Work Related Project (CO599)

Week 7:

—— Quality

– Evaluation

—— Further Modelling

Contents

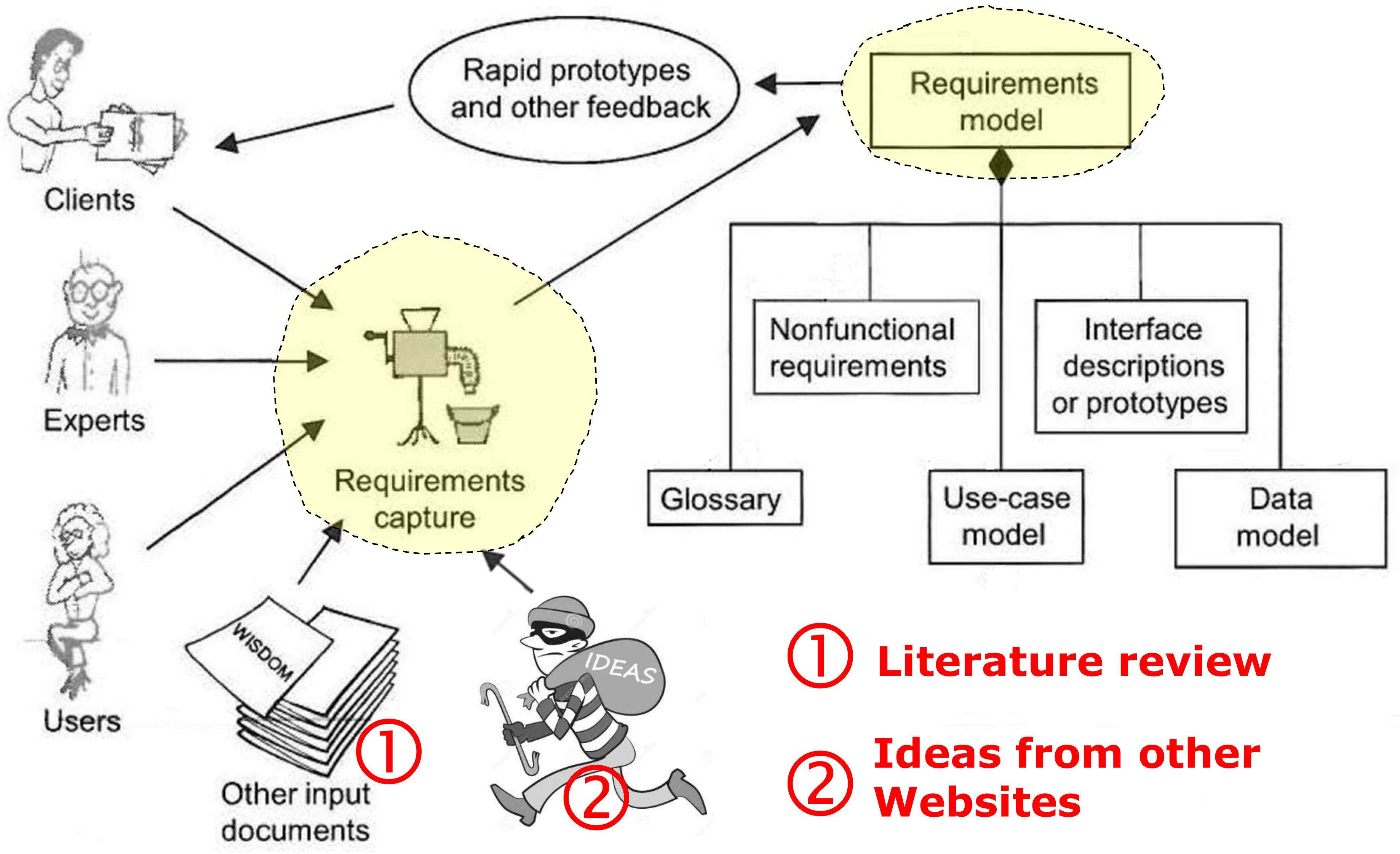
- Recap about the Whaler Project case study
- Reflect upon functional and non-functional requirements, as points of reference for quality
- Usability and usability testing
- Further modelling – activity diagrams

Recap and Context

The Whaler Project

- Analysis of the information available
- Specification of requirements
- Reviewed other examples of relevant Websites, and good design principles, to come up with your own design
- Modelling interactions of different types of users with the “system” (use cases and use case diagrams)

<u>Requirements</u> Based on the Analysis of Information from Different Sources



① **Literature review**

② **Ideas from other Websites**

The Same Ideas and Processes can be Applied to your own Project

- **Development of:**
  - CV creation/management Website
  - Web-based asset management system
  - Social networking “skill swap” Web service
  - Web-based content management system for setting up Websites
  - Mobile App for creating, storing and reading recipes
  - Web service to upload geotagged videos
  - Web-based system for programming tutorials for kids
  - Library system for managing book stock
  - Technology-enhanced learning system for teaching computing

The Quality and Appropriateness of the Design/Build

Interfaces

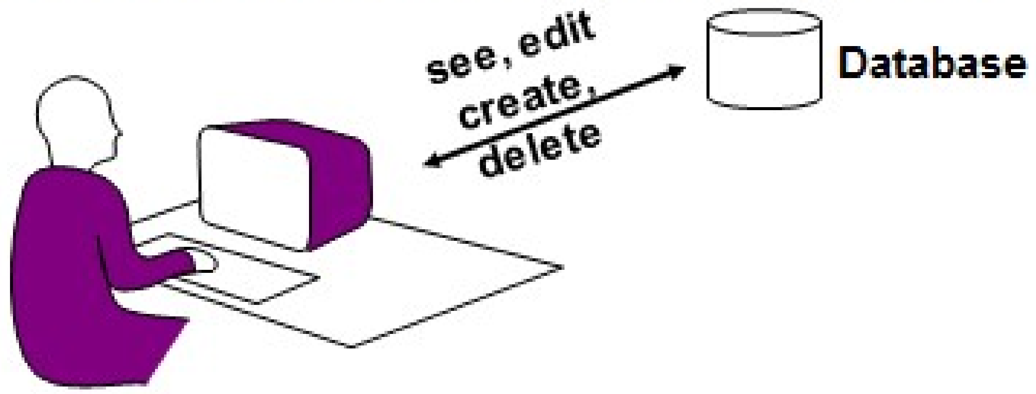
- **USER INTERFACE**
  - The part of the system that the user sees, hears and feels
  - Other parts of the system are hidden to the user (such as the database for storing information) but the user has some imagination or understanding of what is going on
- **TECHNICAL INTERFACE**
  - The system may have interfaces to other systems

Design of User Interfaces

- **EASY IN PRINCIPLE**
  - Just make it possible for the user to see the data/information, to input/change data, to send commands (as appropriate to their access rights)
- **THE CHALLENGE FOR DEVELOPERS**
  - Must deliver adequate functionality as well as adequate quality

Developers Might Focus on Functionality & Not Quality Factors

Easy to make a user interface:
Just give access to the database

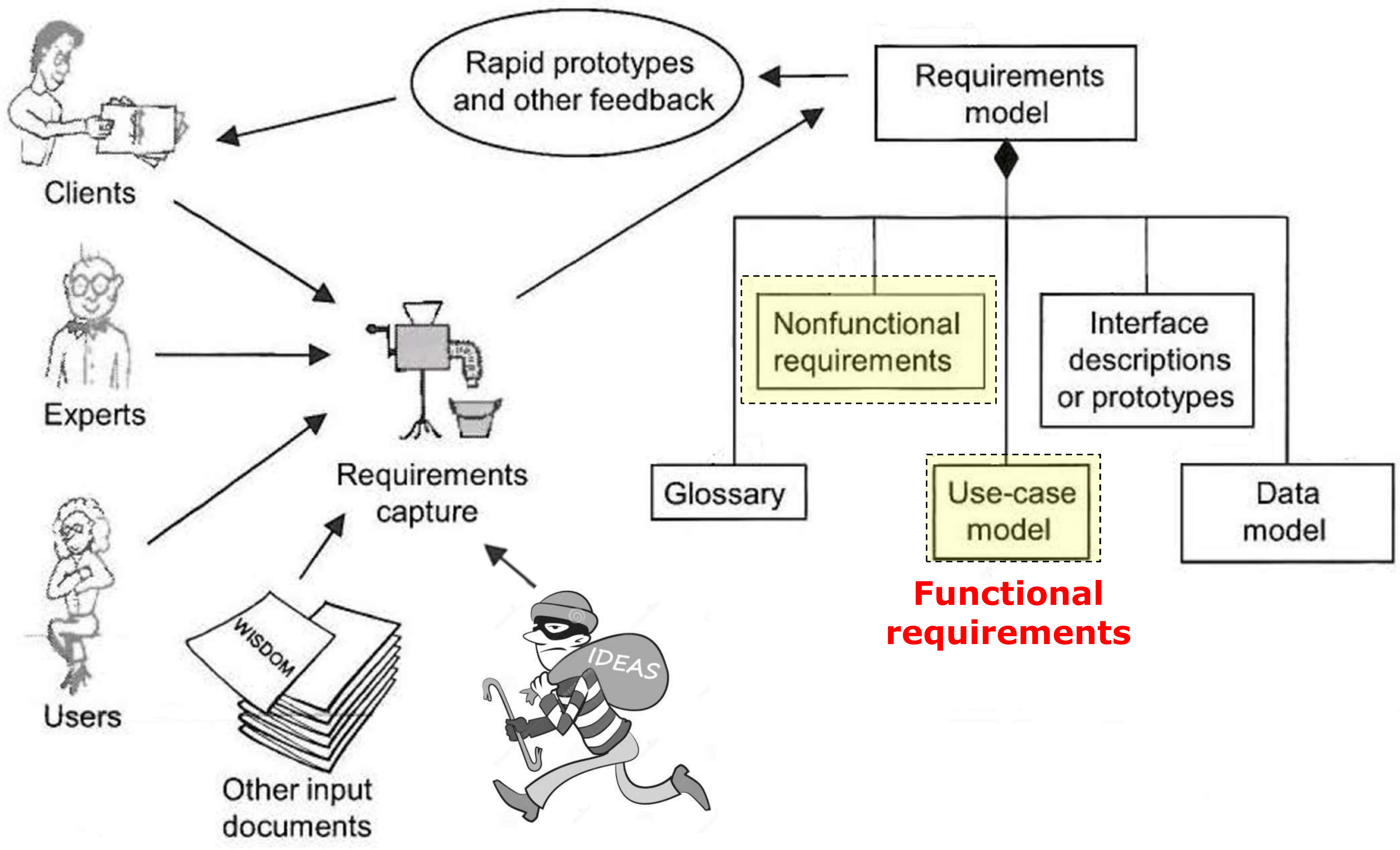


Hard to make
a good
user interface

Functionality: ✓
Necessary features

Quality factors: ?
Non functional requirements

Design Must Address Functional and Non-Functional Requirements



Consider a Company Specific Quality Model for Software and Systems

Hewlett-Packard: F.U.R.P.S.

