CHALMERS |  UNIVERSITY OF GOTHENBURG

DIT045 H17 Requirements and User Experience

Review & Exam Info

Jennifer Horkoff
Email: jenho@chalmers.se

Today

- Review and summary of the course

- About the Exam

Lectures

1. Course Intro

RE Lectures

2. Requirements & Concepts
3. Documentation & Quality
4. Agile Requirements Engineering (Guest Lecture)
5. Requirements Modeling
6. Elicitation
7. Personas, Scenarios, Creativity, Prioritization

UX Lectures

8. Intro to UX & Prototyping
9. UX Design Patterns 1
10. UX Design Patterns 2
11. Usability & Intro to Usability Testing
12. Usability Testing & Field Studies
13. Verification and Validation

RE Example 4: Denver Airport

- "The Denver International Airport tried to build a very sophisticated version of such a system (Baggage Handling System) several years ago. The system used PCs, thousands of remote controlled carts, and a 21-mile-long track. Carts moved along the track carrying luggage from check-in counters to sorting areas and then straight to the flights waiting at airport gates. After spending $230 milling (USD) over 10 years the project was cancelled. Much of the failure can be attributed to requirements engineering mistakes."
- Issues:
  - Poor performance
  - Poor reliability
  - Poor understanding of complexity, novelty

- De Neufville, Richard. "The baggage system at Denver: prospects and lessons." *Journal of Air Transport Management* 1.4 (1994): 229-236.
- http://www.denverpost.com/2014/12/31/united-express-has-major-baggage-issues-at-denver-airport/

(Laplante)

System Failures

- Solution 1:
  - Build and deploy system incrementally
  - Heavy user involvement
  - Prototype
  - (a more Agile method)

Even this solution involves RE methods and tools!

- Solution 2:
  - Spent more time upfront understanding the problem, domain, users, environment, existing systems…
  - Focus on scoping
  - Capture knowledge in a structured and understandable way
  - Verify knowledge
  - (a more traditional Requirements Engineering method)

- Both can be good or bad depending on the type of system, size of system, size of team, nature of domain…
- Best solution can be in between

What is Requirements Engineering (RE)? (1)

- "Requirements Engineering can be defined as the systematic process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of the understanding gained." (Pohl, 1993)

- Systematic
- Iterative
- Co-operative
- Representations
- Checking accuracy (validation)

Why is RE important?

- Requirements form the basis for:
  - Project planning
  - Risk management
  - Trade off
  - Acceptance testing
  - Change control

(Dick et al.)

- Avoid system failures (see previous examples)
- Organizational memory
- Liability

What does RE include?

- One view (IREB):
  - System and system context (Domain)
  - Elicitation
  - Requirements Documentation
    - Text, Models
  - Validation and Negotiation
  - Management
  - Tool Support
- Another view (Easterbrook):
  - Risk, Ethics, Feasibility
  - Stakeholders, goals, boundaries, scoping
  - Elicitation
  - Modeling
  - Non-functional Requirements
  - Verification and Validation
  - Prioritizing Requirements
  - Software Evolution
- Another (Nuseibeh & Easterbrook):
  - Context and groundwork
  - Eliciting requirements
  - Modeling and analyzing requirements
  - Agreeing requirements
  - Evolving Requirements
- Another (Cheng & Atlee)
  - Elicitation
  - Modeling
  - Requirements Analysis
  - Verification & Validation
  - Requirements Management

