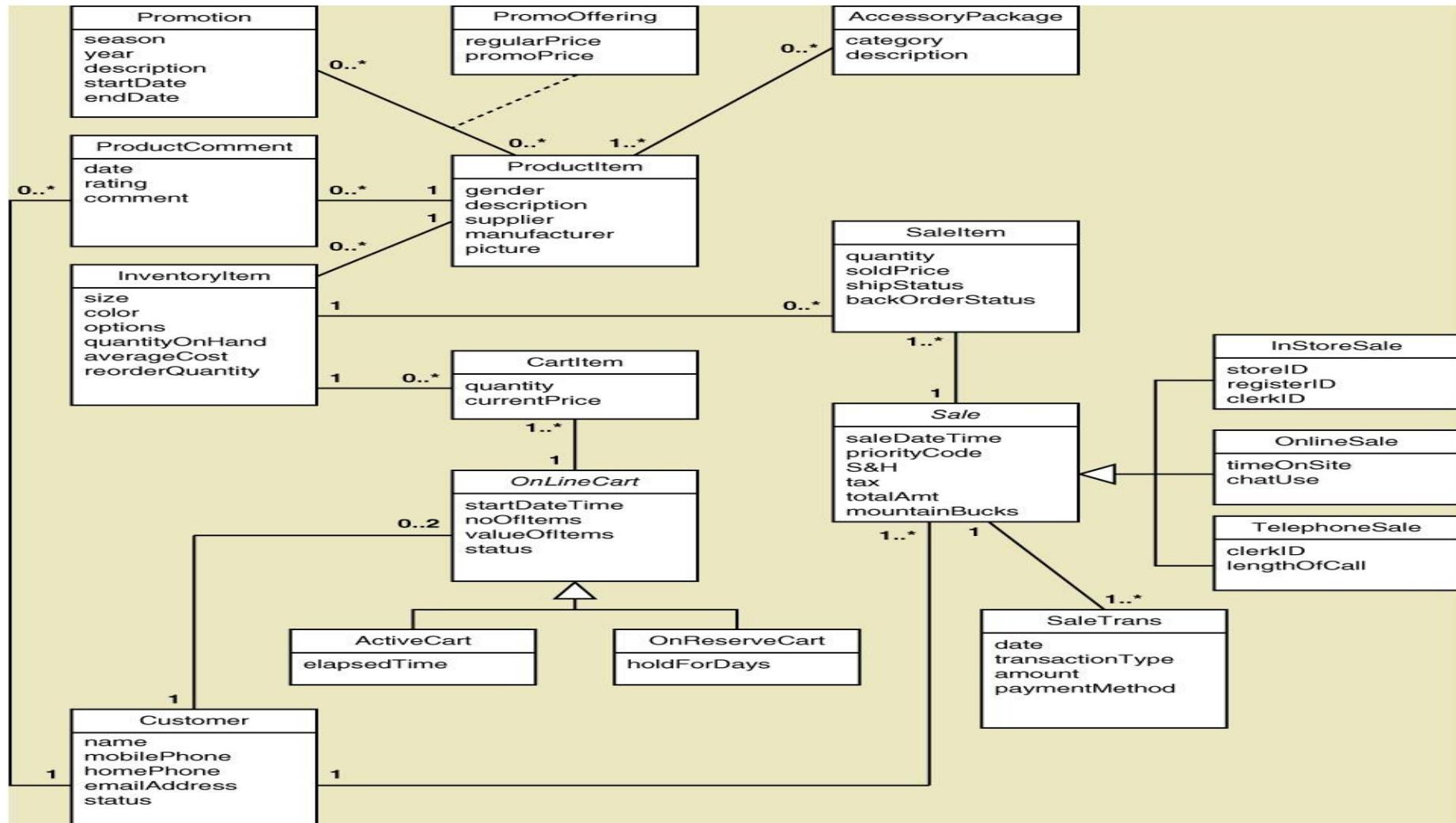
- UML class diagram showing class names and relationships
- Do not include attributes, cardinality and operations(methods)



9. Evolution of UML class models as a system is developed

2. Domain model class diagram (often referred to as a class diagram)

initial domain model which also includes relationship multiplicities, attributes



9. Evolution of UML class models as a system is developed

3. Design class diagram includes attribute types, methods and navigation

Domain class diagram

Student
studentID name address dateAdmitted lastSemesterCredits lastSemesterGPA totalCreditHours totalGPA major

Design class diagram

«Entity» Student
<p>-studentID: integer {key} -name: string -address: string -dateAdmitted: date -lastSemesterCredits: number -lastSemesterGPA: number -totalCreditHours: number -totalGPA: number -major: string</p> <p>+createStudent (name, address, major): Student +createStudent (studentID): Student +changeName (name) +changeAddress (address) +changeMajor (major) +getName () : string +getAddress () : string +getMajor () : string +getCreditHours () : number +updateCreditHours () <u>+findAboveHours (int hours): studentArray</u></p>



Workshop Preparation

Start getting ready for your Assignment 2
Story Mapping Workshop

Thanks for watching
See you next week

Resources:

Prescribed text:

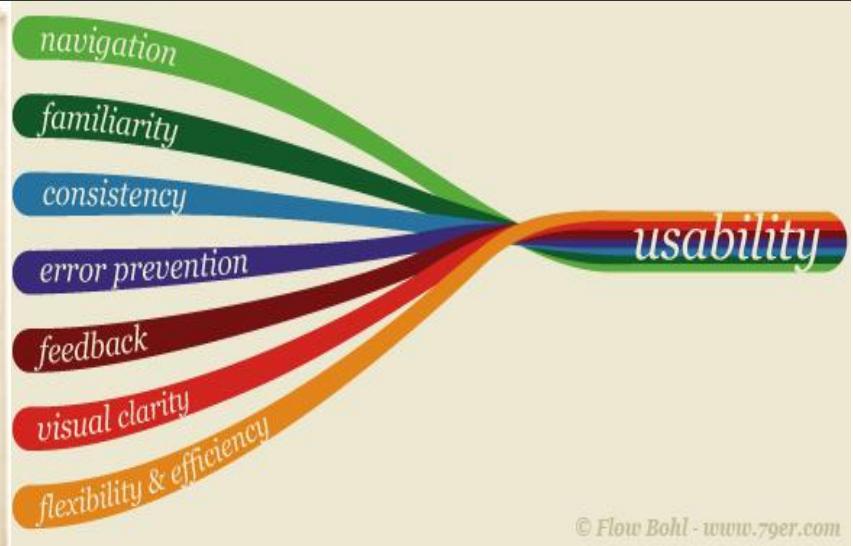
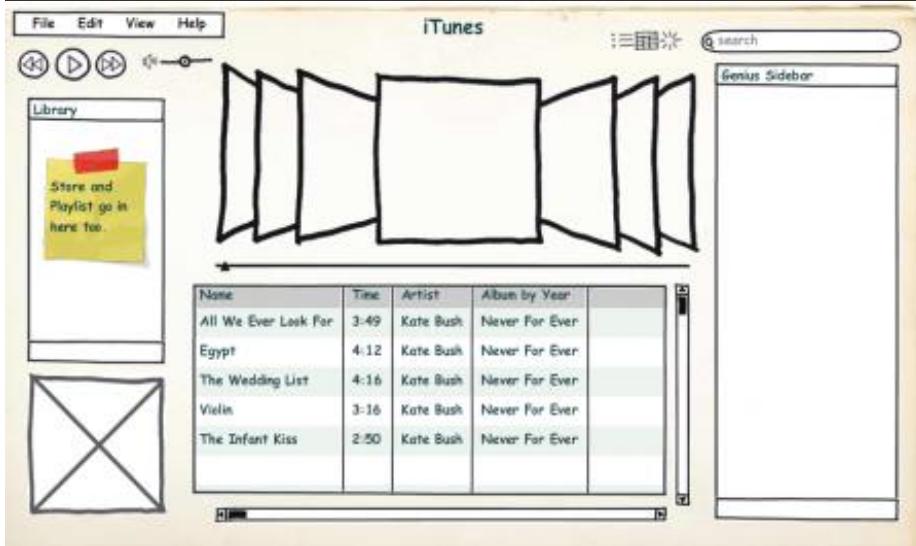
- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 4 (pp. 94-100, 103-114)



FIT2001 – Systems Development

Seminar 7.1 Investigating system requirements – Prototyping Usability of systems

Chris Gonsalvez



Our road map:

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- A range of Requirements gathering and documentation techniques: User Story Mapping, User Stories, Activity Diagrams, Use Case Diagrams, Domain Class Models

Some more Requirements gathering techniques:

- Prototyping
- Usability of systems

At the end of this seminar you will:

- Be able to explain the role of prototyping in systems development – the advantages and disadvantages
- Understand the process for developing suitable prototypes for a given scenario
- Appreciate the need to develop usable systems
- Understand the benefits of considering usability in interface design

FIT2001 Student feedback

"I wanted to thank you for teaching me as your subject gave me a solid foundation for the work that I've been doing during my IBL placement at PwC. A significant portion of the content from FIT2001 has been relevant in some way. At first I was working on the Fraud and Forensics team as a data analyst and was able to apply what I'd learnt about stakeholder management and communicating with non-technical employees. A couple of times, I tried to get some experience with the Digital Change team and I think that I was welcomed onto the UX team (a sub-team within Digital) during the last six weeks of my placement because I had been able to demonstrate a good understanding of UX and the agile methodology

FIT2001 Student feedback (cont.)

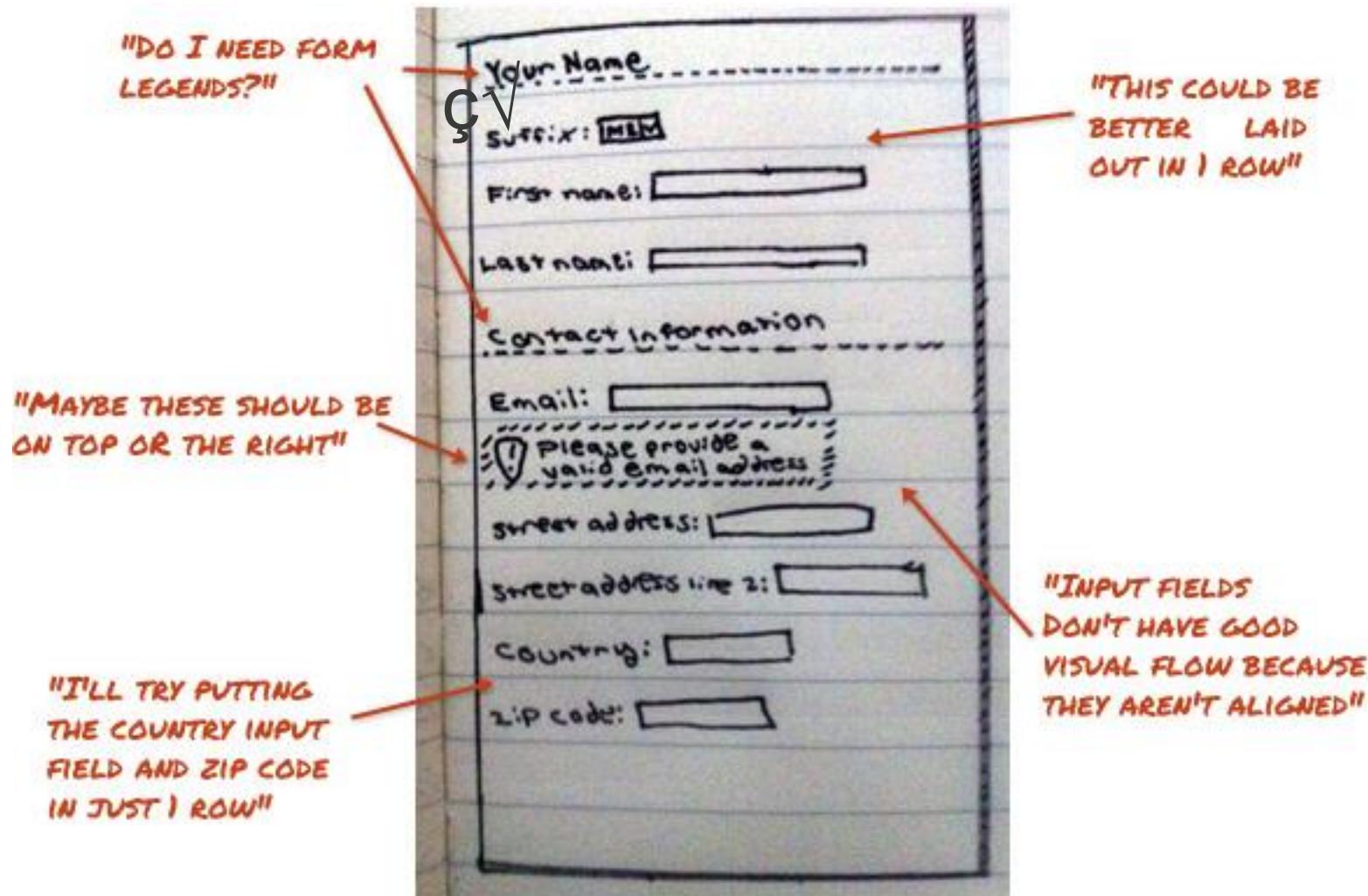
... I was able to follow the "UX lingo" with ease and required very little training when it came to conducting research, documenting business findings, creating personas/user journeys/user stories and designing prototypes. It was fantastic to join in on real stakeholder interviews and help conduct client workshops to understand the business and user requirements. After two weeks with the UX team, a contractor asked me about my background in UX, assuming I'd been in the field for a while, and I admitted that I'd only ever learnt the theory in University - this was my first UX project. I think that says a lot about the preparation I received.

I think it's fair to say that without FIT2001 and its well-structured curriculum I would not have been as successful during my placement, given such interesting tasks and maybe not have had the chance to work with the UX team at all."

Lecture Outline:

1. Prototyping Overview
2. Prototyping Process
3. Prototyping Example
4. Usability Issues
5. Usability Definition
6. Usability Importance
7. Usability Poor Examples
8. Usability Evaluation
9. Usability Cost

Prototyping – a picture speaks a thousand words



What is Prototyping?

- The process of quickly mocking up the future system functionality
- Uses visuals to describe thousands of words worth of design and development specifications that detail how a system should behave and look.
- It can be throw-away (experimental) or evolutionary
- It can be horizontal or vertical

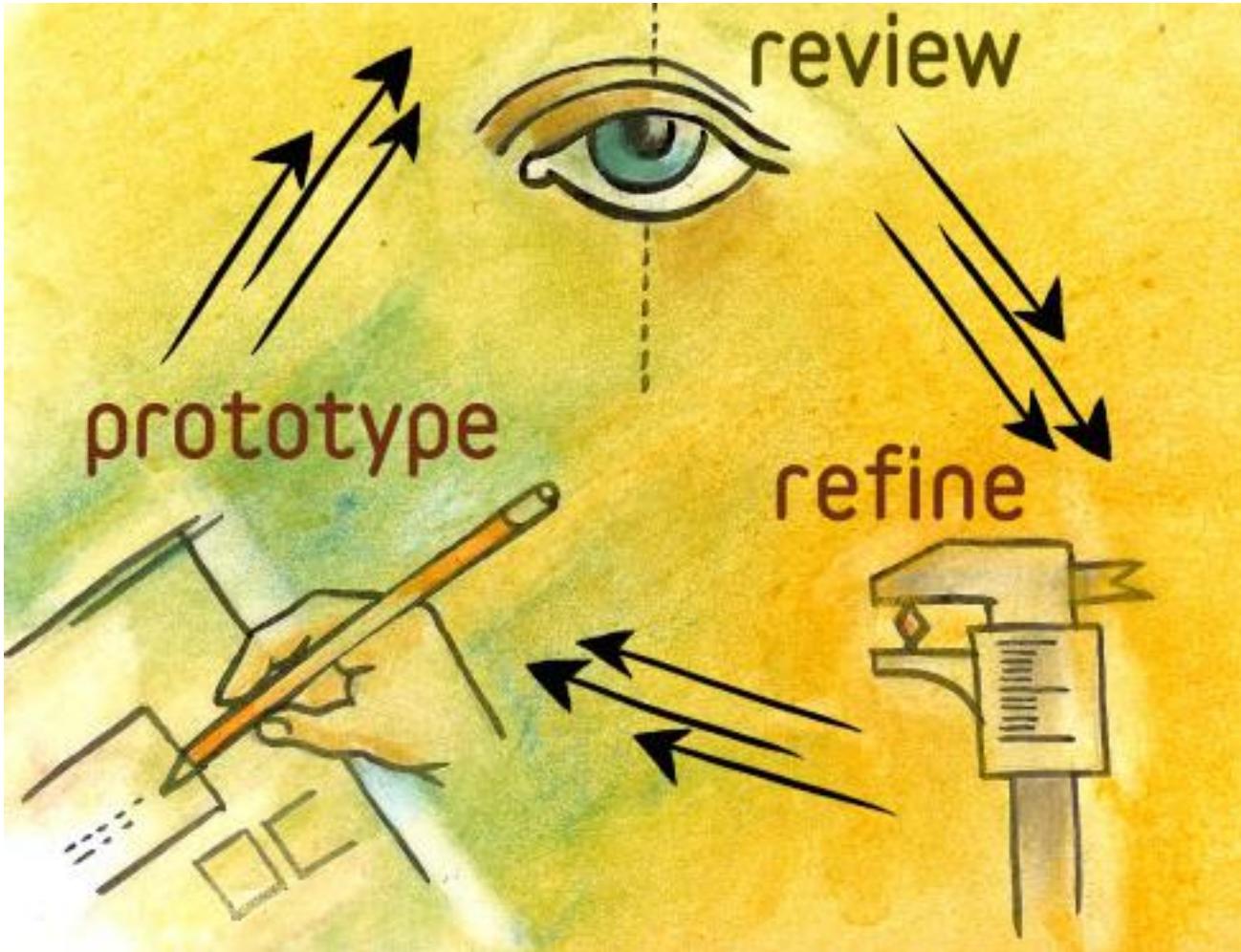
Why Prototype?

- Explore ideas before you invest in them
 - Improved communication, risk reduction, reduced maintenance, greater user satisfaction
- Saves time and money
- Proof of concept
- Design exploration
- Technical exploration

Some dangers of prototyping

- Prototyping might:
 - Make the users think the system is developed
 - Have to manage expectations carefully
 - Create a system that doesn't scale
 - Waste time (as developers spend a great deal of time making throw away prototypes look good)

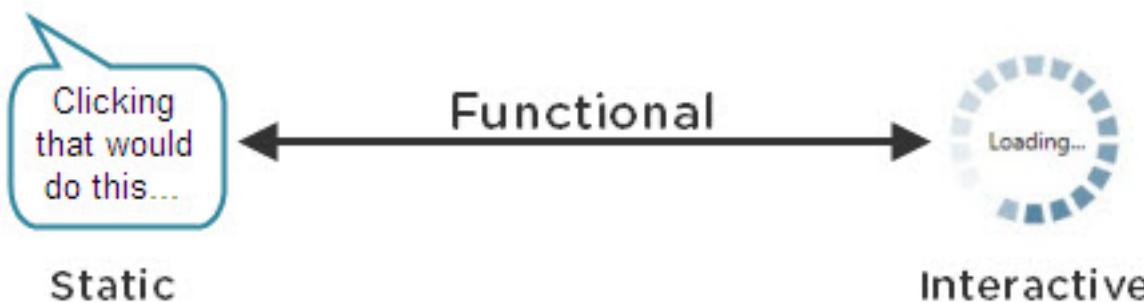
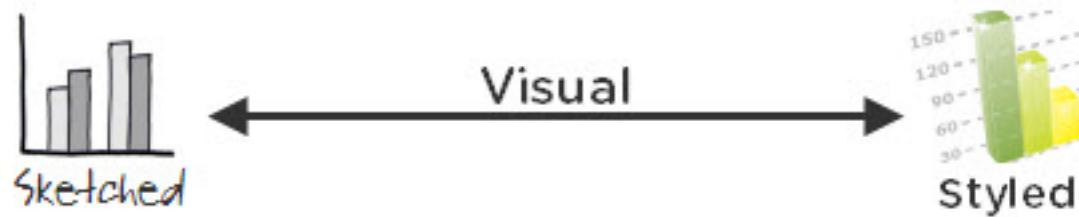
The Prototyping process



Scoping a Prototype

- What needs to be prototyped?
 - Complex interactions, new functionality, changes in workflow
- How much should be prototyped?
 - The functionality that will be used most of the time
- Find the story
 - Prototype scenarios – user stories, different persona experiences
- Plan your iterations
 - Start broadly, then drill down for some functionality
- Choose the appropriate fidelity
 - Visual (style), Functional (Interactions), Content (real)

Choose the appropriate fidelity



The Prototyping spectrum

- Low fidelity
 - create rough paper based mock-ups
 - gets feedback on design approaches and concepts
 - lets you make changes easily and quickly



The Prototyping spectrum

- Medium fidelity
 - Increased fidelity with computer based tools
 - demonstrates behaviour of the application – simulates interactions

The Prototyping spectrum

- High fidelity
 - Most realistic, often mistaken for final product
 - Excellent reference for developers
 - Great for usability testing and training
 - Learning curve for developers

The screenshot shows the VMS Video Downloads website interface. At the top, there's a navigation bar with links for 'Return to VMSVideo', 'Welcome Guest', 'Sign In', 'Create Account', 'Cart 0 Items Subtotal: \$0.00', and 'REVIEW CART/CHECKOUT'. On the left, there are three columns of filters: 'BROWSE CATEGORIES' (Action Adventure, Animation, Classics, Comedy, Documentary, Drama, Foreign, Independent, Music & Performance, Mystery & Suspense, Romance, Sci-Fi & Fantasy, Special Interest, Westerns, View All Categories), 'BROWSE TV NETWORKS' (HBO, CBS, FOX, WB), and 'BROWSE TV SHOWS' (The Sopranos, The OC, Deadwood, Entourage, Grey's Anatomy). The main content area features a 'TV Time Machine' section with a heading 'Filter Shows By: All Genres TV Ratings'. It includes a text block about tuning in to long lost classics, followed by a grid of show cards for the '2000's', '1990's', '1980's', '1970's', '1960's', and '1950's'. Below this, a section titled 'Most Popular Shows from the 90's' displays cards for 'DOOGIE HOWSER, M.D.', 'Vee-Zee', 'Kids in the Hall', 'BRAIN CANDY', and 'MURKIN'. A 'Select a Year from the 90's:' dropdown menu lists years from 1990 to 1999. The bottom of the page shows a footer with links for 'About Us', 'Privacy', 'Terms & Conditions', and 'Help', along with a page number 'Page 1 of 2'.

Selecting Prototyping Tools

- Evaluate tools – feature set and strengths What are your needs and requirements?
 - How easy is it to learn and use the tool?
 - Is it flexible to support all types of applications?
 - Is there a repository of reusable stencils, templates or widgets available?
 - How easy is it to share for review? Can feedback be captured using the tool?
 - How easy is it to make changes on the fly or to incorporate feedback?
 - Does it have any collaboration features, such as allowing multiple people to work on it at the same time?
 - What are the licensing terms and costs?

Do

- Work collaboratively with all stakeholders
- Avoid “prototype creep” - set expectations – why are you doing it?
- When creating interactive high-fidelity prototypes and simulations, build in realistic delays
- Reuse, reuse, reuse – save template for future projects
- Begin every prototype review session with the disclaimer that this is **just a prototype, a mock-up, not the actual solution.**

Don't

- Don't **prototype features or functionality that cannot be implemented** – understand your technology
- Don't take every change or request that comes out of a **prototype review** as a new requirement. Be aware of scope creep
- Be very specific about the type of feedback you are looking for – Are the steps logically arranged? Is the navigation clear and intuitive?
- Don't be a perfectionist - just good enough to give everyone a common understanding
- Don't prototype everything – just enough to understand what is required

ON THE SPOT COURIER SERVICES

Bill Wiley – Pick-up and Delivery function

When Bill Wiley receives a request for pickup, he enters the pickup information on a form and processes the payment. If the payment is approved, he contacts his courier staff with the pick up and delivery information. When they pick up the package they ring Bill to inform him that they have picked up the package and he notes it on the form. They also ring him when the pack is delivered, which Bill also notes on the form. The package has to be signed by an approved person before it is delivered.

Create low fidelity rough mock-ups for the Pick-up and delivery function for On the Spot Courier Services



3. Prototyping Example

Pick-up and Delivery function: Low Fidelity Mock-up

PICK-UP / DELIVERY REQUEST

CLIENT AT
ATARI Consultants
Atlas' Gyges/
Atboshu Pty. Ltd.

DATE 22/3/14 AVAILABILITY

Msg .. No Couriers Available today.

PICK-UP LOCATION

DELIVERY LOCATION

INSTRUCTIONS

PAYMENT PAYPAL CREDIT CARD

ALLOCATED

PICK-UP CONFIRMED -/-/-

DELIVERY CONFIRMED -/-/-

Usability



How usable are these products ?



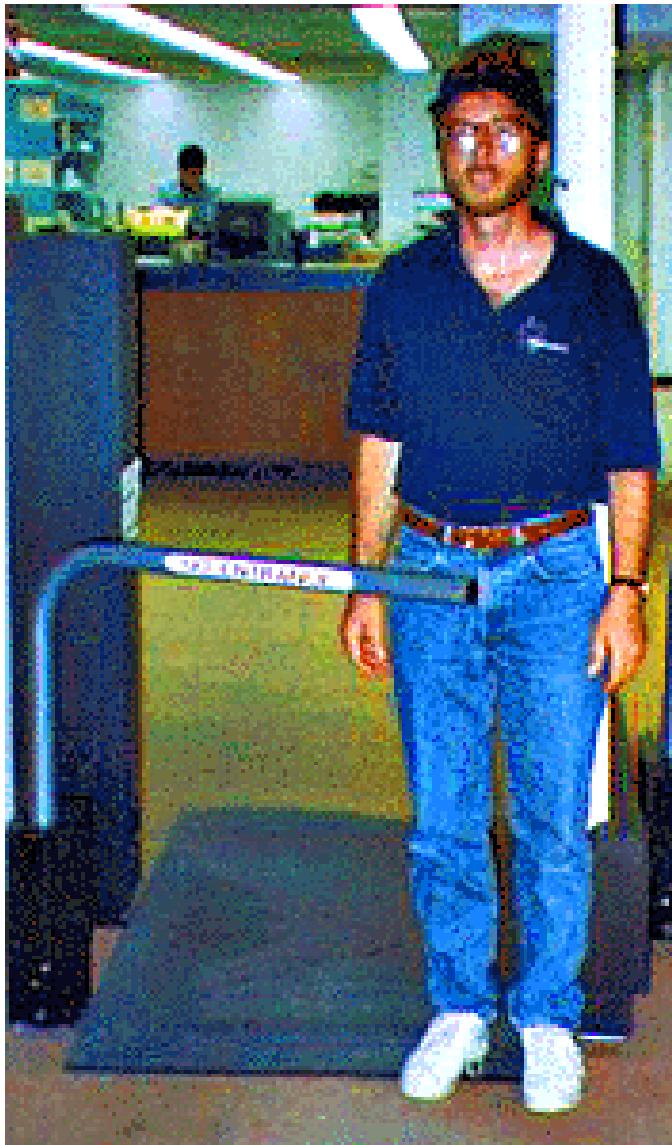
What about
these
instructions?





When trying to open this file cabinet users found themselves pulling the handle on the top (See arrow).

Guess what happened?



Ouch !!!!!

Don Norman – The Godfather of UX

User Experience

<https://www.youtube.com/watch?v=RIQEoJaLQRA>

We want to develop **USABLE**
information systems that are of
real value to our clients

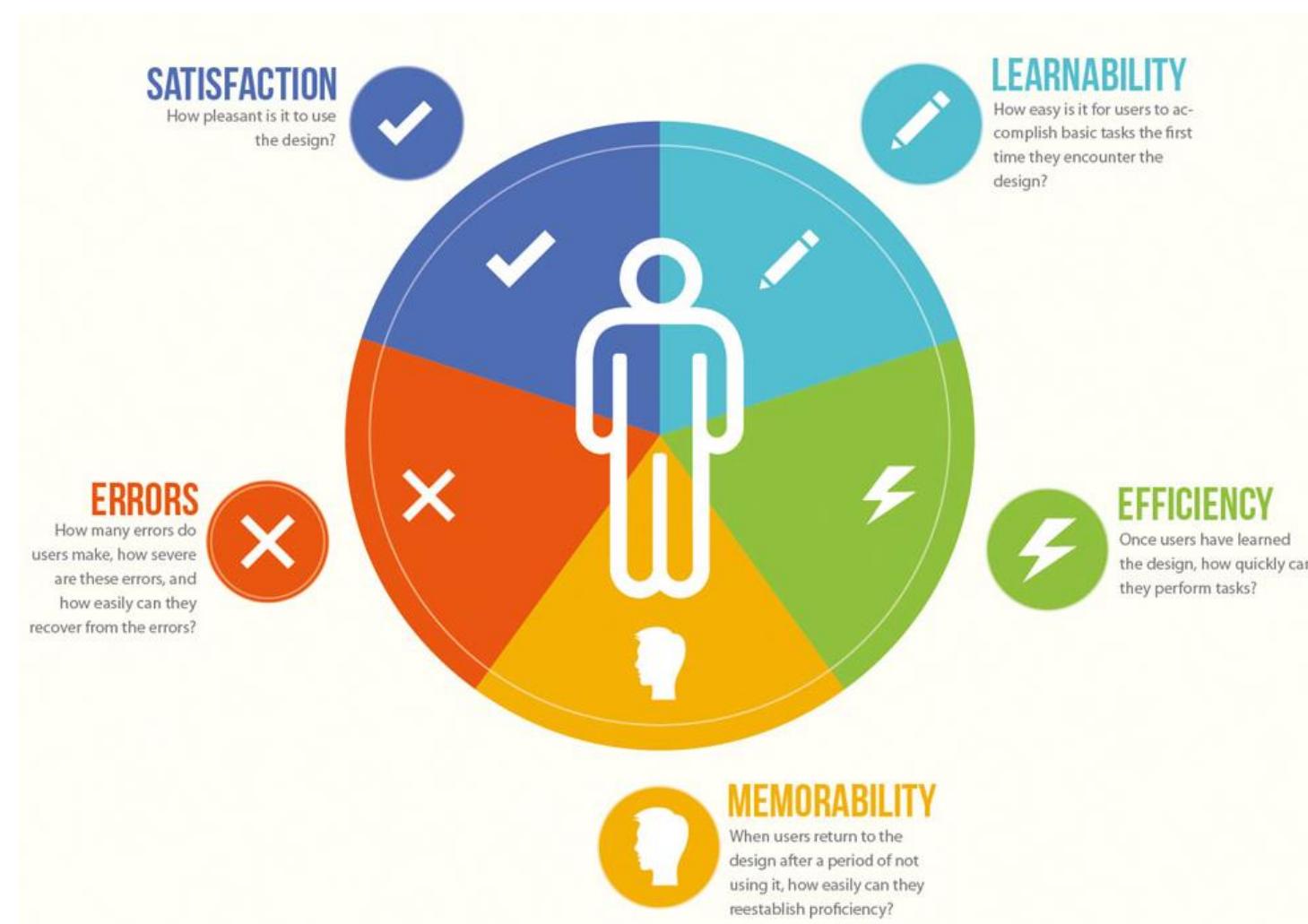
Which three measures are needed to establish how usable a product is?

What is Usability?

User Effectiveness, Efficiency and Satisfaction

- ISO 9241-11: **Usability is:** The extent to which a product can be used by specified users to achieve specified goals with **effectiveness, efficiency, and satisfaction** in a specified context of use.
- **Effectiveness:** accuracy and completeness with which users achieve specified goals.
- **Efficiency:** resources expended in relation to the ‘effectiveness’ with which users achieve goals.
- **Satisfaction:** the comfort and acceptability of the work system to its users and other people affected by its use.

Evaluating Usability?



Evaluating Usability – 5 criteria?

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- **Memorability:** When users return to the design after a period of not using it, how easily can they re-establish proficiency?
- **Errors:** How many errors do users make? How severe are these errors? How easily can they recover from the errors?
- **Satisfaction:** How pleasant is it to use the design?

Usability problems

Design a birthday form field in contact form

This screenshot shows a contact form with a "Geburtsdatum*" (Birthday) field. The input field contains the date "16.12.1961". A red circle highlights the right edge of the input field, indicating it is too short to accommodate the full date string.

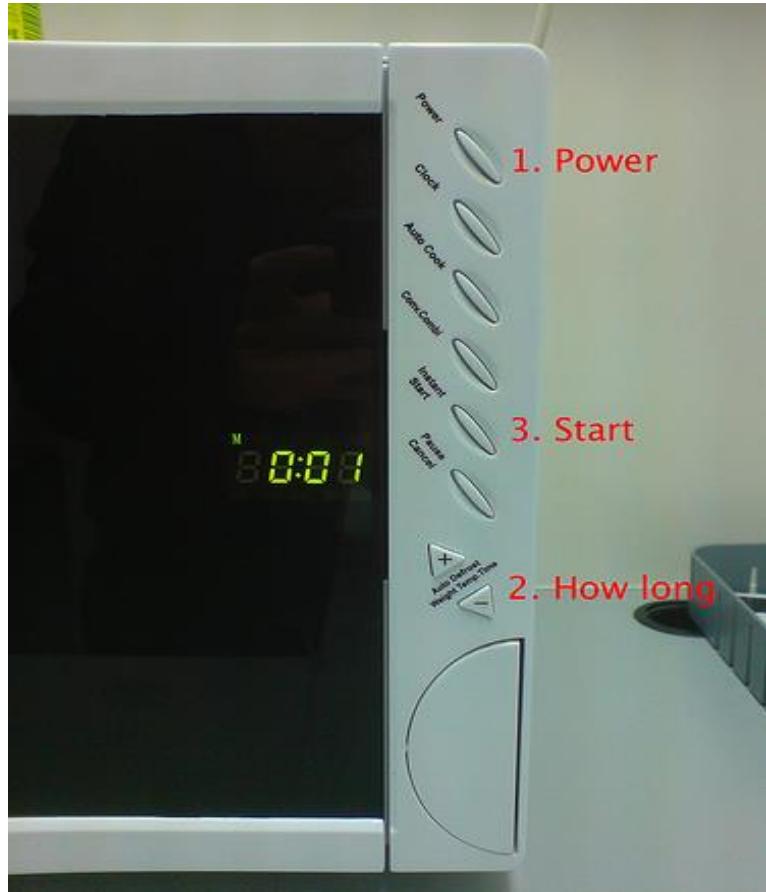


This screenshot shows a contact form with a "Geburtsdatum*" (Birthday) field containing the date "16.12.1966". To the right of the input field, the text "Format: TT.MM.JJJJ" indicates the date format. A red circle highlights this text, showing that the format is clearly communicated to the user.

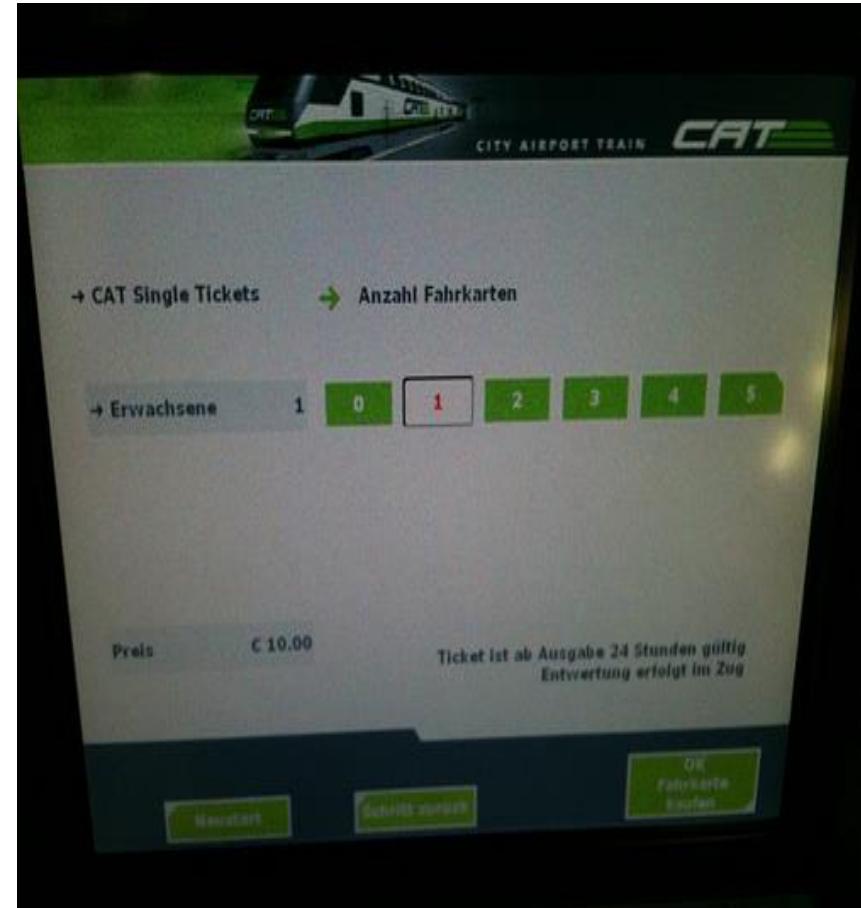
Issue: Birthday field is not long enough to show a full birthday date like 16.12.1966

Issue: A date-picker does not make much sense for a birthday date (it even contains a very non-useful “Today” button).

Usability problems



Sequencing is important



At Vienna Airport, a screen showing the number of tickets (**including 0**) a passenger can buy to travel to city via the City Airport Train

Usability should NOT be ...

- Expensive
- Time consuming
- A creativity killer
- Focus groups
- Customer satisfaction surveys

“It is far better to adapt the technology to the user than to force the user to adapt to the technology.”

- Larry Marine

Why is usability important?

- Helps improve user efficiency
- Can make users feel more in control
- Can improve user satisfaction
- Helps improve sales of commercially available software products
- Helps improve actual usage of systems (e.g. ERP, e-commerce)



Why is usability important?

For e-commerce usability is essential to survive:

- If systems are difficult to use, people leave.
- If users get lost, they leave.
- If systems (web sites) are hard to read or don't answer users' key questions, they leave.
- If users cannot find the product on web sites, they cannot buy it.
- If users don't know how to buy the product, they cannot buy it.
- If users cannot find the price of a product, they will not buy it.

“The joy of an early release lasts but a short time. The bitterness of an unusable system lasts for years.”

- *Anonymous*

Usability: Is it important?

THE DESIGN SESSION

We could make our site nice and simple, easy to understand...



Or we could make the user choose among thousands of meaningless options and blind them with Flash movies and blinking adverts.



Bear in mind that we never have to actually meet these people.



© Usability by Design, 2002

Make it so their browser crashes each time they move the mouse.

Stupidity is rife some examples

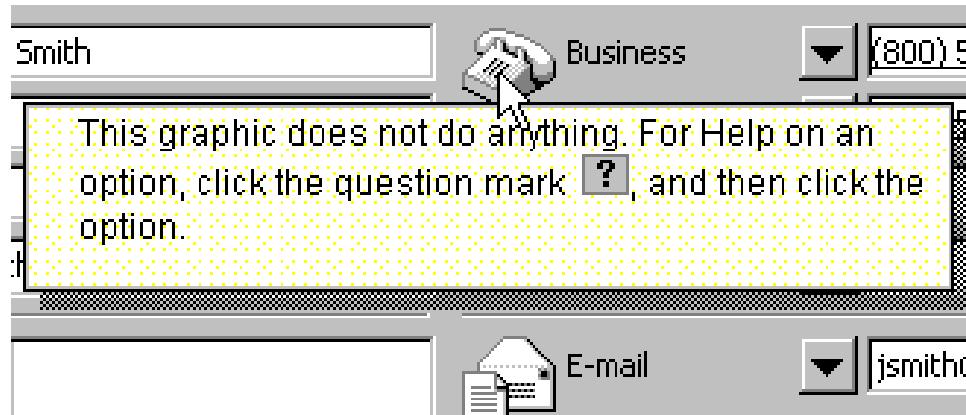
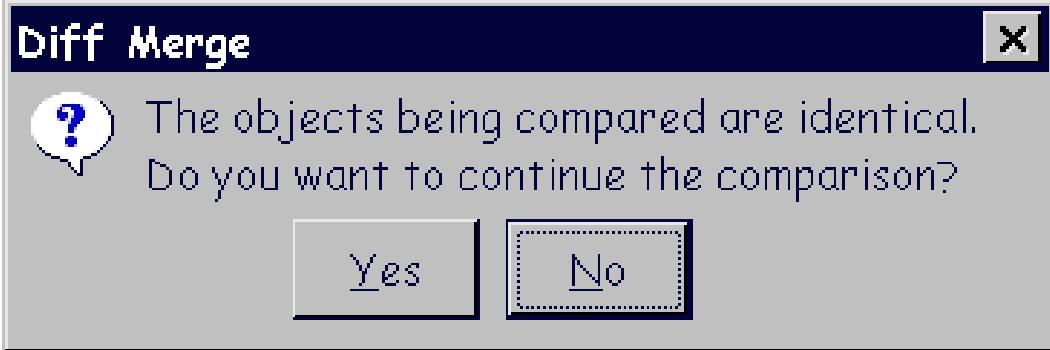


7. Usability Poor Examples

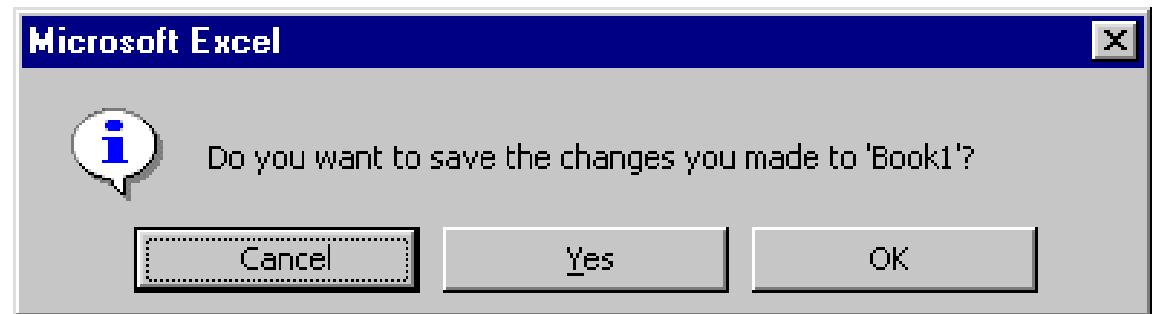
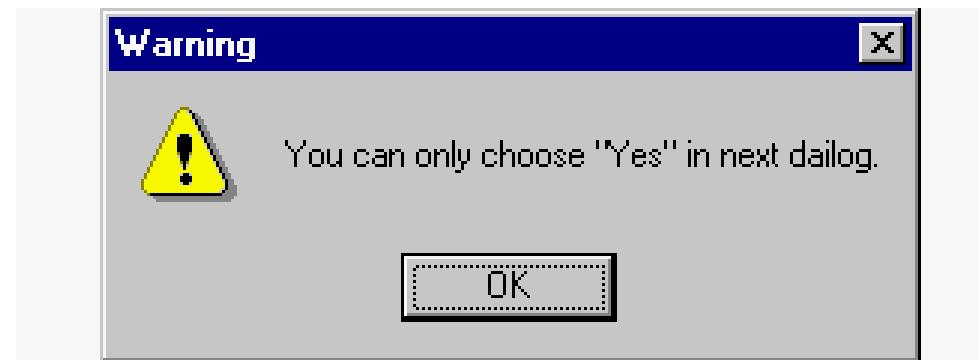
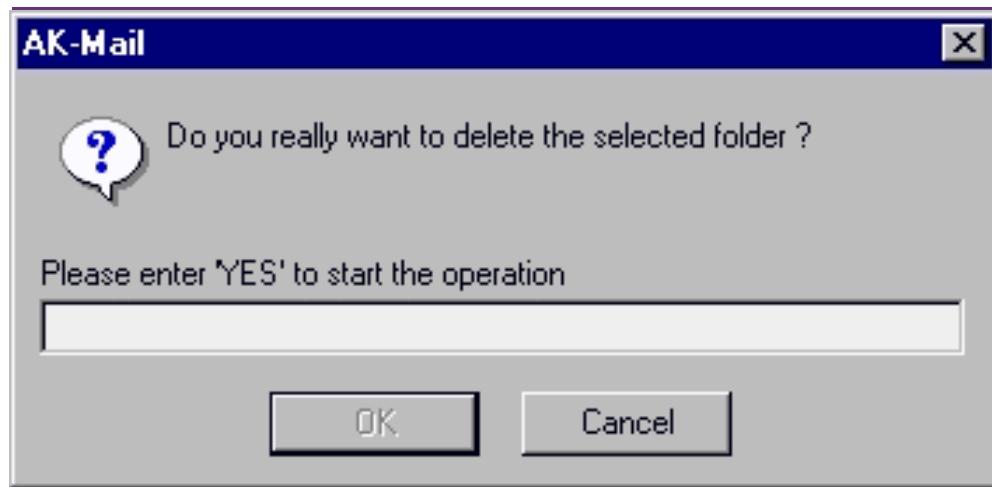
A screenshot of a highly cluttered and confusing website layout, likely from the early 2000s. The top navigation bar is filled with numerous categories in a grid-like pattern, including Auctions, Art & Collectibles, Books, Music, DVD & Video, Electronics, Software, Tools & Hardware, Toys & Video Games, and more. Below this is a secondary navigation bar with links like WELCOME, HOW TO ORDER, GIFT IDEAS, DEALS OF THE DAY, COMMUNITY, and FREE E-CARDS. A central message reads "Hello, Dack Ragus. We have recommendations for you in Books, Music, and more." To the left, there's a sidebar with a search bar and a browse menu. The main content area features a large "amazon.com" logo, a "New for You" section with a "Dack Ragus" recommendation, and a "Relive the Saga" section for Star Wars products. The overall design is chaotic and lacks a clear visual hierarchy.