Result of a statistical project survey at Hewlett Packard 1987 to improve its products based on

FIVE FACTORS…

- **Functionality:** functions it performs, their generality and security
- **Usability:** aesthetics, consistency, documentation
- **Reliability:** frequency and severity of failure, accuracy of output
- **Performance:** response time, resource consumption
- **Supportability:** can it be extended, adapted, corrected?
- FURPS is originally a company specific quality model

F
U
N
C
T
I
O
N
A
L

Hewlett-Packard: F.U.R.P.S.

Result of a statistical project survey at Hewlett Packard 1987 to improve its products based on

FIVE FACTORS…

- **Functionality:** functions it performs, their generality and security
- **Usability:** aesthetics, consistency, documentation
- **Reliability:** frequency and severity of failure, accuracy of output
- **Performance:** response time, resource consumption
- **Supportability:** can it be extended, adapted, corrected?
- FURPS is originally a company specific quality model

N
O
N
|
F
U
N
C
T
I
O
N
A
L

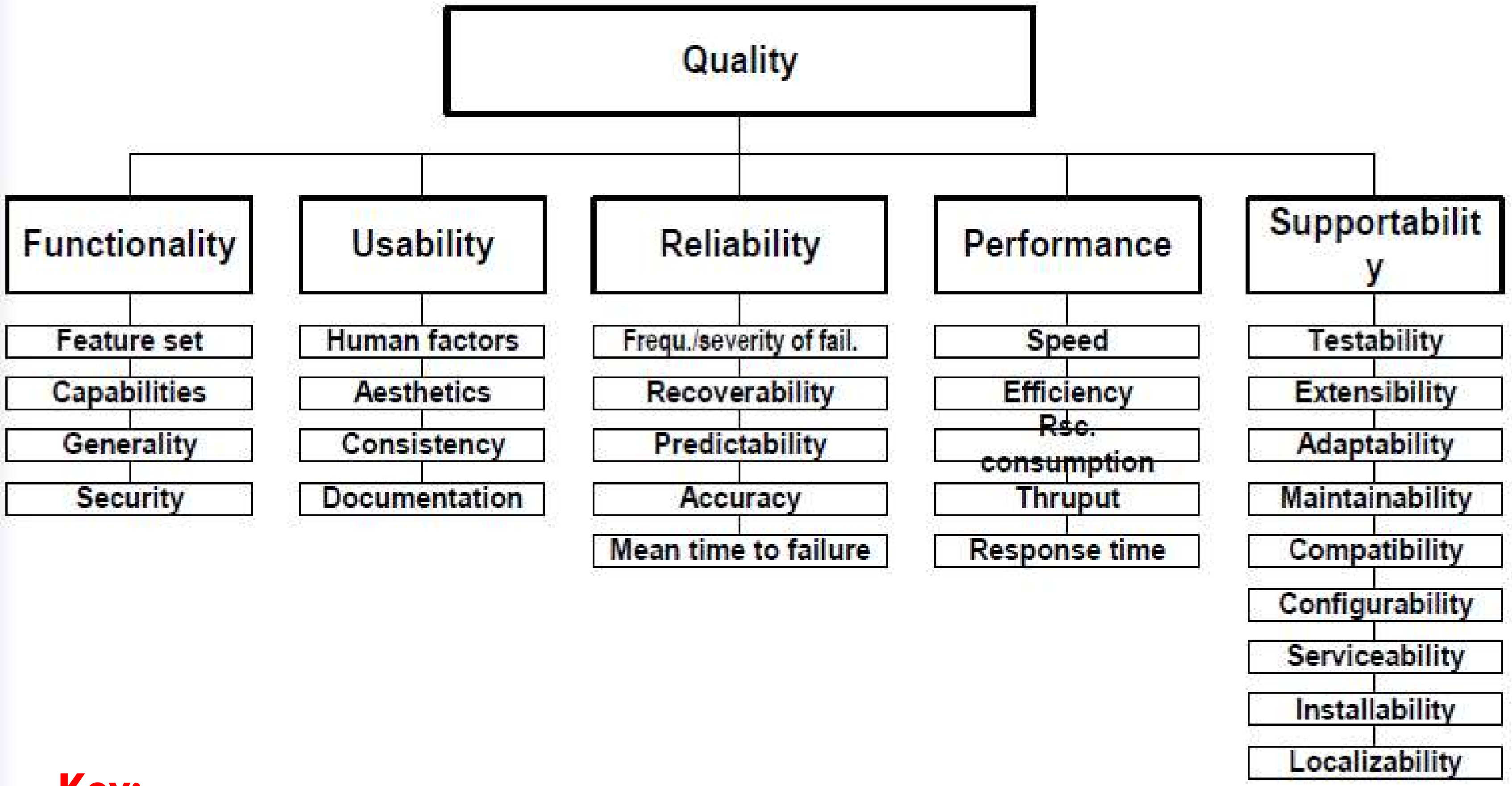
Hewlett-Packard: F.U.R.P.S.

Result of a statistical project survey at Hewlett Packard 1987 to improve its products based on

FIVE FACTORS…

- **Functionality:** functions it performs, their generality and security
- **Usability:** aesthetics, consistency, documentation
- **Reliability:** frequency and severity of failure, accuracy of output
- **Performance:** response time, resource consumption
- **Supportability:** can it be extended, adapted, corrected?
- FURPS is originally a company specific quality model

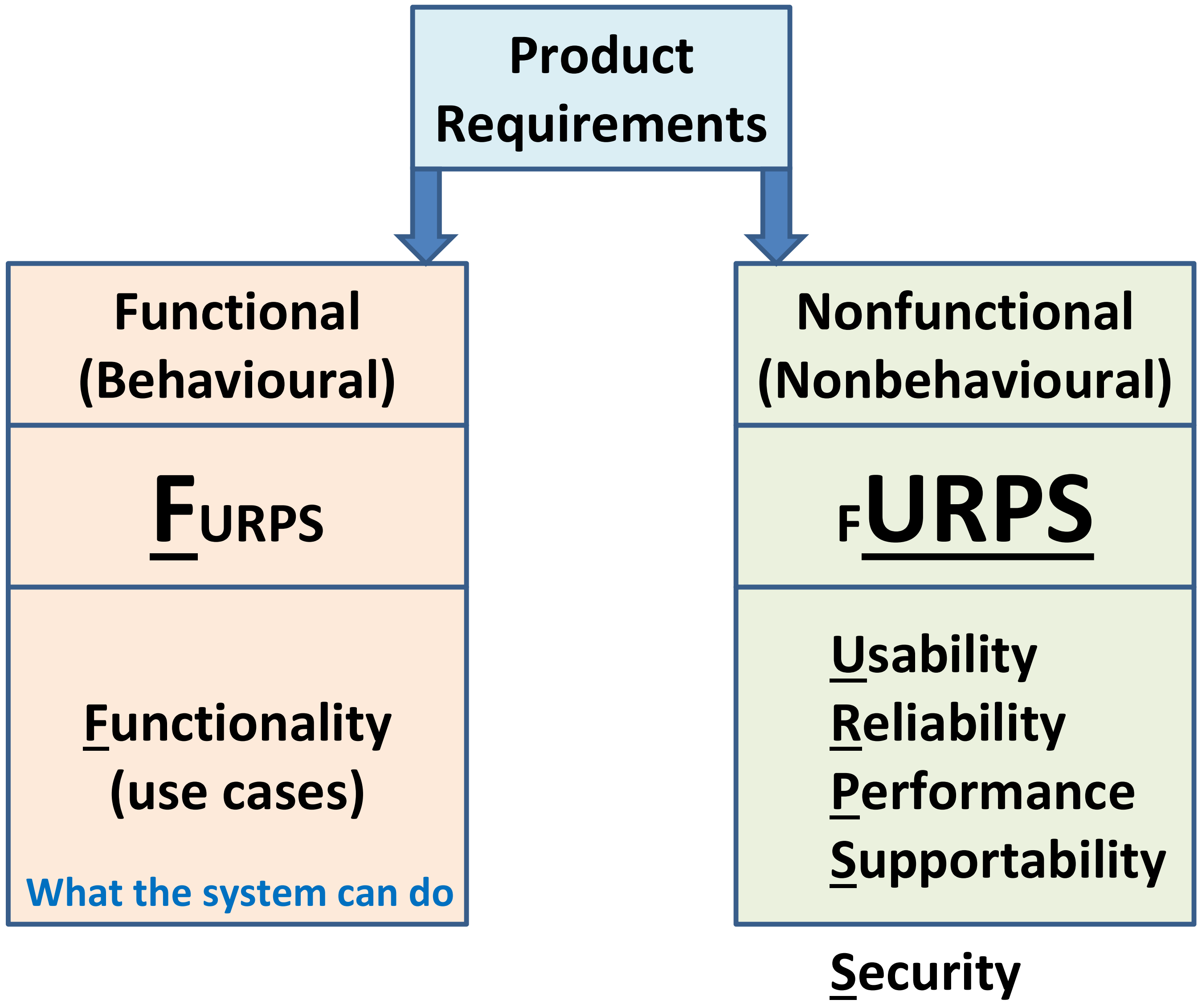
F.U.R.P.S. Sub Categories



Key:

Rsc. Consumption = Resource consumption/usage
Thruput = Throughput

Now Widely Used: FURPS+



Important Reference Points

- **The specified functional and non-functional requirements (FURPS) provide a reference point, against which we can:**
  - Design and build the “system”
  - Test and evaluate the “system”
  - Measure the quality and fitness-for-purpose of the “system”

Focus on Usability

Definition(1) of Usability [Lauesen, 2005]

- **Usability consists of SIX factors**
  - **Fit for use (functionality):** The system can support the user's tasks.
  - **Ease of learning:** How easy is the system to learn for various groups of users?
  - **Task efficiency:** How efficient is it for the frequent user?
  - **Ease of remembering:** How easy is it to remember for the occasional user?
  - **Subjective satisfaction:** How satisfied is the user with the system?
  - **Understandability:** How easy is it to understand what the system does?

Definition(2) of Usability

- **Usability is the measure of the quality of a user’s experience when interacting with a website. It’s about making a website that is easy, efficient, and pleasant for your visitors.** [Felke-Morris, 2015]
- **This can be broken down into several factors that affect the user’s experience: see overleaf……..**

Usability Factors [U.S. Dept of HSS, 2016]

Usability is not a single, one-dimensional property of a product, system, or user interface; it is a combination of factors including:

- **Intuitive design**: a nearly effortless understanding of the architecture and navigation of the site
- **Ease of learning**: how fast a user who has never seen the user interface before can accomplish basic tasks
- **Efficiency of use**: How fast an experienced user can accomplish tasks
- **Memorability**: after visiting the site, if a user can remember enough to use it effectively in future visits
- **Error frequency and severity**: how often users make errors while using the system, how serious the errors are, and how users recover from the errors
- **Subjective satisfaction**: If the user likes using the system

Basics of Usability Testing

- **Usability tests are an effective way to discover usability problems**
  - **Think-aloud test:** allow a user to try out realistic tasks on a real system or a mock up of it, asking them to explain and describe their actions
  - **Real system:** the user might use a real (or largely finished) system to carry out tasks
  - **Prototypes and mockups:** user examines different representations of the product at different project stages
  - **Test team:** could involve facilitator, note taker and observer (as well as the tester)

1. Plan the Test

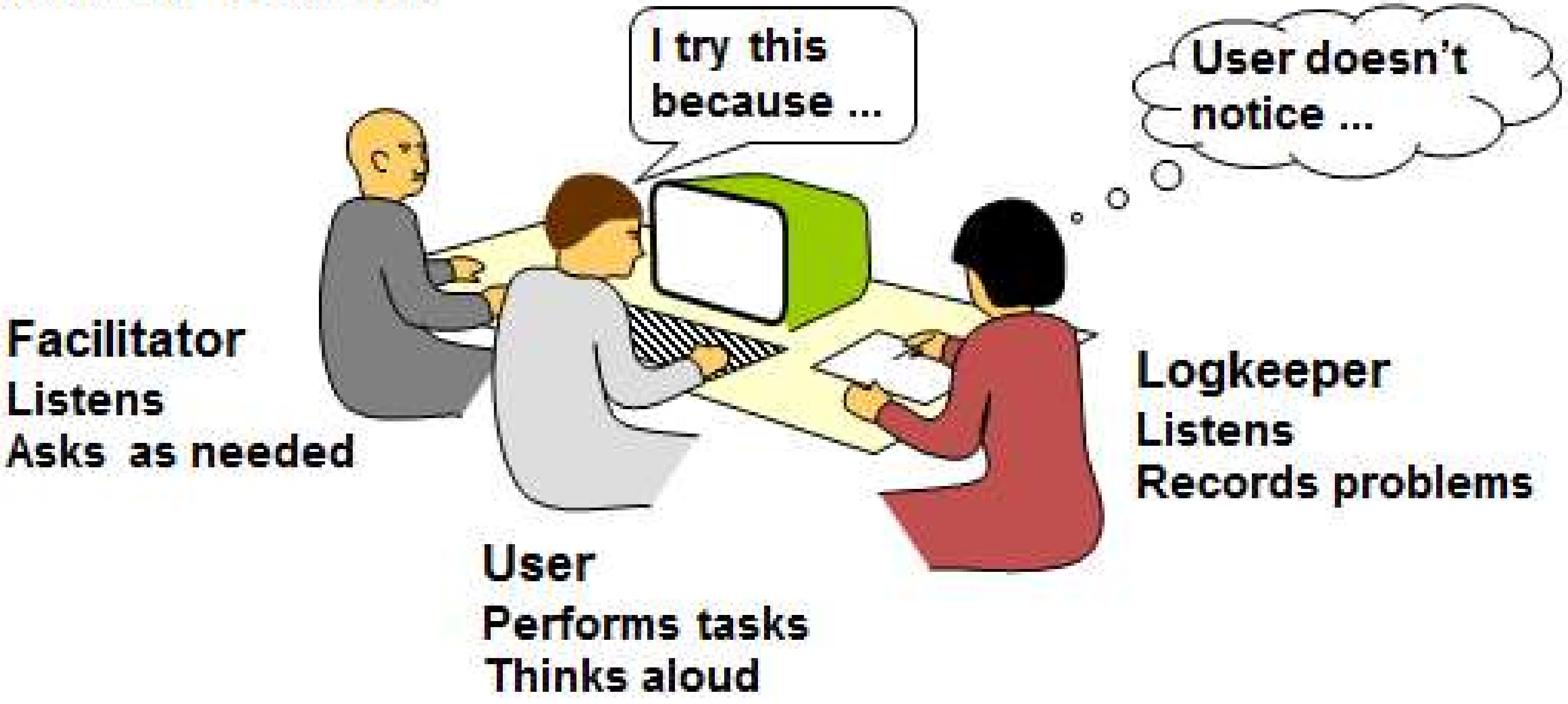
- **Test users**
  - Choose people that represent typical users…
  - …with appropriate level of “IT” and “context” knowledge
- **Test Tasks**
  - Invite the test user(s) to do realistic tasks based around a stated scenario, without giving them too many hints, for example, make a boat reservation
- **Study the system**
  - You should learn and understand the system yourself (as test facilitator) in order that you can understand the user attempts to do and understand things

2. Carry out the Test

- **The arriving test user might be a bit nervous and scared of looking stupid**
  - Clearly explain the purpose of the test
  - Provide reassurance and try to relax them
  - Give the user the first test task (‘think aloud’)
  - Observe, listen, take notes (e.g. problems)
  - Ask questions about what they are doing (how and why)
  - If the user gets stuck, you may provide hints

2. Carry out the Test

Purpose:
Find usability problems



3. Reporting the Test

- **After the test**
  - You have to write a list of the problems the user encountered in a clear, understandable manner for, within, say, 12 hours of the actual test (before you forget!)
  - These notes and observations will underpin the next enhancements to the product/prototype

Foundations of UX: Usability Testing

http://www.lynda.com/User-Experience-tutorials/What-usability-testing/421803/445264-4.html?

- What is usability testing?
- What you can usability test?
- Planning your first test

Heuristic Evaluation

- **Involves letting a usability specialist examine and comment upon the interface**
  - A usability expert or developer examines the screen
  - They provide their professional opinion (on good and bad aspects), possibly using heuristic guidelines
  - Several evaluators may deliver individual lists or a combined list of defects and issues
  - Can supplement (but not replace) a usability test.
- **Heuristic evaluation can find lots of problems but some may be false (in the sense that they do not cause problems to real users)**
- **<u>First Law of Usability:</u> Heuristic evaluation may only have a 50% hit rate**

Heuristic Guidelines: The Eight Golden Rules of Interface Design (1)

1. Strive for consistency

– Frequently violated but can be tricky to follow because many forms of consistency

2. Enable frequent users to use shortcuts

– Want to reduce the number of interactions and increase the pace of interaction (short response times, fast display rates)….abbreviations, special keys, hidden commands

Heuristic Guidelines: The Eight Golden Rules of Interface Design (2)

3. Offer informative feedback

- System feedback for every user action (modest for frequent & minor actions; substantial for infrequent & major actions)

4. Design dialogs to yield closure

- Sequences of actions organised into groups with a beginning, middle and end; informative feedback after completion of actions gives operators a sense of accomplishment and relief

Heuristic Guidelines: The Eight Golden Rules of Interface Design (3)

5. Offer error prevention and simple error handling

– Design the system to eliminate or minimise errors; advice given where errors are made

6. Permit easy reversal of actions

– As much as possible, actions should be reversible; knowledge that errors can be undone encourages exploration

Heuristic Guidelines: The Eight Golden Rules of Interface Design (4)

7. Support internal locus of control

– Experienced operators strongly desire to be in charge of the system

8. Reduce short-term memory load

– In relation to the limitations of human memory

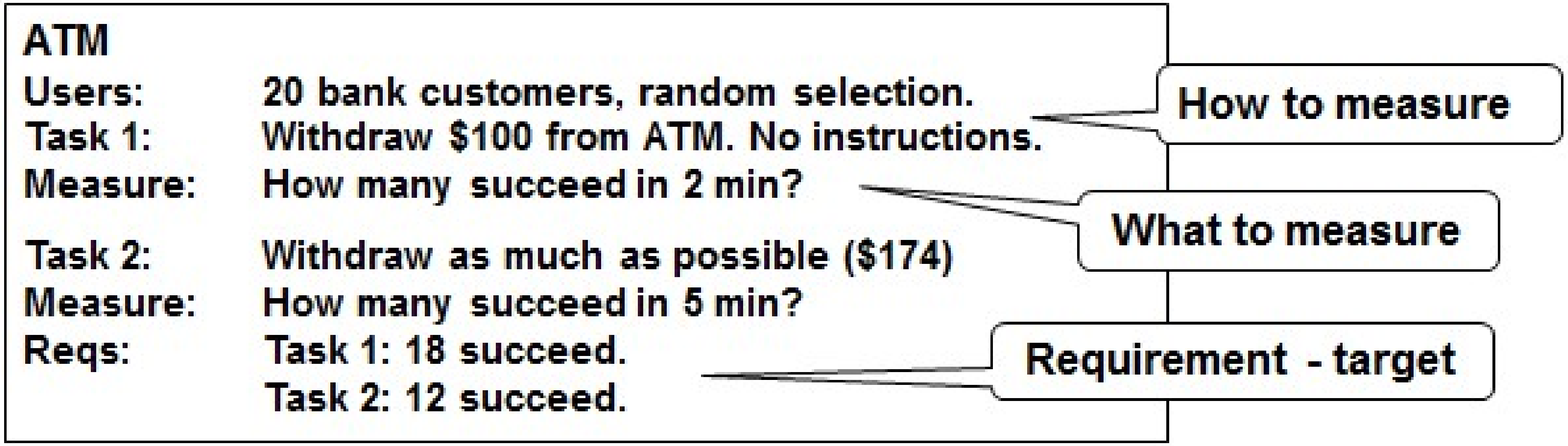
– Requires the display/interface to be simple, sufficient training time, etc.