7. Usability Poor Examples



7. Usability Poor Examples



Thank you!

JavaScript



Thank you for your interest in browsing our catalog! It's Easy and it's Efficient! Adobe Acrobat Reader 4.0 uses a 'Pointing Finger' with a 'W' for a mouse pointer whenever you encounter an area where a 'Selection' can be made. When the catalog index page appears, you will notice that the 'Pointing Finger' will appear when you pass over an index item (Product Type) that is selectable. If you click on an item, the pages related to that product will be downloaded to you. Each page has been modularized so that typical download times with a V.90 modem will not exceed 60 seconds with the average download time less than 20 seconds. Depending on your Browser, you may not see a timeline, just be patient and the pages will appear. In some cases another index page will appear requiring further selection. The same process should be followed. Using the pager in Acrobat Reader is easy and efficient and in a short time you will be an expert at it. To return to the previous index, simply click your Browser 'Back' button. Two other configurations of mouse pointers are also used by Acrobat Reader. An 'Open Hand' for moving the page around and a 'Magnifier' for zooming in and out while viewing the page. You may select either one from the tool bar at the upper part of the screen. Please carefully jot down the Model Numbers of interest so that they can be entered accurately in the on-line ordering system.

How do you evaluate usability?

- Formative evaluation
- Summative evaluation

Formative Evaluation

- Let the users experience prototypes and identify usability problems
- Users provides feedback based on a review of functionality and interface
- Takes place during development
- Types of formative evaluations:
 - Users review the product and influence the final outcome
 - Evaluation by HCI experts
 - Heuristic evaluation, Cognitive walkthrough - learnability

Summative Evaluation

- Takes place post development:
 - Via lab experiments ...
experts observe users using
the interfaces through *one-way mirrors*
- Quantitative results collected
 - The current usability of an
interface is measured by
things like task times,
completion rates and
satisfaction



Usability measures

- Time to learn
 - How long does it take to learn the task
- Speed of performance
 - How long does it take to perform the task
- Rate of errors by users
 - How many errors and what kinds of errors are made?
- Retention over time
 - Frequency of use and ease of learning help user retention
- Subjective satisfaction
 - Allow for user feedback

Usability testing with eye-tracking

- The process of measuring the point of users' gaze
- Special devices are used to track user's eye movements as users use software
 - from headsets to simple web cams
- Produces a “**heatmap**” that shows how long users looked at each section of the screen

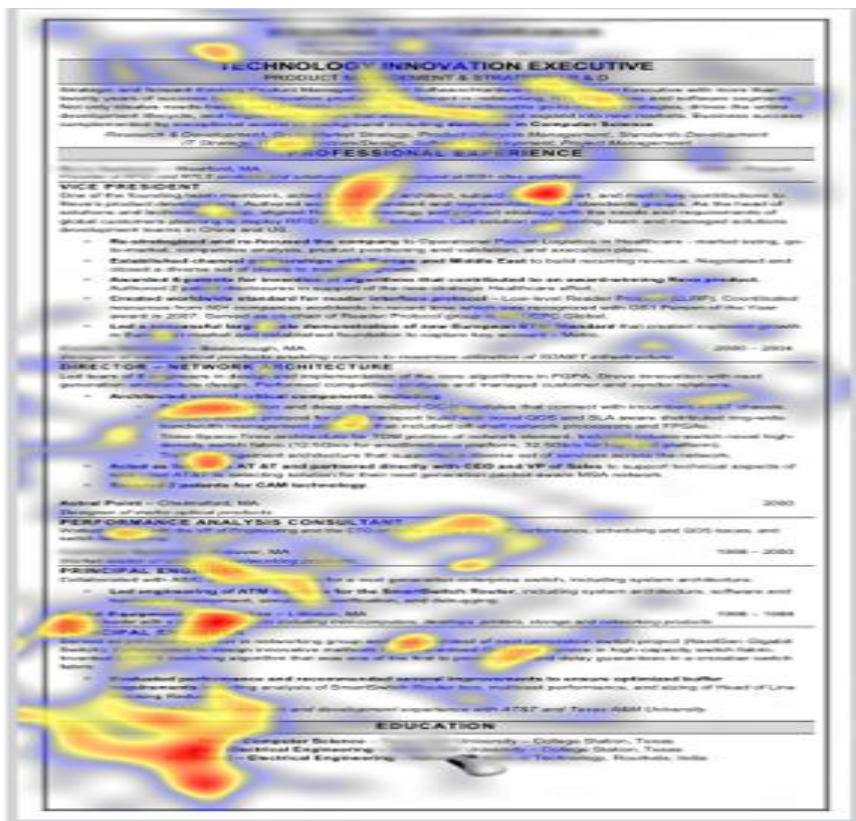


www.sr-research.com

www.tobi.com

Sample “heatmaps” of recruiters reviewing resumes

- In the six seconds they spend on a resume, recruiters focus on name, current title and company, current position start and end dates, previous title and company, previous position start and end dates and education



Sample “heatmaps” of how people look at your Facebook

- When potential dates, employers and friends glance at your online social profiles, what do they see?
- [EyeTrackShop](#), a startup runs eye-tracking studies for advertisers. They find the following:
 - profile pictures matter
 - who you know gets noticed
 - content on top wins
 - the further something is down a page, the fewer number of people look at it.



Do a five second test

- Five second tests help a software developer understand people's first impressions of their designs.
- By finding out *what a person recalls about your design in just 5 seconds*, developers can ensure that their message is being communicated as effectively as possible.
- <http://fivesecondtest.com/>

How much does **Usability cost?**

- Cost: Best practice - spend **10% of project budget** on usability
 - **More than doubles** a web site's desired quality metrics
 - **Slightly less than doubles** an intranet's quality metrics
- Benefits :
 - For internal users,
 - Cuts training budgets in half or more
 - Reduces bottlenecks by enabling more non-specialized personnel to perform duties
 - For external users
 - Doubles (or more) the number of registered users
 - Doubles (or more) number of products ordered

Workshop Preparation

Focus on Assignment 2 and working
collaboratively with your team

Thanks for watching
Hope you are enjoying your
break

Resources:

Prescribed text:

- Not covered in the text

Other resources:

Prototyping

- <http://scottberkun.com/essays/12-the-art-of-ui-prototyping/>

Usability

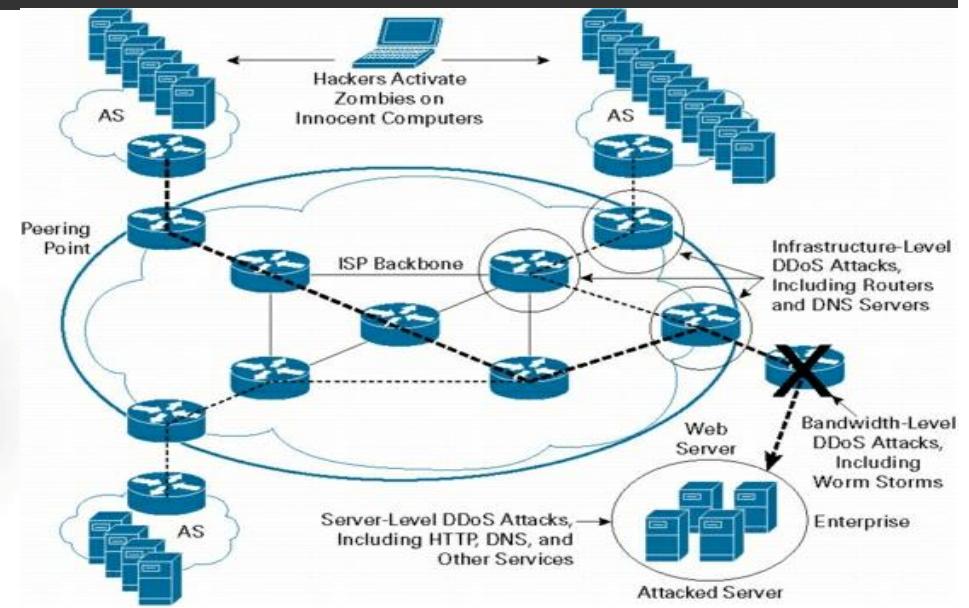
- <http://www.nngroup.com/articles/usability-101-introduction-to-usability/>



FIT2001 – Systems Development

Seminar 7.2 Design Overview

Chris Gonsalvez

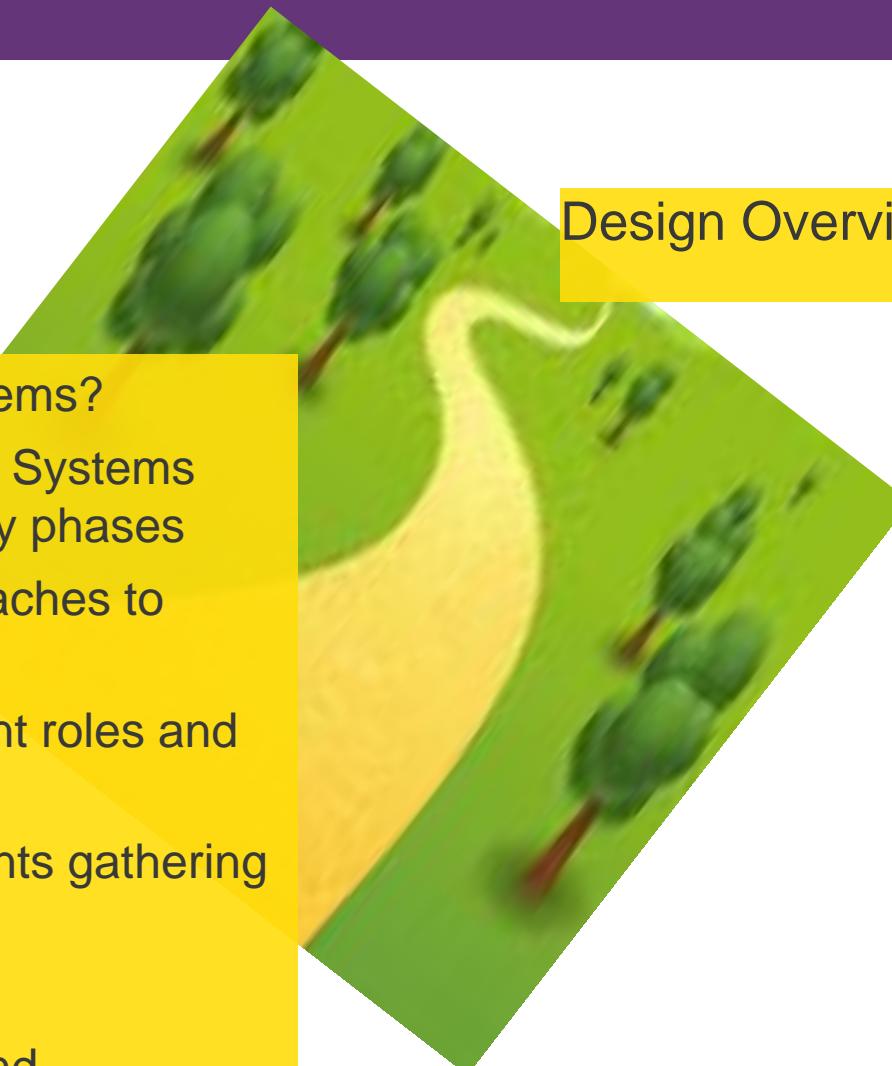




Our road map:

Design Overview

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering and documentation techniques
- Prototyping & Usability





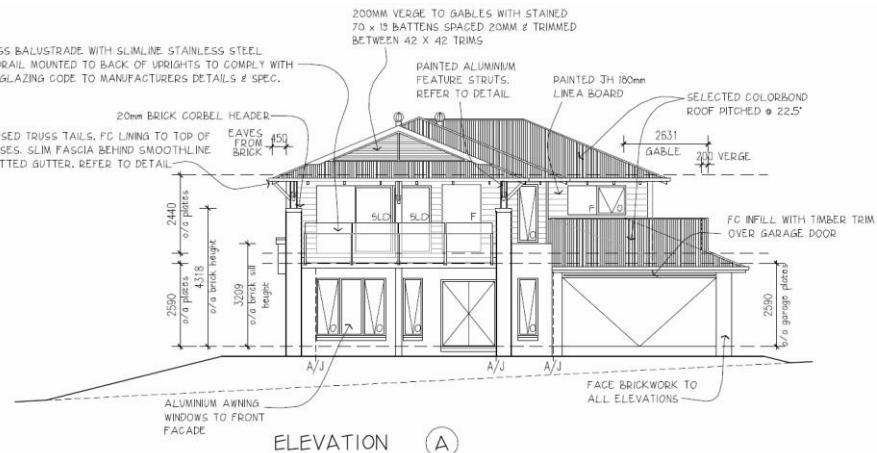
At the end of this seminar you will be able to:

- Describe the difference between systems analysis and systems design
- Understand broadly each major design activity

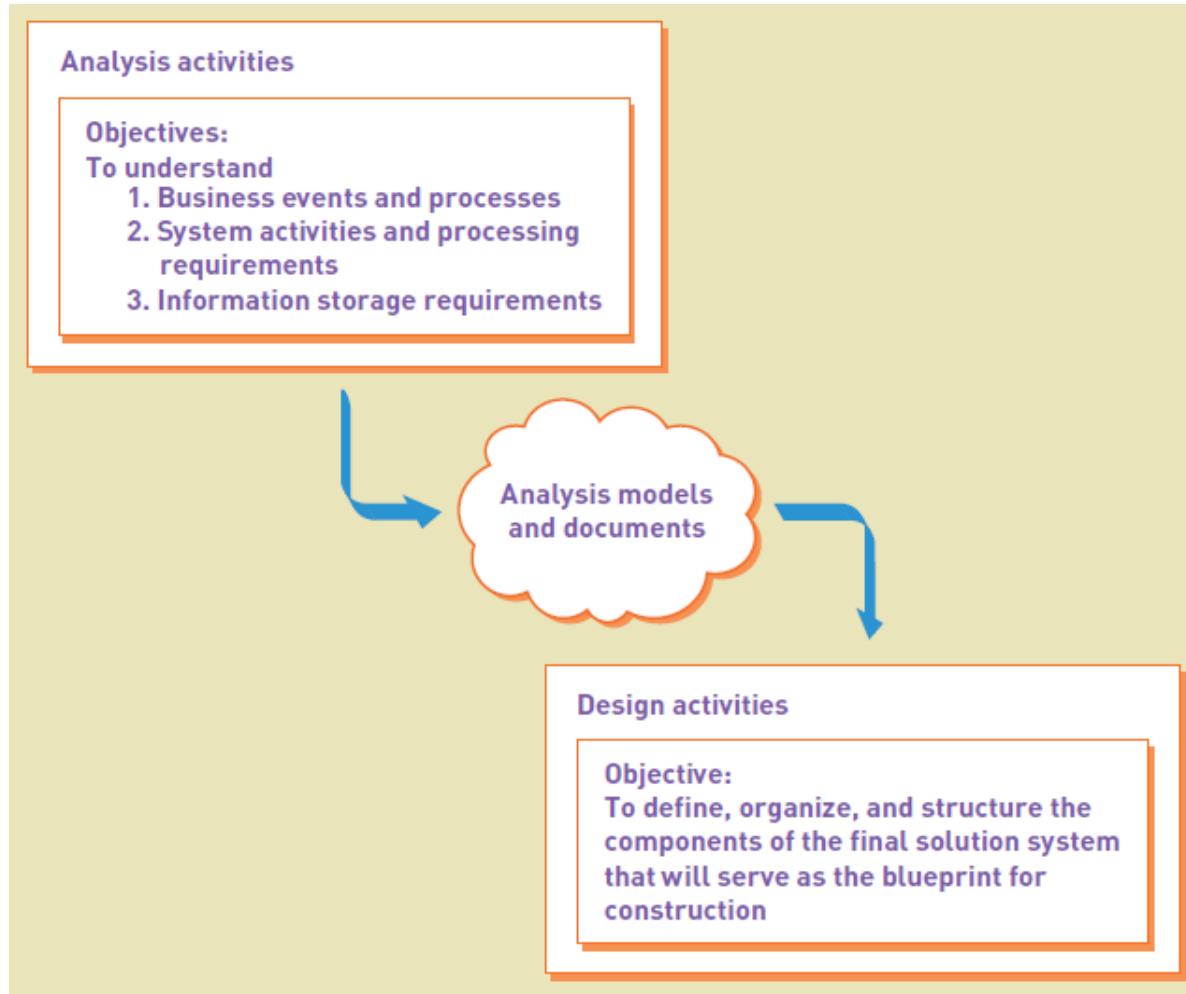
What is Design?

- The bridge from REQUIREMENTS to SOLUTION
- Focuses on:
 - HOW the system will be built
 - *Unlike Analysis which focuses on WHAT the solution needs to do*
 - What the structural components of the new system will be

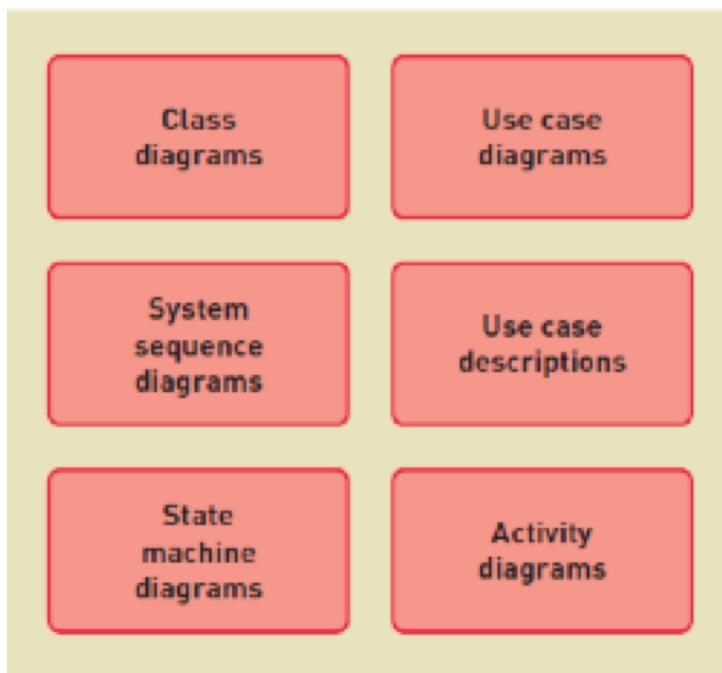
..... a blueprint for
development



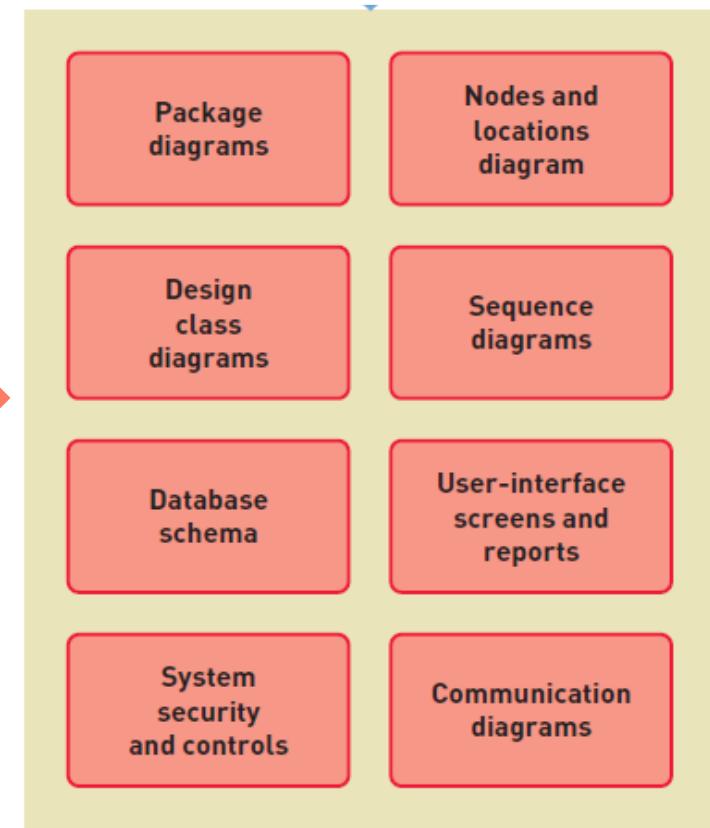
Inputs and Outputs of the design process?



Analysis Models



Design Models



Levels of Design

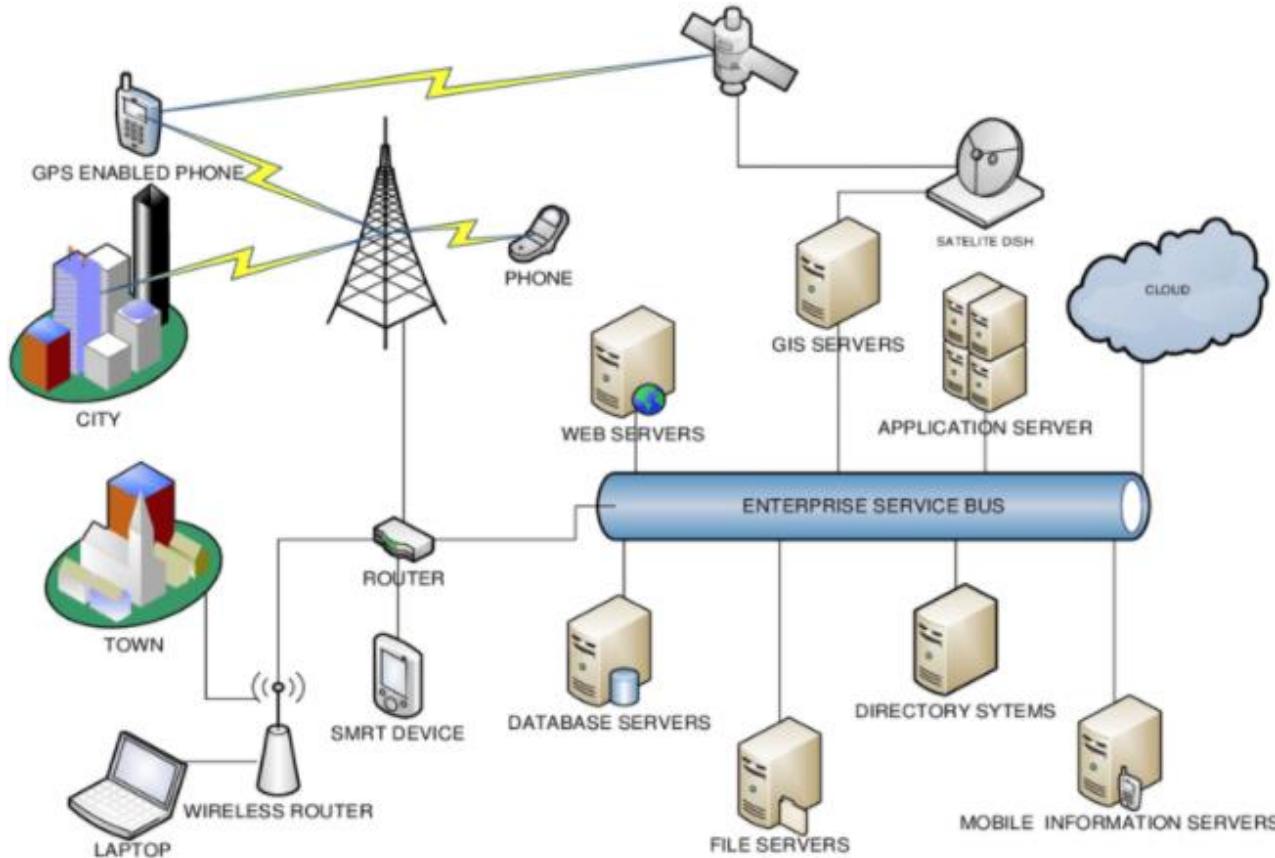
- Architectural Design *called General or Conceptual Design*
 - Broad design of the overall system structure
- Detailed Design
 - Low level design that includes the design of the specific program details
 - Design of each use case
 - Design of the database
 - Design of user and system interfaces
 - Design of controls and security

Design the Environment

How will the new system interact with other systems and with the organisation's current computer system architecture?

- Done at an organizational level
 - ... individual designer will not have control
- The new system either adapts to the current environment or if needed the designer requests changes to the environment

Design the Environment



Covered in detail in FIT1047 Introduction to computer systems, networks and security

Design the application components

What are the key parts of the information system and how will they interact when the system is deployed?

- Application component – a unit of software
 - Can vary in size
 - Programming language chosen will affect what the components are
 - How will they interact with the technology to meet functionality
 - Can be built or bought
 - Purchased separately or made available by a third party provider just as AusPost for tracking

Design the User Interfaces

How will users interact with the new system?

Covered in detail in Seminar 8 and you can choose to do an elective FIT3175 – Usability



Design the Database

How will data be captured, structured and stored for use by the new system?

***Covered in detail in FIT2094 or FIT3171
Databases and FIT3176 or FIT2104 for
advanced database electives***



Design the Software Classes and Methods

Detailed description of how the software works so coding can be done

Models created in analysis are extended to include software specific elements, and additional models such as sequence diagrams are created (*covered in Seminar 9*)



Design System Controls and Security

How will you mitigate the wide range of risks so that your system is safe and functions as expected

Covered in detail in Seminar 10

Workshop Preparation

Focus on Assignment 2 and working
collaboratively with your team

Thanks for watching
Hope you are enjoying your
break

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 6



Information Technology

FIT2001 – Systems Development

Seminar 8: Designing the User Interface (HCI)

Chris Gonsalvez

The image displays two hand-drawn wireframe sketches of user interface designs.

Left Sketch: A desktop application window titled "iTunes". It features a "Library" tab with a yellow sticky note that says "Store and Playlist go in here too.". Below it is a table showing music tracks:

Name	Time	Artist	Album
All We Ever Look For	3:49	Kate Bush	Never For Ever
Egypt	4:12	Kate Bush	Never For Ever
The Wedding List	4:16	Kate Bush	Never For Ever
Violin	3:16	Kate Bush	Never For Ever
The Infant Kiss	2:50	Kate Bush	Never For Ever

Right Sketch: A web-based application window titled "MobileDynamics". The main title bar says "Student Info > Jason Smith". The page shows student details (First Name: Jason, Last Name: Smith), addresses (Home and Work), and phones (Email and Home). On the right is a photo of a smiling man. Below this is a "Communications" section listing interactions like "Send transcripts for Jason" and "Jason called regarding his transcripts". At the bottom are sections for "Mailings" and "Email Response", "Email Student", "Retail Invoice", "Student Invoice", "Master List", and "Button".

Our road map

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- A range of Requirements gathering and documentation techniques
- An overview of Design activities
- Designing systems that our clients want - Usability

Designing systems that our clients want

- Interface Design principles
- Interface Design Tips



At the end of this topic you will be able to:

- Use the persona method to explore users of the system to assist with interface design
- Understand the principles of good interface design
- Use best practice tips for interface design

What is a user persona?

A user persona should be a fictional individual used to represent the demographics, goals, values and pain points of a large proportion of a product's user group - correct answer. The primary persona should represent your largest group of users as your system should primarily cater for them.

It should not be the description of a single important user's behaviour and personality regardless of how important they are as it should represent the majority of your users.

It should not be about the behaviours, characteristics and goals which the UX designer hopes from a group of users. What the designers hope for is irrelevant - the persona should be based on the representative behaviours, characteristics and goals of the users gathered via suitable research methods.

It should not be based on the collection of behaviours, characteristics and goals which belong to a small group of your actual users, as you would then have way too many personas to use and it would be impossible to design and interface to meet all the competing needs of small groups of users.

Lecture Outline:

1. Design Introduction
2. User Interfaces – Overview
3. UI Design Guidelines
 - 3.1 User centred approach, Personas
 - 3.2 Ben Shneiderman: 8 Golden Rules
 - 3.3 Jakob Nielsen: 10 heuristics for Interface Design
 - 3.4 Metaphors
 - 3.5 Affordance & Visibility
 - 3.6 Example
4. Practical Interface Design Tips to review in your own time

User Interface (UI) - Definition

- “All components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system.”

ISO 9241-11

- “That part of a computer system with which a user interacts in order to undertake his or her tasks and achieve his or her goals”

Stone et al, User Interface Design and Evaluation, 2005

User interfaces

- Usability can be drastically enhanced by carefully designing user interfaces
- User interface
 - require human interactions
 - varies depending on:
 - purpose (input, dialog box, report)
 - user characteristics (users with disability, novice/experienced)
 - device (e.g. mobile phone screen size)

Guidelines for designing UI

A wide range of guidelines available. Some important ones:

- User Centred Design
- Ben Shneiderman – The eight golden rules of interface design
- Jakob Nielsen – 10 usability heuristics for interface design
- Donald Norman's guidelines based on Affordance and Visibility

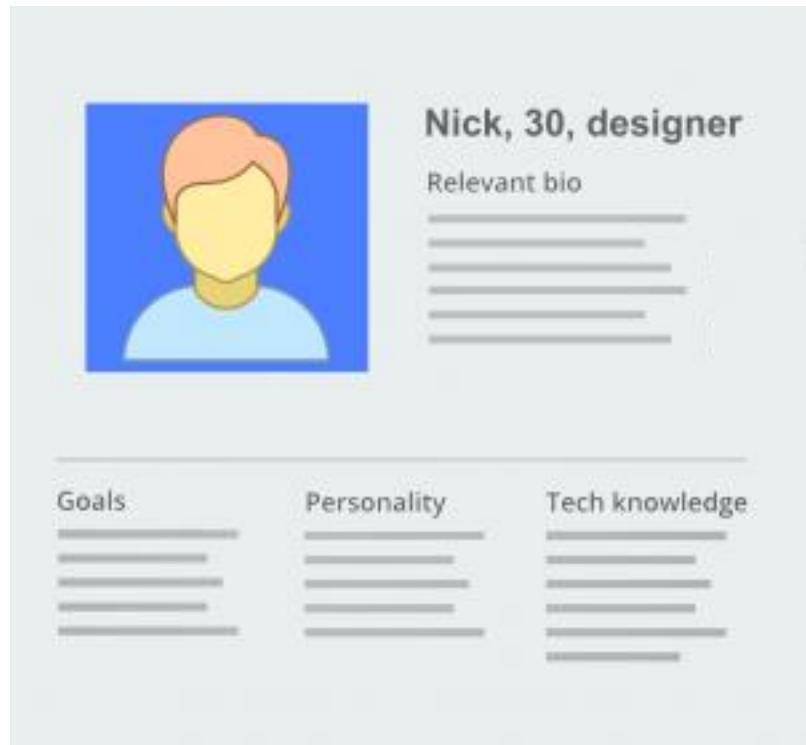
Use a User-centred design approach

Three key principles:

- Early and continuous focus on users and their work ... Personas are a useful tool
- Evaluate all designs to ensure usability
 - Use prototypes to observe behaviour
- Design iteratively

Personas - Designing for your users

- The personas method allows you to explore the psychology of an imagined user's interaction with the product.
- Creating products for specific NOT generic users, provides a clear vision rather than unfocussed goal



What is a Persona?

- “Archetypical descriptions of user behaviour patterns into representative profiles, to humanise design focus, test scenarios, and aid design communication.” Cooper, A. (2004)
- Create representations of key audience segments for reference throughout the design process

How do you create Personas?

1. Collect information about your users
 - understand the target audience's mindsets, motivations, and behaviours ..
Use research - interviews, workshops, questionnaires
2. Identify behavioural patterns from research data
 - find patterns in data to help group users
3. Create personas and prioritise them
 - assemble personas around patterns – add just enough detail to characterise the user base
 - if you have multiple personas define the primary persona (the most relevant) and follow the rule "*design for the primary – accommodate the secondary.*"
4. Identify relevant scenarios for the personas
 - by pairing the personas with the scenarios, you can gather requirements and design relevant solutions
5. Share your findings and socialise personas among stakeholders
 - team should see the value in them, they should be front and centre of the design process

Persona – What information do you include?

- Persona Name and photo
- Demographics - Gender, age, place of residence, Profession and field of work, Marital status, Financial status – fictional personal details to make more realistic
- Personality - Hobbies, Favourite brands, Do they follow trends?, Media consumption habits, hours spent online, What kind of gadgets do they use and how? Quote or slogan that captures the personality
- Behaviour patterns, Goals, Skills, Attitudes, Frustrations or pain points, Environment they operate in
- Product context information - Do they have previous knowledge about the product? In what context do they use the product? What are their motivations? Why would they use the product? Context specific details eg. For a banking app financial details

Persona example for Travel booking site



Amelie Alexander

Looking for a site that will simplify the planning of my business trips.

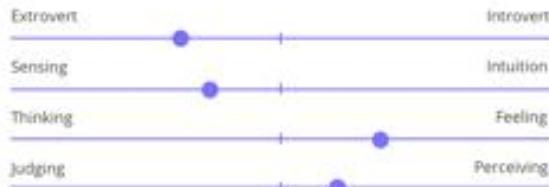
Demographics

Age :	35
Occupation :	Regional Director
Status :	Married
Location :	United States
Tier :	Frequent Traveler
Archetype :	The Planner

Bio

Stella is a Regional Director who travels 2-3 times each month for work. She has a specific region in which she travels to & often visits the same cities and stays in the same hotel. She's frustrated by the fact that no matter how frequent she takes similar trips, she spends hours of her day booking a reservation. She expects her travel solutions to be organized as she is.

Personality



Goals

1. To spend less time booking a reservation.
2. To save her searches & favourite hotels.
3. To narrow her options when needing to search.
4. To see recommendations based on her past bookings and interests.
5. To book the reservation through customised budget plans.

Type of Travel Planner



Frustrations



Too much time spent on booking.



Always need to search for favourite hotels.



Planning the proper timeline for the total journey.

Motivations



Why are Personas important?

Build Empathy

- Helps users seem more real - designers empathise and build for their users

Provide Direction For Making Design Decisions

- Helps focus design decision on users – don't build it for yourself or a generic user

Communicate Research Findings

- Team on the same page, communicates information in an easy to understand format

Why Personas can fail

Personas were created but the team did not use them

- If team have had a bad experience they see them as a waste of time and are loathe to try again

No buy-in from the team

- They think they know their users well so don't need it

Personas were not developed collaboratively

- If users and team not involved they think you have created pretty pictures of fake people

Communication failure

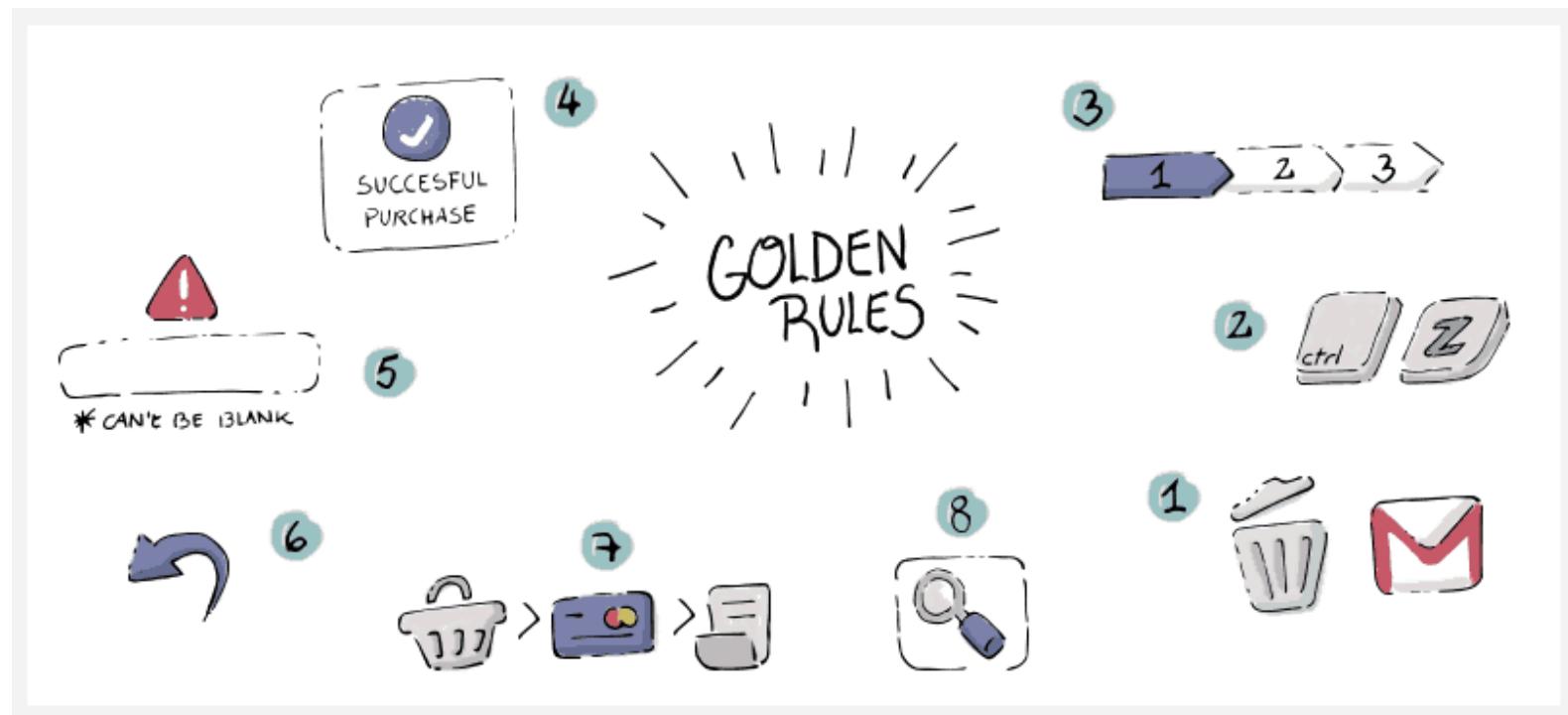
- The team does not understand what Personas are or their purpose

The personas are flawed

- The persona is not built to reflect the scope of work it is meant to impact.