User Review

- **Normally made with expert users; the facilitator goes through the process with them**
  - Show how the different screens are to be used, discuss how various tasks are carried out.
  - Such reviews are important for finding missing functionality
  - The expert user can point to things that the system cannot do but may not be able to comment upon ease-of –use problems

Usability Measurements and Requirements

- **We must be able to measure the quality factors**
  - The time taken to carry out various tasks **(performance)**
  - Requirement = target



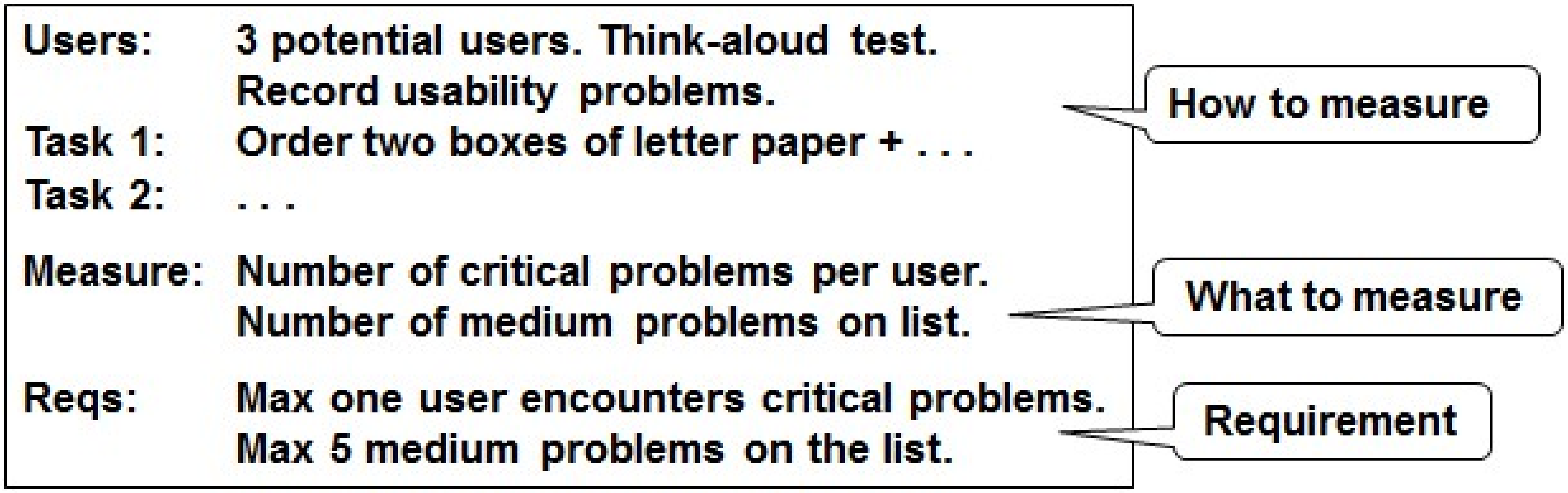
ATM

Users:	20 bank customers, random selection.
Task 1:	Withdraw $100 from ATM. No instructions.
Measure:	How many succeed in 2 min?
Task 2:	Withdraw as much as possible ($174)
Measure:	How many succeed in 5 min?
Reqs:	Task 1: 18 succeed.
	Task 2: 12 succeed.

How to measure
What to measure
Requirement - target

Usability Measurements

- **Problem counts**
  - Instead of measuring task times, we could list and number the problems raised during think-aloud tests

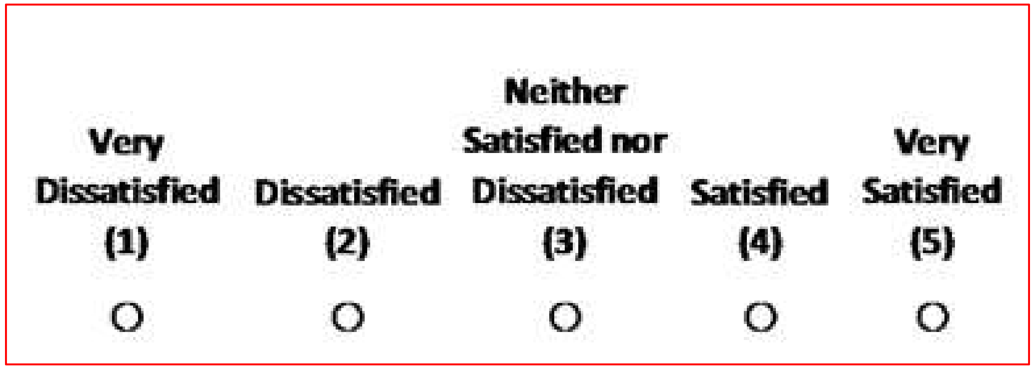


Users:	3 potential users. Think-aloud test. Record usability problems.
Task 1:	Order two boxes of letter paper + . . .
Task 2:	. . .
Measure:	Number of critical problems per user. Number of medium problems on list.
Reqs:	Max one user encounters critical problems. Max 5 medium problems on the list.

How to measure
What to measure
Requirement

Other Measurements

- Keystroke counts, mouse clicks + task time
- Opinion polls



Very Dissatisfied (1)	Dissatisfied (2)	Neither Satisfied nor Dissatisfied (3)	Satisfied (4)	Very Satisfied (5)
○	○	○	○	○

- Score for understanding (understandability test)
- Measure how well the system adheres to guidelines

Many Examples on t'Internet etc.

- Web Usability article
- Website evaluation checklist
- 5 Ways to Evaluate the Quality of Your Website Design

[ADDITIONAL MATERIAL]

Mashable UK – 22 Essential Tools for Testing Your Website's Usability

- http://mashable.com/2011/09/30/website-usability-tools/#nWyUlheE5mqE

Mashable UK

1. User Task Analysis

- Test whether users are able to accomplish their tasks and goals, in the best and most efficient way possible
- Evaluate task performance against:
  - Learnability, intuitiveness, efficiency, preciseness, fault tolerance, memorability, affordance

2. Readability

- Hinges on the following considerations:
  - Ease of comprehension, legibility, reading enjoyment

Mashable UK

3. Site Navigability

- The user must be able to move through multiple Web pages as easily as possible
  - **Information architecture:** How well are the Web pages categorised and organised?
  - **Findability:** Are there sufficient and helpful site features, such as search boxes, links and navigation features?
  - **Efficiency of navigation:** How fast and how many actions (mouse clicks, how much text) does it take to get to the page of interest?

Mashable UK

4. Accessibility

- A Website should be accessible to everyone, including special categories of users
  - Users with disabilities, different browsers, different devices

- Cross-Browser/Cross-Platform Compatibility
- Semantic HTML Markup
- Colour Choices
- Use of HTML Accessibility Features

Mashable UK

5. Website Speed

- Web users care deeply about how fast they get access to the needed information
  - Web page response time
  - Web page size (file size)
  - Code quality

6. User Experience

- How pleasant a Website is to use
  - Fulfillment, usefulness, enjoyment, positive emotions

F – U RPS

FURPS: Reliability

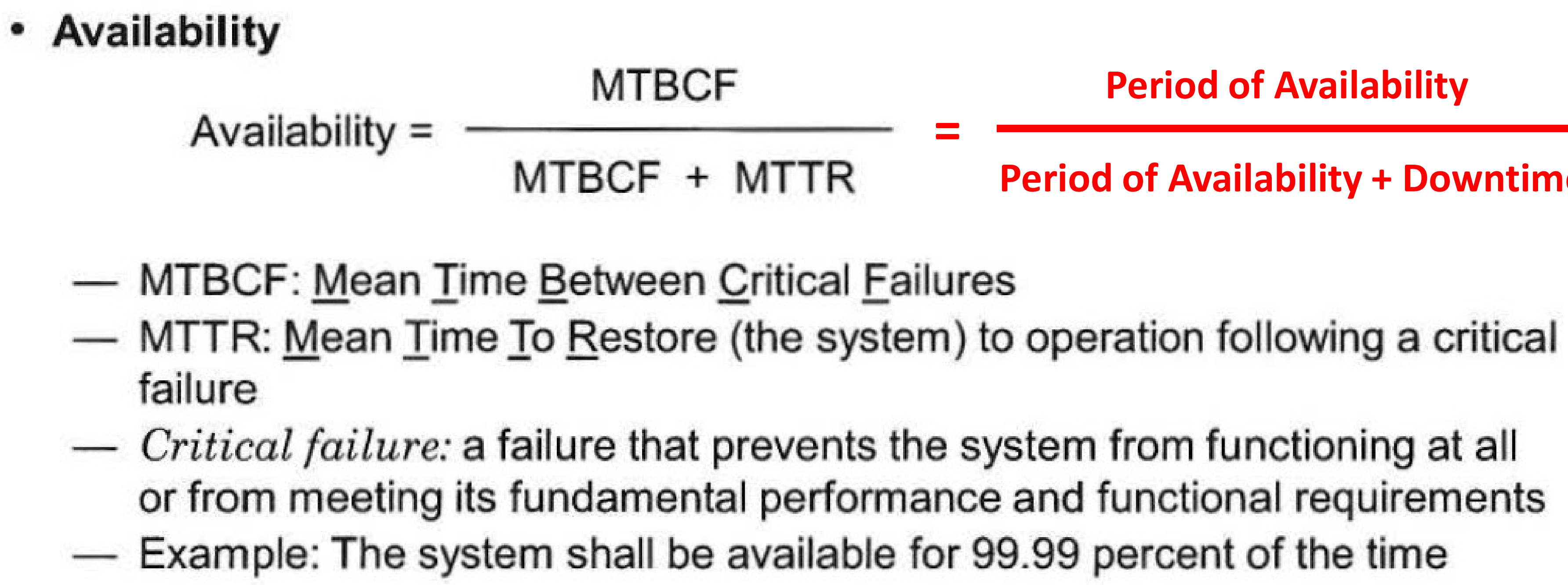
Reliability is the probability of the system executing without failure

Measures of reliability include

- Availability
- Predictability
- Accuracy and precision
- Type and severity of failures
- Failure recovery

Dependability of system – hardware and software elements

Reliability Measure: Availability



- **Availability**

$$\text{Availability} = \frac{\text{MTBCF}}{\text{MTBCF} + \text{MTTR}} = \frac{\textbf{Period of Availability}}{\textbf{Period of Availability + Downtime}}$$

  - MTBCF: Mean Time Between Critical Failures
  - MTTR: Mean Time To Restore (the system) to operation following a critical failure
  - *Critical failure:* a failure that prevents the system from functioning at all or from meeting its fundamental performance and functional requirements
  - Example: The system shall be available for 99.99 percent of the time

Reliability Measure: Accuracy & Precision

Accuracy of the inputs and outputs

- Data integrity
- Referential integrity
- Example: When entering an address, the ZIP/ postal code shall be correct for the given address attributes

Precision

- Example: All interest calculations on accounts shall be rounded to the nearest cent

FURPS: Performance

Throughput

- Number of transactions handled per unit of time
- Example: average number of ATM transactions that can be completed concurrently in one minute

Response time

- Time a particular transaction takes from start to getting the system's response
- Response time can be critical in real-time systems

Efficiency

- Example: System shall be able to achieve its other performance requirements over a 56-KB dial-up connection

FURPS: Supportability

Supportability is

- Testability
- Extensibility
- Adaptability
- Maintainability
- Compatibility
- Configurability
- Serviceability
- Stability
- Robustness
- Flexibility

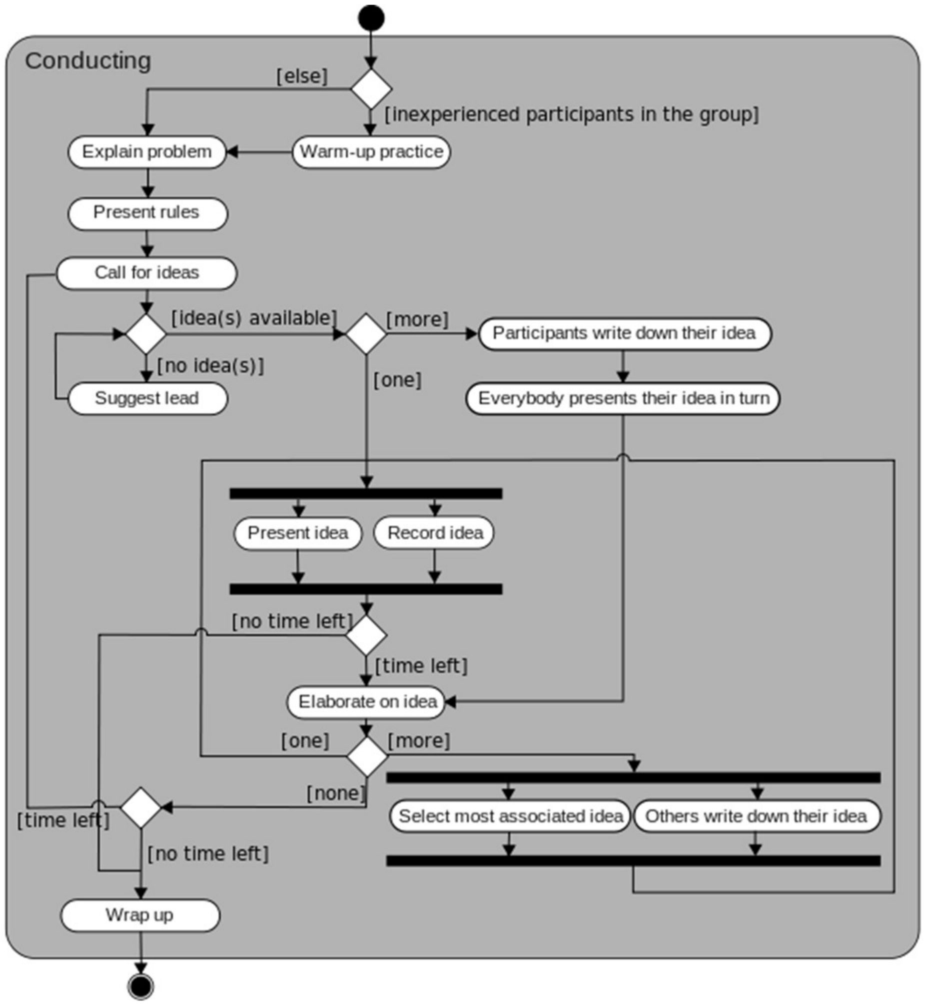
Examples:

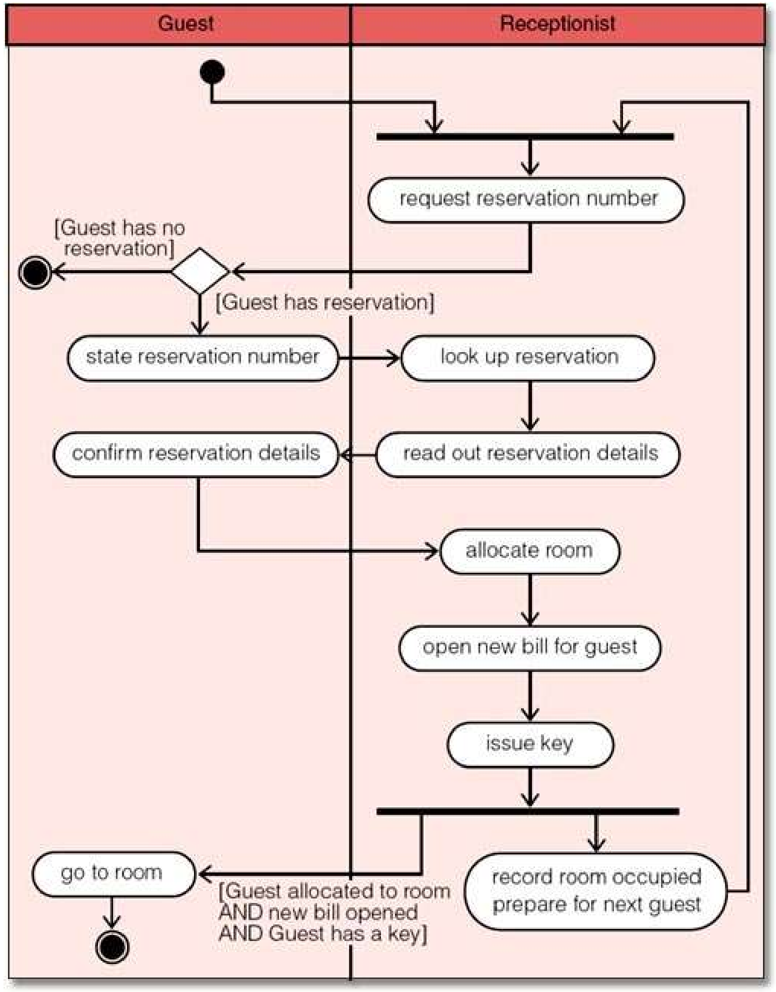
- Compatibility: The ATM shall be able to connect to any banking system for any bank card inserted
- Extensibility: The ATM shall be able to offer phone-card replenishment

Further Modelling – Activity Diagrams

- Learning Tree Course – User & System Requirements for Successful Software Development

Activity Diagrams




Guest
Receptionist
request reservation number
[Guest has no reservation]
[Guest has reservation]
state reservation number
look up reservation
confirm reservation details
read out reservation details
allocate room
open new bill for guest
issue key
go to room
[Guest allocated to room AND new bill opened AND Guest has a key]
record room occupied prepare for next guest