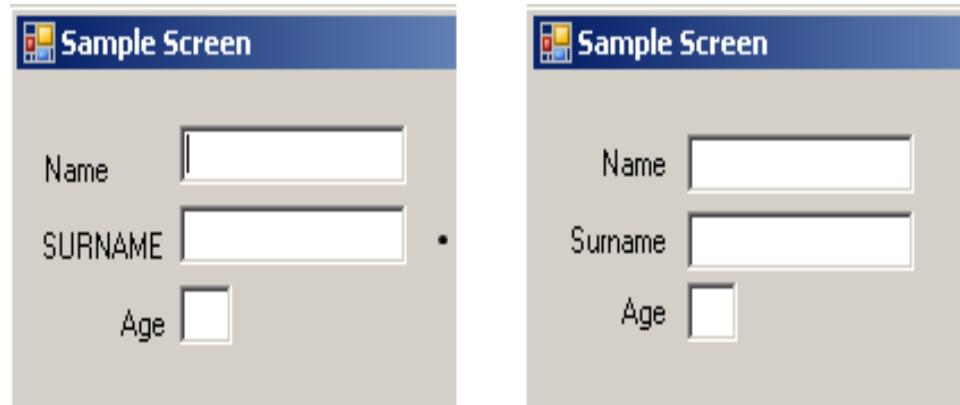
Ben Shneiderman's 8 Golden Rules

Ben was asked to distill effective user interface design into a few key principles – this was the birth of the 8 Golden rules



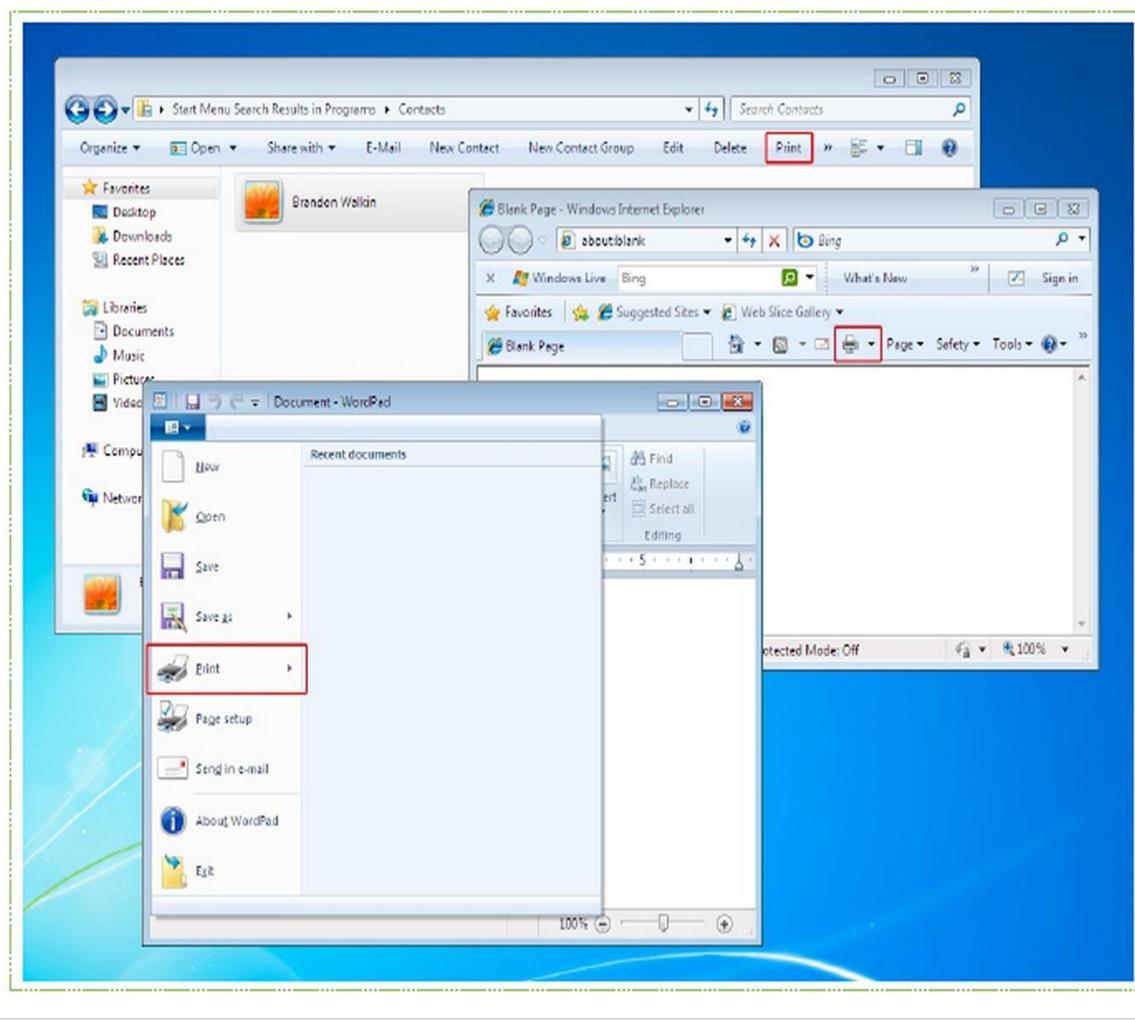
1. Strive for consistency

- Consistency refers:
 - to the way information is arranged on forms
 - the names, and arrangement of menu items
 - the size and shape of icons
 - and the sequence followed to execute tasks should be consistent throughout the system
- Inconsistency in interface results
 - Longer time to learn
 - Will be harder for users to remember



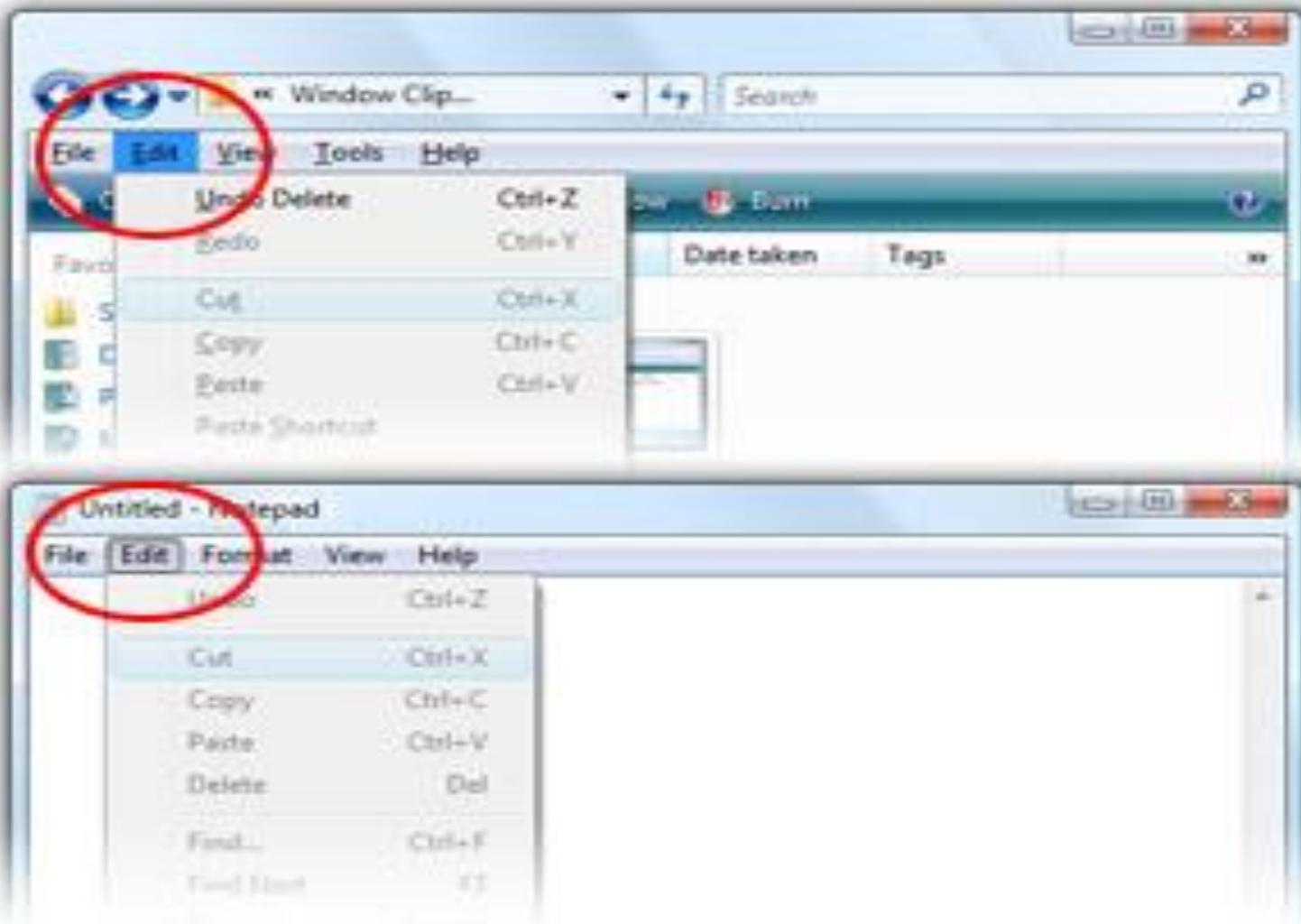
Just small changes can lead to feeling unsettled when using an interface

Inconsistent examples

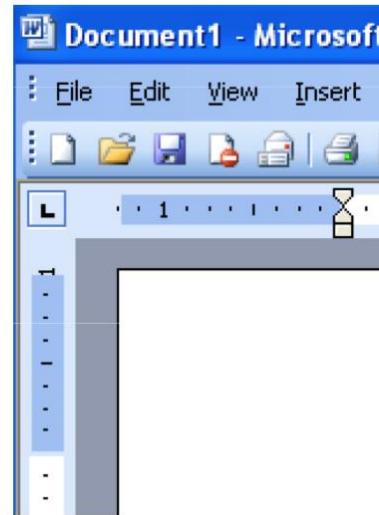
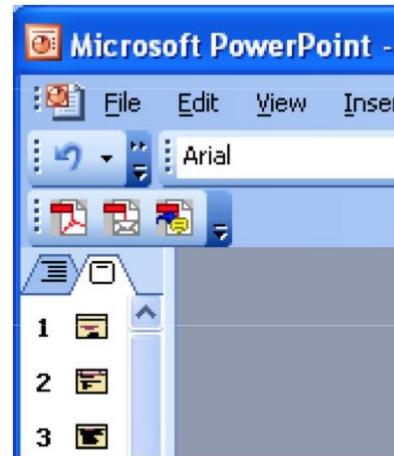
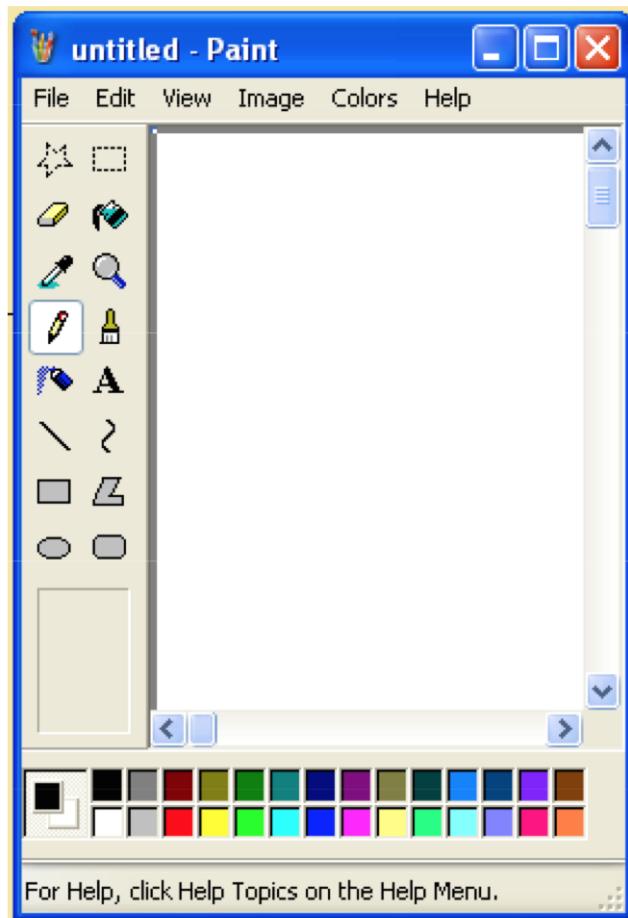


7

Inconsistent examples



Consistency example - Windows

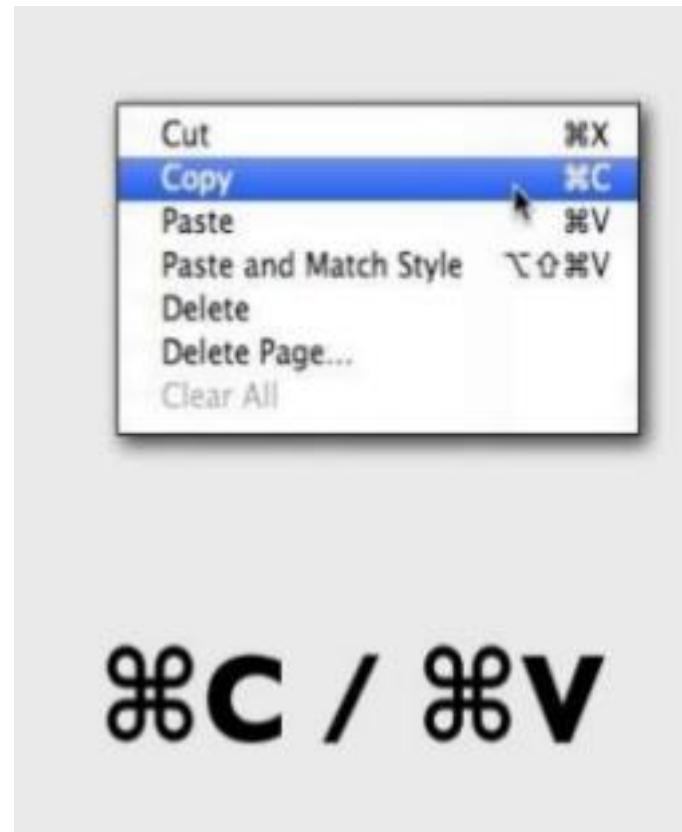
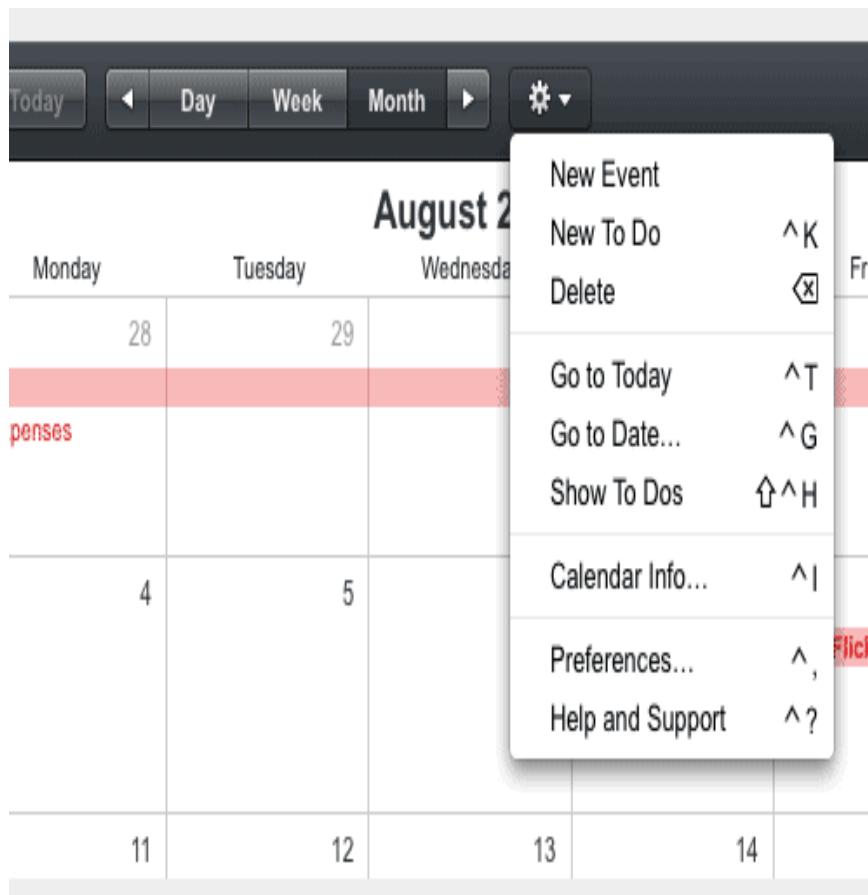


Windows consistent menus – File, Edit, View

Even for new applications user knows where to go to load and save files, cut and paste, change view, etc.

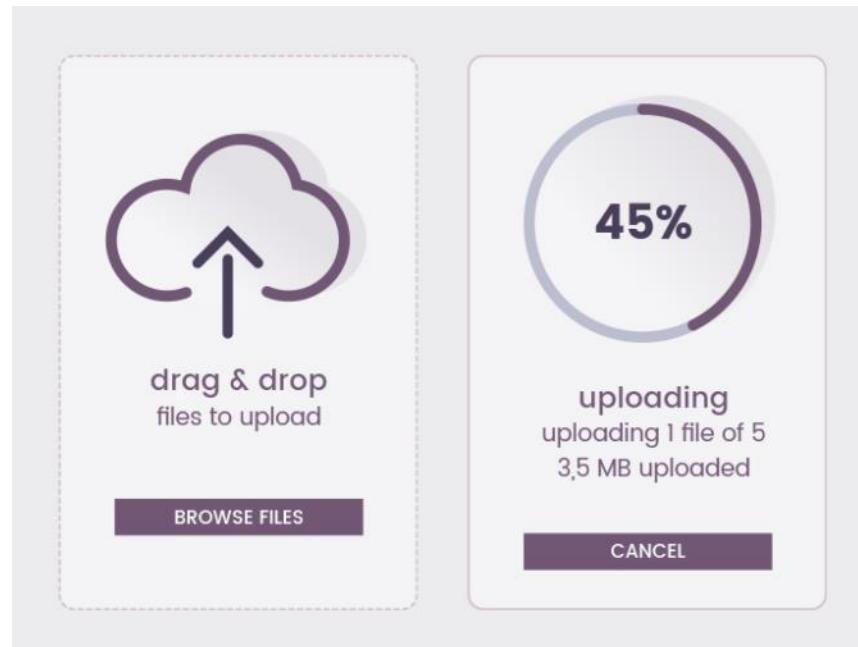
2. Cater for diverse users

Eg. Offer experienced, frequent users shortcuts



3. Offer informative feedback

- Inform user that their actions was received
- Include feedback when something is complete
- Make sure that the feedback is:
 - Informative
 - Clear
 - Concise



Feedback examples

The screenshot shows a user interface for updating a profile. At the top, there is a dark header bar with the WORKFU logo on the left and several navigation links on the right: "Update Profile" (highlighted in orange), "Message" (with a red notification badge showing '2'), a gear icon, and a "Logout" button. Below the header is a light-colored navigation bar with links: "Update your Bio" (highlighted in a white box), "Refine your Keywords", "Add other Networks", "Reading List", and "Preview your Profile". A green banner across the middle of the page displays the message "Your profile has been updated successfully". On the left side, under the heading "Edit your profile", there is a form with the placeholder text "Use the form below to edit and manage your personal details.". To the right of the form is a user profile card for "Winnie Lim". It features a thumbnail image of a woman, her name "Winnie Lim" in bold text, and a "Change" button at the bottom.

Feedback examples



4. Design dialogues that yield closure

- Organise sequences of actions:
 - Beginning
 - Middle
 - End

- Ensure that users know when a conversation or task is at end:
 - Users should know when a task is completed
 - User should be put at ease

Examples of Closure

Delivery Information is done
Then Payment Information
Then Confirmation

The screenshot shows a user interface for an Amazon checkout. At the top, there is a green header bar with the text "Examples of Closure". Below this, the text "Delivery Information is done", "Then Payment Information", and "Then Confirmation" is displayed. The main part of the screen shows a progress bar with four steps: "Delivery Information", "Payment Information", "Confirmation", and "Finished". Above the progress bar, there is a message: "Continue Checkout Procedure to confirm this order." and a red "continue" button. At the bottom of the screen, the Amazon logo is visible along with navigation links: "SIGN IN", "SHIPPING & PAYMENT", "GIFT-WRAP", and "PLACE ORDER".

5. Prevent errors

Message types

- **Error messages:** alerts users of a *problem that has already occurred*.
- **Warning:** alerts users of a *condition that might cause a problem in the future*.
- **Information:** highlights a statement or fact

Effective error messages:

- Inform users that a problem occurred
- Explain why it happened and provide a solution so users can fix the problem



Your password must have:

- ✓ 8 or more characters
- ✓ Upper & lowercase letters
- ✓ At least one number

Strength: strong

Avoid passwords that are easy to guess or used with other websites.

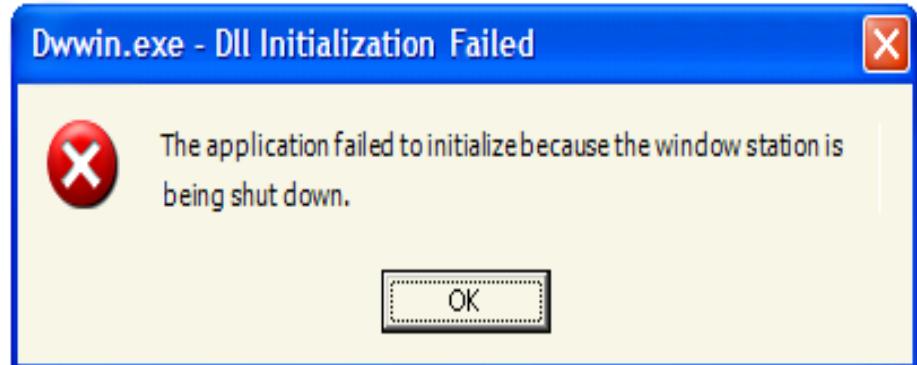
Error 404

http://www.ted.com/talks/reenny_gleeson_404_the_story_of_a_page_not_found

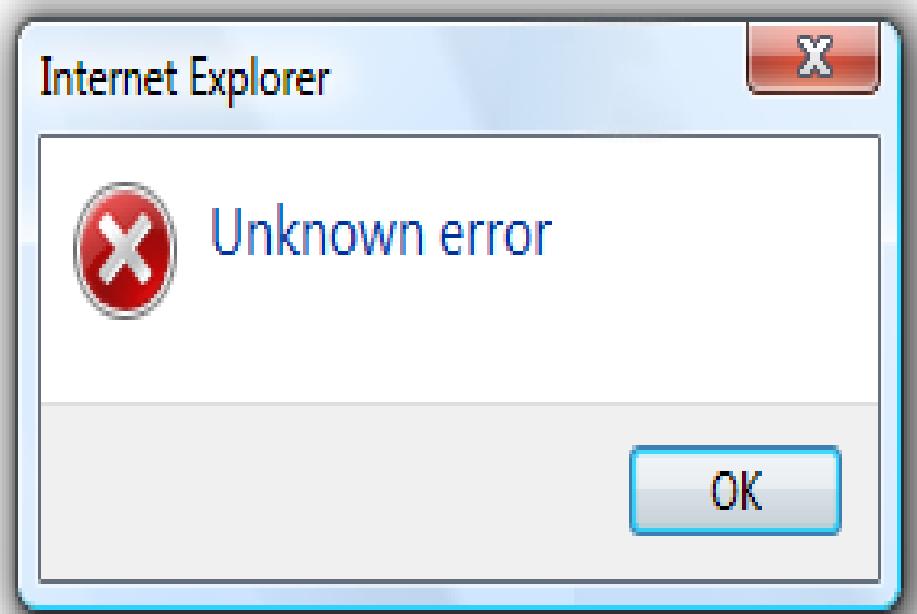
"Little things done right matter
... Well designed moments build brands"

Characteristics of poor error messages

- **Unnecessary error messages**

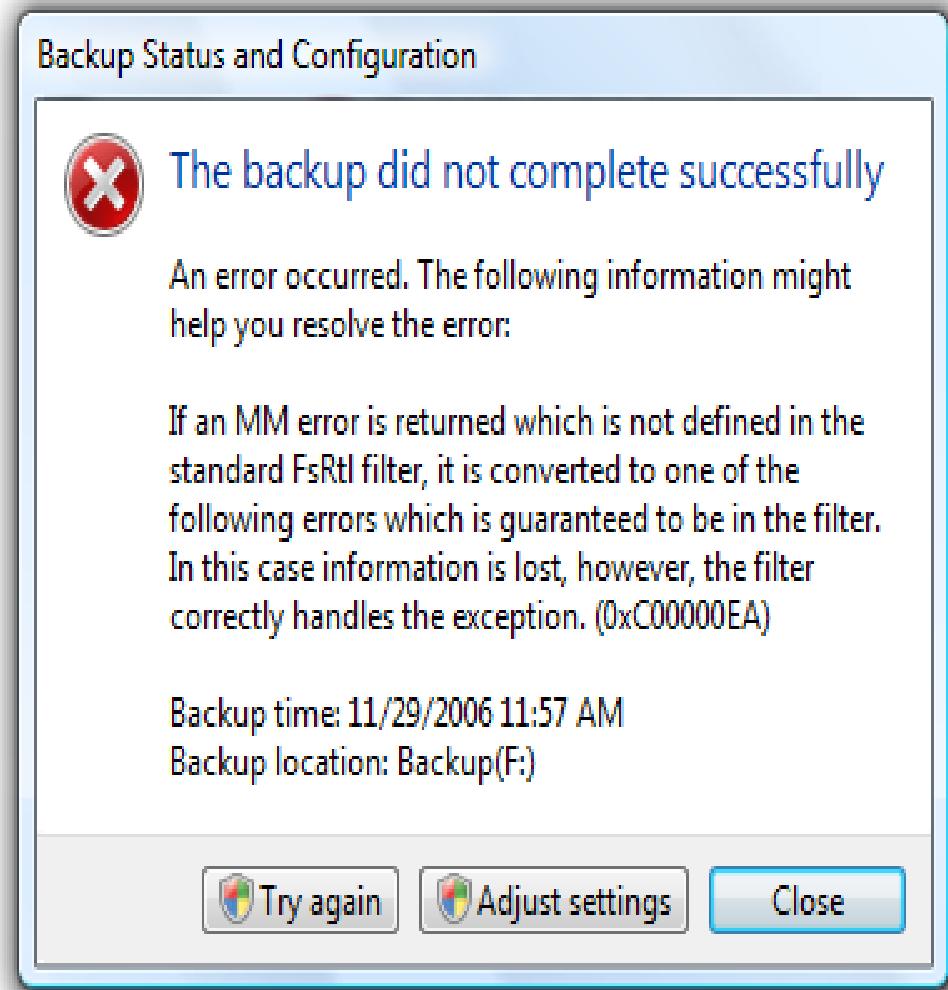


- **Meaningless message** - users learn that there is an error but:
 - has no idea what it is
 - or what to do about it



Characteristics of poor error messages

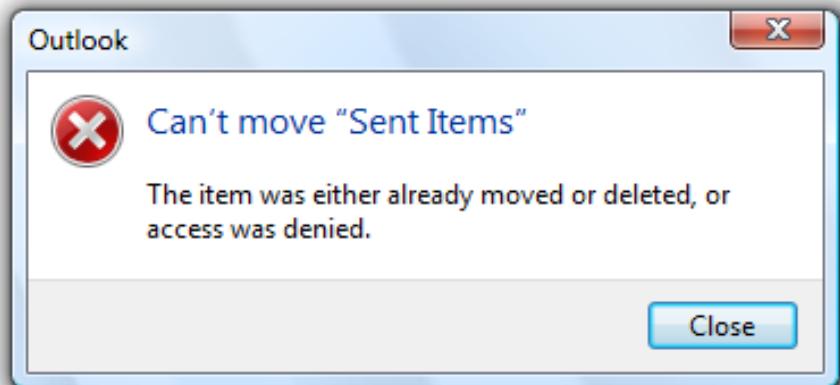
- **Incomprehensible error messages**



Characteristics of poor error messages

- Irrelevant issues in problem statement

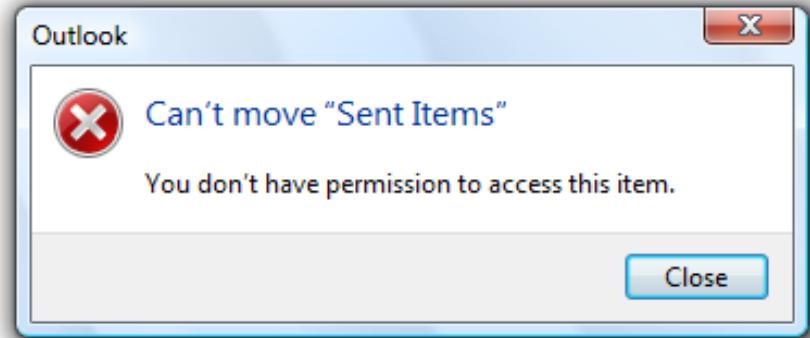
Incorrect error message



A better way:

The program can determine if access was denied, so this problem should be reported with a specific error message.

Correct error message



Characteristics of poor error message

- **Avoid over-communicating**

- Generally, users don't read, they scan
- reduce the text down to its essentials
- Should not require motivation to read

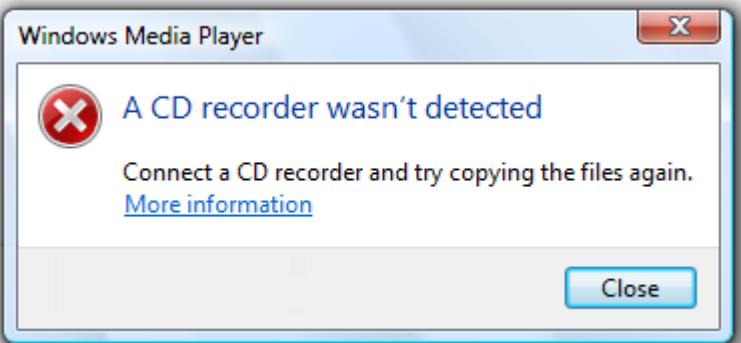
Incorrect error message



A better way:

The program can determine if access was denied, so this problem should be reported with a specific error message.

Correct error message



User input errors

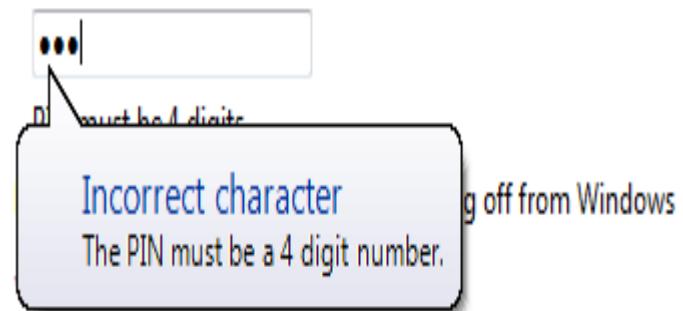
- Whenever possible, prevent or reduce user input errors by:
 - Using controls that are constrained to valid values
 - Providing good format examples

- Use **balloons** for non-critical, single-point user input problems detected while in a text box.
 - If error message placed immediately after the text box difficult to see.

Example 1: Incorrect input
(No control)

Speaker volume:

Example 2: Incorrect input



User input errors

- Use in-place errors for delayed error detection
- There can be multiple in-place errors at a time

Sign in

Please type your e-mail address in the format `yourname@example.com`.

 Windows Live ID:
(example555@hotmail.com)

Please type your password.

Password:

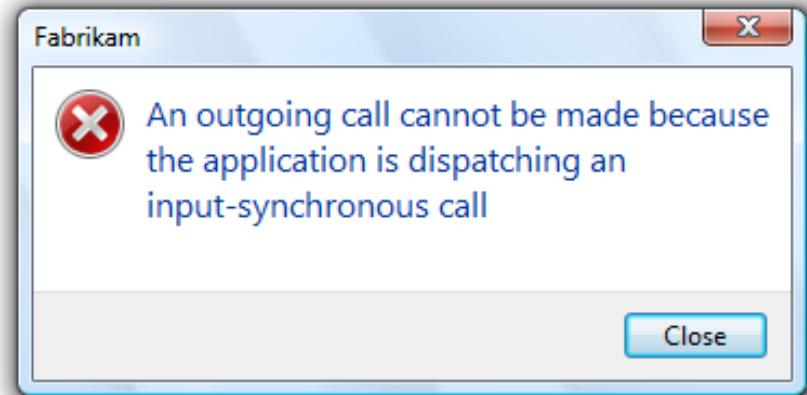
[Forgot your password?](#)

The use of sound and text in error messages

- **Sound**

- Generally, error messages should not be accompanied with a sound effect or beep
- Doing so is jarring and unnecessary

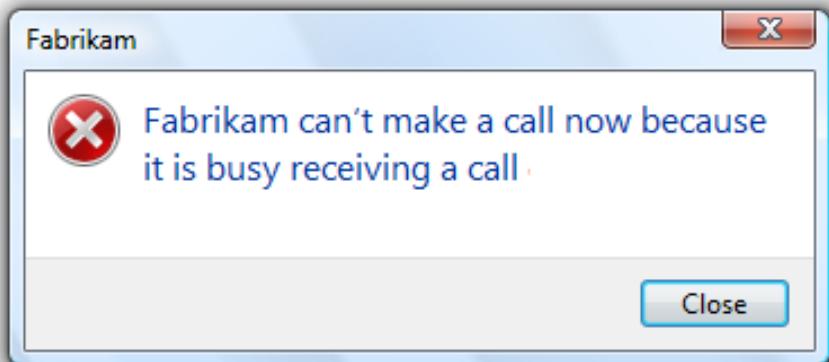
Incorrect error message



- **Text**

- Remove any redundant text
- Avoid technical jargon

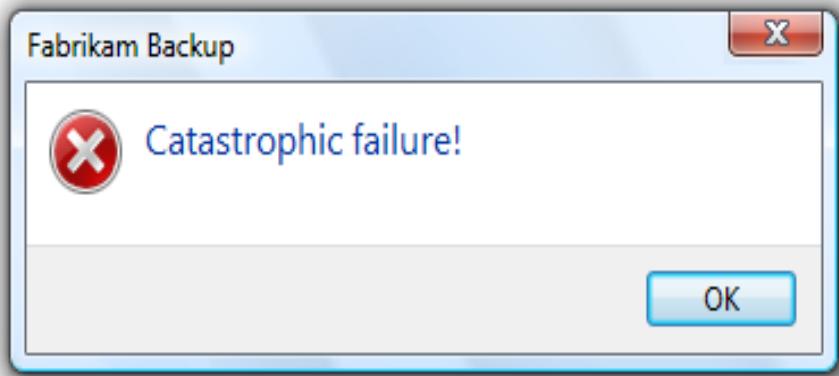
Correct error message



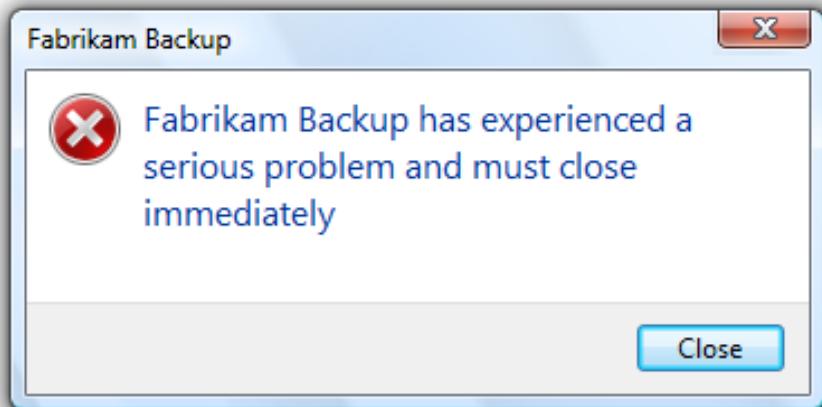
The use of text in error messages

- When designing error messages, use an **ENCOURAGING** tone
- Avoid using the following words:
 - Error, failure
.. use *problem*
 - Failed to
.. use *unable to*
 - Illegal, invalid, bad
.. use *incorrect*
 - Abort, kill, terminate
.. use *stop*
 - Catastrophic, fatal
.. use *serious*

Example: Inappropriate error message



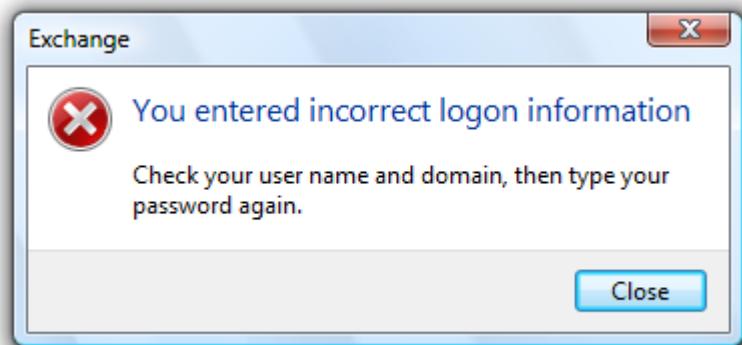
Example: More appropriate error message



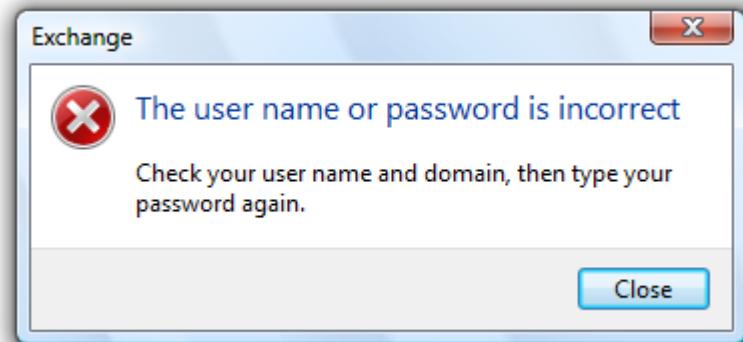
The use of text in error messages

- Don't use phrasing that ***blames*** the user
- Avoid using ***you*** and ***your*** in the phrasing.
- Use the passive voice when the user is the subject

Example: Inappropriate error message – it blames the user by using the active voice



Example: Appropriate error message



7. Reduce short-term memory load

Humans have limited capacity for processing info in short-term memory

- 7 +/- 2 chunks of information, 20-30 seconds
- Avoid interfaces where users have to remember information from one screen to the next



Hi, I'm Dory! I have short term memory loss!

Hi, I'm Dory!

Demotivation.us

Short-term memory: Implications for UI design

- Highlights where you are and shows sequence of actions

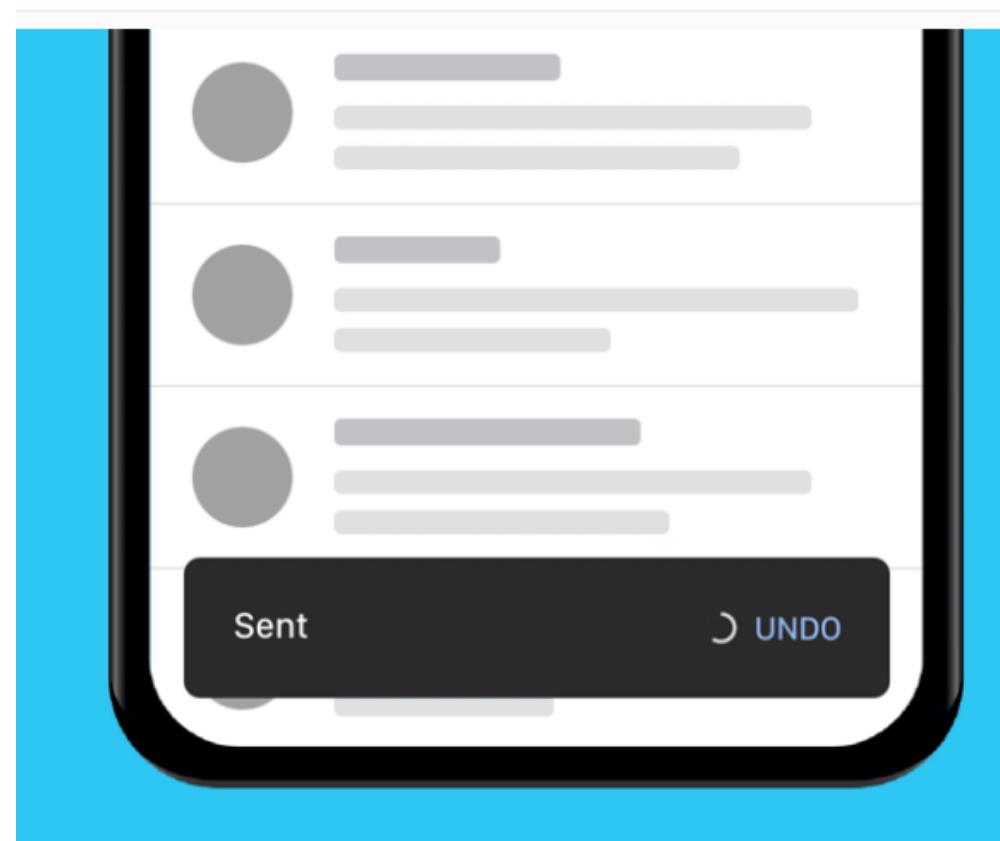
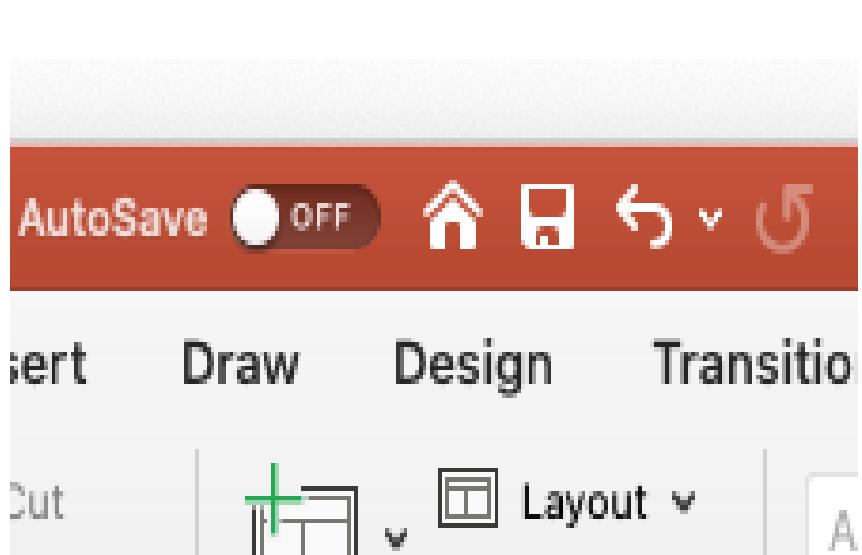
The screenshot shows a step 1 of a two-step checkout process. At the top, there are three red buttons: "cart", "checkout", and "receipt". Below them are three small icons: "safe", "easy", and "fast". The main area is titled "step 1: your email". It has a text input field labeled "Email:" and a placeholder "Please enter your email address.". Below the input field are two radio buttons: one selected for "Checkout as a Guest" and one unselected for "Create or use an Account". To the right is a table showing the purchase details:

item	quantity	price
AKG Q460 Quincy Jones -Black Code: 082-012-0462 Weight: 1.1 LB	1	\$109.00
		Subtotal: \$109.00
		Shipping & Handling: TBD
		Order Total: \$109.00

At the bottom of the page is the Amazon logo and a navigation bar with links: "SIGN IN", "SHIPPING & PAYMENT", "GIFT-WRAP", and "PLACE ORDER".

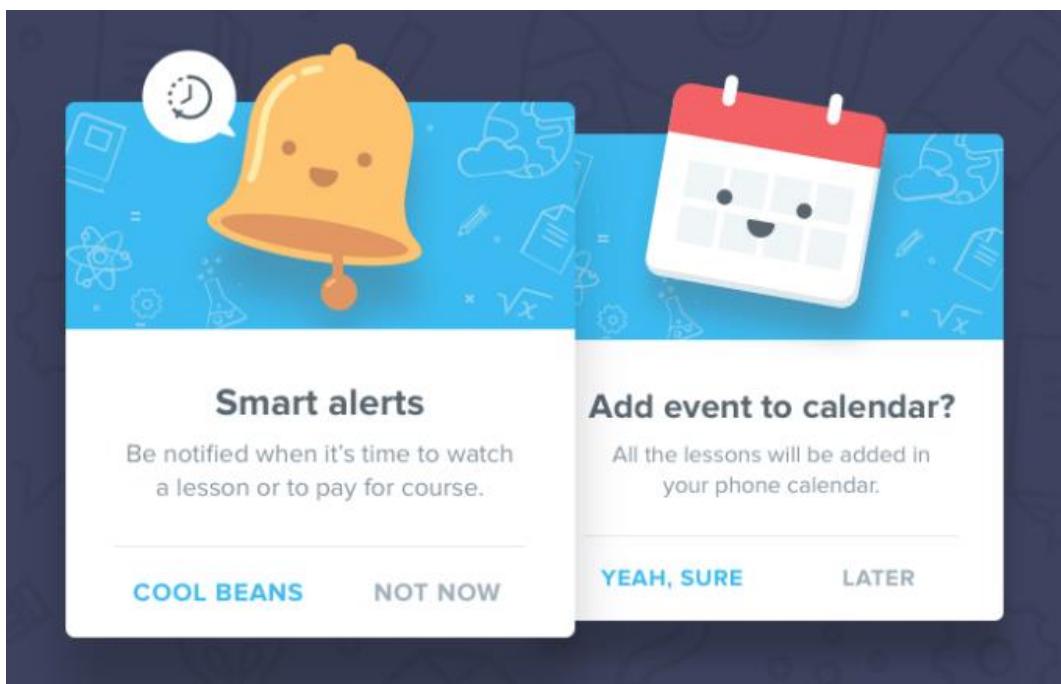
6. Permit easy reversal of actions

- Users need to feel that they can cancel or reverse an action
- Provide tools for reversal of their actions – buttons or menu bar options

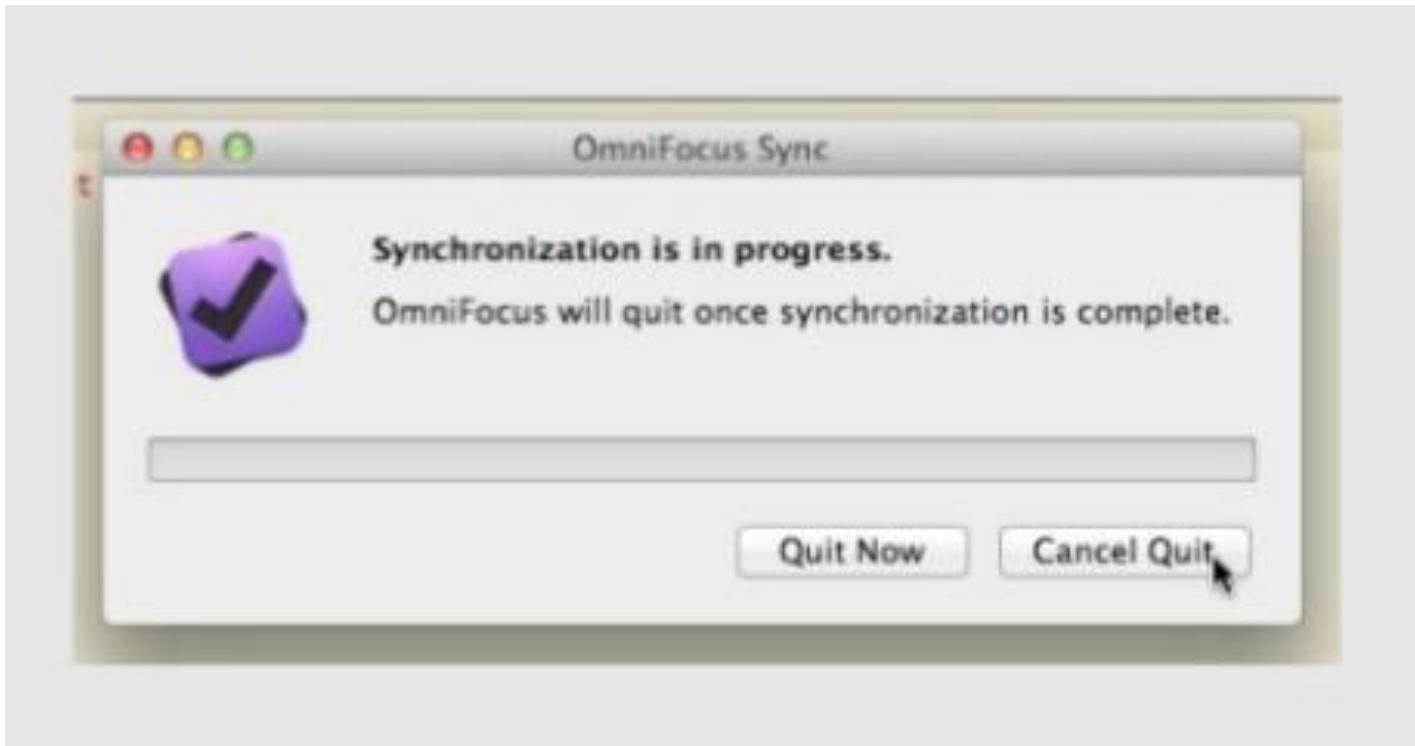


8. Support internal locus of control

- Experienced users want to be in charge while interacting with the system
- They do not like:
 - Tedious data entry
 - Surprising system actions
 - No help messages
 - Complex error message
 - Being forced to remember



The user decides what to do next



Jakob Nielsen has 10 heuristics for Interface Design (a few similar to the 8 Golden rules)

- Visibility of system status (3)
- Match between system and the real world
- User control and freedom (8)
- Consistency and standards (1)
- Error prevention (5)
- Recognition rather than recall (7)
- Flexibility and efficiency of use (2)
- Aesthetic and minimalist design
- Help users recover from errors (5)
- Help and documentation

... and some other things to think about

Match between the system and the real world Metaphors

- Metaphors are *analogies between features of the UI and some aspects of physical reality* that users are familiar with.
- Use of a concept or word from one setting (e.g., real world) to convey meaning in another (e.g., digital world)
 - physical analogies (e.g., trash, spreadsheet, file cabinet)
 - cultural standards (e.g., colour, words)
- Help reduce cognitive load for user and improves ease of learning



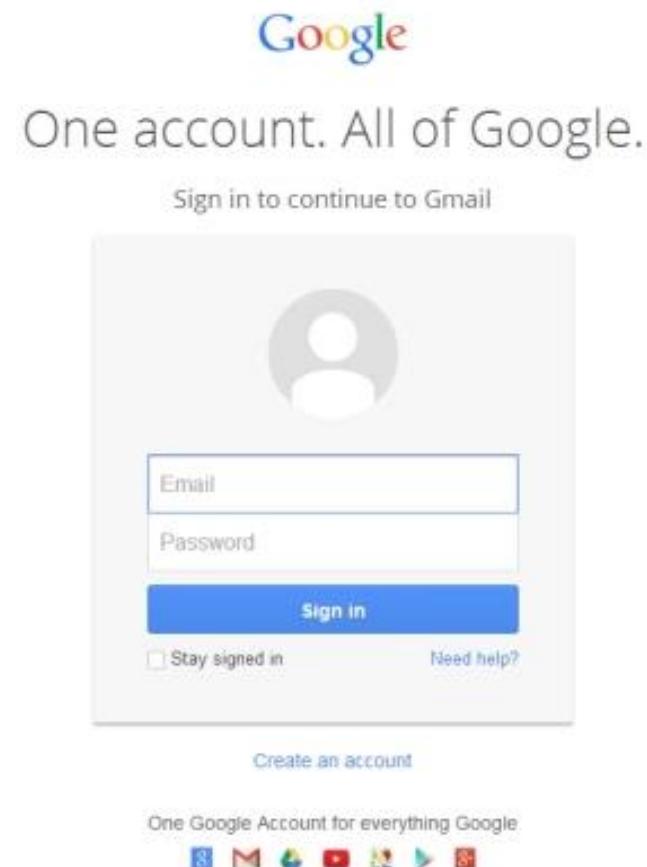
Examples of Metaphors in GUI

- MacIntosh's bin icon to delete files
- Form fill-ins (paper-based forms as a metaphor)
- Digital camera software (photo album as a metaphor)
- Tabs in a GUI (physical filing system as a metaphor)
- To-do list
- Calendar
- Shopping cart metaphor for e-commerce applications

Aesthetics and Minimalist design - Bad



Aesthetics and Minimalist design - Good



Guidelines for designing UI

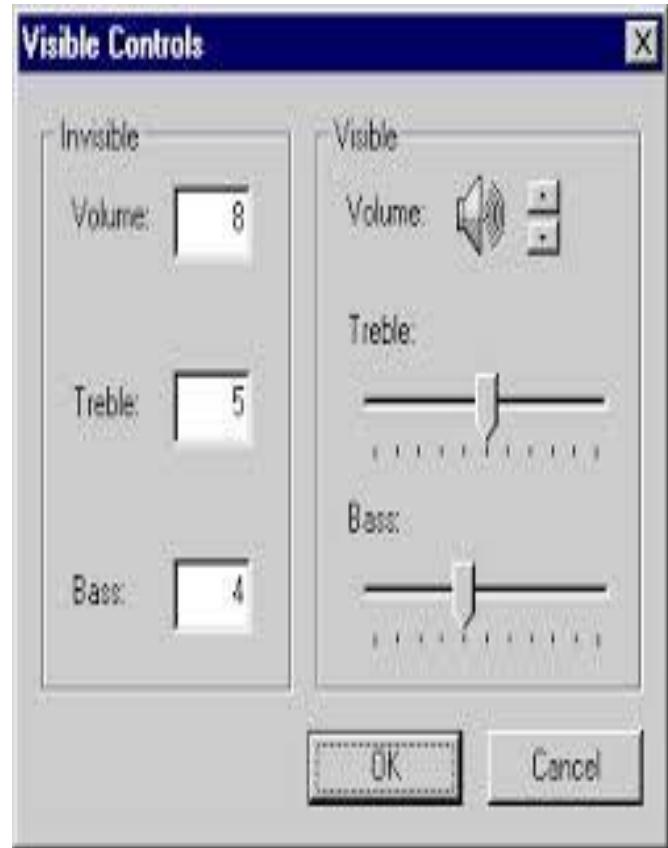
Donald Norman advises designing UI based on: Affordance and Visibility

▪ Affordance

- the appearance of a specific control should suggest the purpose for which it is used (i.e. functionality)

▪ Visibility

- The user shown know how to operate something by just looking at it
- All controls must provide immediate feedback to indicate control is responding



Assess this online form

[https://register.monash.edu.au/
enquiry/](https://register.monash.edu.au/enquiry/)

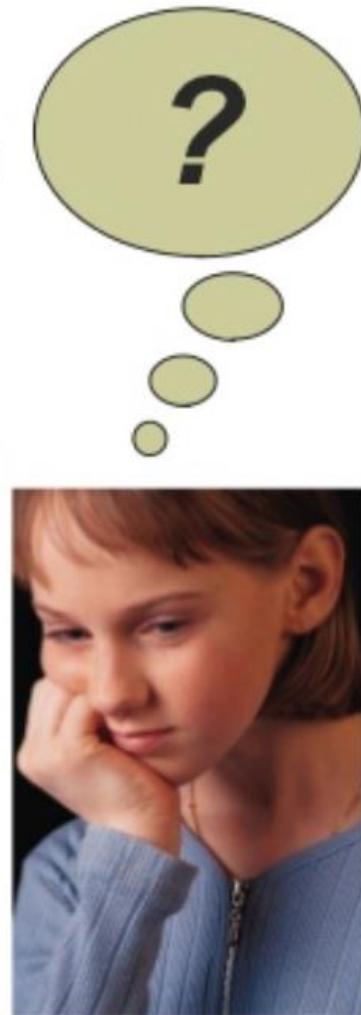


Design principles

- Minimise the pain
 - No one likes filling in forms
 - Smart defaults, inline validation, forgiving inputs
- Illuminate a path to completion
- Consider the context
 - Familiar vs. foreign
 - Frequently used vs. rarely uses
- Ensure consistent communication
 - Errors, Help Success
 - Single voice despite many stakeholders

Don't make me think

- Eliminate questions in user's heads like:
 - Why did they call it *that*? Names of things should be obvious
 - Is it clickable? Buttons should look like buttons; links should look like links.
 - How to search? – use a search box labeled Search or a box with a button that says “Search” next to it.
 - Where am I?
 - Where should I begin?
 - Where did they put _____?
 - What are the most important things on this page?