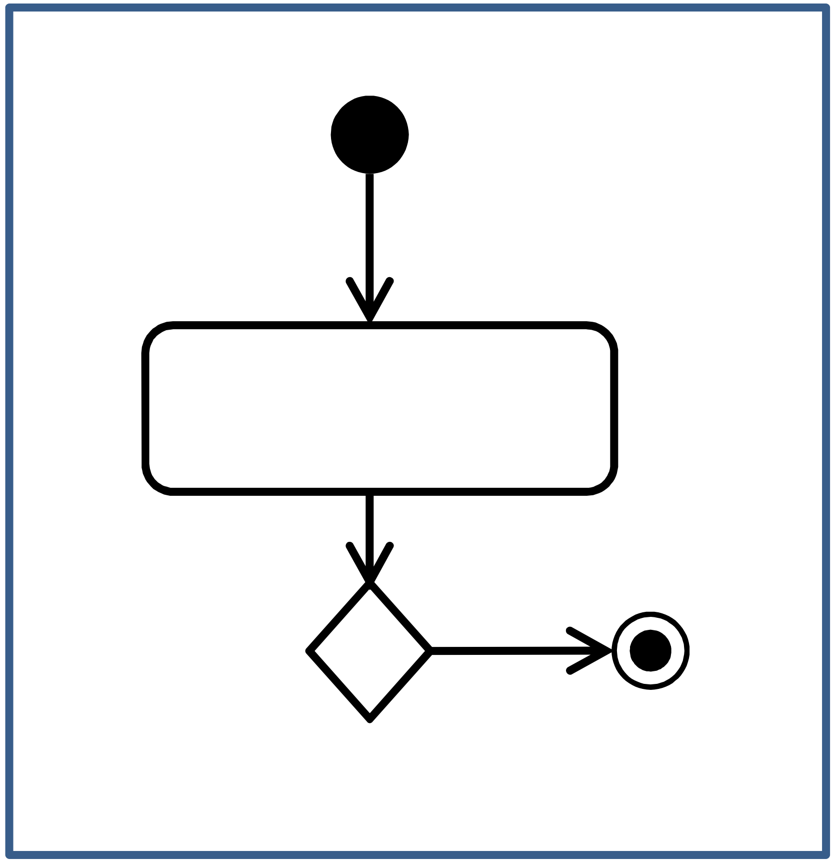
Activity Diagrams

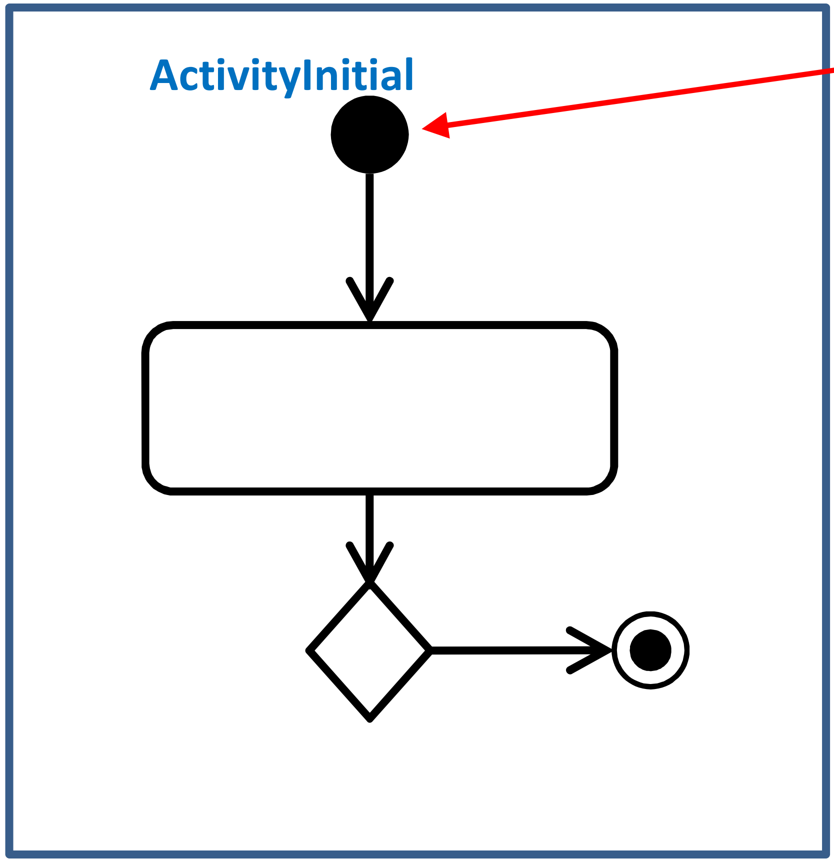
- **An activity diagram represents a dynamic view of the model and depicts**
  - Activities that are carried out by the system or other entities interacting with the system
  - The flow of control from one activity to the next
  - The conditions governing the movement between activities
  - Decision points where two or more activities branch or merge

Activity Diagrams

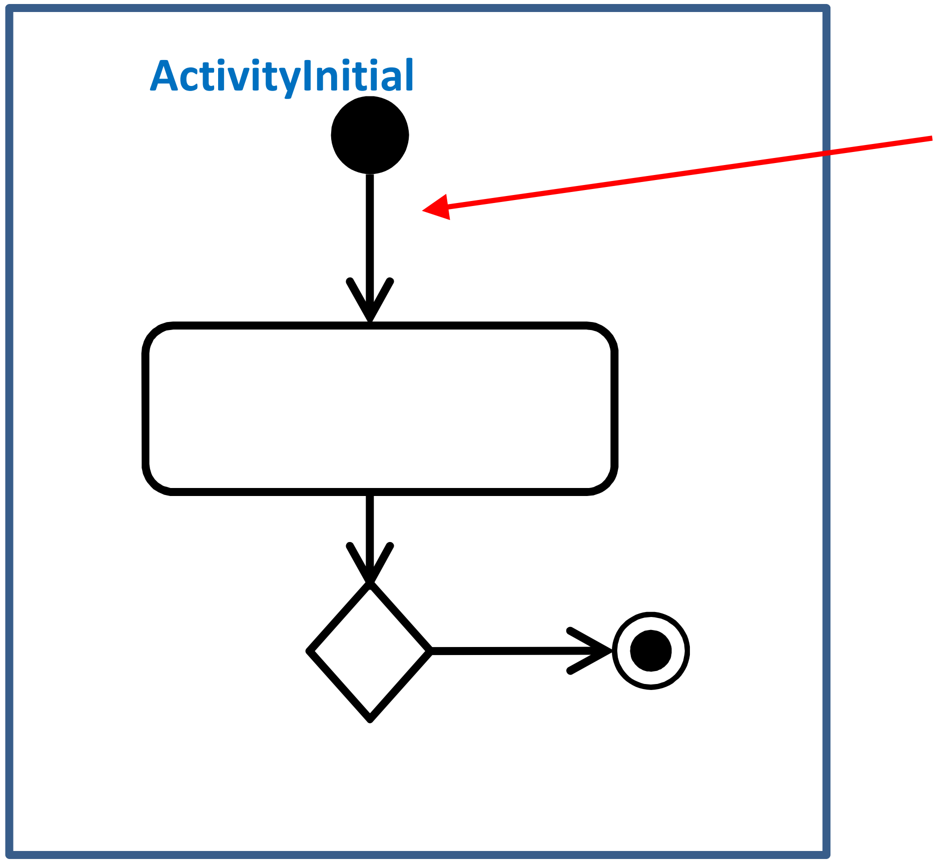
- **Used to show how different workflows in the system are constructed**
  - The start and the finish
  - The potential decision points
  - The parallel processing
- **Use cases can be supplemented by activity diagrams modelling the flow**

Parts of an Activity Diagram

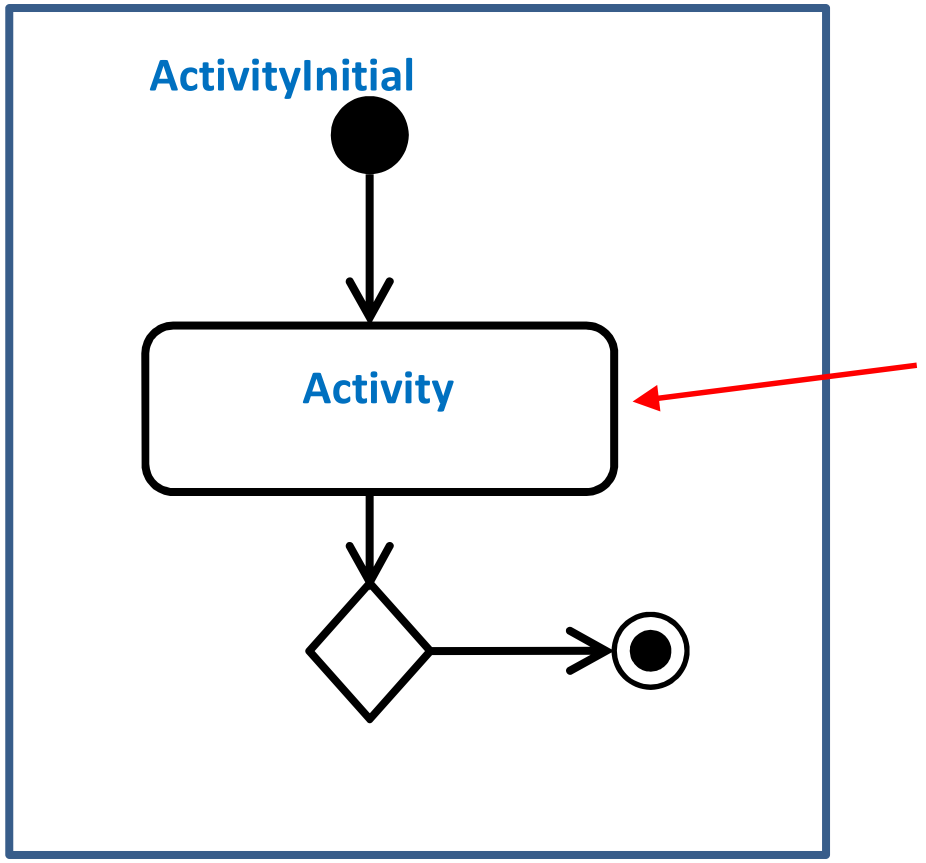




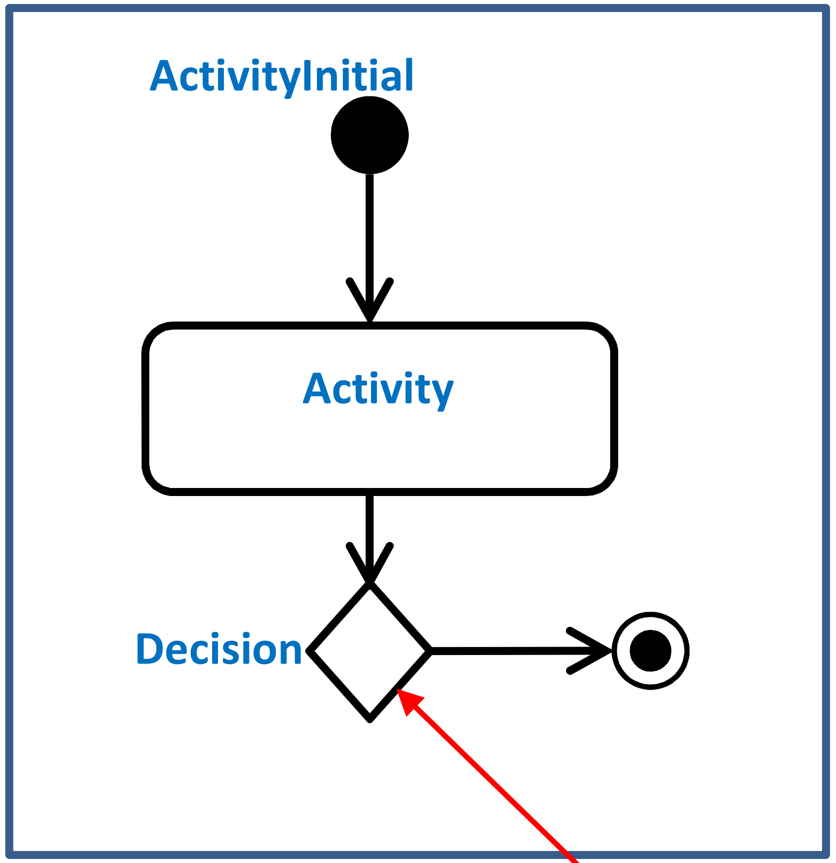
A black filled circle represents the start or **initial state** of the workflow



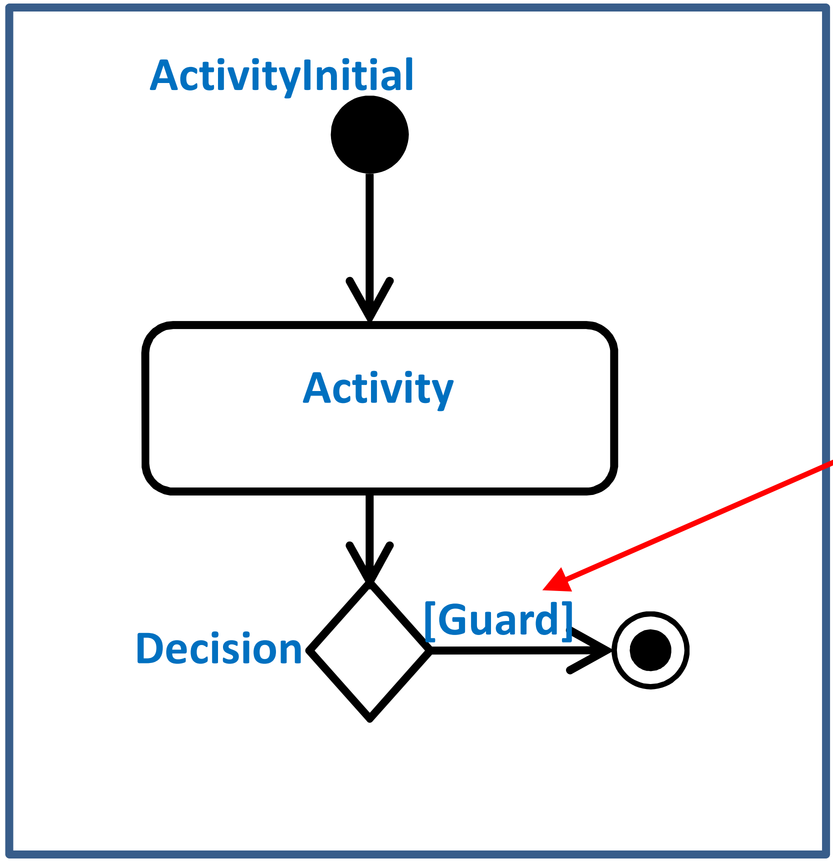
Arrows run from the start to the end, and represent the order in which activities happen, and the **flow of control** from one activity to the next



Rectangles with rounded corners represent **actions or activities**

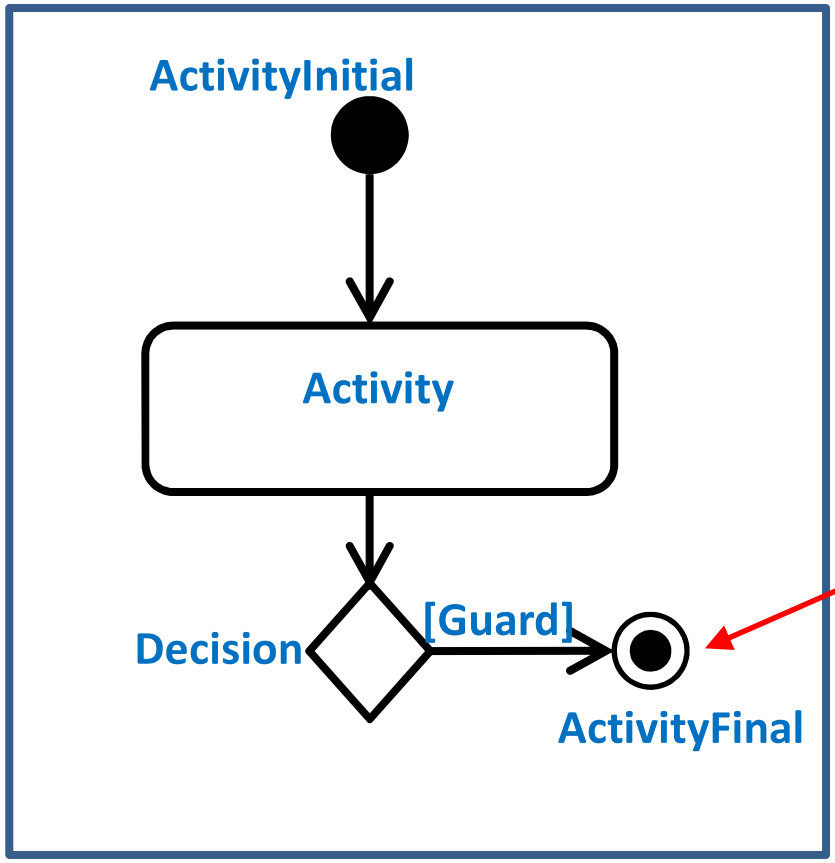


Diamonds represent **decision points**



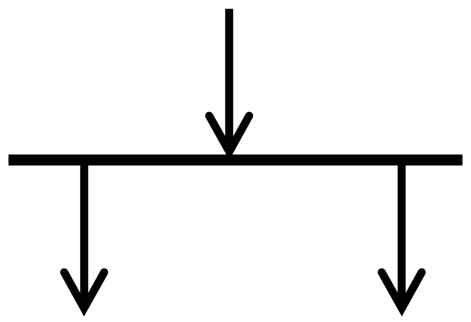
Guard condition, e.g.

- Yes/No
- Valid/Invalid

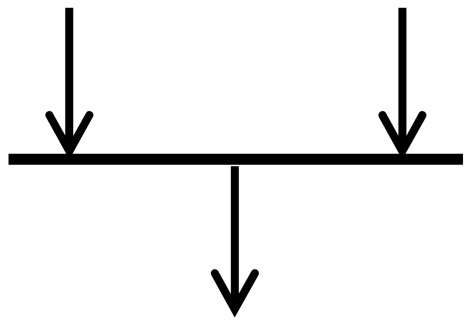


A large circle around a smaller black disc represents the end or **final state** of the workflow