Points from Steve Krug's book: *Don't make me think*

Workshop Preparation

Make sure to look at the additional material at the end of the seminar

Thanks for watching

Resources:

Satzinger, J. W., Jackson, R.B., Burd, S.D. and R. Johnson
(2016) Systems Analysis and Design in a Changing World, 7th
Edition, Thomsen Course Technology, *Chapter 8*

- Jakob Nielsen - 10 Heuristics for Interface design
 - <http://www.nngroup.com/articles/ten-usability-heuristics/>
 - <http://www.whatwasithinking.co.uk/2009/02/27/explaining-usability-heuristics-a-quick-guide/#.Uy-NjNwVdFw>
 - <http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5>
- Ben Shneiderman – 8 Golden rules for Interface design
 - <http://www.cs.umd.edu/~ben/goldenrules.html>
 - <https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces>
- Examples: <http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5>

Practical Interface Design Tips

by Luke Wroblewski

Ref: <http://www.slideshare.net/lukew/best-practices-for-form-design-81133>

Design principles

- Minimise the pain
 - No one likes filling in forms
 - Smart defaults, inline validation, forgiving inputs
- Illuminate a path to completion
- Consider the context
 - Familiar vs. foreign
 - Frequently used vs. rarely uses
- Ensure consistent communication
 - Errors, Help Success
 - Single voice despite many stakeholders

Top aligned labels

- When data being collected is familiar
- Minimize time to completion
- Require more vertical space
- Spacing or contrast is vital to enable efficient scanning
- Flexibility for localization and complex inputs

Vertical Labels

The image displays two wireframe interface designs for 'Vertical Labels'. Both designs feature a 'Label' input field at the top, followed by a dropdown menu labeled 'Longer Label' with the placeholder 'Select Value'. Below the dropdown is another input field labeled 'Even Longer Label'. At the bottom of each design is a radio button group for 'One More Label' with options 'Value 1' and 'Value 2', and a yellow 'Primary Action' button.

Advantage: Rapid Processing

Advantage: Adjacent Label and corresponding Input field

Disadvantage: Increased vertical space

Right aligned labels

- Clear association between label and field
- Requires less vertical space
- More difficult to just scan labels due to left rag
- Fast completion times

Right-Justified Horizontal Labels

Label

Longer Label

Even Longer Label

One More Label Value 1
 Value 2

Primary Action

Disadvantage: Reduced readability

Advantage: Adjacent Label and corresponding Input Field

Label

Longer Label

Even Longer Label

One More Label Value 1
 Value 2

Primary Action

Advantage: Reduced vertical space

Left aligned labels

- When data required is unfamiliar
- Enables label scanning
- Less clear association between label and field
- Requires less vertical space
- Changing label length may impair layout

Left-Justified Horizontal Labels

Label:

Longer Label:

Even Longer Label:

One More Label: Value 1
 Value 2

Primary Action

Disadvantage:
Adjacency of Label and corresponding Input field

Advantage: Easy to scan labels

Label:

Longer Label:

Even Longer Label:

One More Label: Value 1
 Value 2

Primary Action

Advantage: Reduced vertical space

Labels: Best practice

- **Left-aligned labels**
 - Easily associated labels with the proper input fields
 - Excessive distances between labels inputs forced users to take more time
- **Right-aligned labels**
 - Reduced overall number of fixations by nearly half
 - Form completion times were cut nearly in half
- **Top-aligned labels**
 - Permitted users to capture both labels & inputs with a single eye movement'
 - Fastest completion times

Required/Optional form

- **Indication of required fields is most useful when**
 - There are lots of fields
 - But very few are required
 - Enables users to scan form to see what needs to be filled in
- **Indication of optional fields is most useful when**
 - Very few fields are optional
- **Neither is really useful when**
 - All fields are required

Required/Optional form fields - Example

Form with Required Fields

Label

Long Label *required

Longer Label *required
 Select Value

Even Longer Label

One More Label
 Value 1
 Value 2

Primary Action

Secondary Action

Label

Long Label (optional)

Longer Label
 Select Value

Even Longer Label

One More Label
 Value 1
 Value 2

Primary Action

Secondary Action



Required/Optional form fields: Best Practice

- Try to avoid optional fields
- If most fields are required: indicate optional fields
- If most fields are optional: indicate required fields
- Text is best, but * often works for required fields
- Associate indicators with labels

Field Lengths

- Field lengths can provide valuable affordances
- Appropriate field lengths provide enough space for inputs
- Random field lengths may add visual noise to a form

Enter Your Information (Already registered? Sign in)

Please enter your U.S. address and email address to create your account.

First Name Last Name

Street Address

City

State ZIP Code Country or Region U.S. addresses only, please.
United States

Phone Number
I - ext: Needed if there are questions about your order

A valid email address is required to communicate with you.

Email address

Re-enter Email address

Create Password How secure is your password?
Must be at least 6 characters, including a number or special character. Example: eXp0r55
Check your password strength - the higher, the better!

Re-enter Password

By clicking "Register" you agree to eBay Express's privacy policy and terms of use. You also agree to be contacted for marketing purposes, but you can change your notification preferences in your account.

Field Lengths: Best Practice

- When possible, use field length as an affordance
- Otherwise consider a consistent length that provides enough room for inputs

Content Grouping

- Content relationships provide a structured way to organize a form
- Groupings provide
 - A way to scan information required at a high level
 - A sense of how information within a form is related

Separating Related Content

Label:	<input type="text"/>
Longer Label:	<input type="text" value="Select Value"/>
Even Longer Label:	<input type="text"/>
One More Label:	<input checked="" type="radio"/> Value 1 <input type="radio"/> Value 2
Label:	<input type="text"/>
Longer Label:	<input type="text" value="Select Value"/>
Even Longer Label:	<input type="text"/>

Primary Action

Bad example – too much visual noise

The screenshot shows the Office Depot USA login page. At the top, there's a navigation bar with links for Home, En Espanol, Tech Depot, and Link Depot. Below the navigation, there's a banner asking if you shop with them by phone, fax, or have a tax-exempt account, with 'Yes' and 'No' buttons. A 'New Customer Checkout' section follows, containing fields for First Name, Middle Initial, Last Name, Address, City, State/Province, Zip/Postal Code, Country (USA), Phone, and Fax. There are also checkboxes for sending emails in HTML format and for exclusive discounts. The 'Billing Details' section contains similar fields for a different address. To the right, there's a 'Shipping Info' section with fields for Business Name, First Name, Middle Initial, Last Name, Address, City, State/Province, Zip/Postal Code, Country (USA), Phone, and Shipping Email. There are also checkboxes for sending emails in HTML format and for exclusive discounts. Below these are sections for 'Optional Info' (with a note about special offers) and 'Customer POF' (Customer Ref ID). At the bottom, there's a 'Shopping Cart' section with an order summary for 'Office Depot® Multipurpose Paper, Assorted Colors, Pack of 100'. The cart shows a quantity of 1, a unit price of \$2.00, and a total price of \$2.00. It also includes a 'Coupon' link. The page footer contains links for Site Index, Other Depots, Customer Services, Company Info, and Specials, along with terms and privacy policy links.

Ref:
form-



Good example

Other Costs (if they apply) ▶ Learn more

Who will pay the county transfer fee?
 Buyer
 Seller

Who will pay the city transfer fee?
 Buyer
 Seller

Who will pay the home owner's association transfer fee?
 Buyer
 Seller

Who will pay for the home owner's association transfer documents?
 Buyer
 Seller

Home Warranty

Do you want to order a home warranty?

Who will pay for the home warranty?
 Buyer
 Seller

How much home warranty coverage?

Which home warranty options do you want?
 Air conditioner Well
 Roofing Pool
 Pool Washer / Dryer / Refrigerator

Other

Liquidated Damages

Liquidated damages can be assessed if the buyer fails to complete the purchase because of default. If the buyer agrees to pay liquidated damages in case of default, then the seller retains the deposit initially paid by the buyer.

If you default, do you agree to pay liquidated damages?
 Yes
 No

Dispute Resolution

Rather than having disputes resolved in courts, buyers and sellers can agree to have all disputes resolved by arbitrator as provided by California law.

Do you agree to submit disputes to neutral arbitration?
 Yes
 No

Expiration

When do you want your offer to expire? (Commonly 3 calendar days after the buyer signs and dates the offer)

This offer shall officially expire, be deemed revoked, and the deposit shall be returned, unless the offer is signed by the seller and a copy of the offer is personally received by the buyer at 5 p.m. on the third day after this offer is signed by the buyer.

If the seller makes a counter-offer, your Redfin Agent will help you respond appropriately.



Content Grouping: Best Practice

- Use relevant content groupings to organize forms
- Use the minimum amount of visual elements necessary to communicate useful relationships

Actions



- **Not all form actions are equal**
 - Reset, Cancel, & Go Back are secondary actions: rarely need to be used (if at all)
 - Save, Continue, & Submit are primary actions: directly responsible for form completion
- **The visual presentation of actions should match their importance**

Actions - Examples

TYPICAL WEB FORM

Personal Information

First Name

Last Name

Contact Information

Address

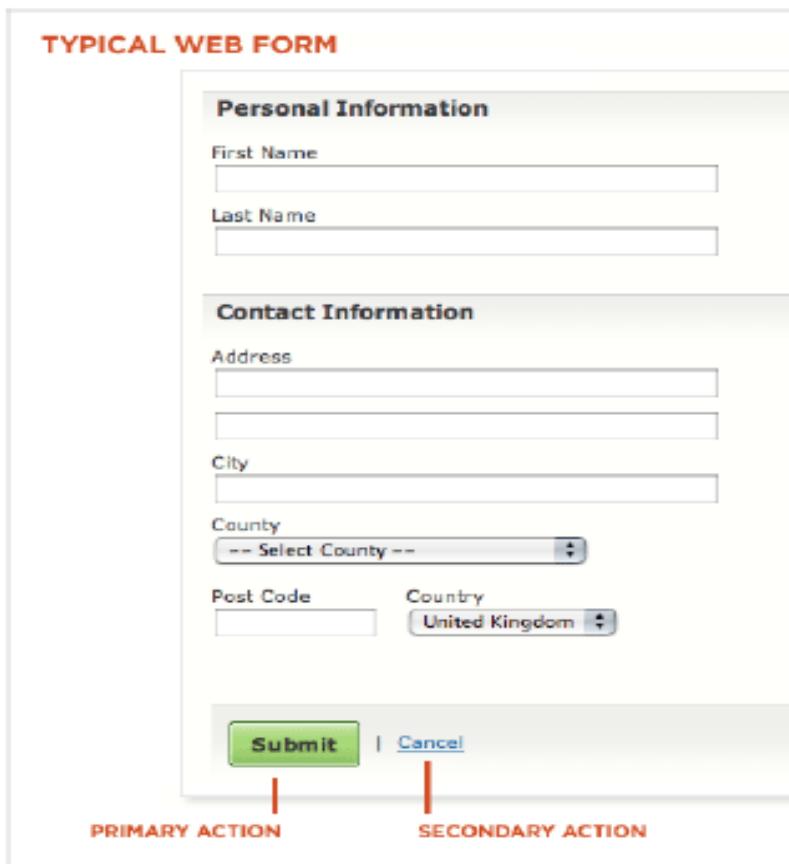
City

County

Post Code Country

Submit | **Cancel**

PRIMARY ACTION **SECONDARY ACTION**



A

Post Code Country

Submit | **Cancel**

B

Post Code Country

Submit **Cancel**

C

Post Code Country

Submit **Cancel**

D

Post Code Country

Submit **Cancel**

E

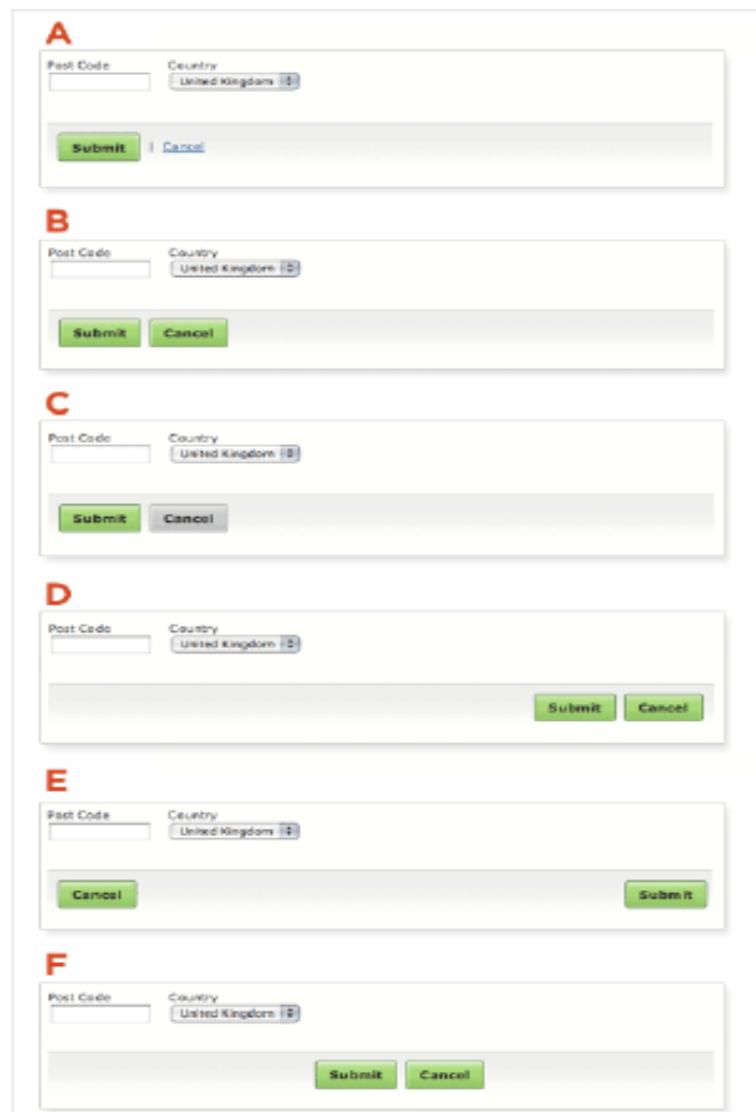
Post Code Country

Cancel **Submit**

F

Post Code Country

Submit **Cancel**



Actions – Heat tracking map

A

Post Code: Country:

Submit Cancel

B

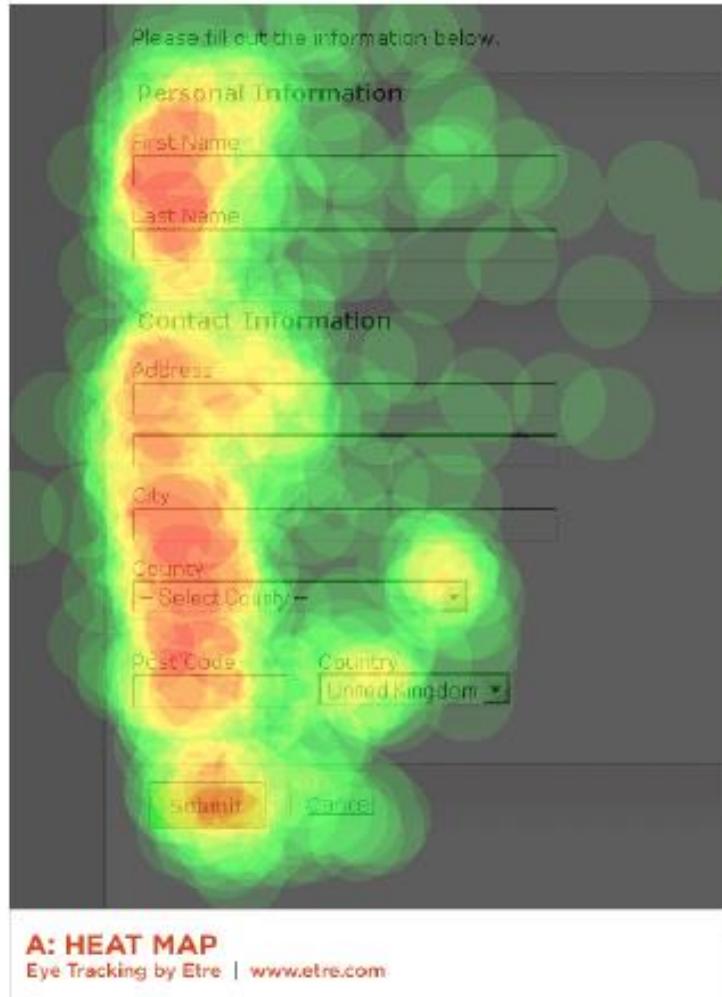
Post Code: Country:

Submit Cancel

C

Post Code: Country:

Submit Cancel



Actions: Best practice

- **Avoid secondary actions if possible**
- **Otherwise, ensure a clear visual distinction between primary & secondary actions**
- **Align primary actions with input fields for a clear path to completion**

Providing Help & Tips

- Help & Tips are useful when:
 - Asking for unfamiliar data
 - Users may question why data is being requested
 - There are recommended ways of providing data
 - Certain data requests are optional
- However, Help & Tips can quickly overwhelm a form if overused
- In these cases, you may want to consider a dynamic solution
 - Automatic inline exposure
 - User activated inline exposure
 - User activated section exposure

Help – Automatic inline exposure

The image shows a user registration form from Wufoo. At the top, there's a red header bar with the Wufoo logo. Below it, the title "II. User Registration" is displayed. The form consists of several input fields and associated validation messages:

- 1. Enter Your Email Address:** A text input field with a validation message: "You must supply a valid email address. We will never sell or disclose your email address to third parties."
- 2. Choose a Password:** A text input field.
- Verify Password:** A text input field next to the password field.
- 3. Pick your Wufoo Name / URL:** A text input field containing the placeholder "http://username.wufoo.com".

Below the input fields are two checkboxes:

- Keep me updated about Infinity Box projects.
- I agree to Wufoo's [Terms of Service](#).

At the bottom of the form are two buttons: "Create Account" (green background) and "Cancel" (red background).

At the very bottom of the page, there's footer text: "Wufoo · Infinity Box Inc. · Tampa, FL" and links to "About", "Blog", "Forums", "Terms", "Privacy", "Support", and "Contact". On the right side, there's a small "Wufoo" logo.

Help – User activated inline exposure

Phone Details

IMEI Code: 

PAC Code: 

Phone Details

IMEI Code: 

Explanation of IMEI Code

The International Mobile Equipment Identity (IMEI) number is a unique 15-digit code used to identify an individual GSM mobile telephone. The number can be found on most mobiles by typing in *#06#. If this combination doesn't work on your mobile phone, please call our support centre on +44 (0) 1252 XXXX XXX.

[Back to IMEI input field.](#)

PAC Code: 

Providing Help & Tips: Best Practice

- Minimize the amount of help & tips required to fill out a form
- Help visible and adjacent to a data request is most useful
- When lots of unfamiliar data is being requested, consider using a dynamic help system

Path to Completion

Primary goal for every form is completion

Every input requires consideration & action

- Remove all unnecessary data requests
- Enable flexible data input

Provide a clear path

Enable smart defaults

Clear Path to Completion

The screenshot shows a PayPal 'Check Payment Details' page. At the top, there's a navigation bar with 'My Account', 'Send Money', 'Request Money', 'Merchant Tools', and 'Auction Tools'. On the right, there are 'Log Out | Help' links. Below the navigation, the title 'Check Payment Details' is followed by a 'Secure Transaction' icon. The main section is titled 'Payment Details' and contains the following information:

- Pay To: paypal.jf@spinfree.com (a verified member)
- Amount: \$37.00
- Source of Funds: PayPal balance more funding options
- Email Subject: Here's the cash I owe ya
- Note: Thanks for bailing me out! I also included \$7 for the cab ride. Thanks again!

Below this, the 'Shipping Information' section has two radio button options: 'Ship to' (selected) with the address '400 North May Street, #301, Chicago, IL 60622, USA' and 'Add Address', and 'No shipping address required'. A red arrow points from the bottom left towards the 'Send the \$37' button. At the bottom of the page, there's a yellow footer bar containing the 'Send the \$37' button, 'Edit Transaction', and 'Cancel Transaction' links. The footer also includes links to 'About Us', 'Accounts', 'Fees', 'Privacy', 'Security Center', 'User Agreement', 'Developers', and 'Shops', along with a copyright notice: 'Copyright © 1999-2003 PayPal. All rights reserved.'

Flexible inputs

Flexible Data Input

Phone Number
 (ex. 555-123-4444)

Phone Number
() - -

Phone Number

(555) 123-4444

555-123-4444

555 123 4444

555.123.4444

5551234444

Smart Defaults

Shipping Costs

Shipping Service
Standard delivery

[Add another shipping service](#)

Shipping Insurance
Not offered \$ 0.00
[View insurance rate table](#).

Shipping & Handling
\$

Don't know what to charge? Try the  [Shipping Calculator](#). To offer free shipping, enter 0.00 above.

Sales Tax
I don't charge tax [Change](#)

Path to Completion: Best Practice

- Remove all unnecessary data requests
- Enable smart defaults
- Employ flexible data entry
- Illuminate a clear path to completion
- For long forms, show progress & save

Tabbing: Best Practice

- Remember to account for tabbing behavior
- Use the tabindex attribute to control tabbing order
- Consider tabbing expectations when laying out forms

Progressive disclosure: Best Practice

- Not all users require all available options all the time
- Progressive disclosure provides additional options when appropriate
 - Advanced options
 - Gradual engagement
- Most effective when user-initiated
- Maintain a consistent approach

FEEDBACK

- **Inline validation**
 - Assistance
- **Errors**
 - Indication & Resolution
- **Progress**
 - Indication
- **Success**
 - Verification

Inline validation

- Provide direct feedback as data is entered
 - Validate inputs
 - Suggest valid inputs
 - Help users stay within limits

Inline validation - Example

The screenshot shows a user interface for creating a password. On the left, there is a field labeled "Create Password" containing "*****". Below it, a note says: "Must be at least 6 characters, including a number or special character. Example: eXpr3SS". On the right, there is a section titled "How secure is your password?" with a progress bar consisting of four colored segments (green, yellow, orange, red) and the text "Check your password strength - the higher, the better".

Create Password

Must be at least 6 characters, including a number or special character. Example: eXpr3SS

How secure is your password?

Check your password strength - the higher, the better

Re-enter Password

Inline validation: Best practice

- Use inline validation for inputs that have potentially high error rates
- Use suggested inputs to disambiguate
- Communicate limits

Error handling

- Errors are used to ensure all required data is provided and valid
 - Clear labels, affordances, help/tips & validation can help reduce errors
- But some errors may still occur
- Provide clear resolution in as few steps as possible

Error handling - Example

The image shows a Wufoo user registration form with several error messages displayed in red boxes:

- 1. Enter Your Email Address:** The input field is highlighted in red with the message "Invalid email address".
- 2. Choose a Password:** The "Password" input field is highlighted in red with the message "Password is required".
- 3. Pick your Wufoo Name / URL:** The input field is highlighted in red with the message "Username can only contain letters and numbers." Below it, a placeholder URL "http://username.wufoo.com" is shown.
- Checkboxes:** Two checkboxes are present:
 - Keep me updated about Infinity Box projects.
 - I agree to Wufoo's Terms of Service.A red message "Please accept Wufoo's Terms of Service." is displayed below the checkboxes.

Buttons:

- Create Account** (green button)
- Cancel** (red button)

Page Footer:

Wufoo - Infinity Box Inc. - Tampa, FL
About · Blog · Forums · Terms · Privacy · Support · Contact

Wufoo logo

Error handling: Best practice

- Clearly communicate an error has occurred: top placement, visual contrast
- Provide actionable remedies to correct errors
- Associate responsible fields with primary error message
- “Double” the visual language where errors have occurred

Progress

- Sometimes actions require some time to process
 - Form submission
 - Data calculations
 - Uploads
- Provide feedback when an action is in progress

Progress - Example

Attach a file (each file should be under 10MB)

Choose File ap_beyond...rames.pdf

Attaching files ...

→

[Associate this message with a milestone...](#)

Notify people of this message via email

All of IxDA Volunteers

Jeff Howard Pedro Jorge Adler

All of IxDA Board

<input type="checkbox"/> Carrie Ritch	<input type="checkbox"/> Josh Seiden	<input type="checkbox"/> Mauro Cavalletti
<input type="checkbox"/> Dan Saffer	<input type="checkbox"/> Lada Gorlenko	<input type="checkbox"/> micah alpern
<input type="checkbox"/> David Malouf	<input type="checkbox"/> Lisa deBettencourt	<input type="checkbox"/> Pabini Gabriel-Petit
<input type="checkbox"/> Frank Ramirez	<input type="checkbox"/> Luke W	<input type="checkbox"/> Robert Reimann
<input type="checkbox"/> Greg Petroff		



Progress: Best practice

- Provide indication of tasks in progress
- Disable “submit” button after user clicks it to avoid duplicate submissions

Successful completion

- After successful form completion confirm data input in context
 - On updated page
 - On revised form
- Provide feedback via
 - Message (removable)
 - Animated Indicator

Successful completion - Example

The screenshot shows the AlertLogic Invision Security dashboard. The main navigation bar includes Summary, Dashboard, Threats, Exposures, Defenses, Management, and Reports. The Dashboard section is active. On the left, there's a sidebar with options for Layout (grid, cards, list), Modules (Incidents, Incident Severity, External vs. Internal Incidents), and a Did you Know? note about repositioning modules. The central area has four main sections: Incidents, Exposures, Exposed Hosts, and Attached Hosts. The Incidents section displays a table with columns Threat, Events, Date, and Summary, listing items like '23 misc-activity' and '45 preprocessor'. The Exposures section shows a table with Exposure and Count, including entries like 'E-2000-1200' (Count 32). The Exposed Hosts section lists hosts with their respective exposure counts. A modal window titled 'Edit Incidents' is open, containing a 'Changes Saved.' message with a checkmark icon and a 'Close this window.' button. Below the dashboard, there are links to 'View complete Incidents report', 'View complete Incident Severity report', and 'View complete Exposed Hosts report'.

Successful completion: Best Practice

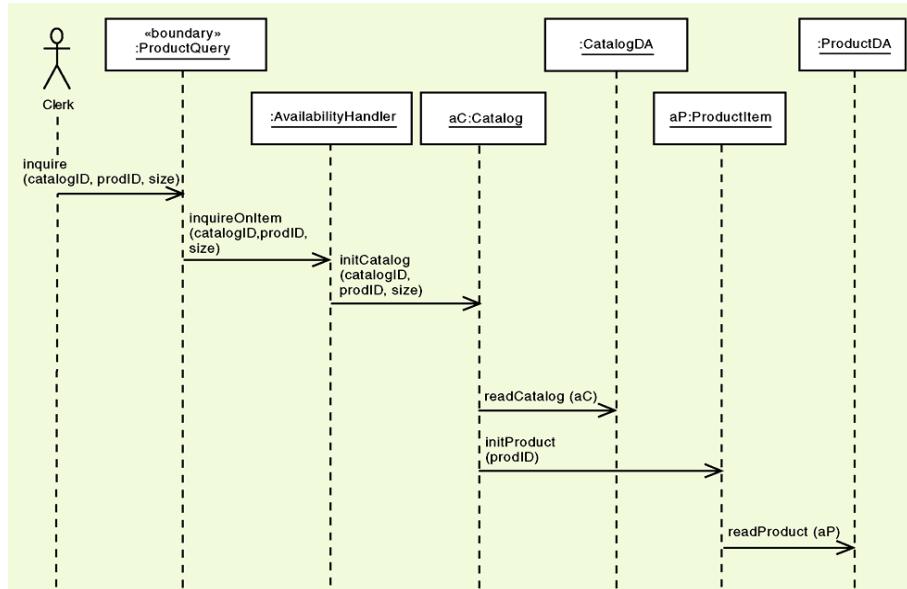
- Clearly communicate a data submission has been successful
- Provide feedback in context of data submitted



FIT2001 – Systems Development

Seminar 9: Use case realisation – Sequence diagrams

Chris Gonsalvez





Our road map:

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- A range of Requirements gathering and documentation techniques
- Designing systems that our clients want – Usability
- Interface design principles & tips

- Use case realisation
- Design Class Diagrams, Sequence diagrams,

At the end of this topic you will:

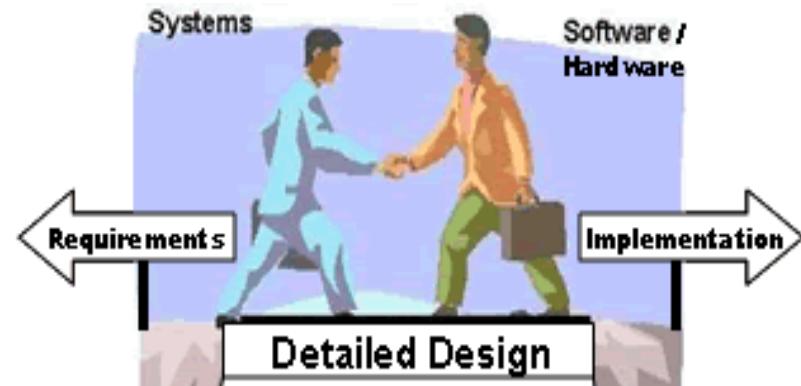
- Be aware of object oriented design fundamentals
- Understand the process of use case realisation and be able to apply the techniques to develop detailed UML models
 - design class diagrams and sequence diagrams

Lecture Outline

1. Overview of Object Oriented (OO) Design
2. Overview of OO Programs
3. Fundamental Design principles
4. Iterative OO Design process
 - 4.1 Create first-cut Design Class Diagram
 - Definition, Notation, Process, Example
 - 4.2 Create Sequence Diagrams
 - 4.2.1 Background information: System Sequence Diagram
 - 4.2.2 First cut sequence diagram – How?, Example
 - 4.2.3 Final cut sequence diagram – How? Example
 - 4.3 Update Design Class Diagram
5. Summary

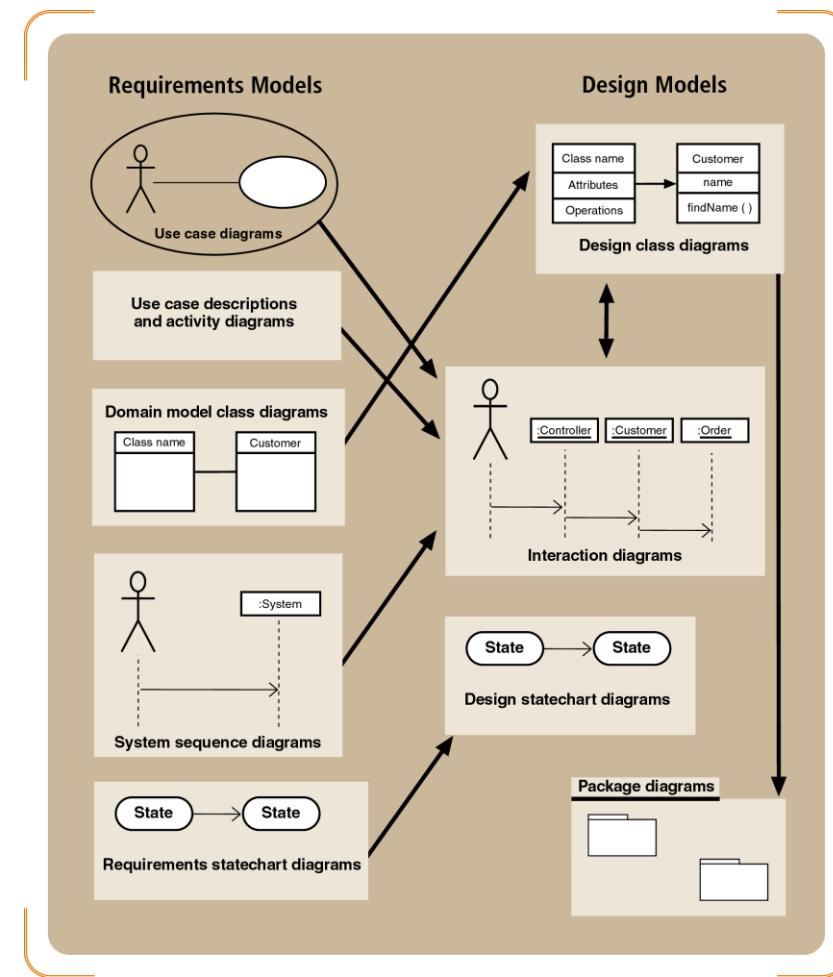
What is Object Oriented (OO) Design?

- The bridge between users requirements and programming for the new system
- A process by which detailed OO Design models (blueprints) are created – extends the requirements models
 - *These models are built only if necessary (Agile tenet)*
- The design models are used to build the new system, develop the database, user-interfaces, networks, controls and security



Relationship between UML analysis and design models

- Models such as activity diagrams, use cases diagrams, class models can inform various design models such as sequence diagrams and design class diagrams



Our focus:

Design models:

- Design class diagrams
- Interaction diagrams – in particular:
 - Sequence Diagrams
- Before we start:
 - Overview of OO programs
 - Overview of fundamental design principles

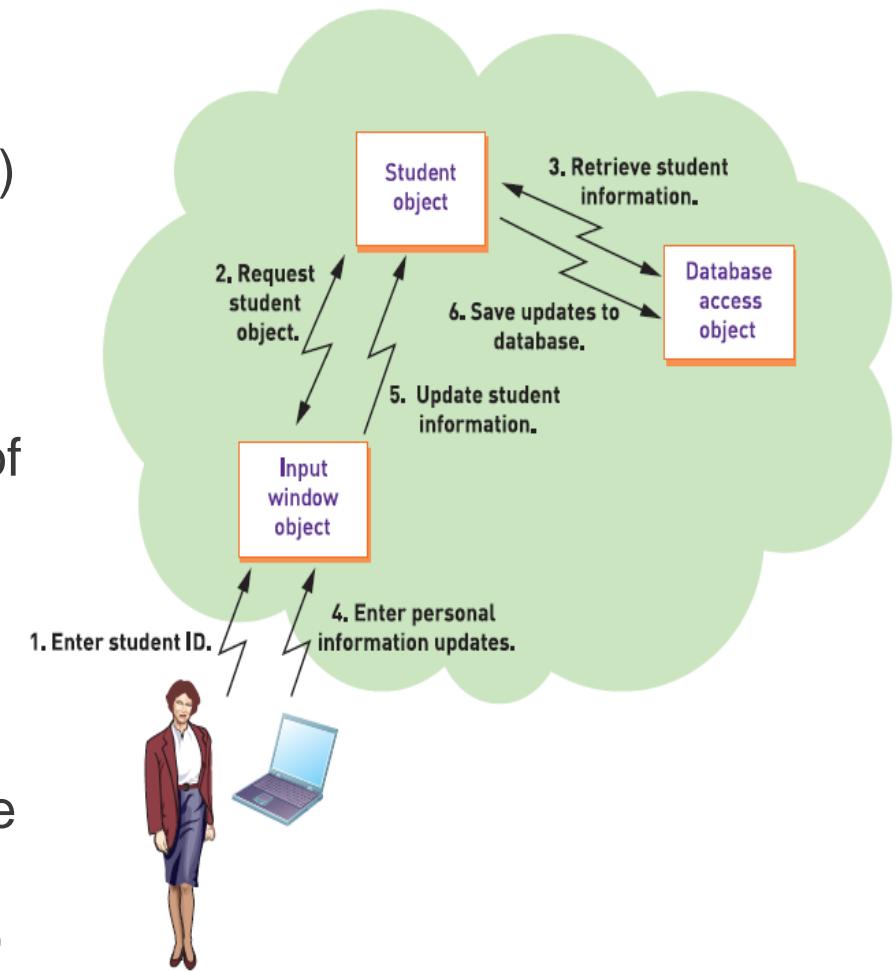
Overview of object-oriented programs

(so that we know what our design is trying to support)

- Set of program objects that cooperate by communicating to accomplish result
- Each object contains program logic and necessary attributes in a single encapsulated unit
- Objects send each other messages and work together to support functions of main program
- OO system design provides detail for programmers
 - Eg. design class diagrams, interaction diagrams – communication diagrams, sequence diagrams

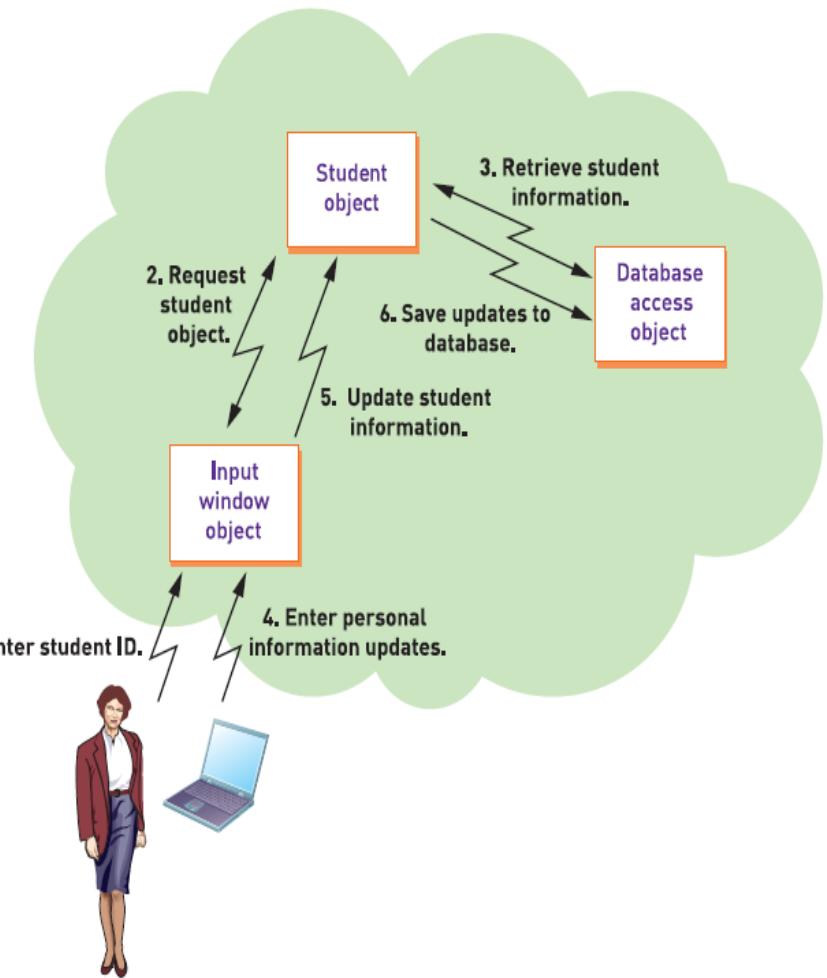
OOO event-driven program flow.1

- Windows object displays a form to enter Student ID and other info. (1)
- After student ID entered, windows object sends a message (2) to the Student Class telling it to create a new Student Object (an instance of Student Class), and to go to the database, retrieve the student information (3) and put it in the Student Object.
- Then the Student Object sends the information back to the Windows Object to display on the screen (2)



OOO event-driven program flow.2

- The student then updates their personal information (4), and another series of messages is sent to update the Student Object (5) in the program and then the student information is used to update in the database (6).
- Analysts define the structure of the program logic and the data fields when they define the class.
- The object only comes into existence when the program runs
- This is called **INSTANTIATION** of the class – making an instance (object) based on the class template