Parts of an Activity Diagram

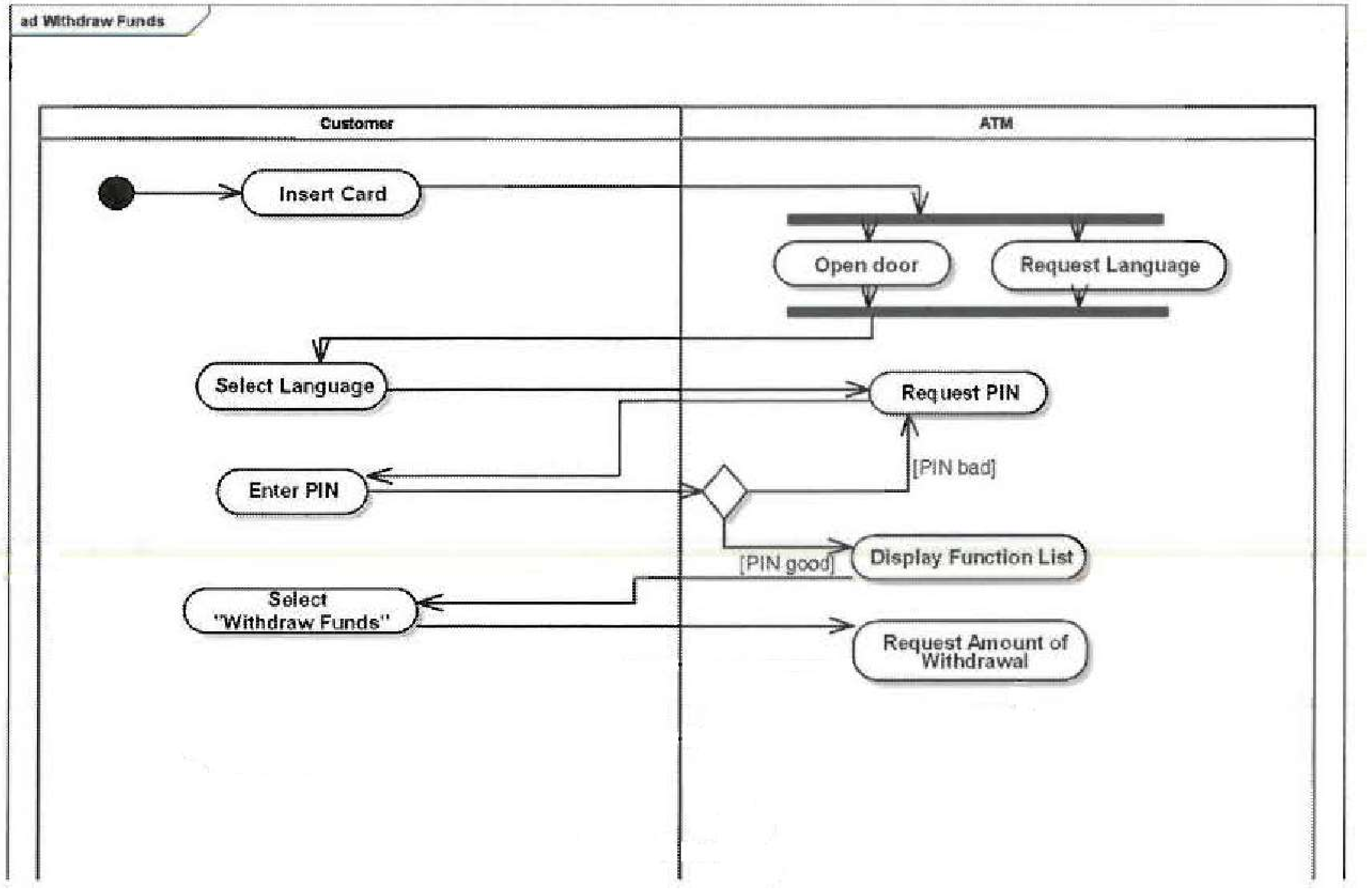


FORK: a single flow transitioning into two parallel flows

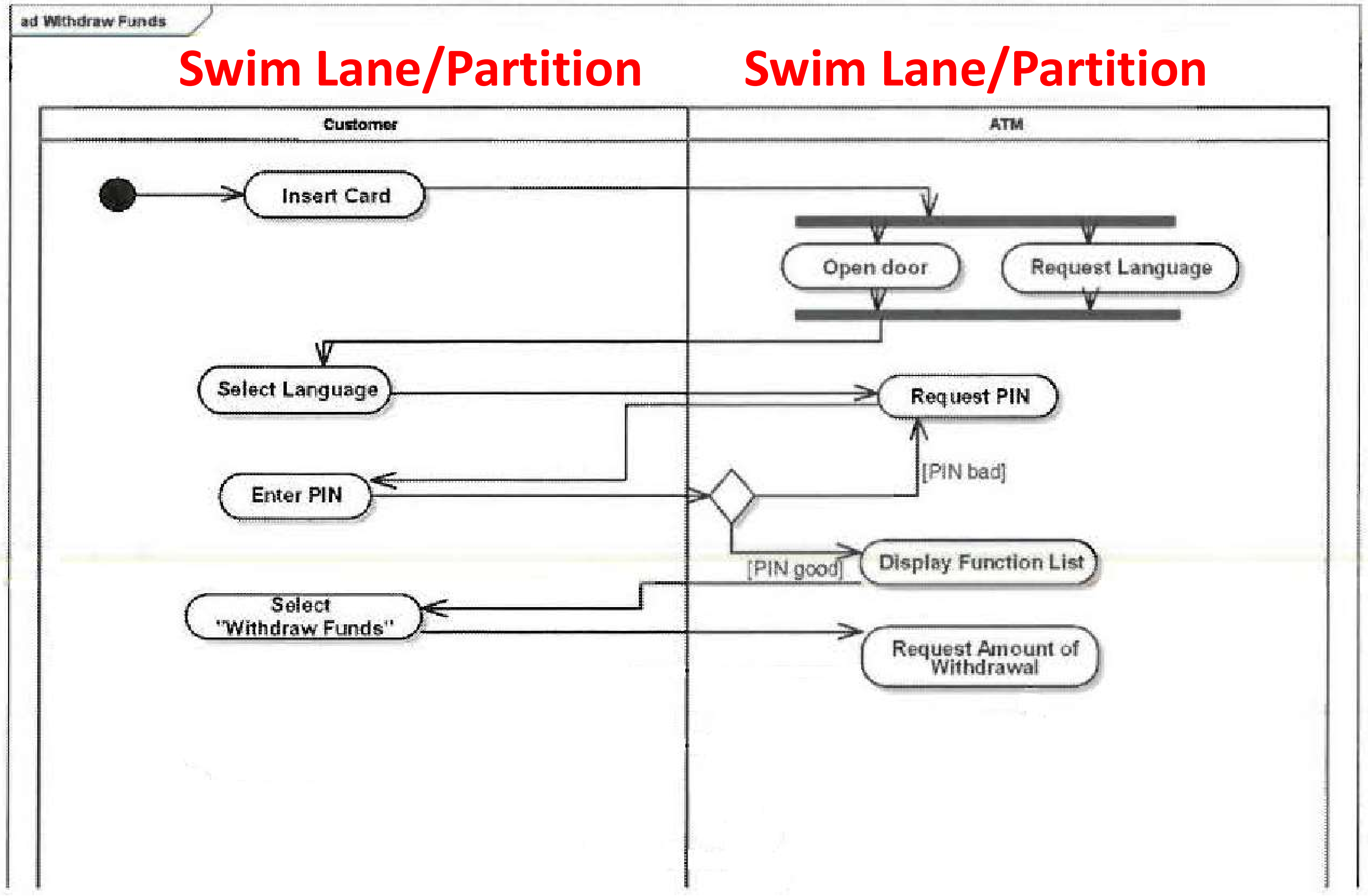


JOIN: two flows merging into a single flow

ATM Example: *Withdraw Funds* Use Case

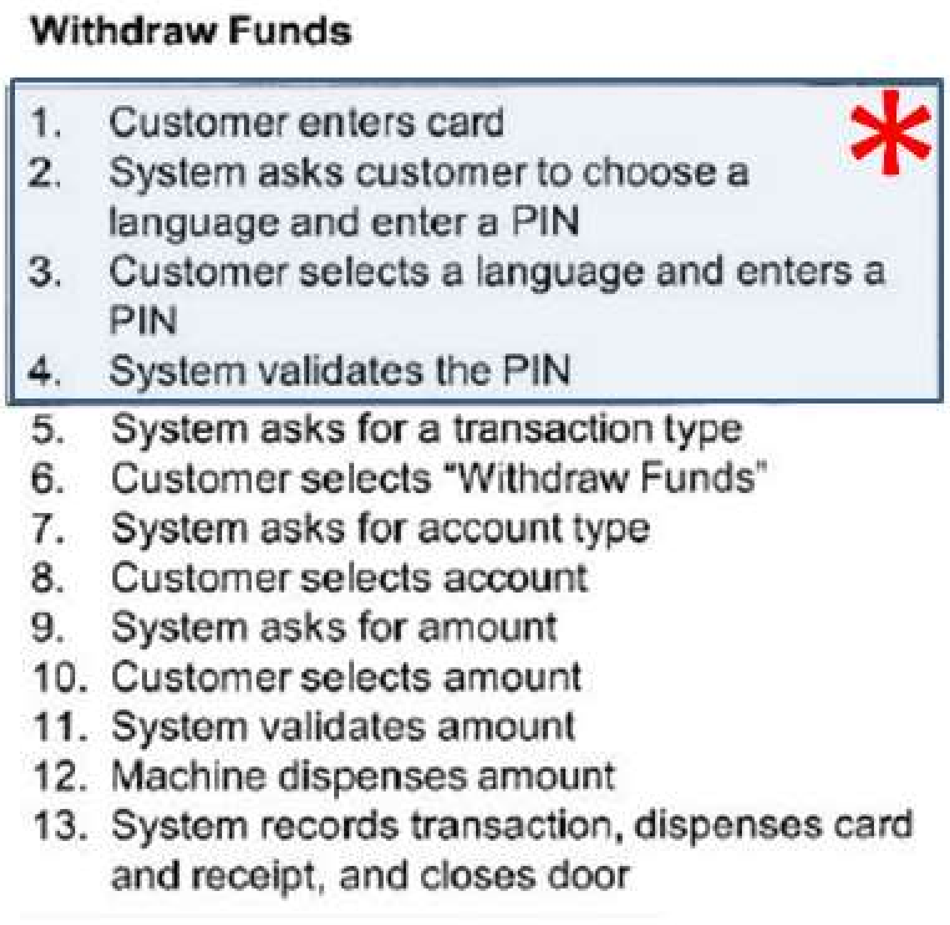


ATM Example: *Withdraw Funds* Use Case



Try to Complete the Diagram

Remember….



Withdraw Funds

1. Customer enters card
2. System asks customer to choose a language and enter a PIN
3. Customer selects a language and enters a PIN
4. System validates the PIN
5. System asks for a transaction type
6. Customer selects "Withdraw Funds"
7. System asks for account type
8. Customer selects account
9. System asks for amount
10. Customer selects amount
11. System validates amount
12. Machine dispenses amount
13. System records transaction, dispenses card and receipt, and closes door

* <u>**Log In**</u> Use Case

Produce an Activity Diagram for at least one Use Case within the Whaler Project, for example *Choose Boat*

References

- Felke-Morris, T. (2015) Basics of Web Design: HTML5 & CSS3. London: Pearson.
- Lauesen, S. (2005) User Interface Design: A Software Engineering Perspective. Boston: Addison Wesley.
- Learning Tree Course (2013) – User & System Requirements for Successful Software Development.
- U.S. Dept of Health and Human Services (2016). Usability Evaluation Basics [online]. Available from: http://www.usability.gov/what-and-why/usability-evaluation.html [Accessed: 18 April 2016].