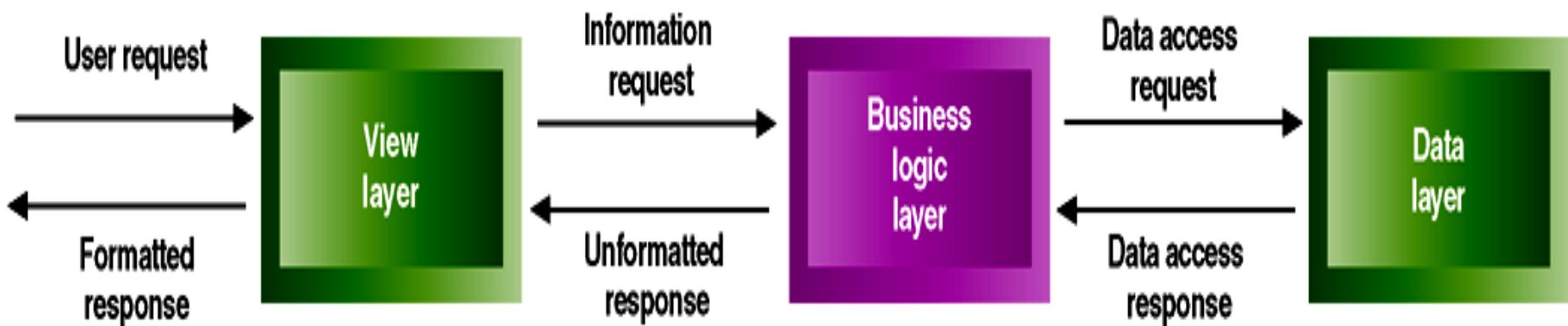
OOO event-driven program flow.3

Three layer architecture

The three objects on the previous slide represent a 3 layer architecture structure

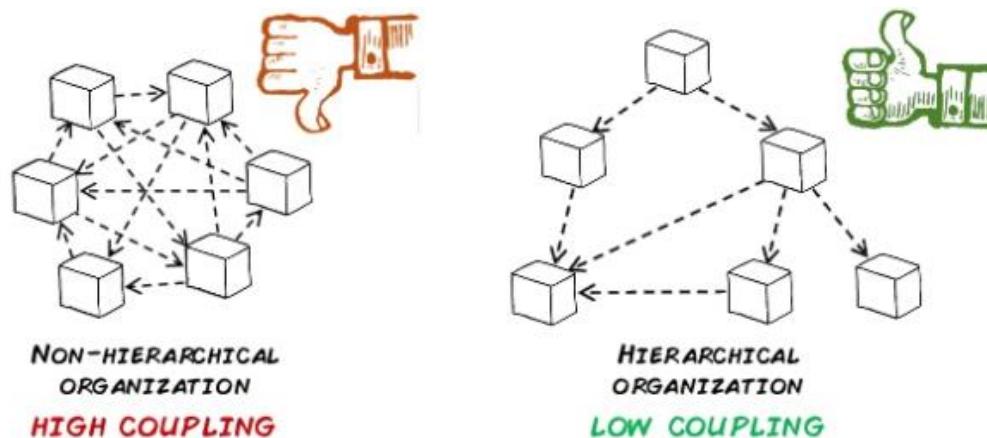
1. **Input window object (View layer)** accepts user inputs and formats/ displays processing results

2. **Student Object (Business logic or domain layer)** implements the rules and procedures of business processing
3. **Database access object (data layer)** manages stored data usually in one or more databases and communicates with the domain layer



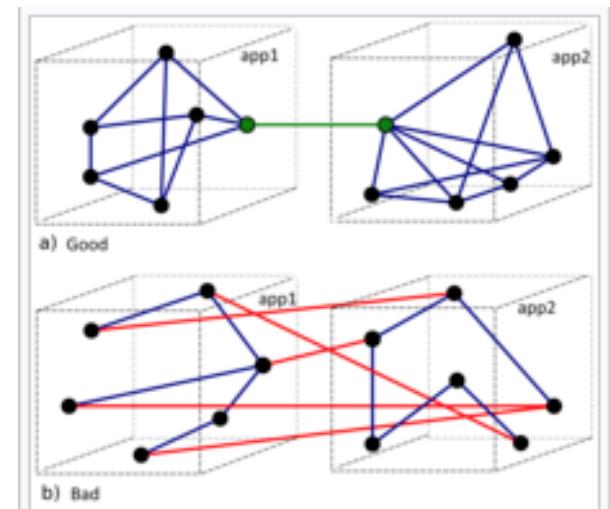
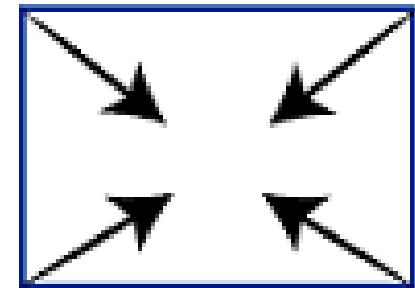
Fundamental Design Principles - Coupling

- Qualitative measure of how closely classes are linked
 - interconnected by the data in messages
 - number of navigation arrows on design class diagram
 - number of parameters passed by methods
- Low or loose coupling makes system easier to understand and maintain, minimal ripple effect



Fundamental Design Principles - Cohesion

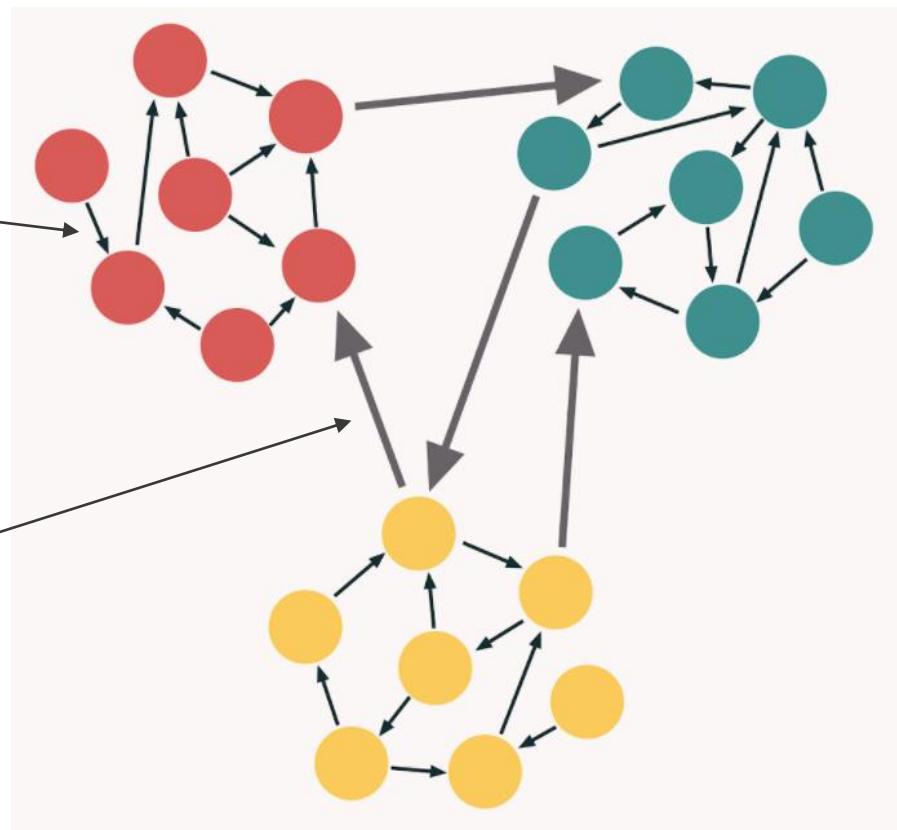
- Measures the consistency of functions in a class ... the unity of purpose of the methods within a class and the class itself
- Ensure clear separation of responsibility – divide low cohesive class into several highly cohesive classes
- Tight coupling often leads to low cohesion
 - increases the extent to which objects are dependent
 - can causes ripple-through effects
 - reduces the ability to reuse parts of the system
 - hard to maintain



Difference between coupling and cohesion

Cohesion
within a
module

Coupling
between
modules



Iterative OO Design process

1. Create first-cut (preliminary) design class diagram
2. Create sequence diagrams, for each use case
...realisation of use cases
3. Return to design class diagram and update based on
design of sequence diagrams

Design class diagrams and detailed interaction diagrams inform each other and should be developed in parallel

Design class diagrams (DCD) – What?

- Extend the domain model class diagram developed during OO analysis
- Design and document the programming classes that will be built for the new system .. describes the design component within the classes:
 - navigation between classes, attribute names and properties, method names and properties
- Shows set of problem domain classes and their association relationships
 - associations focus on navigation not multiplicity
 - direction of messages passed between objects
 - indicate the extent of coupling between objects

Moving from the Domain model class diagram to the Design Class Diagram

Domain diagram Student	Design class diagram Student
<p>Student</p> <p>studentID name address dateAdmitted lastSemesterCredits lastSemesterGPA totalCreditHours totalGPA major</p>	<p>«Entity» Student</p> <p>-studentID: integer {key} -name: string -address: string -dateAdmitted: date -lastSemesterCredits: number -lastSemesterGPA: number -totalCreditHours: number -totalGPA: number -major: string</p> <p>+createStudent (name, address, major): Student +createStudent (studentID): Student +changeName (name) +changeAddress (address) +changeMajor (major) +getName () : string +getAddress () : string +getMajor () : string +getCreditHours () : number +updateCreditHours () <u>+findAboveHours (int hours): studentArray</u></p>

Design class stereotypes

- Domain model class diagrams show a conceptual view of users' business environment
- Design classes define software building blocks
 - When developers build design class diagrams, new classes are abstracted and identified as part of the design process
 - UML uses **stereotype** notation to categorise a type of design class – it has some special characteristic we want to highlight

Design class notation: **Stereotypes**

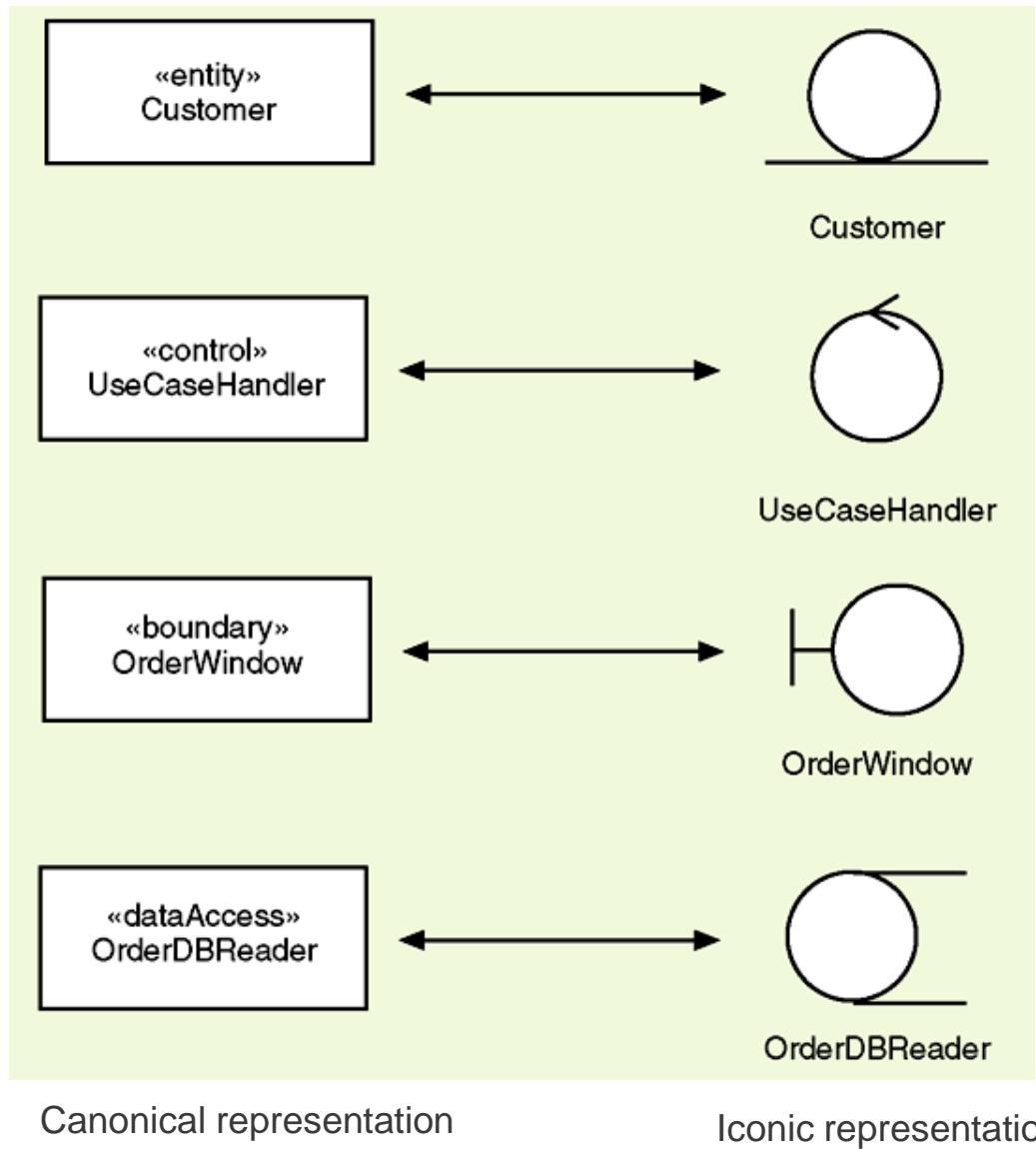
- **Entity** – design stereotype for problem domain class
 - something users deal with when doing their work
 - persistent class .. exists after system is shut down
- **Boundary** – designed to live on system's automation boundary
 - user interface class and windows classes
- **Control** – mediates between boundary and entity classes, between the view layer and domain layer
- **Data access** – retrieves from and sends data to the database

Stereotypes Notation:

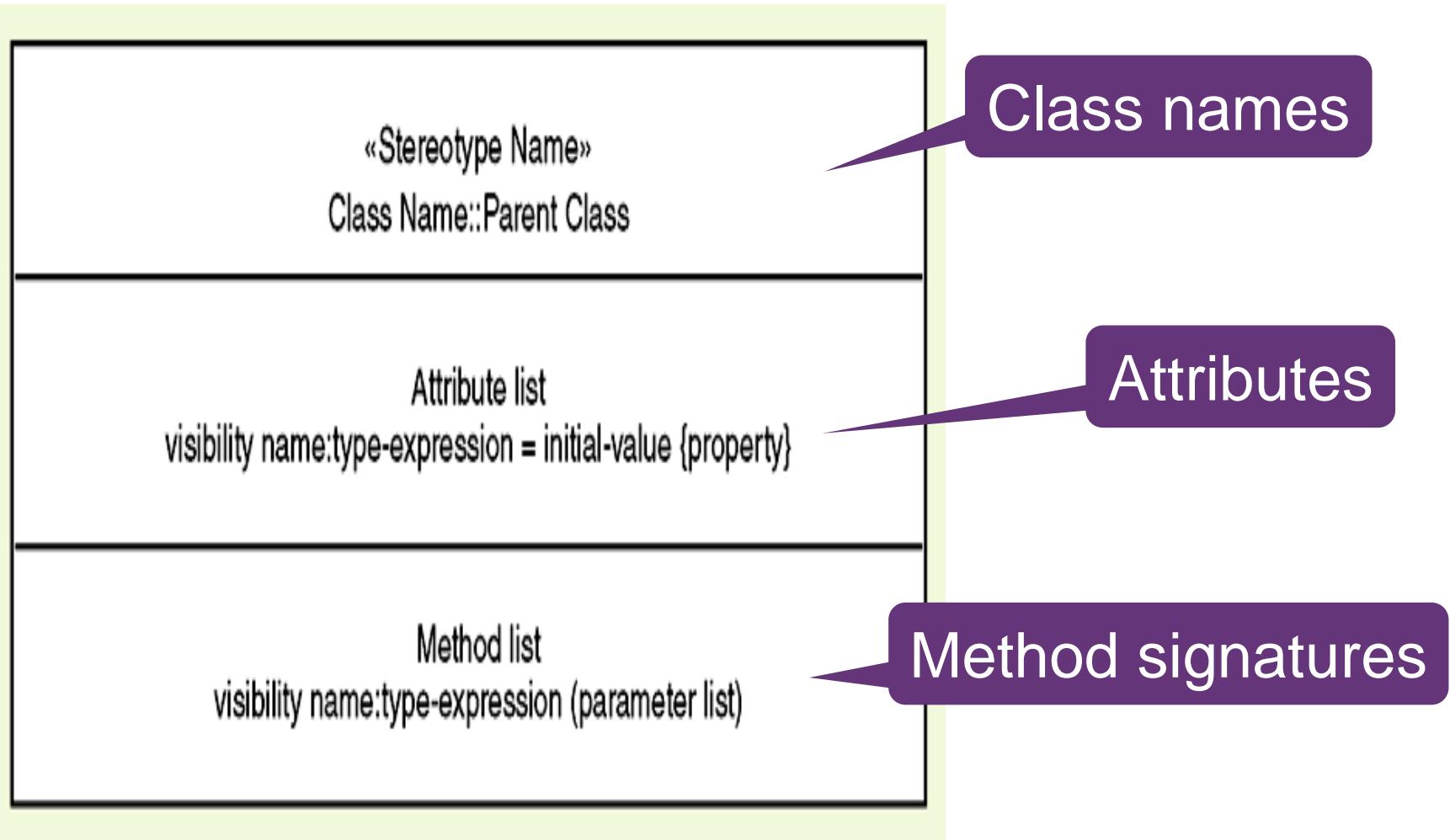
Full notation with «»

OR

Shorthand notation with
circular icons



Design class notation



Design class notation: Class, Attributes

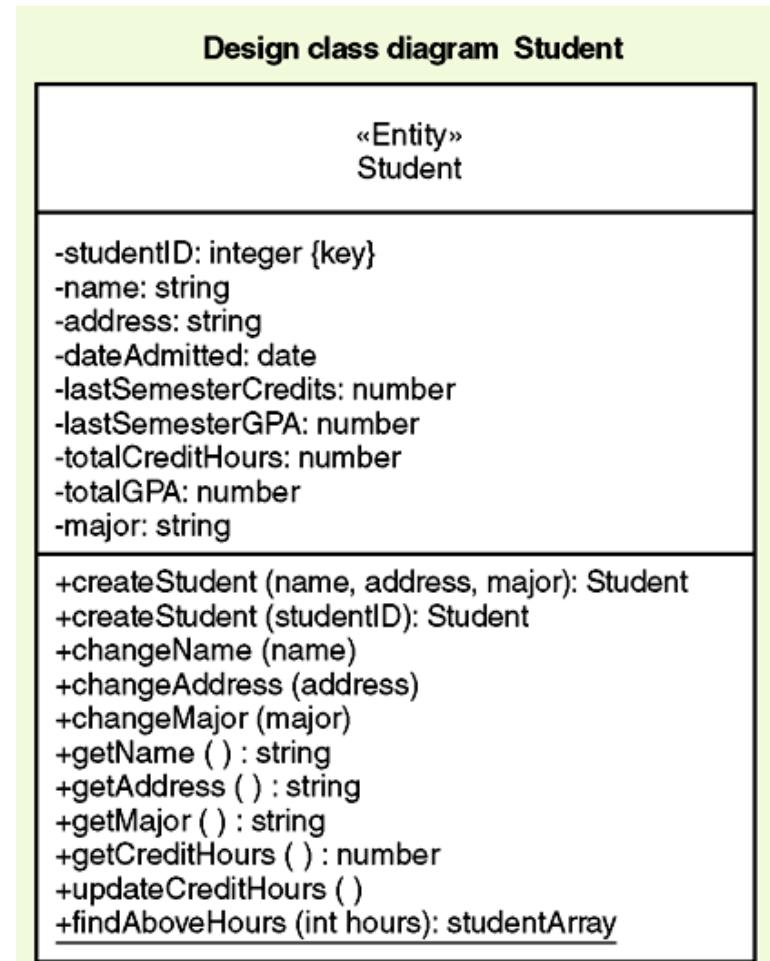
- Stereotype name & Class name
- Attributes
 - Visibility .. Can other objects access the attribute
 - private, public or protected
 - Attribute name
 - Data-type-expression
 - character, integer, string, number, currency or date
 - Initial value (if applicable)
 - Property within curly brackets eg. {key}

Design class diagram Student

«Entity» Student
<pre>-studentID: integer {key} -name: string -address: string -dateAdmitted: date -lastSemesterCredits: number -lastSemesterGPA: number -totalCreditHours: number -totalGPA: number -major: string</pre>
<pre>+createStudent (name, address, major): Student +createStudent (studentID): Student +changeName (name) +changeAddress (address) +changeMajor (major) +getName () : string +getAddress () : string +getMajor () : string +getCreditHours () : number +updateCreditHours () +findAboveHours (int hours): studentArray</pre>

Design class notation: Methods

- Method signature – all information needed to call a method:
 - Method Visibility
 - Method name
 - Method parameter list (input arguments)
 - Return type-expression (return parameter)



Attribute and method visibility

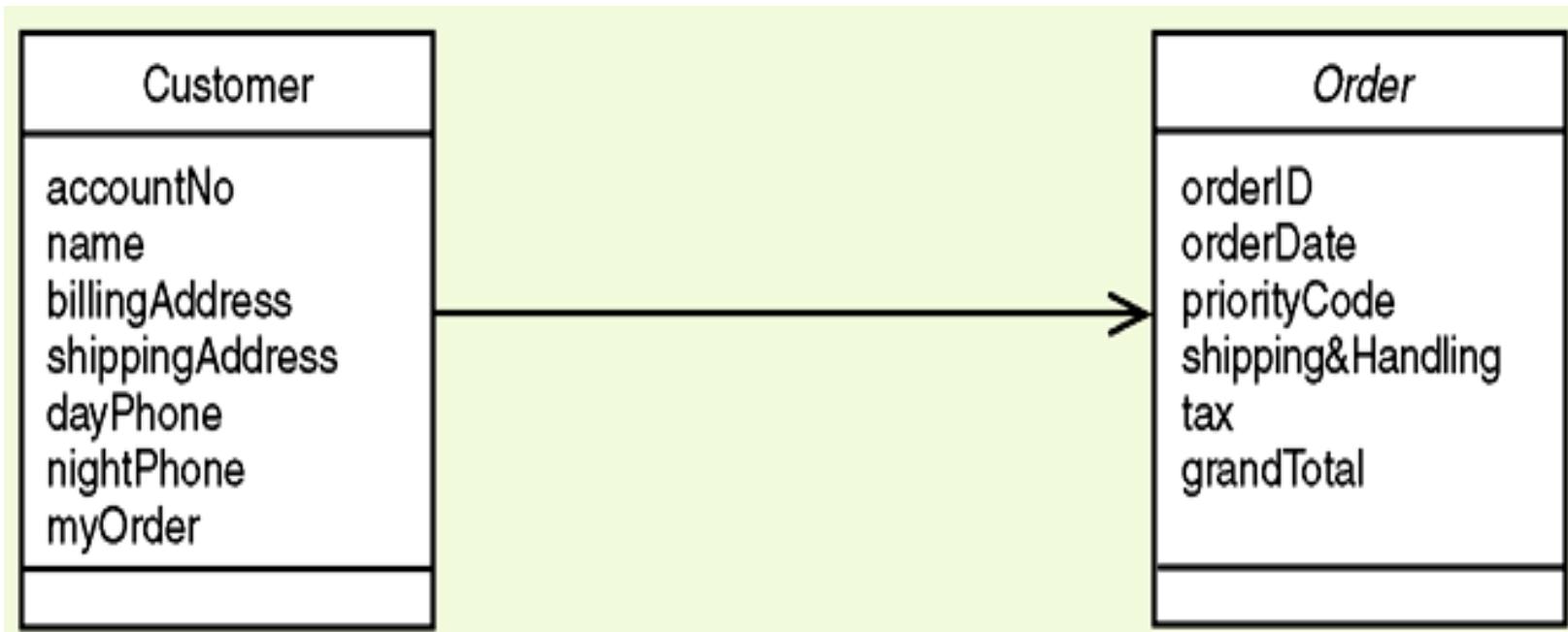
- **Public (+)**
 - available to other objects in system
 - get, set, display, constructor, destructor
- **Private (-)**
 - not accessible by other objects
 - methods only accessible to the object itself
 - often process-specific functions
- **Protected (#)**
 - can only be accessed by the object or its children

Design class diagrams – How?

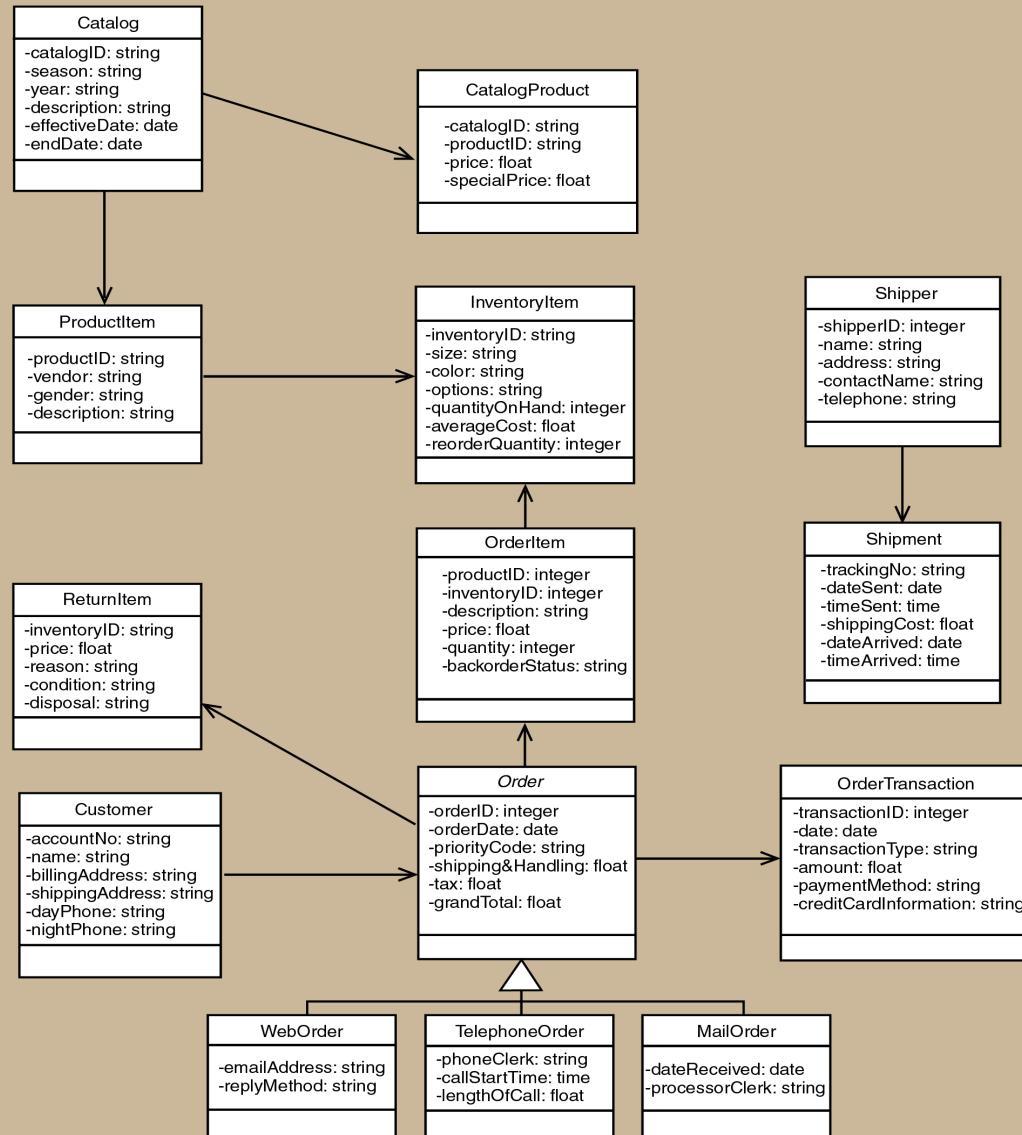
- Create a first-cut Design class diagram
 - Elaborate the attributes
 - add additional attributes if required
 - define their type
 - declare visibility (most should be private)
 - Show navigation visibility (add visibility arrows)
 - The ability of one object to view and interact with another object
 - Can be bidirectional
 - Will need to be updated as design progresses
- The first-cut Design class diagram is used to help develop the sequence diagram which will then be used to refine the Design class diagram

Navigation visibility between classes

- Navigation arrow indicates that the Order object must be visible to the Customer object
- Need to ask: Which classes need to have references to or be able to access other classes?



First-cut Design class diagram: Example



Iterative OO Design process

1. Create first-cut (preliminary) design class diagram
2. **Create sequence diagrams, for each use case
...realisation of use cases**
3. Return to design class diagram and update based on
design of sequence diagrams

*Design class diagrams and detailed interaction diagrams
inform each other and should be developed in parallel*

Use case realisation with interaction diagrams

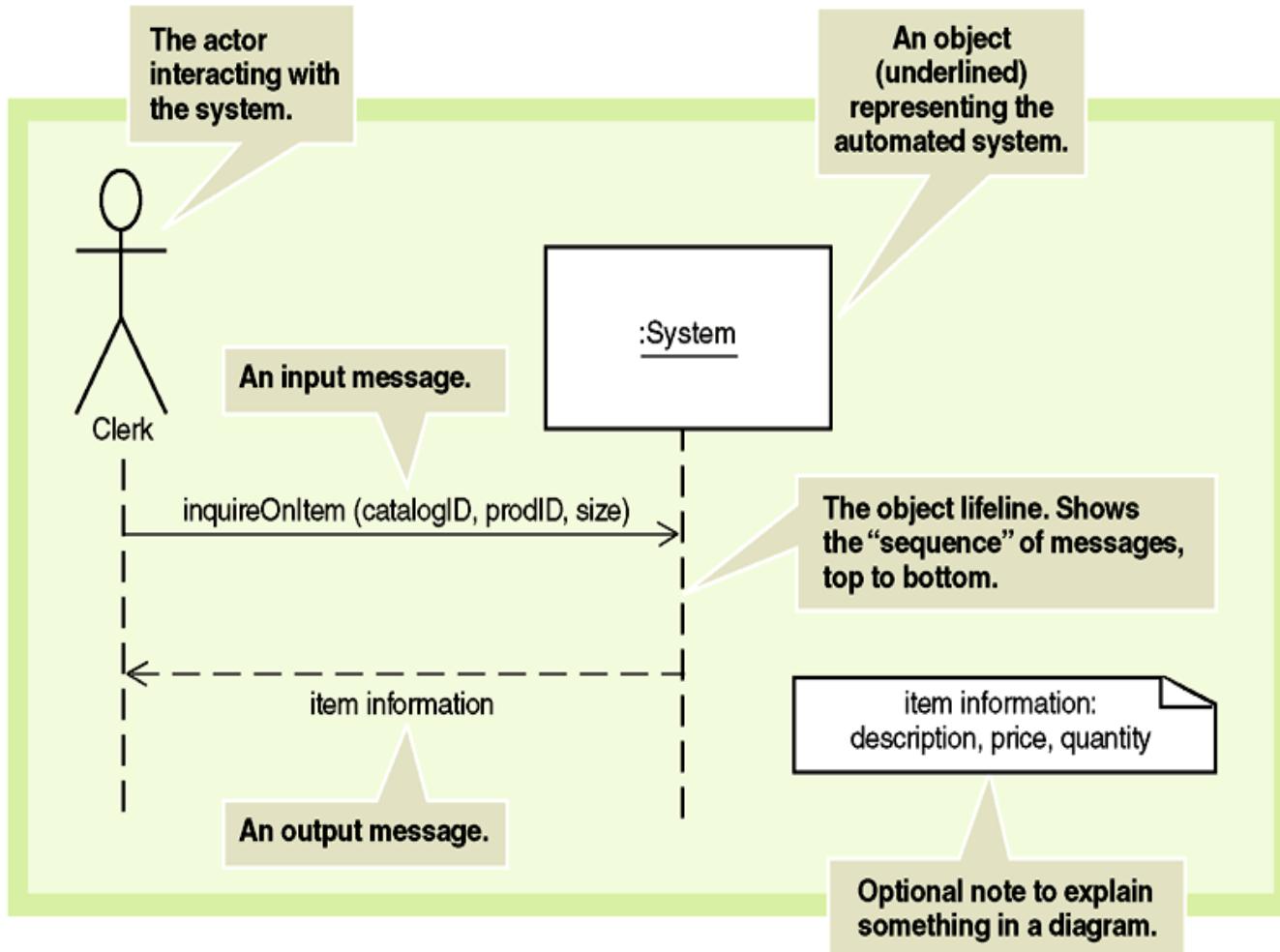
- Interaction diagrams are at the heart of object-oriented design. There are 2 types:
 - Sequence
 - Communication

(we will be doing an overview of Sequence Diagrams only)
- Realisation of a use case - Determine what objects communicate by sending messages to each other

Some background information: The System Sequence Diagram

- System sequence diagram
 - Shows sequence of messages between the actor and the system for a single use case
 - Only shows actor and one object – part of the system
 - Shows input & output messaging requirements for the use case
 - Actor, :System, object lifeline
 - Messages

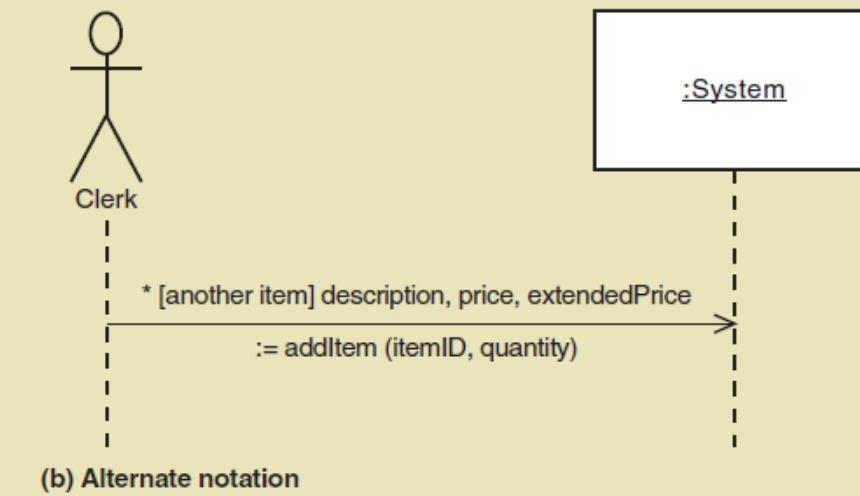
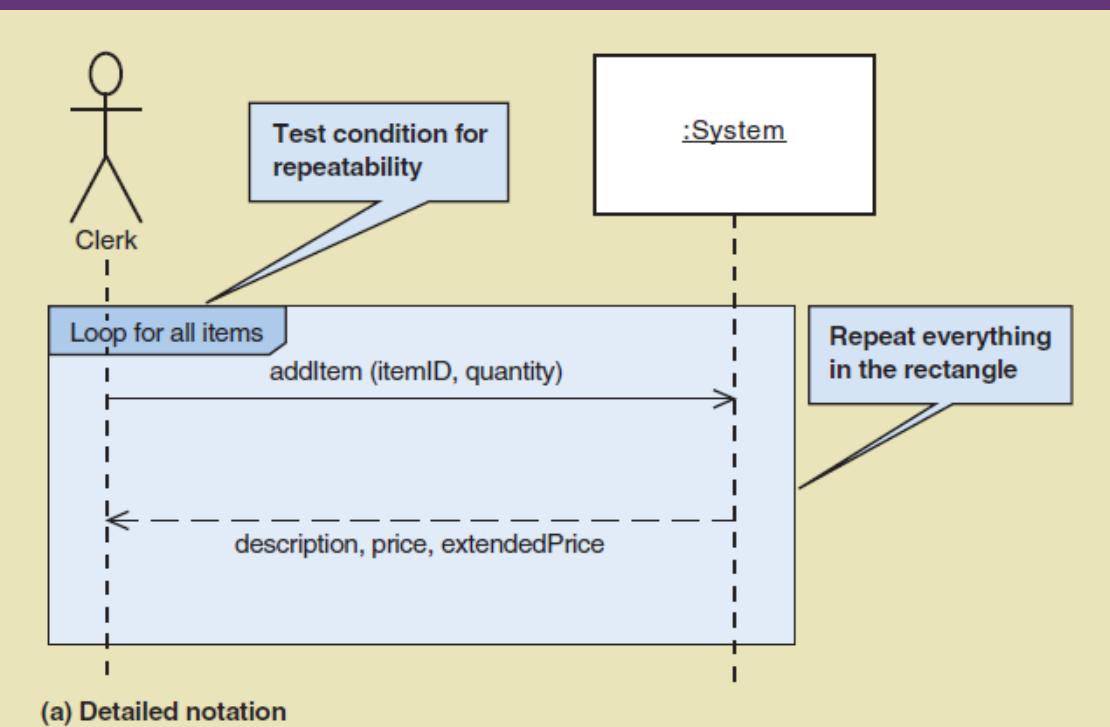
System Sequence Diagram (SSD) - Notation



SSD Notation

- Actor represented by stick figure – person (or role) that “interacts” with system by entering input data and receiving output data
- Object notation is rectangle with name of object underlined – shows individual object and not class of all similar objects
- Lifeline is vertical line under object or actor to show passage of time for object
- Messages use arrows to show messages sent or received by actor or system

SSD Message Examples with Loop Frame



SSD Message Examples

OPT sequence diagram frame

Opt Frame (optional)

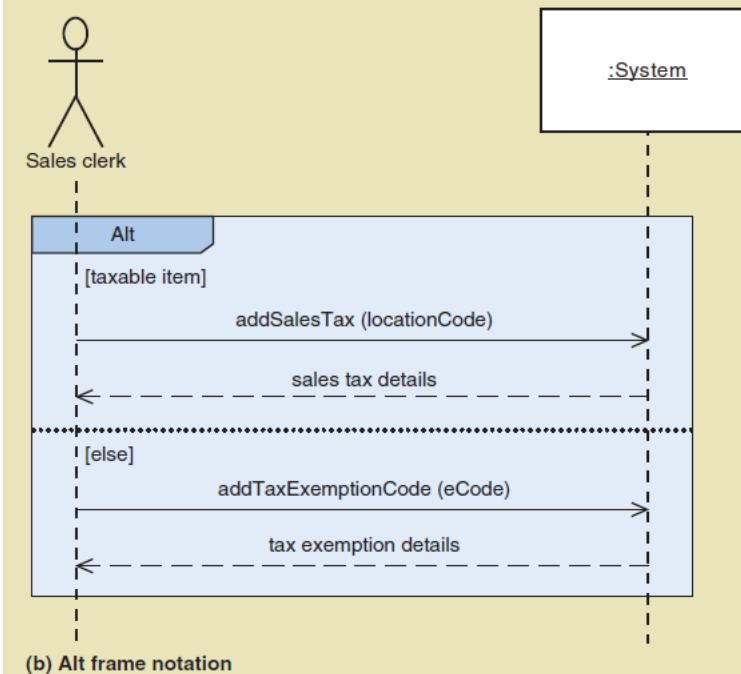
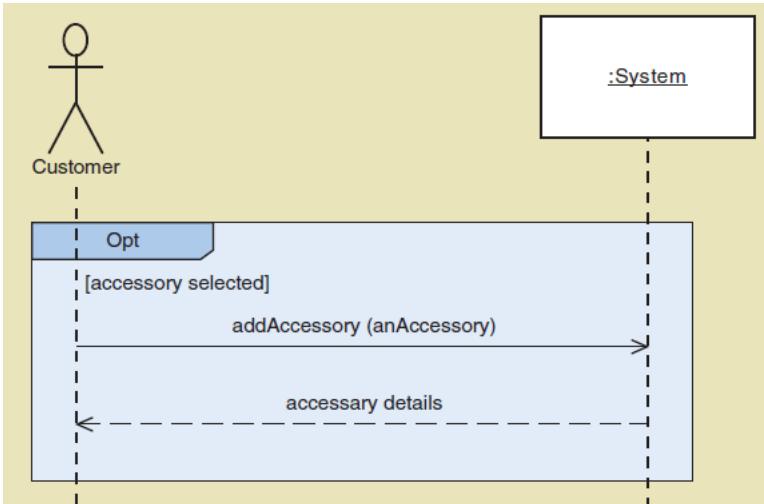
Alt Frame (if-else)

The OPT frame stands for optional and is a portion of an interaction path that may be done. (correct answer)

The ALT frame for alternative shows alternative paths where one of the options will be done

The LOOP frame represents a repeating loop of functionality

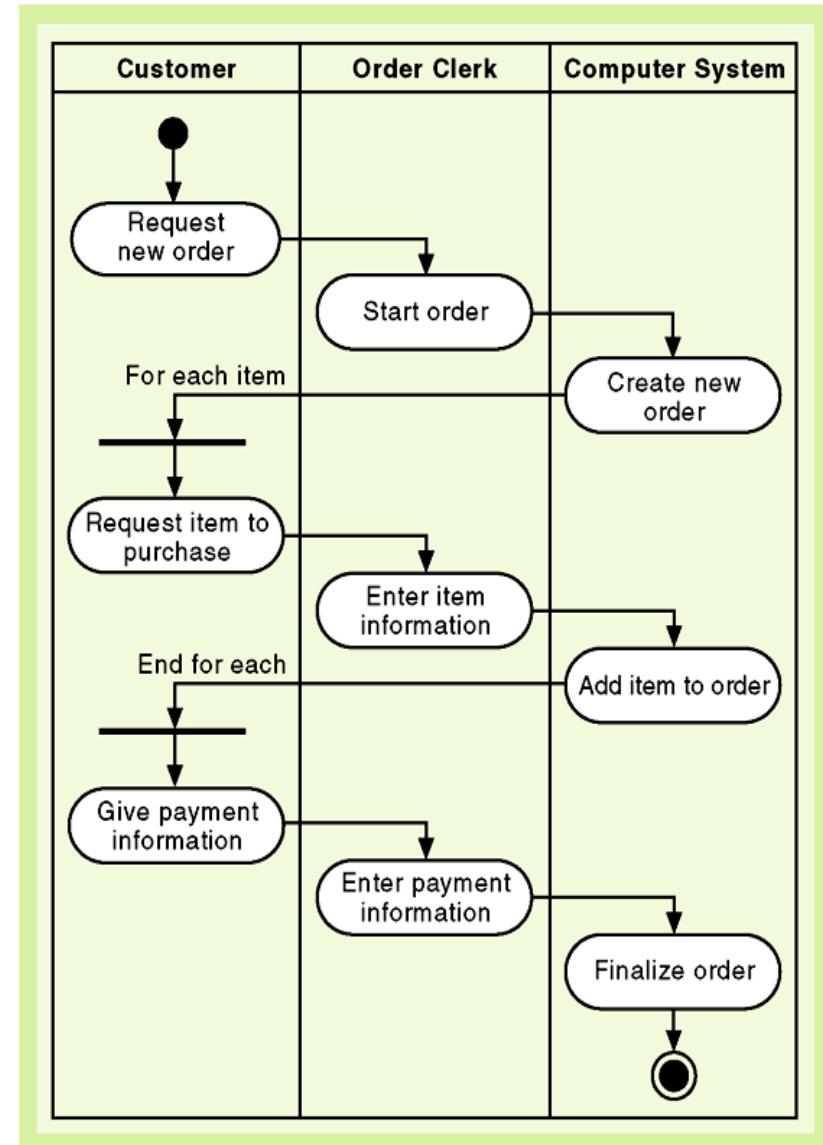
There is no particular frame used to show parallel functionality.



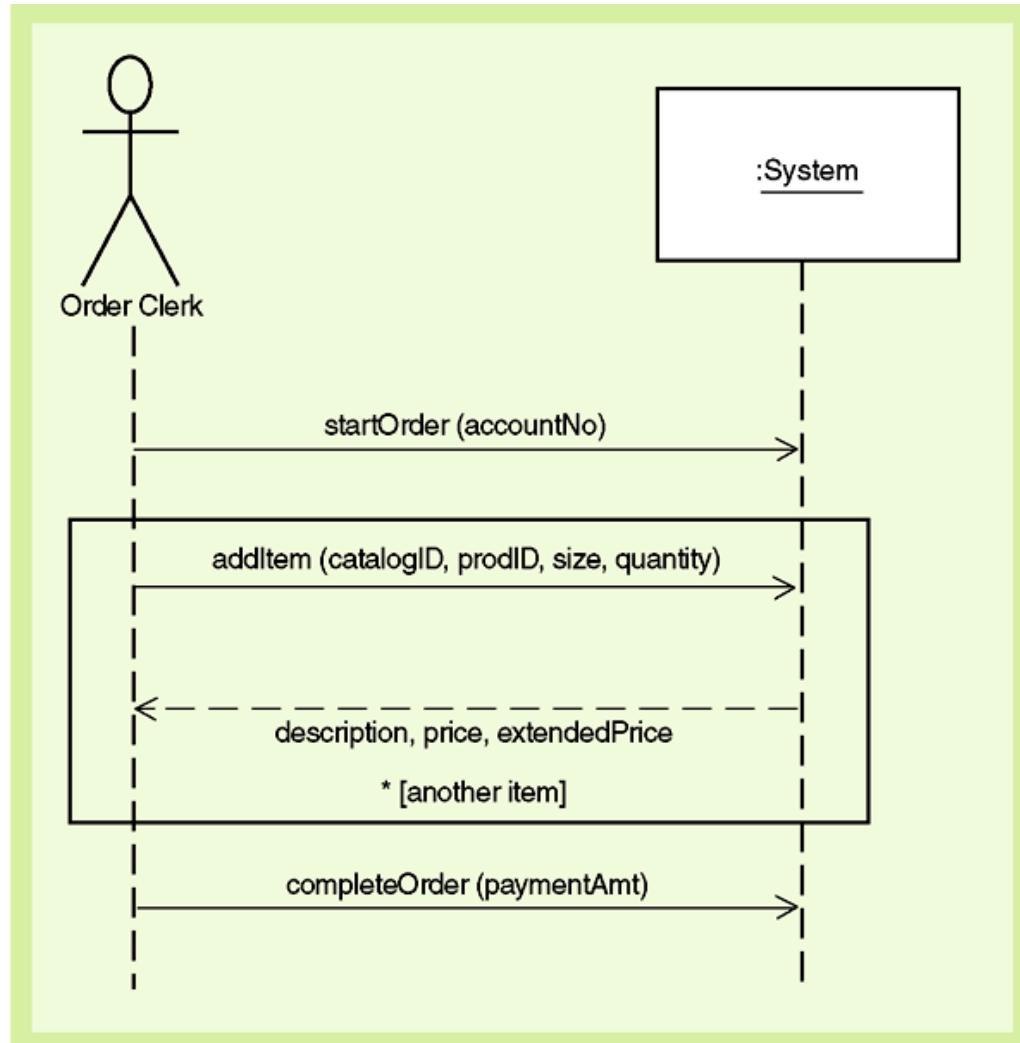
Developing a System Sequence Diagram

- Identify input messages – from use case descriptions or activity diagrams
- Describe message from external actor to system using message notation
 - Name message verb-noun: what the system is asked to do
 - Consider parameters the system will need
- Identify and add any special conditions on input message
 - Iteration/loop frame
 - Opt or Alt frame
- Identify and add output return messages
 - On message itself: aValue:= getValue(valueID)
 - As explicit return on separate dashed line

Simplified Activity Diagram of the Phone Order Scenario



SSD of Simplified Telephone Order Scenario for Create New Order Use Case



Example: use cases to SSD

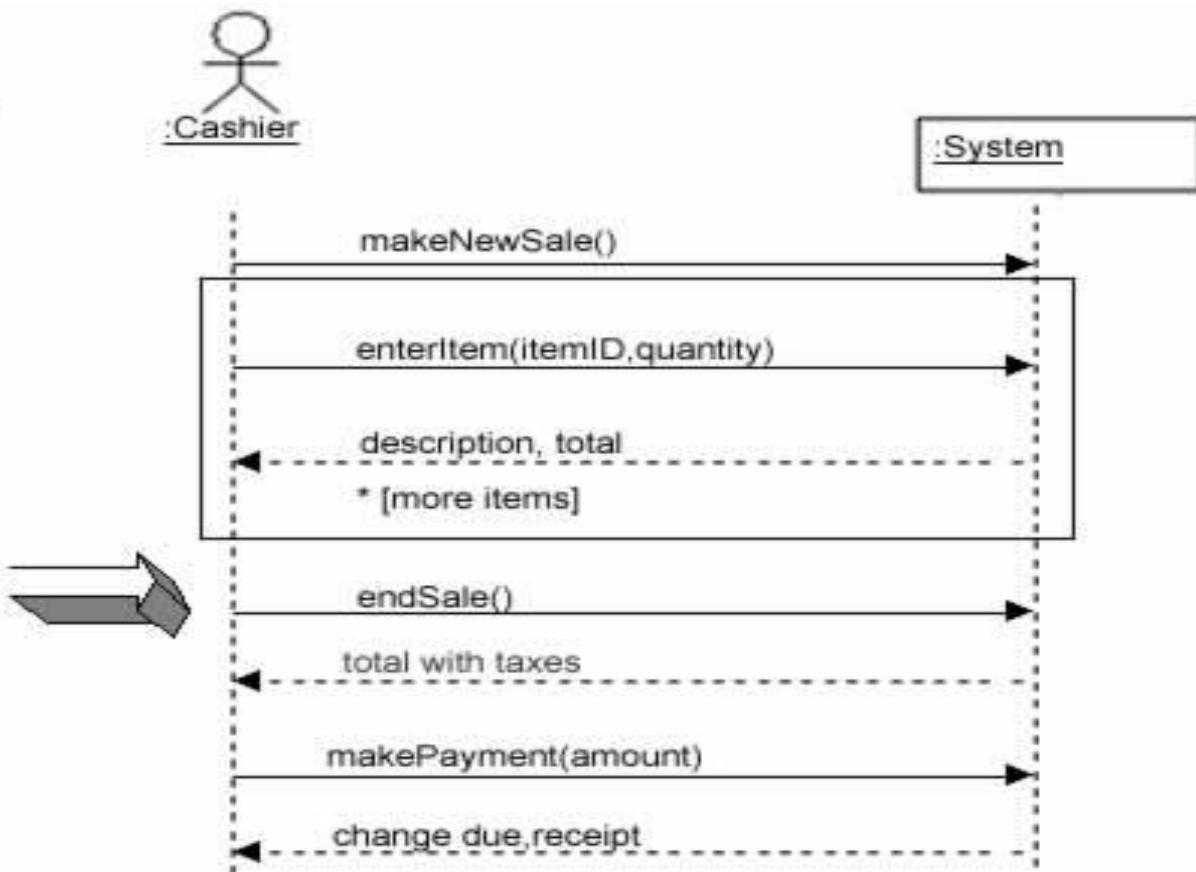
SSD can be created based on a particular use case scenario
(description)

Simple Cash-only Process Sale scenario:

1. customer arrives at aPOS check out with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters a new item identifier.
4. System records new sale line item and presents item description, price and running total.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
- ...

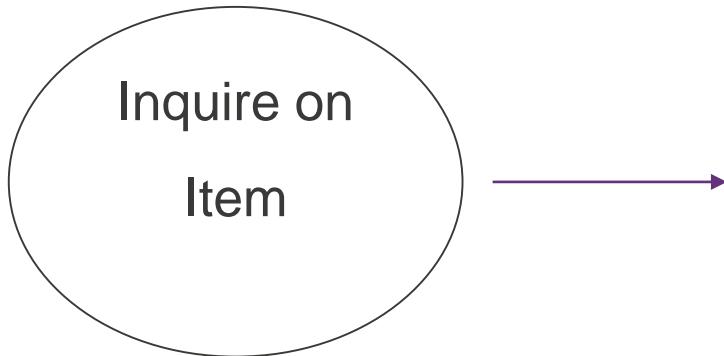


Sequence diagrams: Definition

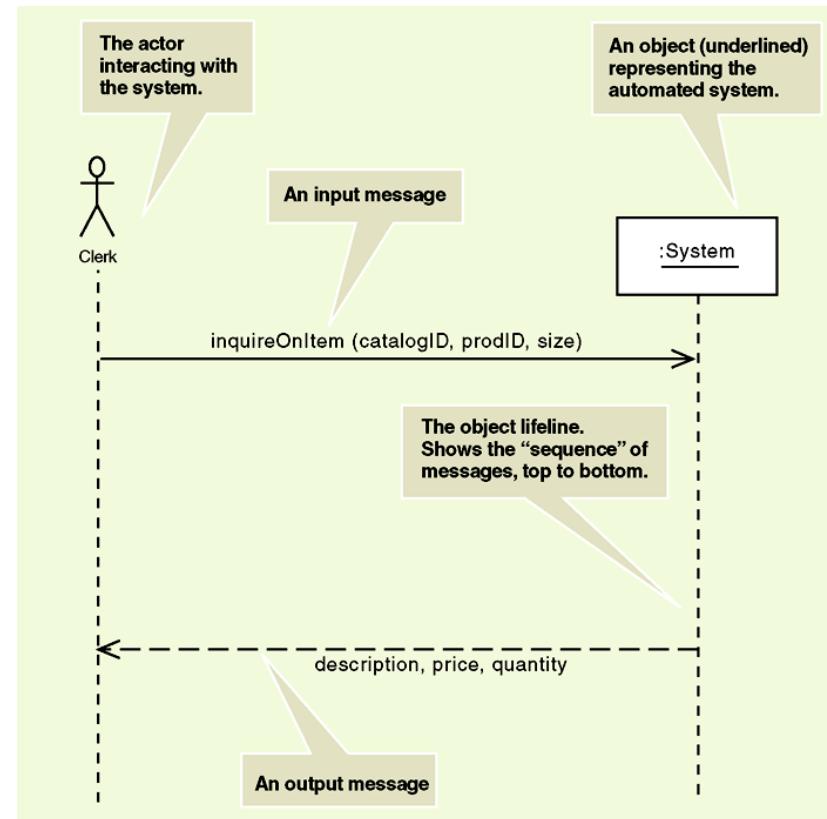
- Sequence diagrams extend the system sequence diagram (SSD) to show:
 - View layer objects
 - Domain layer objects (usually done first)
 - Data access layer objects
- Sequence diagrams used to explain object interactions and document design decisions
- For each use case and scenario
 - documents inputs to and outputs from system
 - captures interactions between system and external world as represented by actors
 - captures and defines specific interactions between collaborating objects within each use case

First-cut sequence diagram: How.1

1. We are going to work with the Use Case for Inquire on Item

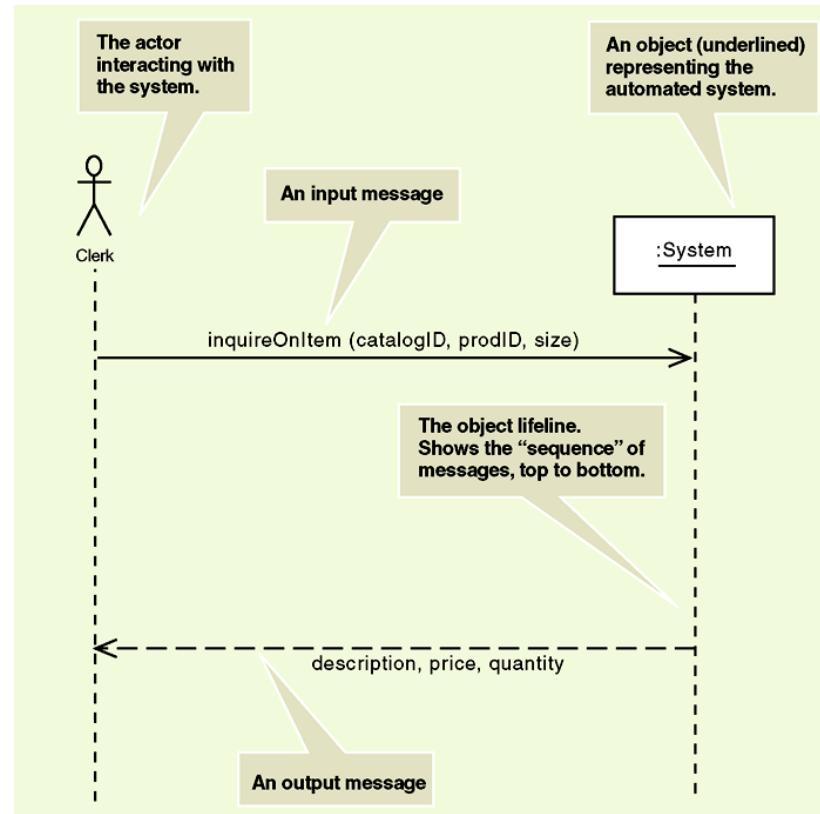


2. Description: For a particular size of product in a catalogue, we want to know the description, price and quantity available



First-cut sequence diagram: How.2

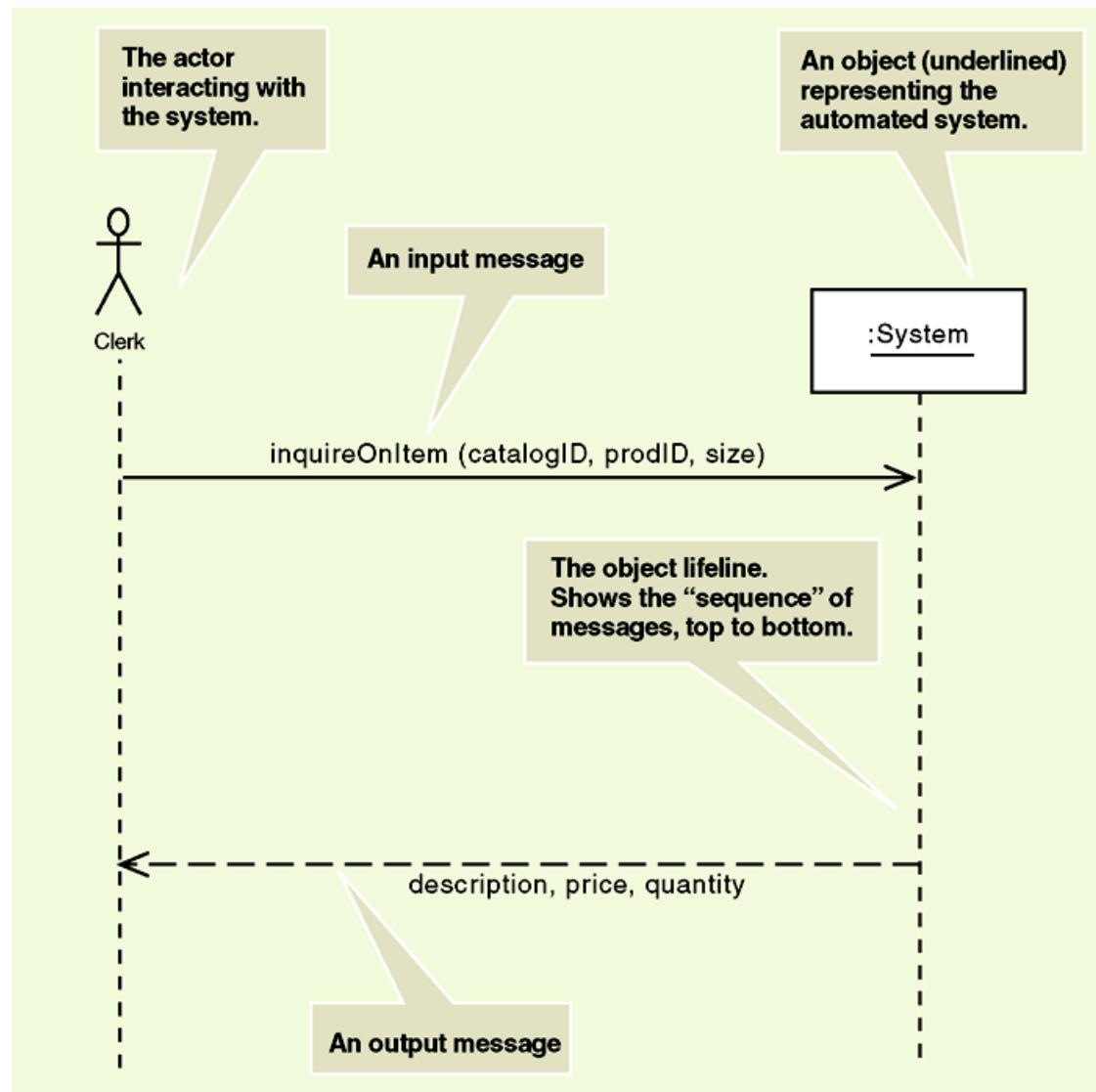
1. Start with elements from System Sequence Diagram (SSD) and first-cut Design Class Diagram
 - a Sequence Diagram uses all elements of an SSD .. System object replaced by all internal objects and messages
2. Replace the :System object with an appropriately named use case controller
3. For each input message
 - determine all internal messages that result from that input



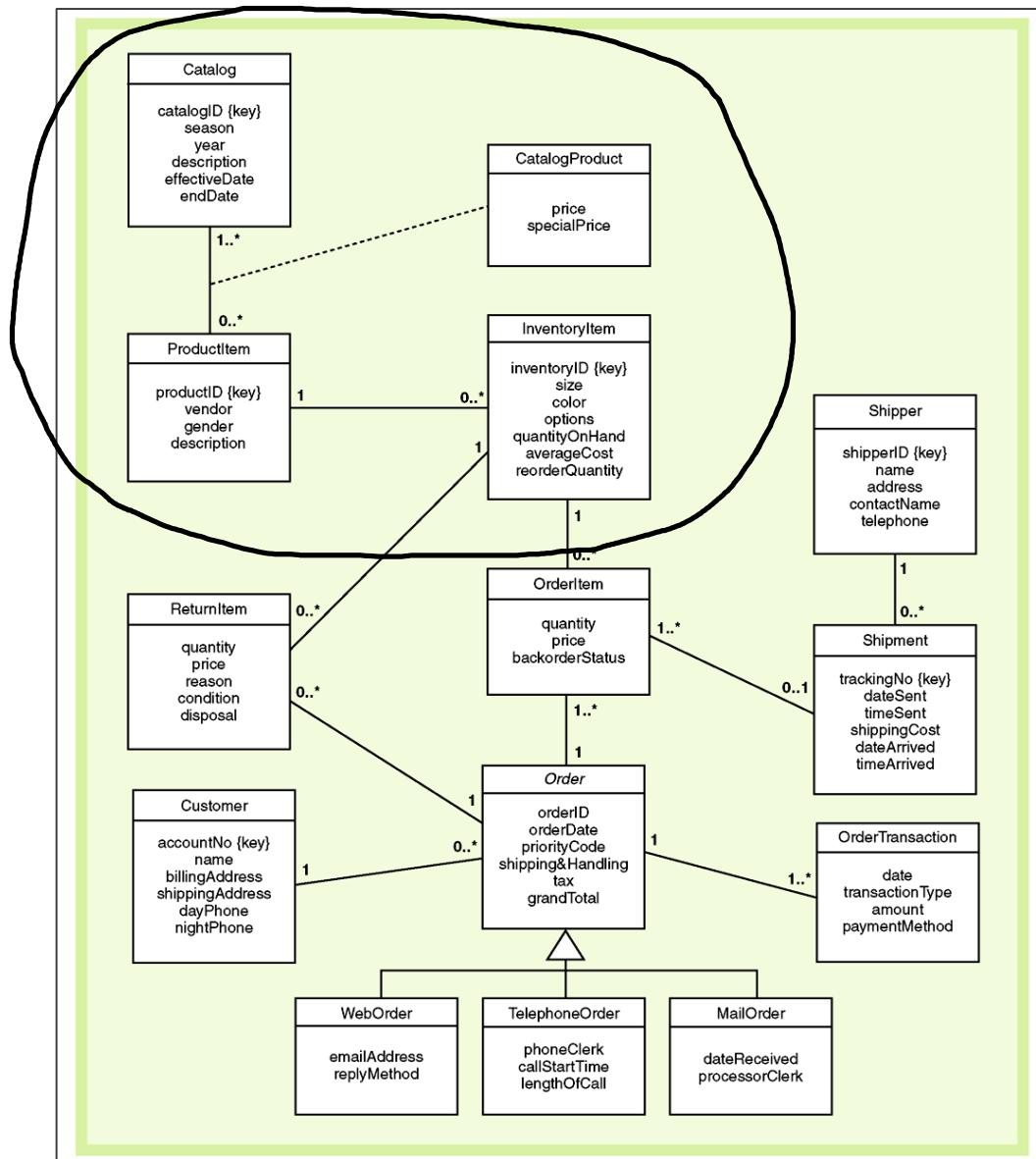
First-cut sequence diagram: How.3

4. For each internal message
 - Determine its objective, what information is needed
 - Identify the complete set of classes affected by the message
 - What class needs it (destination)
 - What class provides it (source)
 - Whether any objects are created as a result of the input
 - Flesh out the components for each message
 - Iteration, true/false conditions, return values, passed parameters

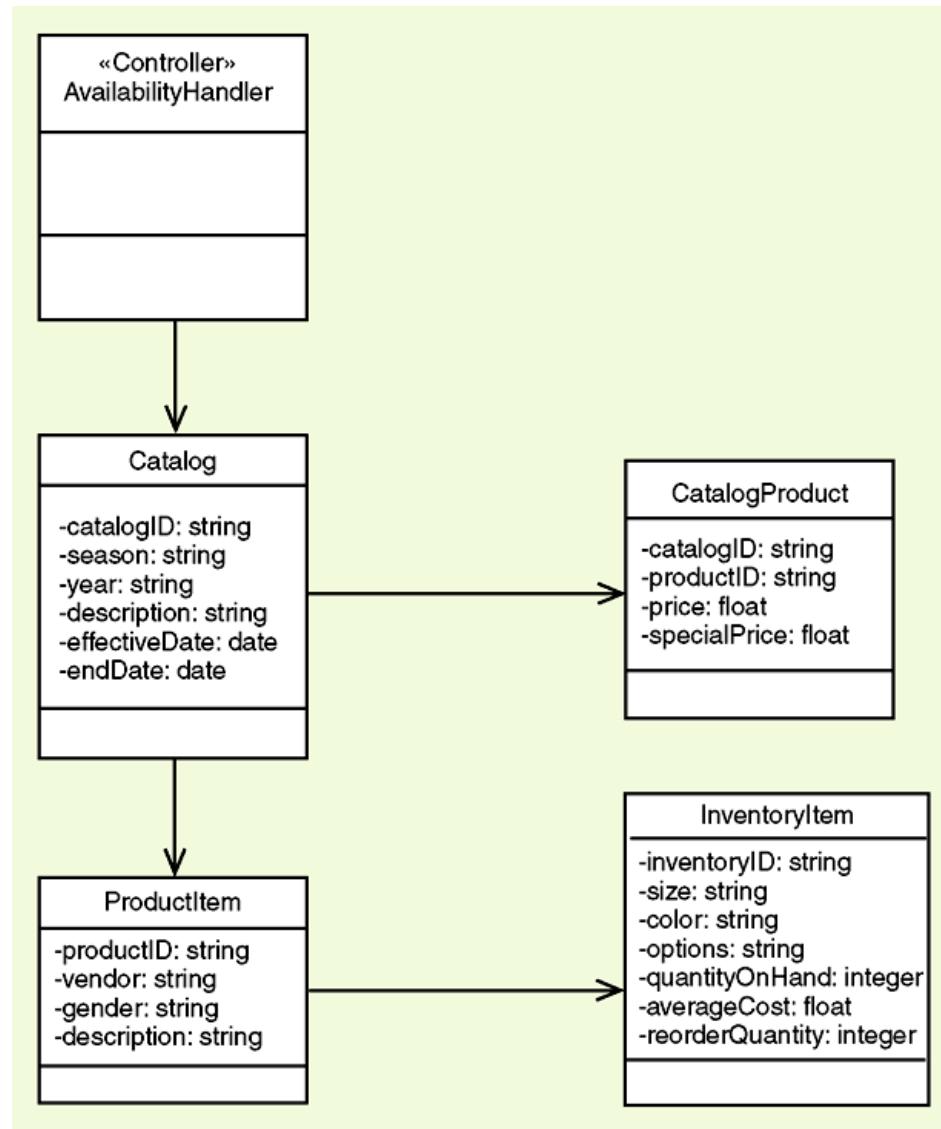
System Sequence Diagram (SSD) for 'look up item availability' use case



Domain class diagram for RMO sales subsystem



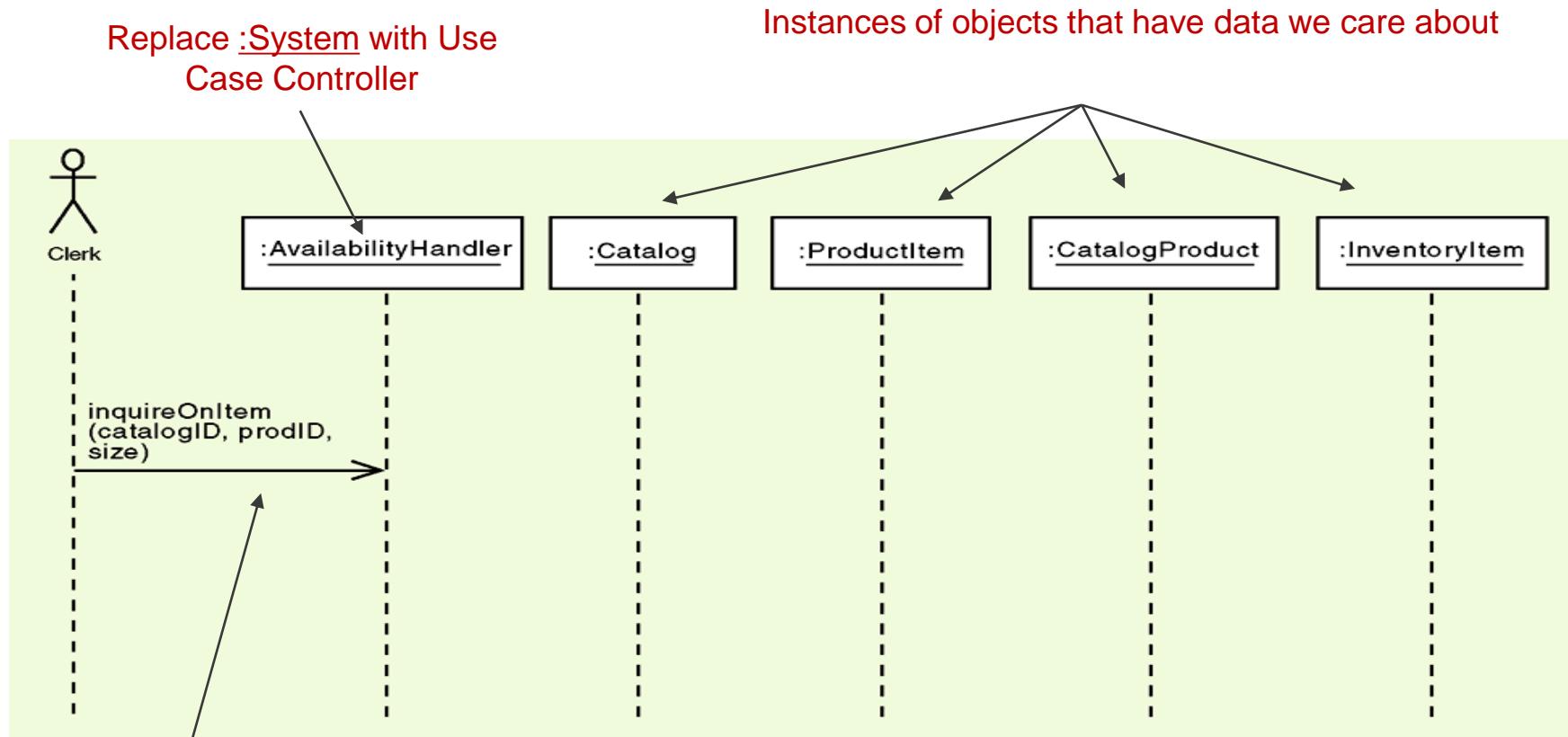
Partial design class diagram for the ‘look up item availability’ use case



Utility class: Use case controller

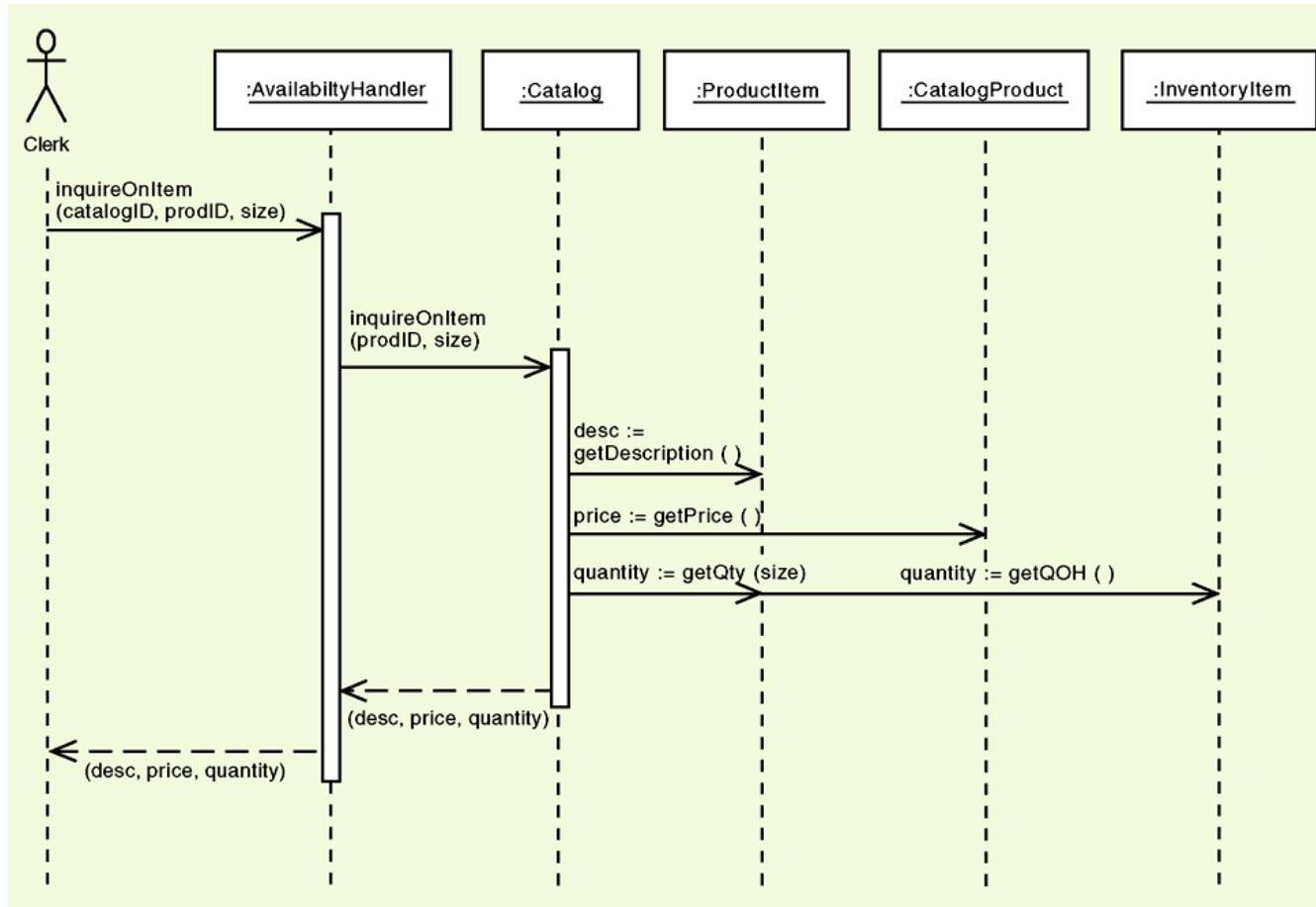
- Design pattern - a repeatable approach to solving a problem .. a recipe
- Use case controller is one of the oldest software design pattern classes
 - a utility class - created to handle necessary system functions, tasks not part of domain object responsibility
 - designed as collection point for incoming messages
 - use case controller acts as intermediary between outside world and internal system
 - contains only that logic necessary to coordinate but not control the execution of the use case

Objects included in ‘look up item availability’ use case



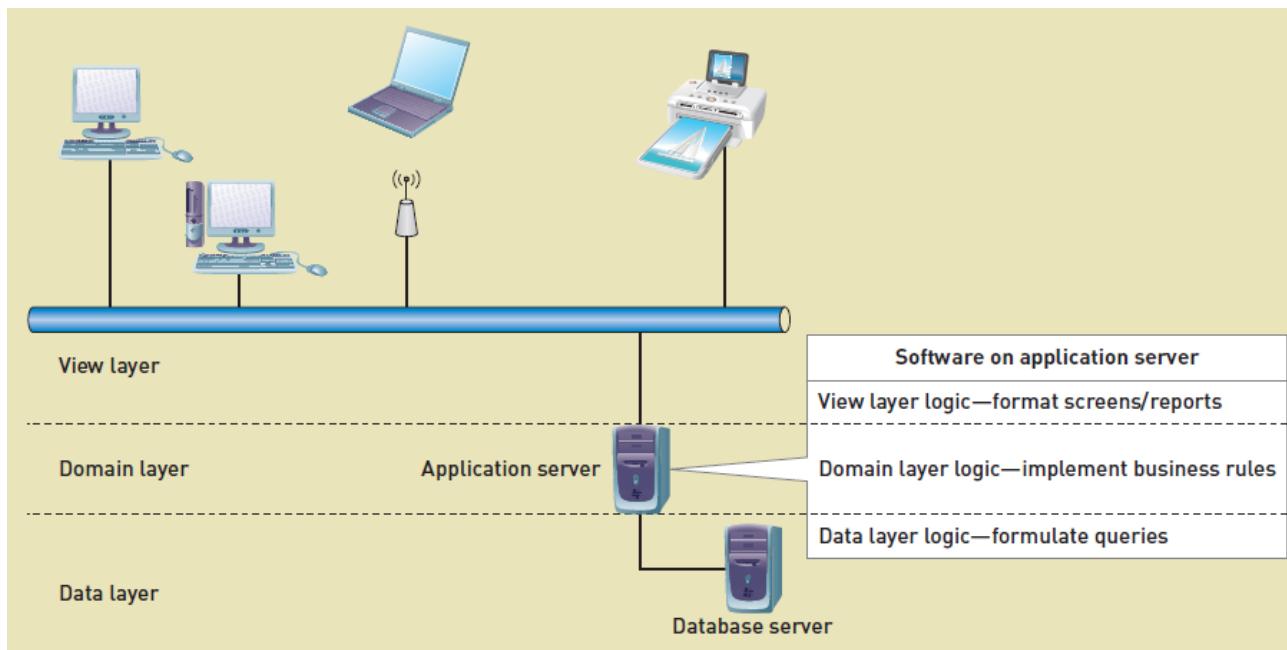
Same message that was originally sent to :System

First-cut sequence diagram for ‘look up item availability’ use case



Final cut sequence diagrams ... multi-layer design – How?

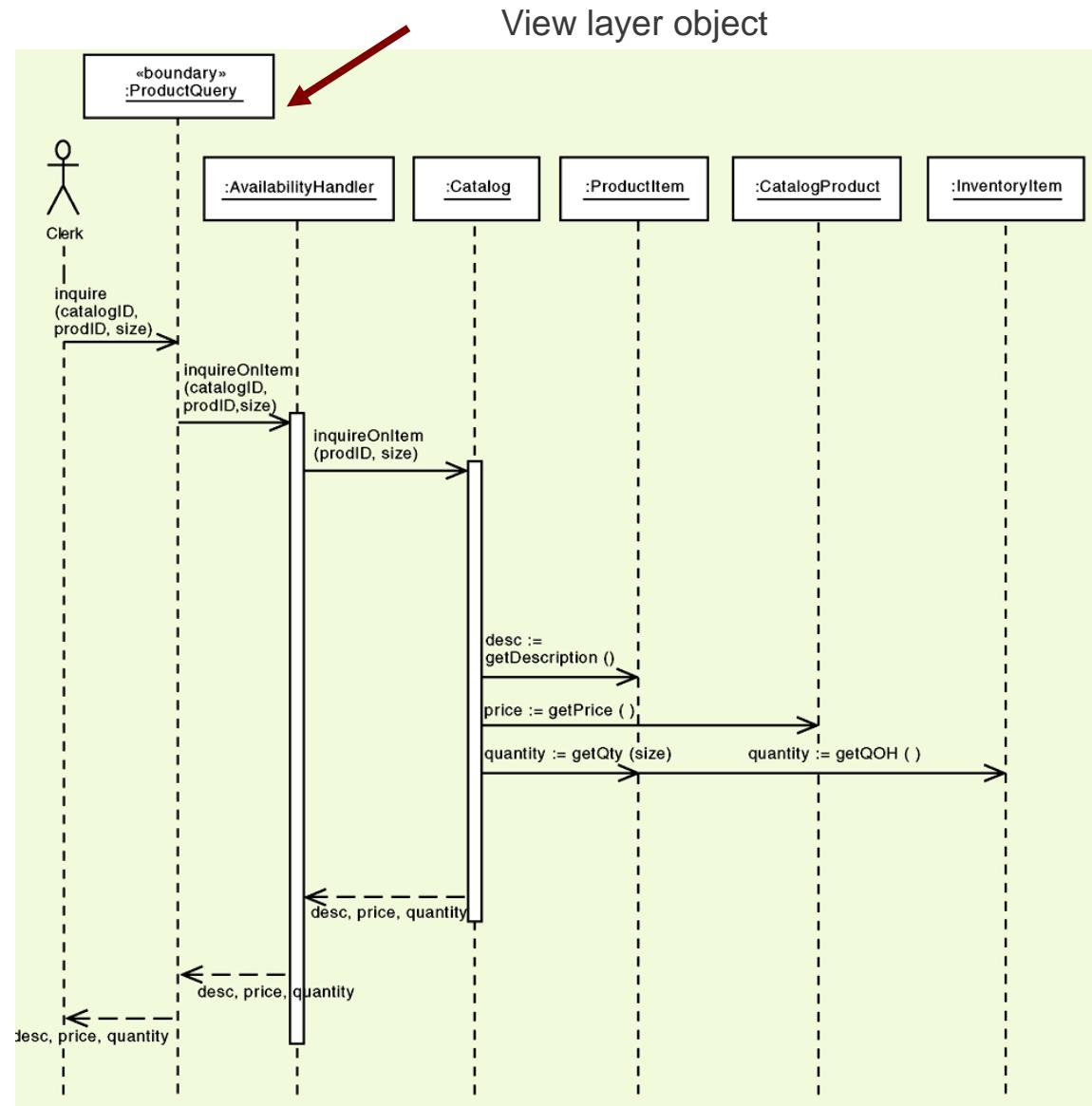
- Final cut sequence diagrams represent a three-layer architectural design



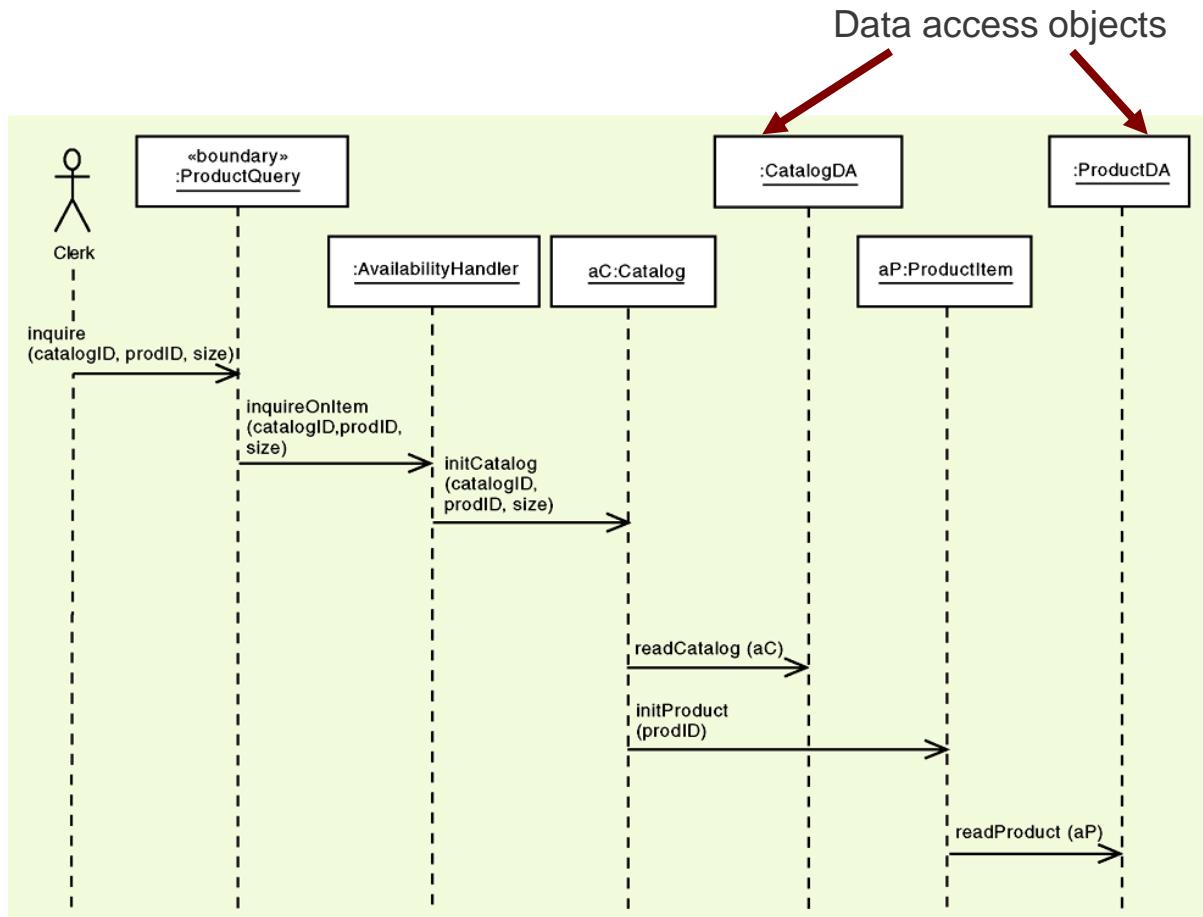
Final cut sequence diagrams ... multi-layer design – How?

- Add a view layer interface class before the controller either as a single GUI class or as Windows classes
- Add a data access class for each problem domain class
 - data access layer should only support database CRUD operations so classes maintain a high level of cohesion and are loosely coupled with the business layer

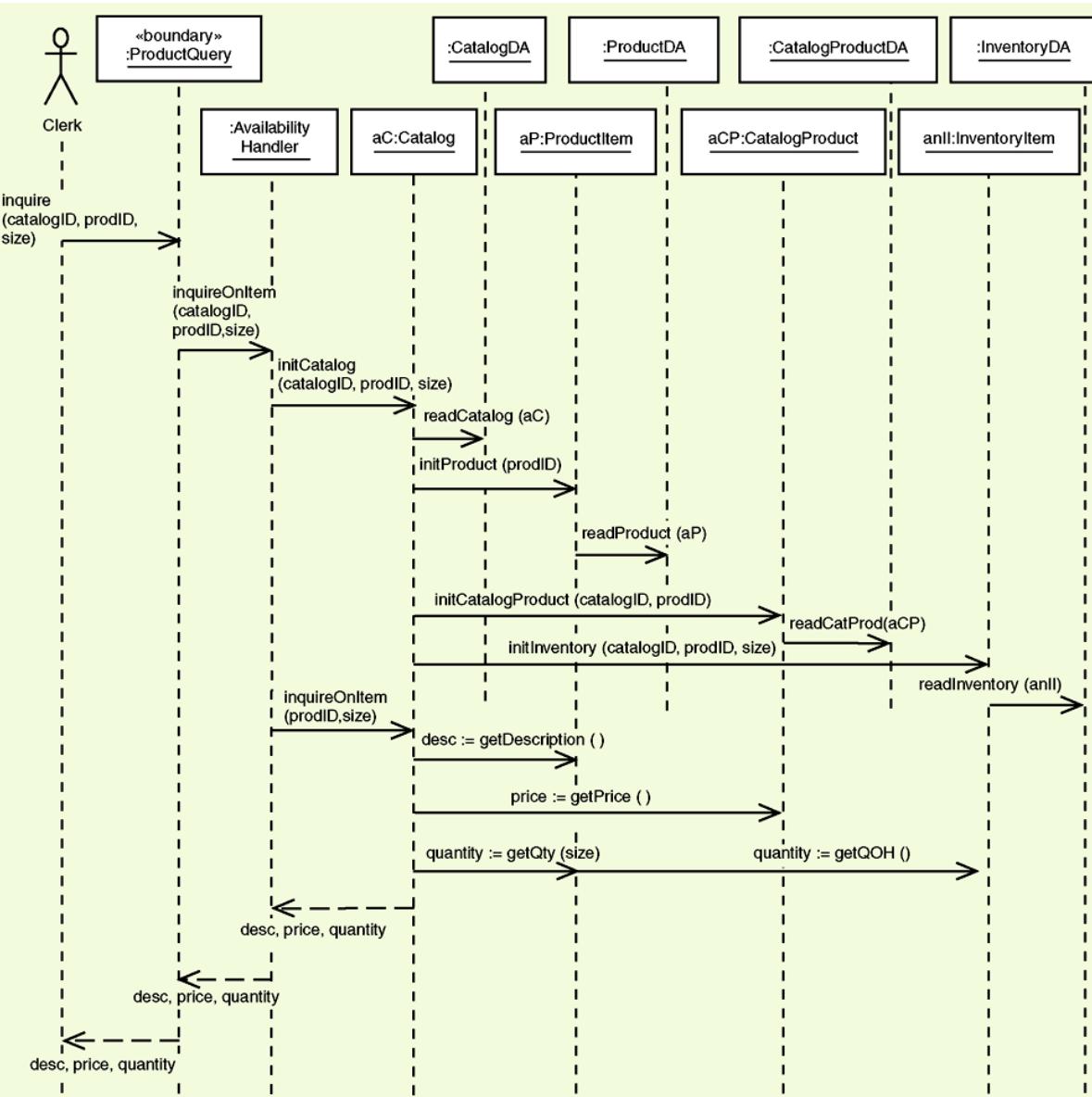
'Look up item availability' use case with view layer and user interface object



Partial three-layer design for ‘look up item availability’ use case with data layer



**Completed
three-layer
design for
'look up
item
availability'
use case**



Iterative OO Design process

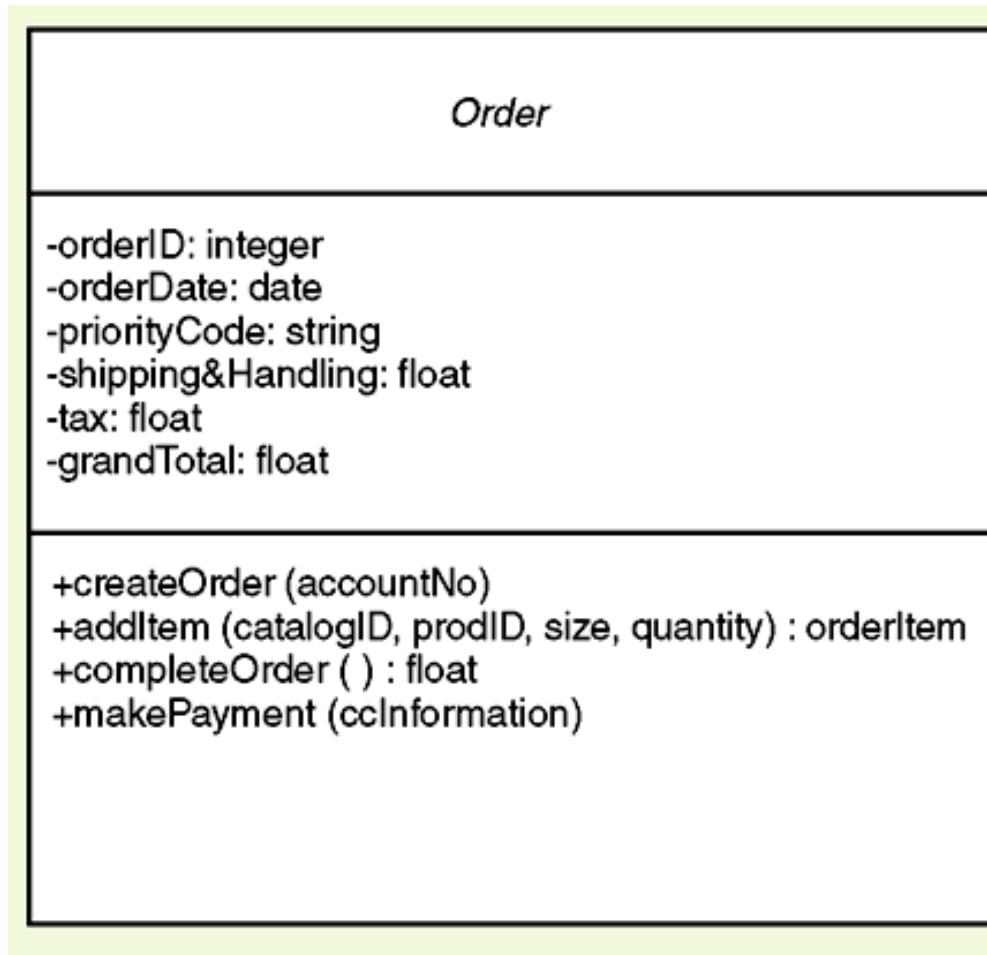
1. Create first-cut (preliminary) design class diagram
2. Create sequence diagrams, for each use case
...realisation of use cases
3. **Return to design class diagram and update based on design of sequence diagrams**

Design class diagrams and detailed interaction diagrams inform each other and should be developed in parallel

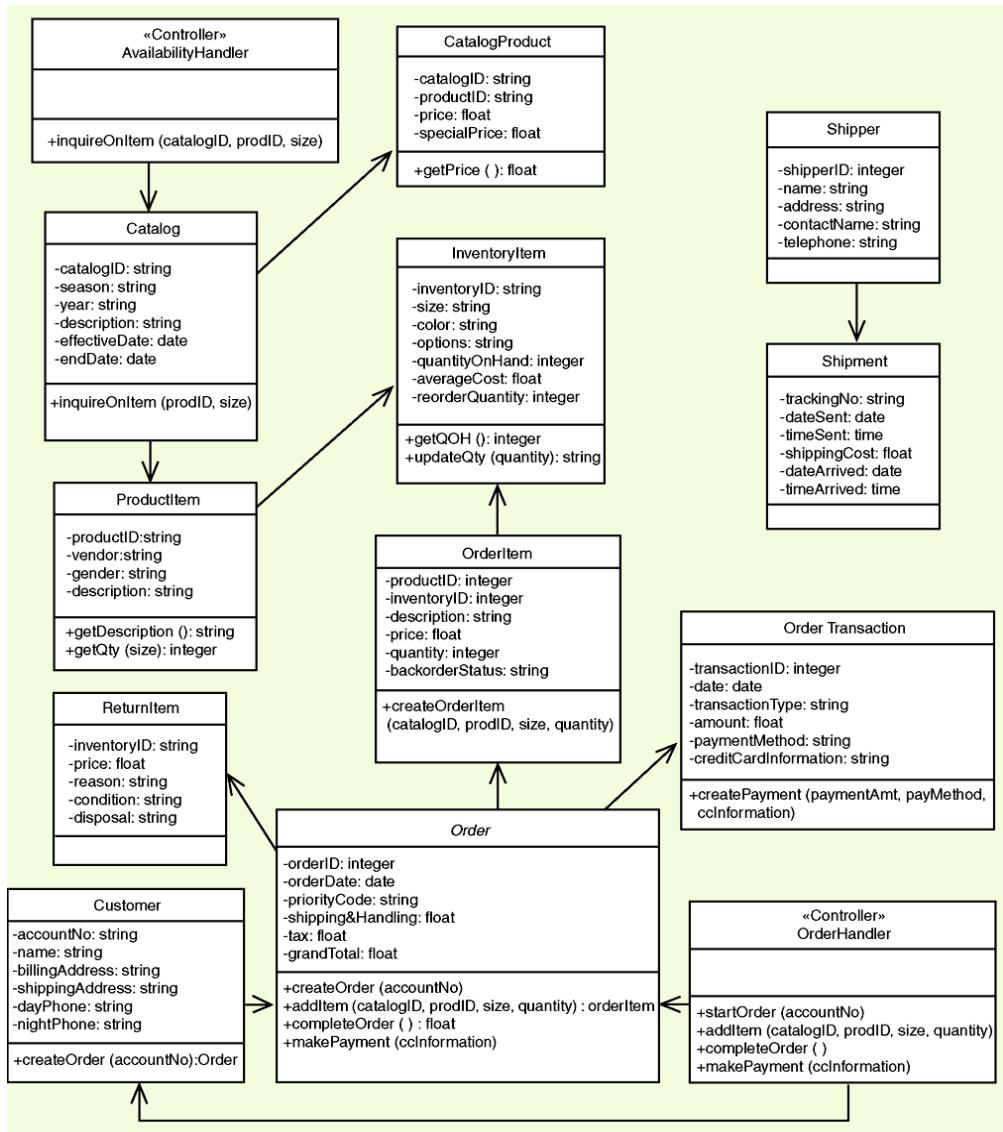
Updating the design class diagram

- Design class diagrams developed for each layer
 - new classes for view layer and data access layer
 - new classes for domain layer use case controllers
- Sequence diagrams messages used to add methods to class diagram
 - constructor, data get/set methods, data access CRUD requests
 - any business-specific methods needed for use case
- Any additional attributes should be added
- Internal consistency of models is crucial

Design class with method signatures



Updated design class diagram for the domain layer



Summary

- Object-oriented design is the bridge between user requirements (in analysis models) and final system (constructed in programming language)
- Systems design is driven by use cases, and focuses on specifying system architecture in detail
 - design class diagrams and sequence diagrams
 - domain class diagrams are transformed into design class diagrams
- sequence diagrams are extensions of system sequence diagrams
- Object-oriented design principles must be applied
 - Coupling: connectivity between classes
 - Cohesion: unity of purpose of an individual class
- Three-layer design used to ensure maintainability

Workshop Preparation

- Watch Seminar 9 before the workshop

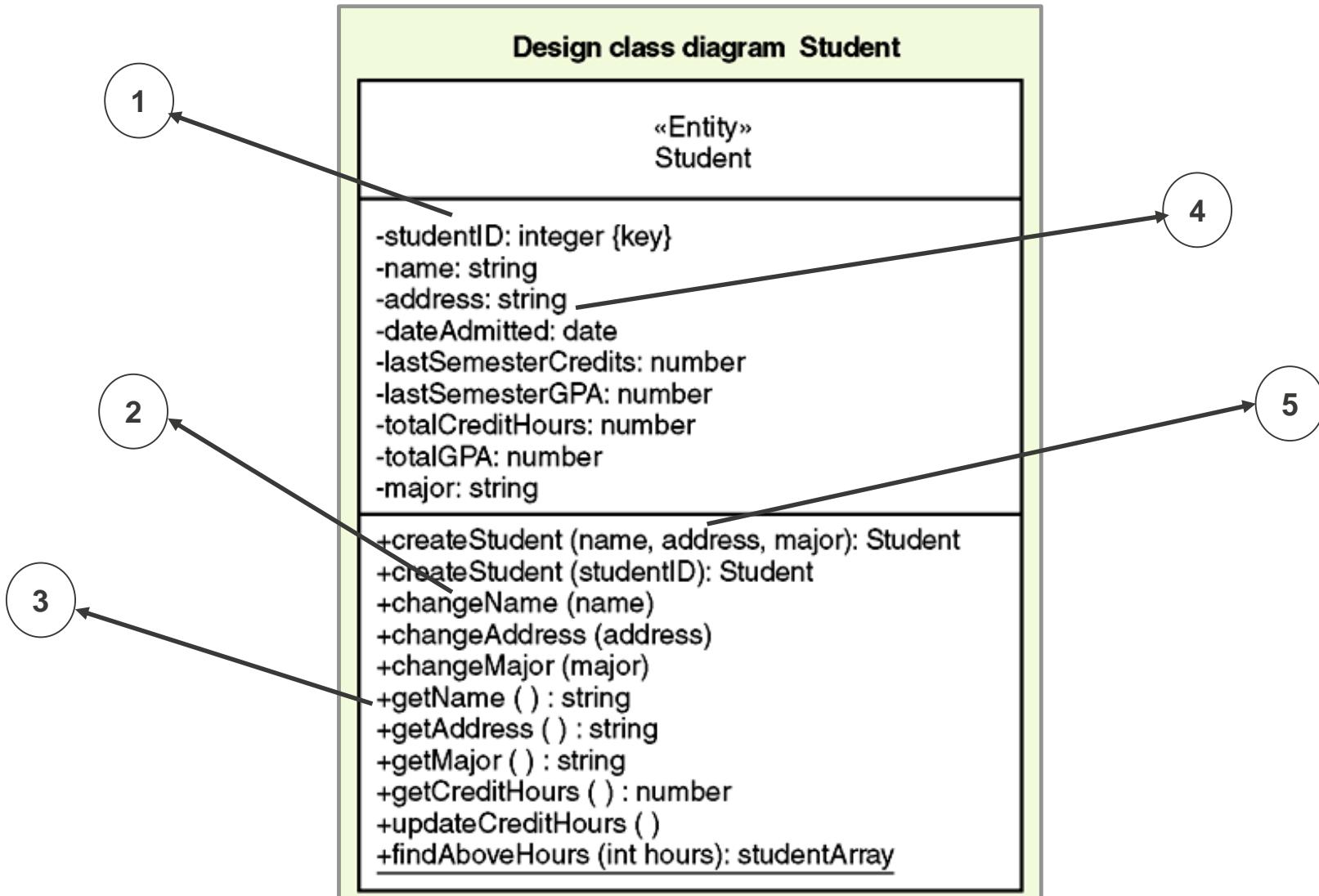
**Thanks for watching
See you next week**

Resources:

Prescribed text:

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 13 (pp. 398-424),
 - For Design Class Diagrams - Chapter 12 (376-382),
 - For System Sequence Diagrams Chapter 3 (139-146)

4.1 Create first-cut DCD: Notation – Methods



For next time

- Next time add types of message and the rectangle boxes on the lifelines
- Clearly explain the purpose of sequence diagrams



MONASH University

Information Technology

FIT2001 – Systems Development

Seminar 10: Security & Testing

Chris Gonsalvez



Our road map

- Security
- Testing

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering and documentation techniques
- Prototyping & Interface Design
- Detailed Design – Design Class Diagrams & Sequence Diagrams

At the end of this seminar you will be able to:

- Understand the importance of ensuring your information system is secure
- Understand the need for testing
- Describe the various types of tests, and explain how and why each is used

Lecture Outline - Security

1. Security – Why?
2. Security model - How?

System Design – Security: Why?

- Information systems need to be secure to be reliable
- Information systems are at risk from:
 - Human error (e.g. wrong data, accidental data deletion)
 - Technical errors (e.g. h/w failure, s/w crash)
 - Accidents and disasters (flood, earthquake, fire)
 - Fraud (e.g. deliberate attempts to corrupt or amend previously legitimate data and information)
 - Commercial espionage (e.g. unauthorised access of sensitive data)
 - Malicious damage (e.g. by internal employees, external hackers)

Information Security Model – How?

The **CIA security model** is the benchmark for evaluating information systems security:

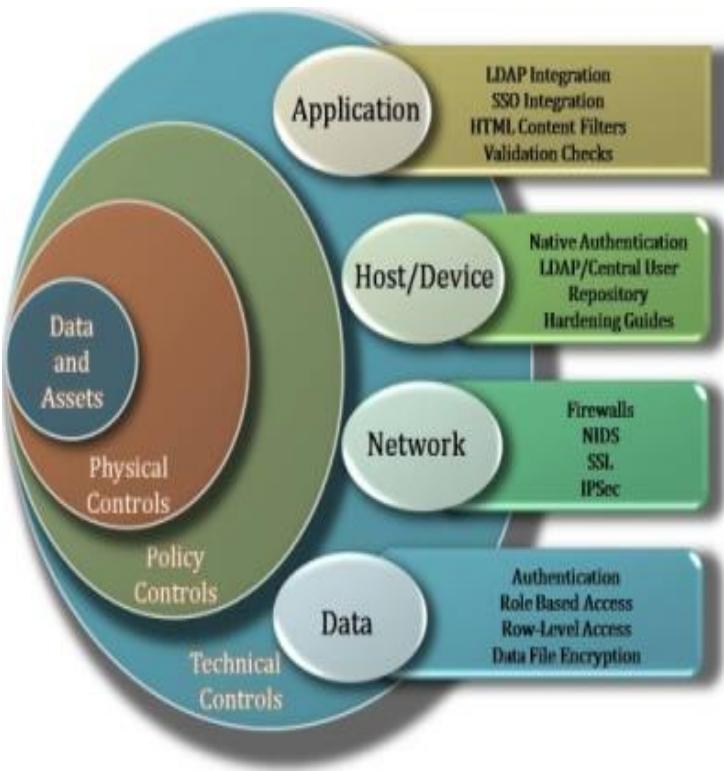
- **Confidentiality** - *Preventing intentional or unintentional unauthorized disclosure*
- **Integrity** - *Prevent unauthorized data modifications*
- **Availability** - *Ensures reliable and timely access to data*

Security – How?

- Information systems can be made more secure by addressing the following factors through a range of controls:
 - ***Prevention*** of errors and breaches
 - ***Detection*** to spot security breaches
 - ***Deterrence*** to discourage breaches
 - ***Data recovery*** to recover lost data or information

Cuts down on losses and legal liabilities
important to every organisation

Security controls



- **Physical controls:** walls, locked doors, guards
- **Procedural controls:** managerial oversight, staff training, defined emergency response processes, fraud controls
- **Regulatory controls:** legislation, policy, rules of conduct
- **Integrity controls:** input controls, access controls, transaction logging, update controls, output controls, fraud controls
- **Redundancy, backup, and recovery controls**
- **Technical controls:** cryptographic software, authentication and authorisation systems, secure protocols

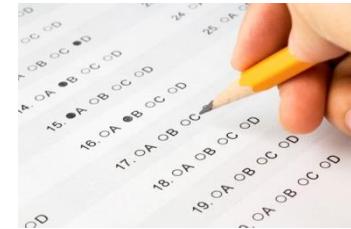
Lecture Outline - Testing

1. Testing – What?
2. Testing – Why?
3. Testing – How?
4. Testing in different methodologies
5. Test methods
6. Testing types
7. Test preparation principles
8. Test planning
9. Test cases
10. Automated testing tools

What is Testing.1?

- Testing represents the process of examining a component, subsystem, or system to:
 - determine if it contains any defects
 - uncovers design flaws and limitations
 - verify that it is ‘fit for purpose’ and meets the needs of the Business/End User
 - reduce the risk of failure

What is Testing.2?



- The primary aim of testing is always:
 - to get the system to fail (i.e. to find errors)
 - rather than to confirm that the system is correct
- Various types of testing exist, and they can be conducted in several ways
- All systems must be tested for:
 - functional
 - non-functional aspects

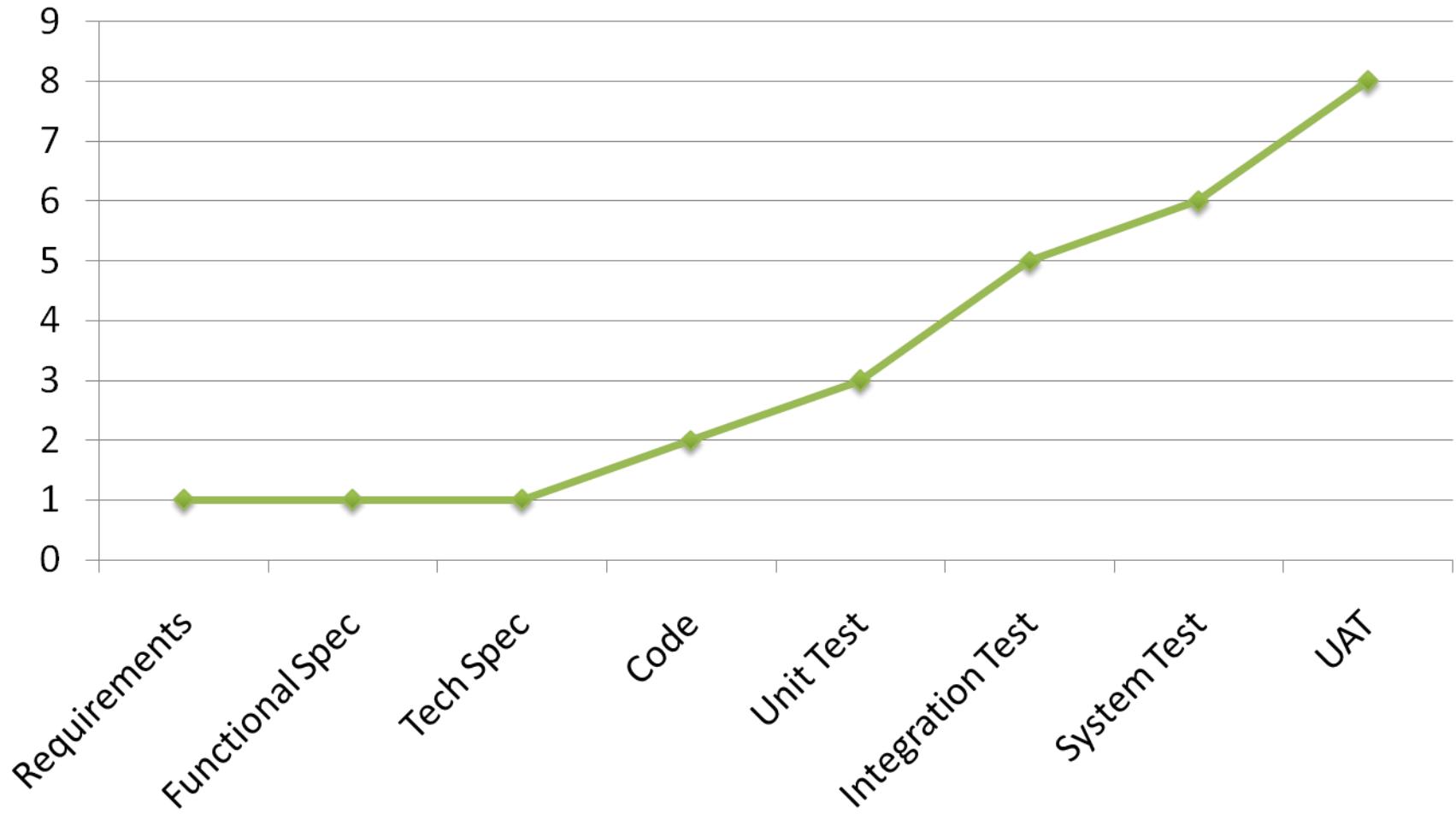
Examples of Software Failures

- MARS Climate Orbiter
 - Simple conversion check was not performed
 - \$125m spacecraft crashed in Martian atmosphere

- Guided-missile Cruiser
 - Use of ‘0’ Zero as a data value in a formula
 - Shut down the ship’s propulsion system
 - Ship was dead in the water for several hours

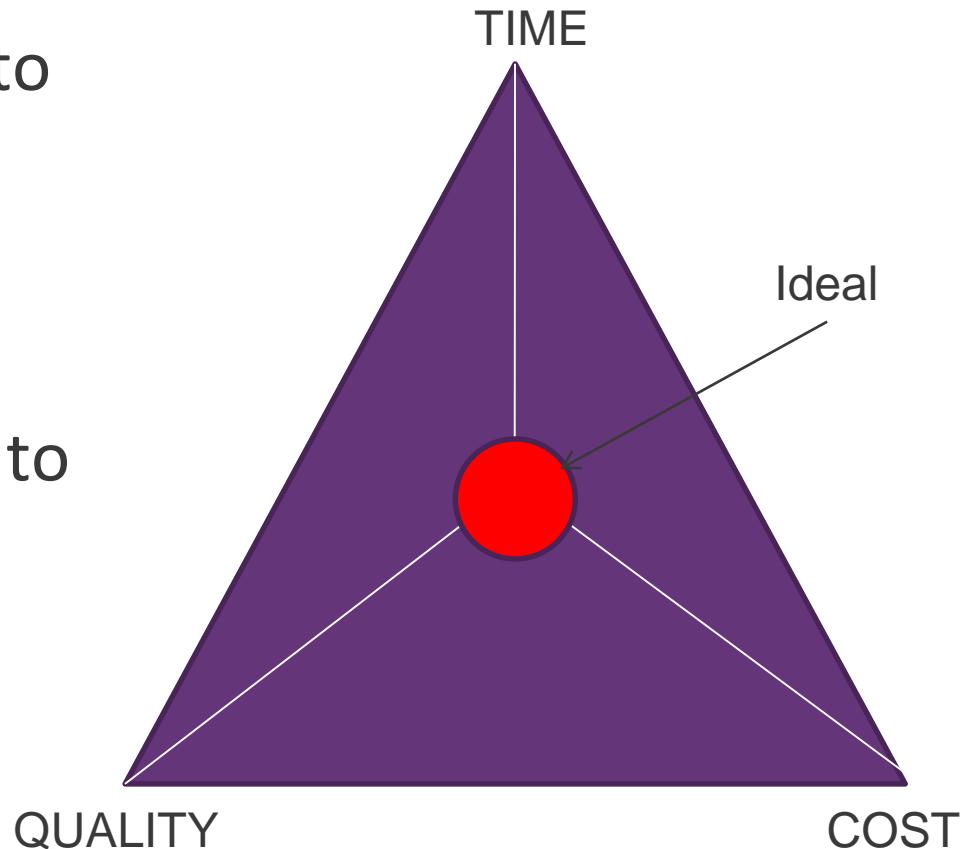


Cost of Errors



We can't find ALL errors ????

- Aim is to reduce the Risk to the business when the solution is implemented
 - Find as many critical problems as possible to reduce impact to clients and their customers
 - Metrics 80/20 rule

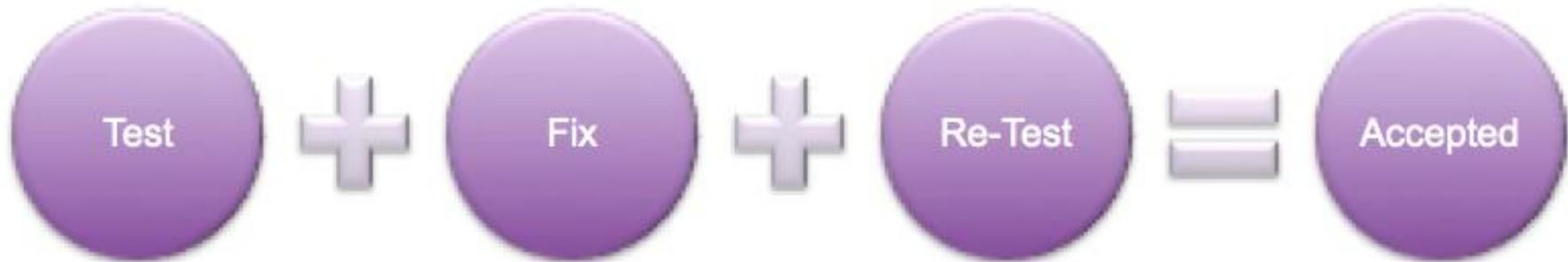


Test Process

- Plan - “What” to test
- Design - “How” you are going to test
- Schedule - “When” you are going to test
- Execute - “What” was the result?

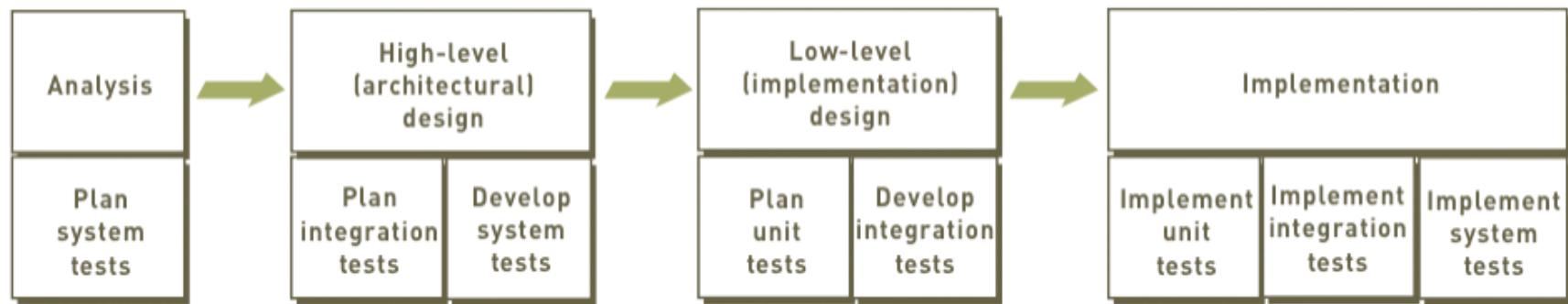


- When defects are found and fixed, must then retest



Waterfall

- Generally all testing and quality control points come late in the project (if time permits)
- Testing is usually scheduled late because people think that testing can only be done on code – NOT TRUE

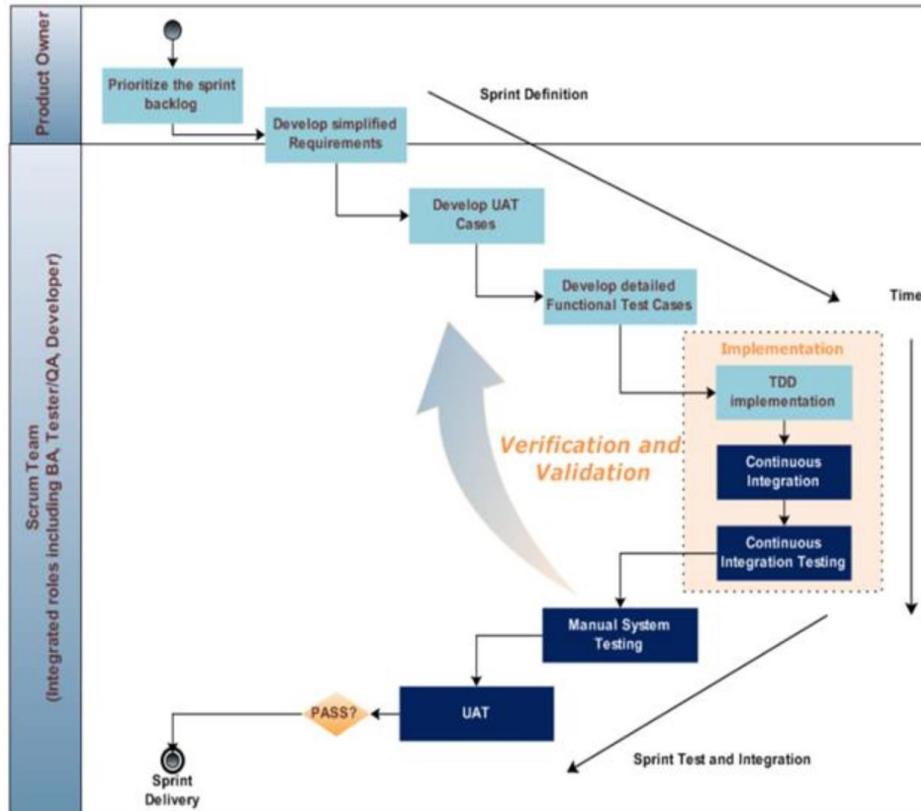


Agile

In **agile development**, testing is integrated throughout the lifecycle; testing the software continuously throughout its development.

- Does not have a separate test phase as such.
- Developers heavily engaged in testing, writing **automated repeatable unit tests** to validate their code.
- Supports the principle of small, iterative, incremental releases.
- Testing is done as part of the build, ensuring that all features are working correctly each time the build is produced.
- **Integration is done as you go.**
- The purpose of these principles is to keep the software in **releasable condition** throughout the development, so it can be shipped whenever it's appropriate.

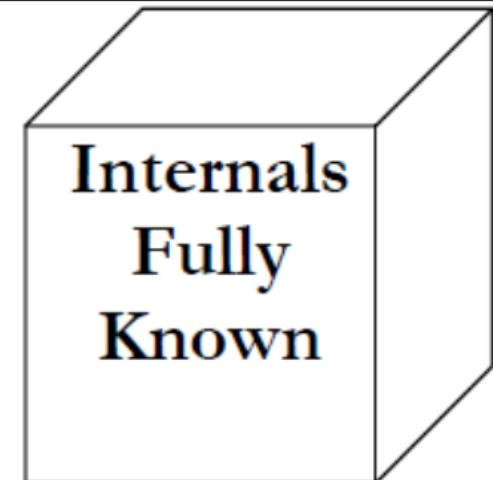
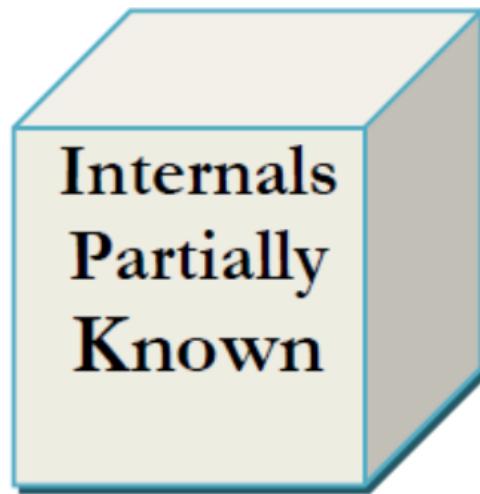
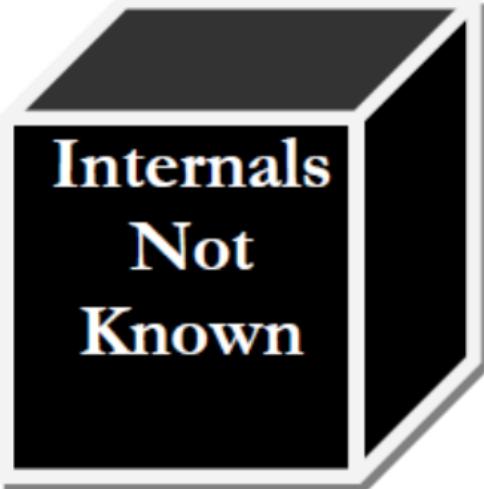
Agile: Testing process



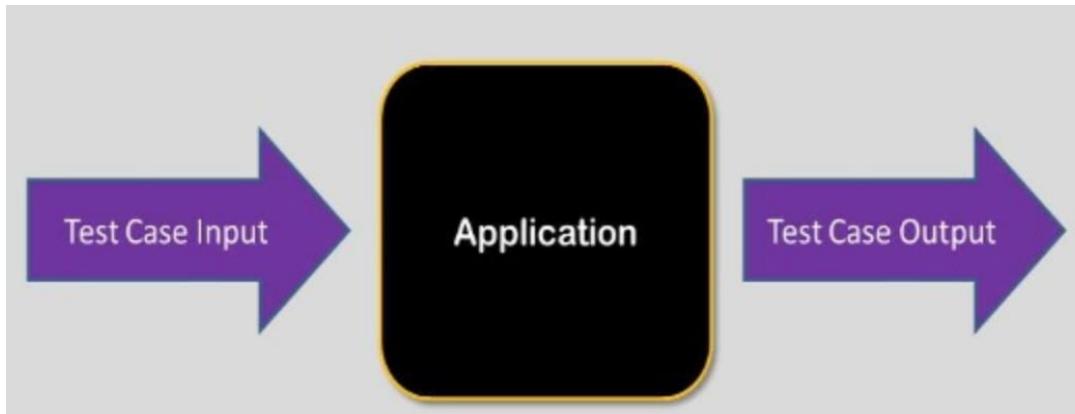
Ref: http://www.codeproject.com/Articles/551161/Agile-Testing-Scrum-and-eXtreme-Programming#_Toc309143137

Test Methods

- The common methods include:
 - Black Box Testing
 - White Box Testing
 - Grey Box Testing

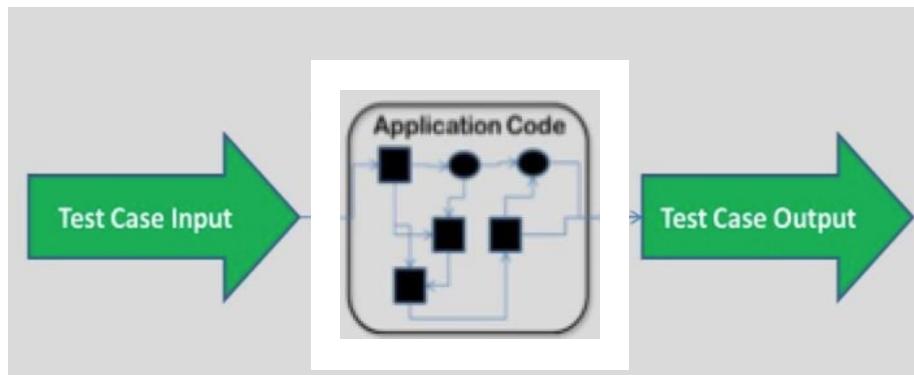


Black Box Testing



- Testing, either functional or non-functional, without reference to the internal structure of the component or system (i.e. code not visible)
- Typically executed in functional test phases i.e. Unit, Integration, System or Acceptance test phases
- Can be independently tested

White Box Testing



- Uses an **internal perspective of the system** to design test cases based on internal structure.
- It **requires programming skills** to identify all paths through the software
- Due to internal perspective, maximum coverage of a scenario is possible

Grey Box Testing



- Is a combination of black box and white box testing
- We look into the “box” being tested just long enough to understand how it has been implemented. Then we close up the box and use our knowledge to choose more effective black box tests.
- Increase testing coverage

5. Test Methods

Comparison between the Three Testing Types

	Black Box Testing	Grey Box Testing	White Box Testing
1.	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2.	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3.	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4.	-Testing is based on external expectations -Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5.	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6.	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7.	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

Testing Types

- The common testing types include:
 - Functional Testing
 - Unit, Integration, System, Acceptance
 - Regression Testing
 - Static & Dynamic Testing
 - Performance, Load & Stress testing
 - Usability testing
 - Accessibility testing
 - Security testing
 - Backup & Recovery testing

Functional Testing

- Testing whereby the system is tested against the requirements specification
- Typically, functional testing involves the following steps:
 - Identify functions that the software is expected to perform.
 - Create input data based on the function's specifications.
 - Determine the output (expected results) based on the function's specifications.
 - Execute the test case.
 - Compare the actual and expected outputs.

Unit Testing

- The testing of individual software components at a coding level
- Unit testing is usually done by the developers themselves, before the module is handed over for integration with other modules of the same solution
- Also known as String, Component or Module testing



Integration Testing



- Integration testing is an interim level of testing applied between unit and system test
- Tests the interaction and consistency of integrated components
- Allows partial system level test without waiting for all the components to be available

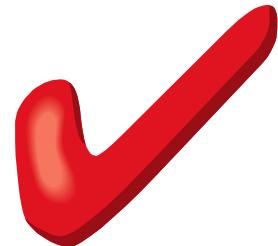
System Testing

- System testing represents testing of the complete system or functionality to be delivered
- System testing is typically performed after all unit and integration testing has been completed
- Includes performance / stress testing
 - Response time, Throughput



User Acceptance Testing (UAT)

- Formal testing with respect to user needs, requirements and business processes
- Conducted to determine whether or not a system satisfies the acceptance criteria – written for user stories
- Enables the user/client to determine whether or not to accept the system
- Acceptance tests often be run by the client themselves with support from the project team
- This is the last stage of testing prior to deployment



Regression Testing

- Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made.
- Can be performed in all test phases

Performance, Load & Stress

- **Performance:** The process of testing to determine the performance and efficiency of a software product or infrastructure.
- **Load:** Testing to evaluate whether a system can handle large quantities of data simultaneously
- **Stress:** Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements

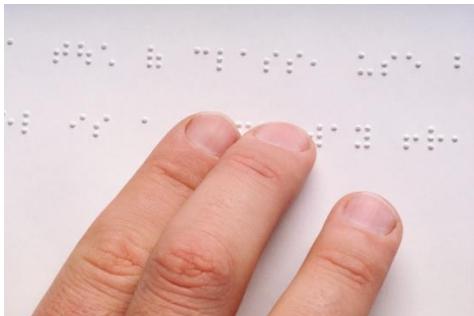


Static Testing

- is a form of software testing where the software isn't actually used.
- It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document.
- It is primarily syntax checking of the code or manually reading of the code or document to find errors.
- This type of testing would primarily be used by the developer who wrote the code, in isolation.

Accessibility Testing

- Testing to determine the ease by which users with disabilities can use a component or system.



Usability Testing

- Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions. [After ISO 9126]
- Typically executed in the later phases of testing when system is more stable i.e. System Test

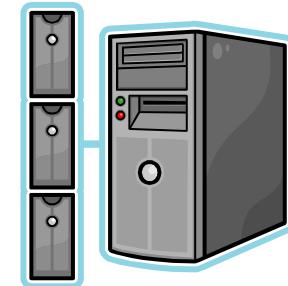
Security Testing



- The process to determine that a System protects data and maintains functionality as intended. The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, authorisation, availability and non-repudiation.

Backup & Recovery Testing

- Taking a backup is the process of taking a copy of an application, its data and environment, such that it may need to be recovered (or restored) in the event of a system failure
- Recovery is the process of reverting to the backup copy of an application, its data and environment such that normal business can be resumed.
- Backup & recovery testing is disruptive so is therefore executed during the quiet times of the later phases of testing

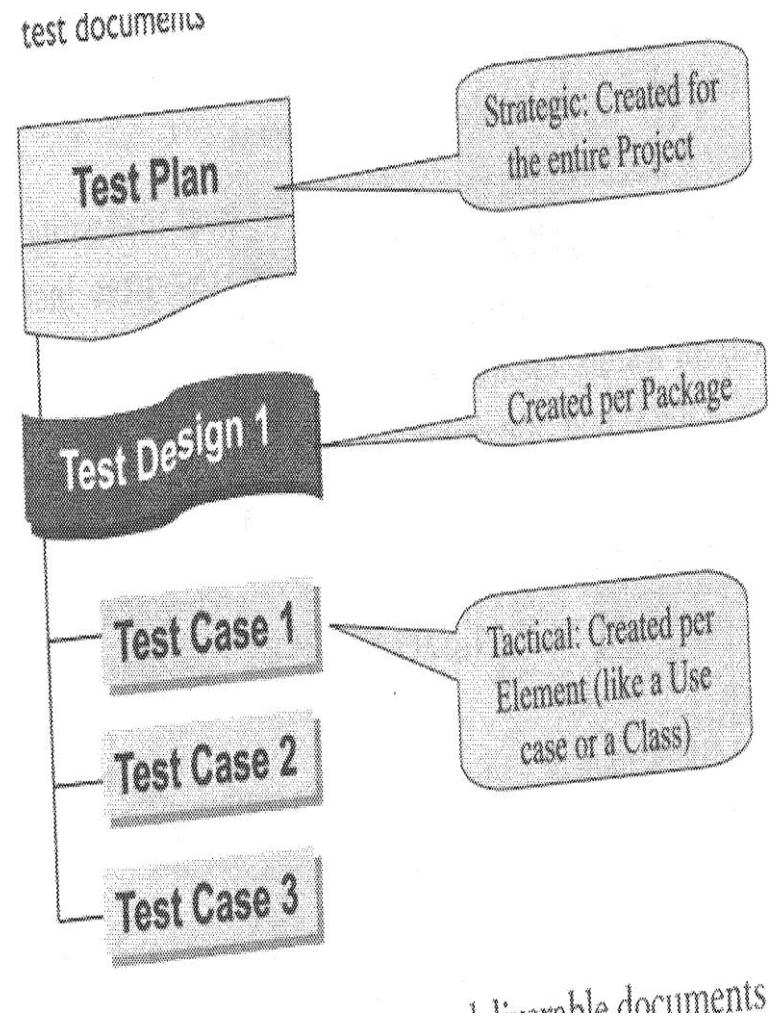


Test Preparation Principles

- Starts early in the project lifecycle
- Produce a Test Strategy Plan first
- Review all documentation
 - Business Requirements
 - Functional Requirements
- Design test requirements/conditions
- Design test cases (manual test) or scripts (automated test) to meet the test requirements
- Prepare the test environment(s) and required data

Test plan & test cases

- Testing needs to be planned in detail
- A **test plan** is prepared that identifies:
 - how and when testing will be done, how long it will take
 - types of tests, things to be tested/not tested
 - resources, staff, facilities, tools, training needed
 - schedule
 - risks
- **Test cases and comprehensive test data** need to be prepared
 - time required for this is often underestimated .. this results in incomplete/inaccurate tests, inadequate test data



Test cases

- A **test case** is a formal description of
 - Starting state
 - Events to which software responds
 - Expected response or ending state
- Important part of testing is specifying test cases and test data
- Analysis phase documentation is useful in preparing test cases (use-case driven)
- **Test data** is defined to be used with a test case

Developing test cases

- Test cases are developed for each use case
 - Requirements traceability
- Test cases represent usage scenarios
 - Test for normal operations
 - Test for significantly different operations
- Test cases comprise the actions to be taken in the steps of the scenario, and the expected system responses

Developing test cases

Test cases are developed for each use case

Test cases can be prepared once the detail of the use case is defined.

The following tasks are involved in developing test cases from use cases:

- **Task 1:** Identify relevant scenarios (event flows)
 - basic flow and alternative flows
- **Task 2:** Identify the variables for each use case step
- **Task 3:** Identify the significantly different options for each INPUT variable
- **Task 4:** Combine options to be tested to form test cases
- **Task 5:** Assign values to variables

Illustrated with an examples in the next few slides

Use case ‘Recording traffic ticket’

Use Case Name:	Record traffic ticket									
Scenario:	Record traffic ticket									
Triggering Event:	Officer sends in new ticket									
Brief Description:	The officer gives the traffic ticket to the clerk. Using the information on the traffic ticket, the clerk first verifies the officer by entering the badge number. Then, the clerk verifies the driver information by entering the driver’s license number. Finally, the clerk enters the ticket information.									
Actors:	Clerk									
Stakeholders:	Manager Officer									
Preconditions:	The officer must exist. The driver must exist.									
Postconditions:	The ticket must exist and be associated with the driver, the officer, and a court.									
Flow of Events:	<table border="1"> <thead> <tr> <th>Actor</th> <th>System</th> </tr> </thead> <tbody> <tr> <td>1. Clerk enters officer badge number.</td> <td>1.1 System reads officer information and displays the name.</td> </tr> <tr> <td>2. Clerk enters driver’s license number.</td> <td>2.1 System reads the driver information and displays the driver name and address.</td> </tr> <tr> <td>3. Clerk enters ticket information.</td> <td>3.1 System displays ticket and driver information.</td> </tr> </tbody> </table>	Actor	System	1. Clerk enters officer badge number.	1.1 System reads officer information and displays the name.	2. Clerk enters driver’s license number.	2.1 System reads the driver information and displays the driver name and address.	3. Clerk enters ticket information.	3.1 System displays ticket and driver information.	
Actor	System									
1. Clerk enters officer badge number.	1.1 System reads officer information and displays the name.									
2. Clerk enters driver’s license number.	2.1 System reads the driver information and displays the driver name and address.									
3. Clerk enters ticket information.	3.1 System displays ticket and driver information.									
Exception Conditions:	1.1 Officer is not found. 2.1 Driver is not found.									

Task 1: Identify basic and alternative event flows

- Basic flow in the recording traffic ticket use case

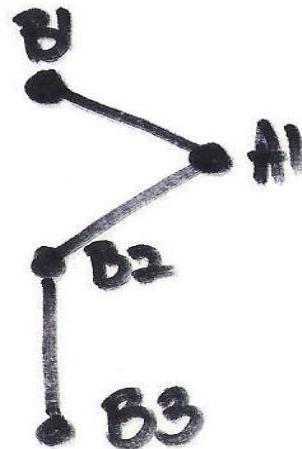
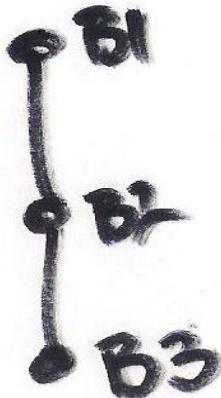
1. Clerk enters officer badge number. (B1)
 - system reads officer information and displays name
2. Clerk enters driver's licence number. (B2)
 - system reads driver information and displays driver name and address.
3. Clerk enters ticket information. (B3)
 - system displays ticket and driver information.

- Alternative flows

- 1.1 Officer is not found (A1)
- 2.1 Driver is not found. (A2)

Task 1: Identify basic and alternative event flows

- Basic flow is a straight line down – sunny day scenario
- While alternative flows are usually deviations from straight line.
- Four possible scenarios are indicated. There are more scenarios than the no. of alternative flows



Scenario 1:
Basic flow
(straight line)

Scenario 2:
Alternative
flow A1

Scenario 3:
Alternative
flow A2

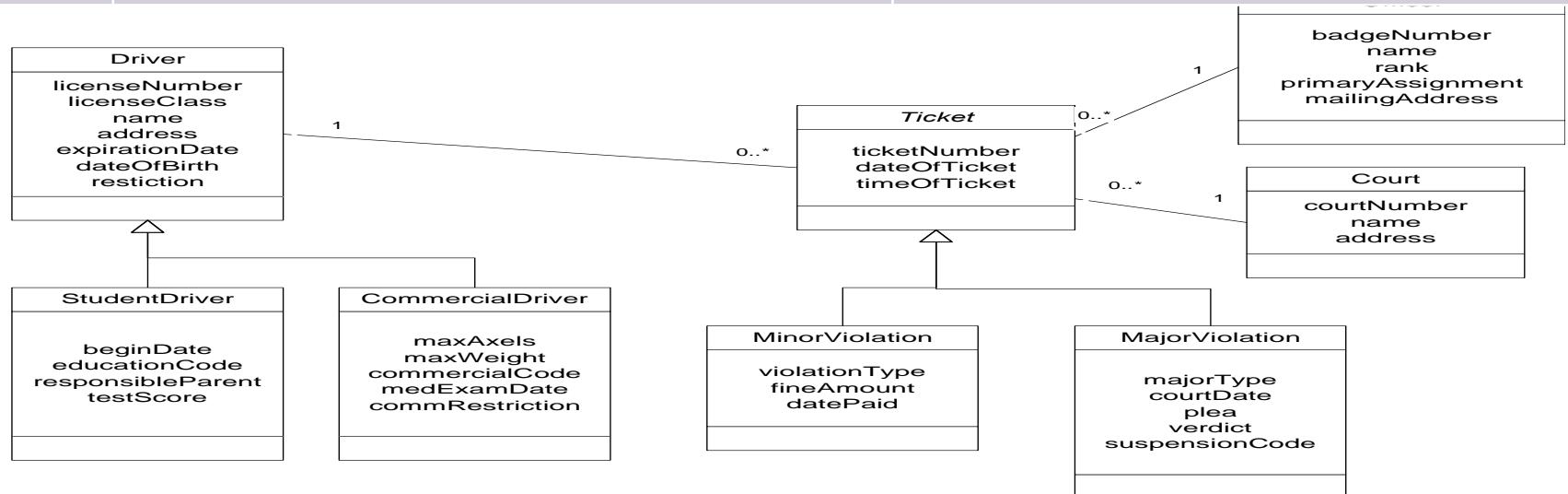
Scenario 4:
Alternative
flows A1,A2

Exhaustive Testing

- This means to test every possible path through the system with every possible data combination.
- This type of testing is not possible.
- Simple Example
 - 20 different inputs, all independent
 - Each value possible 4 variables
 - Exhaustive testing would mean 4^{20} tests
 - This equates to 1,099,511,627,776 test cases
- Other options are available
 - Equivalence Partitioning, Boundary Value Analysis, Sampling and Risk Based Testing

Task 2: Identify the variables at each step

Steps	Variables	Input variable
B1	badge number (input), officer name	badge number (input)
B2	licence number (input), driver name, driver address	licence number (input)
B3	ticket no, date of ticket, time of ticket (input)	ticket no, date of ticket, time of ticket (input)

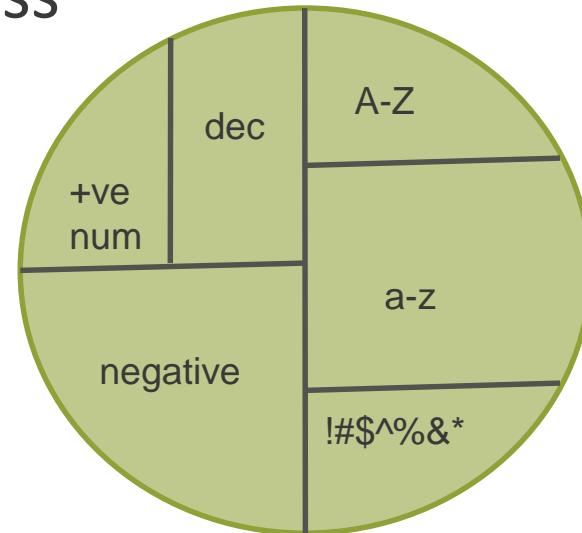


- Use the class model to identify variables

Task 3: Identify the significantly different options for each INPUT variable

Equivalence Partitioning:

- Based on the premise that by testing one value within a class is representative of all values within that field
- Only one data value from a class needs to be tested as it is considered as representative of all values within that partition



Task 3: Identify the significantly different options for each INPUT variable

Badge no. (integer, 8 digits)	Licence number (integer, 10 digits)	Ticket No. (integer, 6 digits)	Ticket date	Ticket time
<ul style="list-style-type: none"> must be greater than 0 Leading 0s must be entered Maximum value 99999999 Value should be in the database (Badge number is a key) No spaces Only numeric 	<ul style="list-style-type: none"> must be greater than 0 Leading 0s must be entered Maximum value 99999999 Value should be in the database (License number is a key) No spaces Only numeric 	<ul style="list-style-type: none"> must be greater than 0 Leading 0s must be entered Maximum value 99999999 No spaces Only numeric 	<ul style="list-style-type: none"> Format must be DDMMYY No spaces Only numeric 	<ul style="list-style-type: none"> Format must be HHMM No spaces Only numeric



Task 3: Identify the significantly different options for each INPUT variable

Boundary Value Analysis:

- Extension of Equivalence, concentrates on maximum and minimum values
- To validate the boundary, tests must be carried out on either side of the boundary with the boundary values increased and decreased

Significantly different option is one when an input variable/action ...

- Triggers a different (alternative) flow
- Triggers an error message
- Changes the user interface
- Causes different options to show in a drop-down list
- Is an input to a business rule
- Border condition
- Changes a default
- Entry format is not clearly defined
- International format differences (dates, currencies)

Options identified: Badge number

- Blank entry, zero entered, negative number, leading 0s not entered, correct number in database, correct number not in database, number too large, alpha only entry, alphanumeric entry

Task 4: Combine options to form test cases

Step	Input variable	Test Case 1 (TC1)	Test Case 2 (TC2)	Test Case 3 (TC3)	Test Case 4 (TC4)
B1	Badge no	Valid badge no	Valid badge no, but not stored in database	Invalid badge no	Valid badge no
B2	Licence No.	Valid licence no.	Valid licence no.	Valid licence no.	Valid licence no. but not stored in db
B3	Ticket info.	Valid ticket info.	Valid ticket info.	Valid ticket info.	Valid ticket info.

Task 4: Combine options to form test cases (cont.)

Step	Input variable	Test Case 5 (TC5)	Test Case 6 (TC6)	Test Case 7 (TC7)	Test Case 8 (TC8)
B1	Badge no	Valid badge no	Valid badge no,	Valid badge no	Valid badge no
B2	Licence No.	Invalid licence no (< 10 digit).	Invalid licence no. (1 digit)	Valid licence no.	Valid licence no.
B3	Ticket info.	Valid ticket info.	Valid ticket info.	Invalid ticket info.(wrong ticket no)	Invalid ticket info. (invalid date)

Task 5: Assign values to variables to TC1

Step	Action	Value	Expected result	Actual result	Pass/Fail	Comments
B1	Enter a valid badge number	98239289	Officer name “Jack Smith” is displayed	Officer name “Jack Smith” is displayed	Pass	
B2	Enter a valid driver's licence number	8112823	Driver name “Jane Doe” is displayed	Driver name “Wendy Smith” is displayed	Fail	
B3	Enter valid ticket information	12345, 24052013, 1505	Error message displayed – date wrong format	Ticket information is displayed	Fail	

Automated testing tools

- Document and create test cases
- Automated running of test cases
 - Including GUI scripts
 - Load testing (might run on hundreds of client machines to test typical load)
- Recording of results
 - Comparison of actual to expected results
 - Comparison of different test trials

Workshop Preparation

We will cover this material in Week 12

The Workshop for Week 11 will be
focussed on Exam Revision

Thanks for watching

Resources:

Satzinger, J. W., Jackson, R.B., Burd, S.D. and R. Johnson
(2016) Systems Analysis and Design in a Changing World, 7th
Edition, Thomsen Course Technology

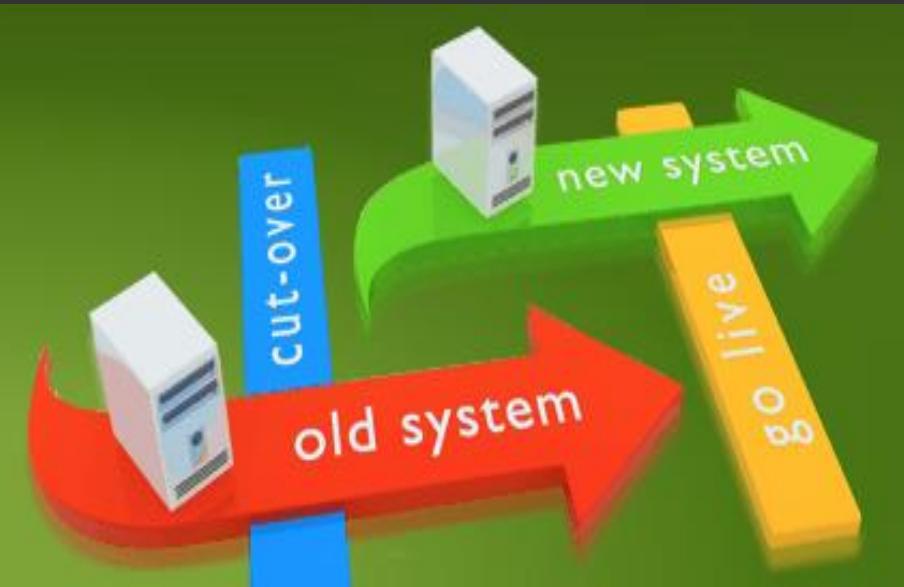
- Security: Chapter 6, pp. 168 - 179
- Testing: Chapter 14, 446 - 453



FIT2001 – Systems Development

S11 - Implementing & Maintaining the System

Chris Gonsalvez



Our road map

- Implementation
- Build/Buy, Outsourcing
- Maintenance / Support

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering and documentation techniques
- Prototyping & Interface Design
- Detailed Design – Design Class Diagrams & Sequence Diagrams
- Testing

At the end of this seminar you will:

- Understand the range of activities in the Implementation phase of the SDLC;
- Understand the advantages and limitations associated with building vs. buying IT applications
- Appreciate the reasons for which many organisations outsource all or parts of their IT function, and recognise the benefits and concerns
- Be aware of the significance of support activity in the work of an IT department and, understand the different types of support activities.

Lecture Outline

Implementing the System – Tasks

1. Implementation Planning
2. Build vs. Buy the System / Outsourcing
3. Testing (covered in Lecture 9)
4. Documentation
5. Get the system ready for production
 - 5.1 Data Conversion / Migration
 - 5.2 Training
6. Deploy the system
7. Wrap up / Transition to support phase
 - 7.1 Maintenance
 - 7.2 Change management
 - 7.3 System closure / Post Implementation review

Implementing the System.1

1. Implementation Planning

- *Review Acceptance Checklist, Prepare Implementation Schedule*

2. Build the System or Buy the System

- *Will result in a varied implementation path*

3. Test the System (covered in Week 10)

- *System Testing (functional and performance), Acceptance Testing*

4. Finalise documentation

- *System documentation, User documentation*

Implementing the System.2

5. Get ready for the System to go into production

- *Data conversion / migration, Configure the production environment, Conduct training*

6. Deploy the system

- *Install / deploy the system, Monitor operations, Benchmark testing, Tune the system,*

7. Wrap up

- *Operations handover, Transition to Support, System Closure, Post-implementation review*

Note: Implementation tasks are carried out throughout the development process

Implementation Planning

- Implementation Plan

- *Developed early in SDLC for Waterfall, at the start of each sprint for Agile or varies depending of when product is going to be released*
- *Needs to be constantly reviewed/updated to reflect changes during development*
- *Requires a great deal of co-ordination - involves many professionals external to the IT development team*
- *Poor planning can cause significant delays to the deadline*
 - *Data not ready*
 - *Training not completed*



Build vs. Buy



Build vs. Buy

BUILD

Custom build IT applications in-house

BUY

Purchase or lease standard IT applications
that are commercially available

- may range from simple to very complex systems

BUY - Useful ...

- When you need an IT application for a generic company function eg. payroll
- When resources for in-house development are in short supply
- When the BUY option is more cost effective than in-house development
- Usually most of the design and implementation tasks are done so there is significant time saving
- Because the system and documentation are usually maintained by the vendor
- Because the design specification is fixed, so no reworking and users have to accept it
- Politically because:
 - external work is often perceived as being superior to an in-house effort so its easier to get a new systems into the company
 - easier to get management support because of fixed costs
 - problems can be attributed to the package rather than internal sources so cuts out a source of internal conflict

BUY – Limitations

- Very rare to find a package that can do everything well that a user wants
- Often need to develop specialised package additions because the multi-purpose packages do not handle certain functions well
- Conversion and integration costs can sometimes be so significant as to render the project infeasible
- Some vendors refuse to support packages which have been customised by the users .. and most packages need some customisation
- Customisation can be so extensive that it would have been cheaper to develop the system in-house

Build vs. Buy? Further considerations

- Functionality
- Cost
- Vendor Support
- Viability of Vendor
- Widespread use by others(client references)
- Flexibility
- Documentation
- Performance and scalability
- Ease of Installation



Information Technology

IT Outsourcing



IT Outsourcing – What?

The practice of turning over some or all of an organisation's IT applications and/or operations to an external provider

- Involves a contractual agreement that involves the exchange of services for payment
- Often includes the transfer of assets (eg staff)

According to Forrester Research:

- Overall investment in tech outsourcing and hardware maintenance in 2019 estimated to be \$703 billion

IT Outsourcing – Why?

Benefits

- Reduced costs
- Strategic – focus on core business activities
- Access – to latest skills and technologies

.... favourable market reaction

– increase in share price

Outsourcing – Need to consider:

- Which aspects of the business will be outsourced – value proposition
- Effect on the business
- Cost to get these services delivered
- Level of required service
- Commitment of the stakeholders
- Board, senior management, staff
- How the relationship be governed - contracts
- Transition
- Selection of external organisation

The Outsourcing process - Phases

The six phases of activity can be defined as follows:

Assess: Define objectives and assess capacity.

Prepare: Service level definition and RFP creation.

Evaluate: Response evaluation and supplier selection.

Commit: Contract development and finalisation.

Transition and Transformation: New service implementation.

Optimise: On-going supplier and vendor management.

Testing

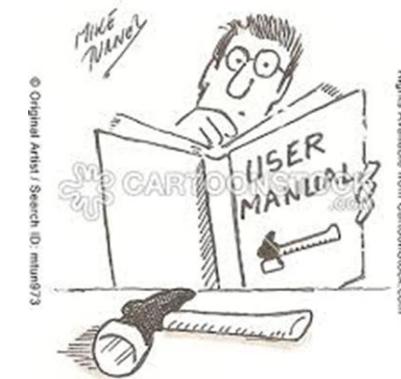
Covered in Lecture 10 last week

System Documentation

- Used to facilitate communication during development
 - *Generated as a by-product of development*
- Required for the day-to-day running, maintenance and enhancement of the system after development
 - *Includes, descriptions of system functions, architecture, and construction details, source code, analysis and design models*
 - *Instructions to run the system for IT Operations staff*
- Documentation has to be kept up-to-date to be useful
 - *Automated tools help*
 - *automatic synchronisation when design models used to generate software*
 - *reverse engineering used to update design models when software is updated*

User Documentation

- *Description of how to use the system for end users*
- *Examples include User Manual, On-line help, Quick Reference guides*
- *The User Manual typically includes (ITIL User Manual template) :*
 - Introduction
 - Overview: A description of the applications and its functions
 - Setup: How to install and configure the application
 - Getting started: Instructions about how to start using the applications
 - Advance procedures: A comprehensive navigation throughout the functions in the application and how to perform advanced tasks
 - Reporting: How to execute and customize the reports available
 - Troubleshooting: Recommendations for solving most common issues
 - References: A comprehensive list of error codes
 - Appendices: A glossary, a list of tables, a list of figures and an index



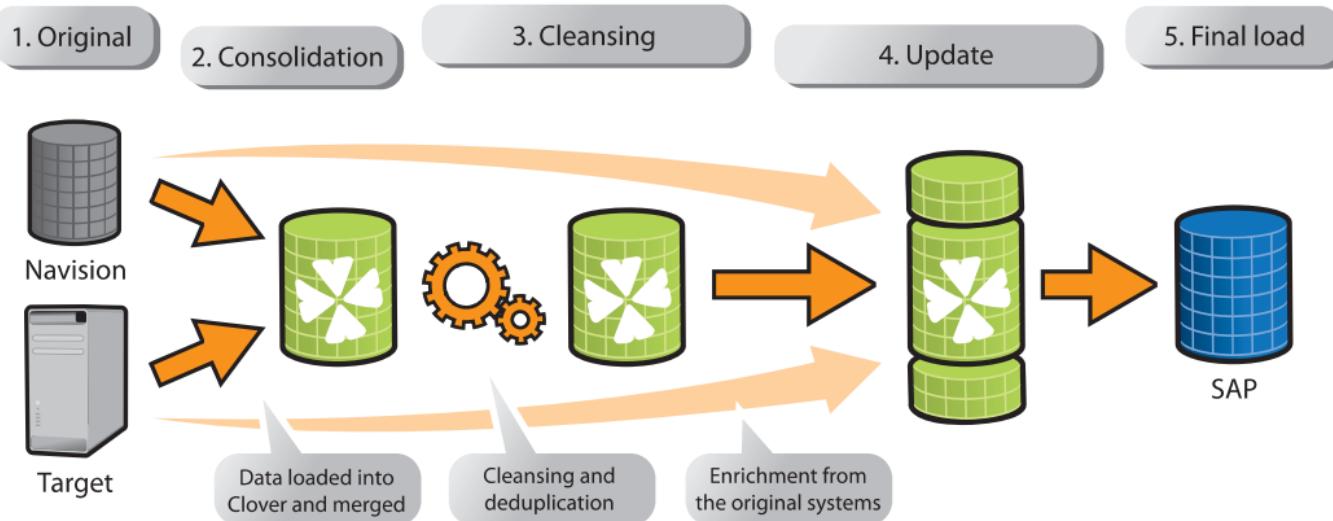
Rights Available from CartoonStock.com

Data conversion / migration



- Process of getting data ready for the new system
 - *A critical, challenging task, that can be very complex and costly*
 - *Can create new or reuse existing data*
 - *Data typically obtained from:*
 - Files or databases of system being replaced
 - Manual records
 - Files or databases of other systems being integrated with new system
 - Interaction with the client during normal system operation
 - *Need to consider:*
 - **Changes in data storage format and content**
 - Conversion process
 - Often needs specially written conversion programs
 - Manual file conversion is a time-consuming task
 - Confirmation of data accuracy

Data conversion / migration



- *Prepare existing files ... no errors, up-to-date*
- *Prepare manual files*
- *Add new data, Cleanse data, Remove duplicates*
- *Build new files and validate*
- *Begin maintenance of new and old files*
 - *can introduce time lag*
 - *files may be out of step*
- *Work towards established cut-off date*
- *Final check of accuracy*

Configure the production environment

- Ensure all facilities are set up:
 - *If infrastructure exists – Is it suitable?*
 - *May require new hardware and system software infrastructure*
 - *must be acquired, installed and configured*
 - *must address any version changes between purchase and delivery*
 - *Consider issues such as:*
 - *adequate space for all resources, ergonomic furniture, noise reduction, privacy, security, appropriate electrical connections, uninterrupted power, etc.*
- Infrastructure requires periodic updates:
 - *Software maintenance releases*
 - *Software version upgrades*
 - *Declining system performance*

Conduct Training

“If you think education is expensive
and time-consuming - try ignorance”

Bok

Conduct Training



- Considerations:
 - *Users*
 - *Number of users, Existing skill levels, On-going usage levels – regular, occasional*
 - *Level of detail to be imparted to the audience*
 - *Who should conduct the training*
 - *Where / When should the training be conducted*
 - *Methods and resources, specialised training documentation – designed to put novice users at ease*
 - *Need supportive User Manager who is committed to allocating time for training*

Conduct Training

- Training aids
 - *Must be easy to use*
 - *Reliable*
 - *Demonstrations and classes*
 - *Training documentation*
 - *especially designed to put the novice user at ease*
 - *On-line help*
 - *Expert users*
- On-going training needs after installation:
 - *Online help*
 - *Resident experts*
 - *Help desk*

IT Training example

From: FITGlobalEmails <fitglobalemails@monash.edu>
Subject: [FIT Announce] Dedicated Faculty BI training - 15th June 2011
Date: 18 May 2011 12:26:06 PM AEST
To: All Faculty of Information Technology Staff <allfit@infotech.monash.edu.au>

Faculty of Information Technology



This notice was approved for distribution on behalf of Michelle Ketchen, School Manager, Caulfield School of IT

Hi everyone,

The following training has now been confirmed for Wednesday 15th June at 10.30 am in H6.90. The session will run for 90 minutes. Please let me know if you would like to attend. If you have already advised me then don't worry.

Course Details:

The course details report provides timely access to course performance information. Course measures such as student course average mark, student course progress rate, student course retention rate, course enrolments and load (EFTSL) are available.

This is at a detailed course level and is viewable across attributes including student course details, student demographics (i.e. country, citizenship type, language, etc.), student academic history (i.e. secondary education, tertiary education, credit etc.), and a student's Monash pathway, that is, if they have completed a 'previous' course at Monash prior to enrolling in their course, if they enrolled into a 'next' Monash course, or if they transferred in or out of a course

Install / deploy the system

- Method of installation / deployment depends on several criteria:
 - **Cost:** *if there are cost constraints certain choices are not viable*
 - **System criticality:** *if system failure would be disastrous, the safest approach should be selected regardless of cost*
 - **Disruption to business:** *what level of disruption to the company and IS operations is acceptable*
 - **User computer experience:** *the more experience the users have, the less necessary it is to delay changeover*
 - **System complexity:** *the more complex the system, the greater the chance of flaws ... a safer approach is better*
 - **User resistance:** *need to consider what the users are best able to cope with*

Installation / Deployment alternatives

Decision 1: *Multiple locations?*

1. Pilot/Single Location installation
2. All Locations

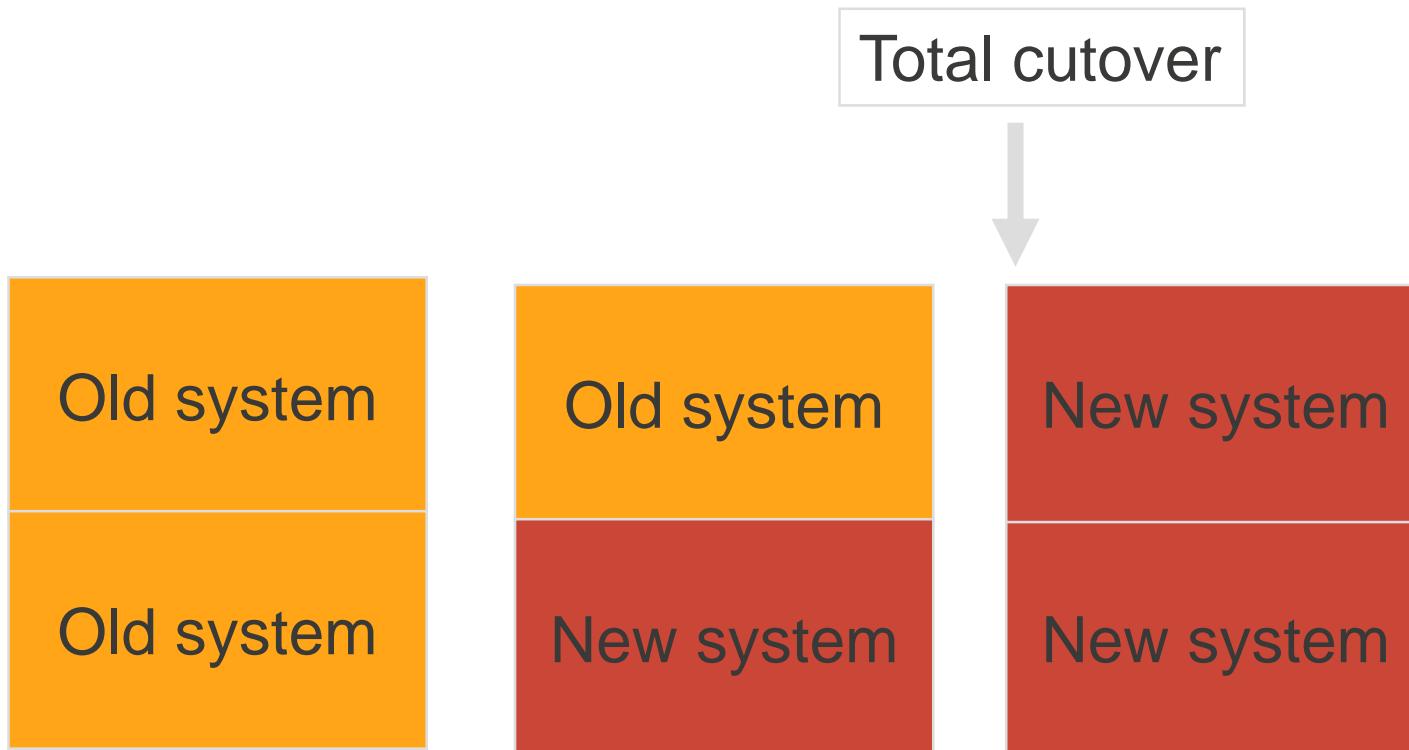
Decision 2: *Multiple functions?*

1. Phased/Staged functionality installation
2. All functionality

Decision 3: *Installation alternatives*

1. Direct/Abrupt installation
2. Parallel installation
3. Phased installation or Staged installation

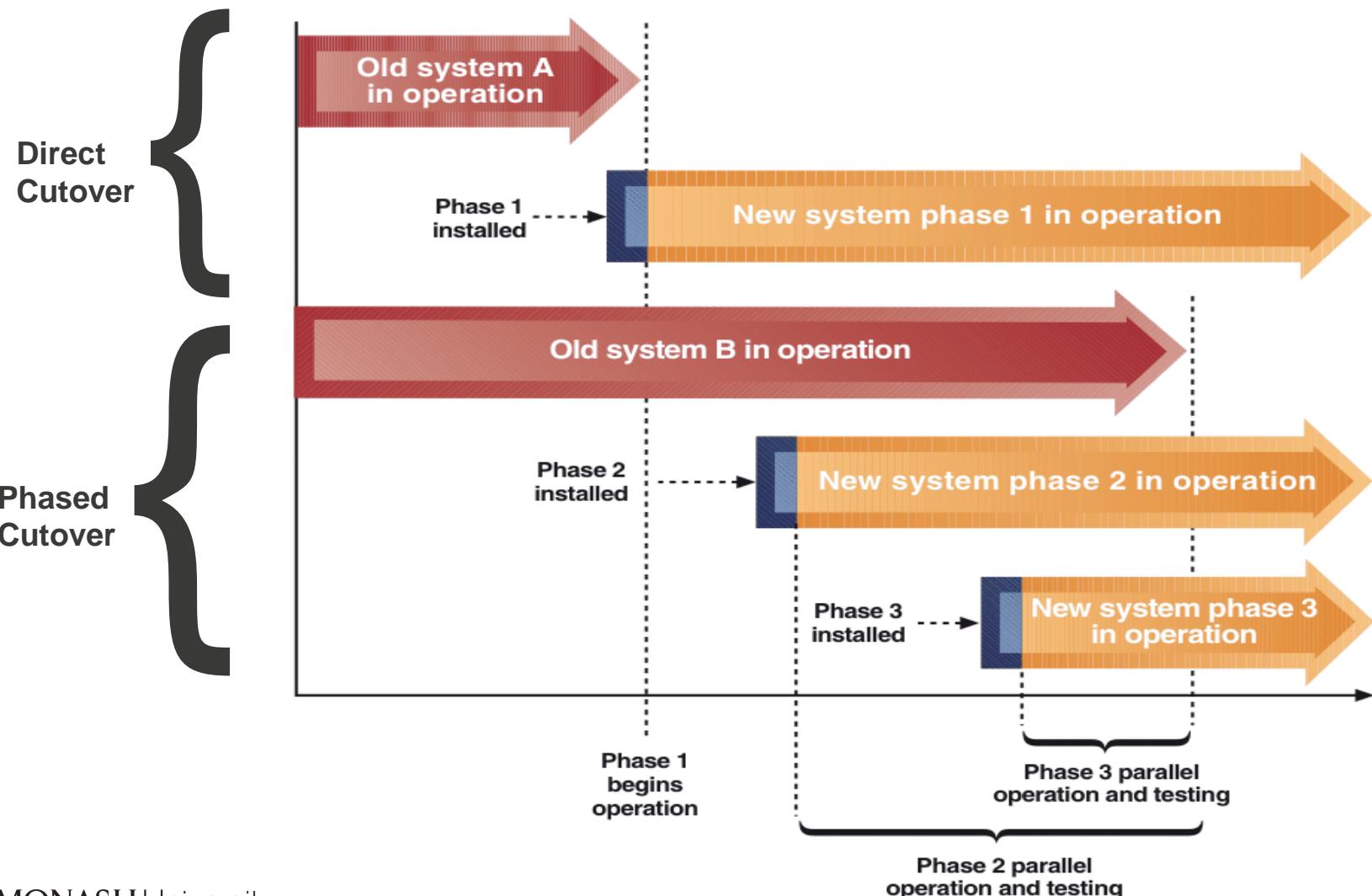
Multiple locations - Pilot Installation



Pilot installation

- Old and new systems operated concurrently
- Only part of the organisation tries out the new system
- The pilot system must prove itself at the test site
- Advantages
 - *Risks relatively low if problems occur*
 - *Errors are localised to pilot site*
 - *Can be used to train users before implementation at their own site*
- Disadvantages
 - *Lack of consistency between different parts of the organisation*

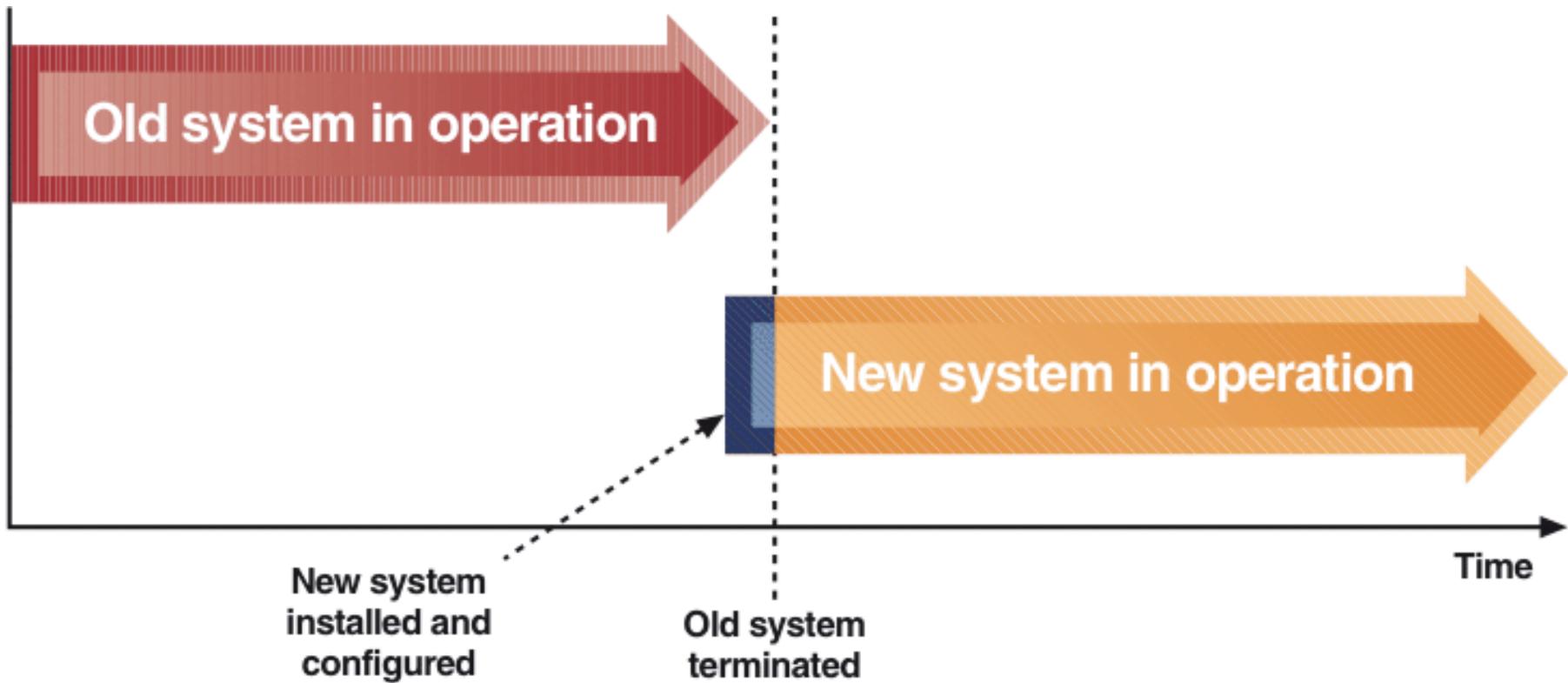
Phased installation



Phased installation

- New system installed in series of steps or phase, where each phase adds components to the existing system
- Advantages
 - Reduces risk because *phase failure is less serious than system failure*
 - Lower costs for earlier results
 - Benefits can be realised earlier
 - Rate of change minimised for users
- Disadvantages
 - Multiple phases cause more activities, milestones, and management complexity for entire effort
 - Close control of systems development is essential
 - Costs associated with the development of temporary interfaces to old systems
 - Limited business applicability

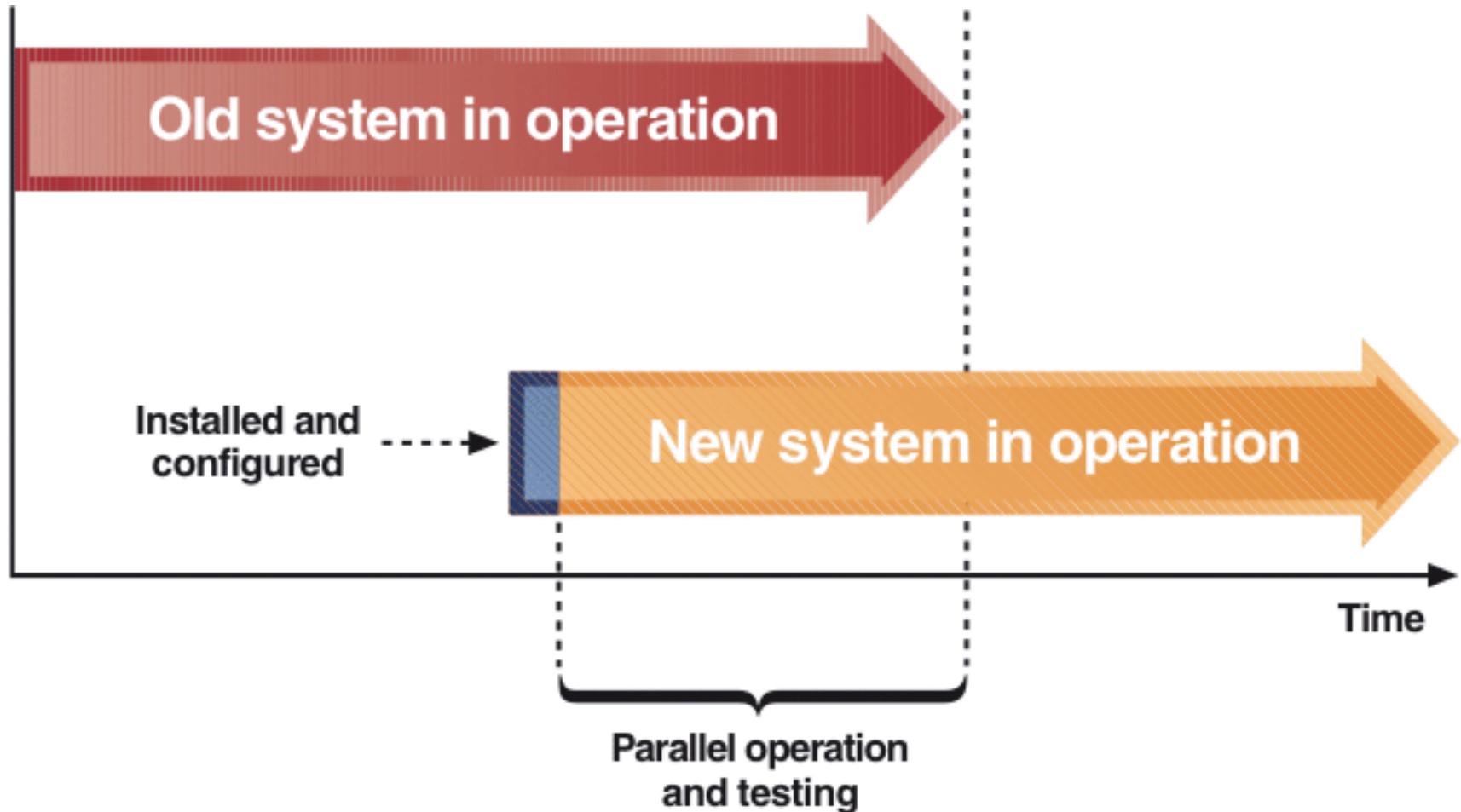
Direct installation / Abrupt cutover



Direct installation / Abrupt cutover

- This approach is **meaningful** when:
 - *the system is not replacing any other system*
 - *the old system is judged absolutely without value*
 - *the old system is either very small and/or very simple*
 - *the new system is completely different from the old and comparisons would be meaningless*
- Advantages
 - *Simple, fewer logistic issues to manage, costs minimised*
- Disadvantages
 - *High risk .. no backup*

Parallel installation



Parallel installation

- Old and new system both operate for an extended period of time. Cut over at the end of a business cycle, after balancing between both systems
- Advantages
 - *Risk low if problems occurs – continual backup*
- Disadvantages
 - *High cost: increased personnel, extra space, increased managerial and logistic complexity*

Example:

Monash are going to update WES in February. Major changes. The University are experiencing financial issues due to reduced Government funding. Staff have been heavily involved with Usability testing and are looking forward to the rollout of the updated system. The system has to be rolled out across all local and international campuses. **What installation/deployment method would you recommend?**



Consider: Cost, System criticality, Disruption to business, User computer experience, System complexity, User resistance

Monitor Operations

- Monitor user satisfaction
 - with functional requirements
 - with system performance
- Run benchmark tests
- Tune system

Wrap up / Transition to Support

Support constitutes:

- Maintenance
 1. *Corrective maintenance*
 2. *Adaptive maintenance*
 3. *Perfective maintenance*
 4. *Preventative maintenance*
- Change management
- Post-implementation review
 - *after the system has been accepted*
- In 2010, maintenance up to 55% of the IT budget
 - Forrester Research – 517 organisations*

Corrective Maintenance

- Corrects analysis, design and implementation errors
 - *most corrective problems arise soon after installation or after major system changes*
 - *should have been isolated and corrected during development*
 - *professional practice during development* should minimise the need (but will not remove it completely)
 - *adds little or no value - focus on removing defects rather than adding anything new*
 - *accounts for up to 75% of all maintenance activity*

Adaptive Maintenance

- To satisfy changes in the environment, changing business needs or new user requirements
 - *changes in tax laws, takeovers and mergers, new OS, etc*
 - *new type of report, new class of customer etc.*
- Less urgent - changes occur over time
- Adaptive maintenance is inevitable, does add value
- Maintenance staff need strong analysis and design skills as well as programming skills
 - *changes often require a complete SDLC*
 - *also need good understanding of the system*

Perfective Maintenance

- To enhance performance, maintainability, usability
 - *adds desired features rather than required*
 - *better run times, faster transaction processing*
- To meet user requirements not previously recognised or given high priority
 - *missed in development or not known about*
 - *considered unimportant*
- Legacy systems (old systems running for at least 10 years) are likely candidates for perfective maintenance

Preventative Maintenance

- Pay now or pay more later
 - defects or potential problems found and corrected before they cause any damage
 - reduce chance of future system failure
 - eg expand number of records beyond needs, standardise formats across platforms
- A natural by-product of maintenance work - identify and fix any potential problems noted while fixing other errors

Change Management Systems

- Overall goal is to manage change effectively
- Organisations implement change management systems in an attempt to reduce the confusion and complexity of developing and maintaining systems

Change Management Systems

- The aims of change management systems are
 - restrict access to production source and object code
 - reduce errors being introduced into production
 - single version of source and object code in production
 - improve quality and reliability of software
 - increase security and control
 - increase software productivity

A change request example

Change Request				
Request Date	2/1/2007	Change Type	<input type="checkbox"/> Error Correction	
Requested By	Wen-Hsu Chang, Comptroller		<input checked="" type="checkbox"/> Modification	
Target System	Customer Accounts - Refunds		<input type="checkbox"/> New Function	
Change (or Error) Description				
U.S. check formats will soon change due to a recently enacted federal law. The new format reserves an area to the right of the current routing number to be used for a security bar code checksum.				
The law requires the new checksum to be printed on checks dated on or after January 1, 2008.				
We currently use a portion of the area in question to print a multicolored security symbol. The security symbol will need to be moved or eliminated, and the security bar code checksum will have to be added.				

A change review form

Change Review			
Change Request ID	2007-11	Date Reviewed	2/7/2007
Priority	<input type="checkbox"/> Critical	<input checked="" type="checkbox"/> Necessary	<input type="checkbox"/> Optional
Hardware Implications need to verify ability of current printers to write a security bar code in mandated area			
Software Implications database will need to be modified to store the security bar code with other check information check writing program must be modified to generate and print the security bar code			
Performance Implications none			
Operating Budget Implications none			
Other Implications none			
Disposition	<input checked="" type="checkbox"/> Approved		<input type="checkbox"/> Rejected
Reason			
Latest Implementation Date	12/31/2007		
Reevaluation Date	n/a	Signature	

System closure

Successful systems closure involves:

- *Gaining stakeholder and customer acceptance of the system - ensure that outcomes are consistent with those defined in the Acceptance criteria (including negotiated variances)*
- *Ensuring all legally binding agreements are met, payments and collections made*
- *Acknowledgement of the success of the team, and dealing with an emotional / career issues team members may face, finalise performance management reports*
- *Internal review of the system to capture lessons learnt*
- *Final Report / Presentation to key stakeholders to formally announce the end of the system development*

Post-Implementation Review

- A PIR analyses what went right and wrong with a project. It is conducted 2 to 6 months after conversion by a team which includes user reps, development staff, internal auditors and sometimes external consultants - development team is not in charge!
 - *look at original requirements and objectives*
 - *evaluate how well they were met*
 - *compare costs of development and operation against original estimates (maintenance costs ??)*
 - *compare original and actual benefits*
 - *new system reviewed to see whether more or original or additional benefits can be realised*

Must not become a witch-hunt



What went wrong ???
Learn for the future !!!

Workshop Preparation

The Workshop for Week 12 will include:

Assignment 3 Presentations

Exam Revision

Moodle Quiz for Week 10 & 11 Seminars

Thanks for watching

Resources:

Satzinger, J. W., Jackson, R.B., Burd, S.D. and R. Johnson (2016)
Systems Analysis and Design in a Changing World, 7th Edition,
Thomson Course Technology, Chapter 14, pp. 454-473

Deloitte (2013): The Outsourcing Handbook – A guide to
outsourcing