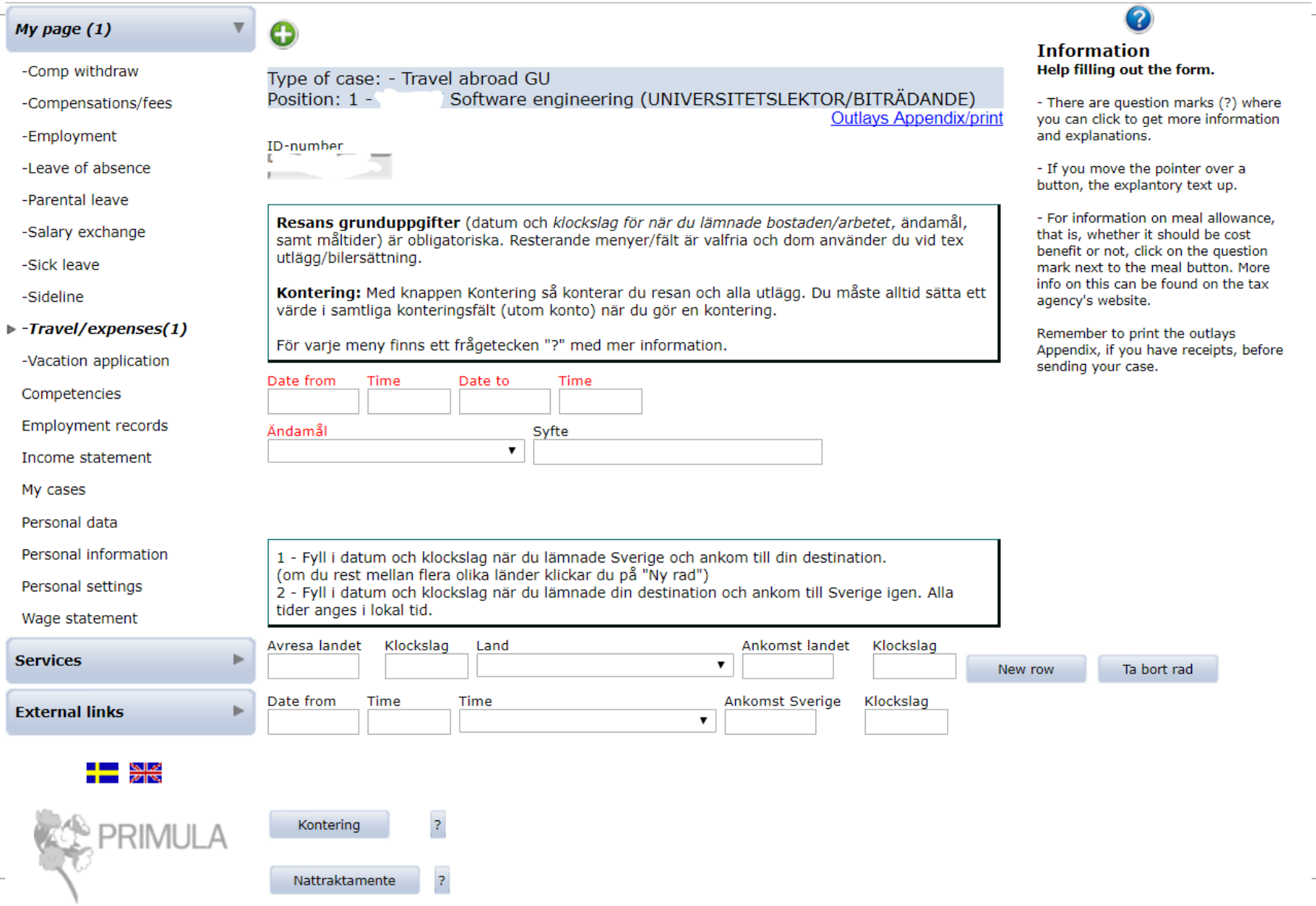
Learning Objectives (RE subset)

- *Knowledge and understanding*
  - describe the process of requirements elicitation, evaluation and prioritization,
  - documentation, validation and development of software requirements,
  - state techniques to acquire and model user demands,

- *Skills and abilities*
  - identify and specify requirements by means of, for instance, scenario-based techniques or goal-oriented techniques,
  - apply techniques to identify personas, scenarios and user stories,

- *Judgement and approach*
  - choose and motivate appropriate methods for involving users in the design process.

UX Example 4 ☹


My page (1)
-Comp withdraw
-Compensations/fees
-Employment
-Leave of absence
-Parental leave
-Salary exchange
-Sick leave
-Sideline
-Travel/expenses(1)
-Vacation application
Competencies
Employment records
Income statement
My cases
Personal data
Personal information
Personal settings
Wage statement
Services
External links
PRIMULA
Type of case: - Travel abroad GU
Position: 1 - Software engineering (UNIVERSITETSLEKTOR/BITRÄDANDE)
Outlays Appendix/print
ID-number
Resans grunduppgifter (datum och klockslag för när du lämnade bostaden/arbetet, ändamål, samt måltider) är obligatoriska. Resterande menyer/fält är valfria och dom använder du vid tex utlägg/bilersättning.
Kontering: Med knappen Kontering så konterar du resan och alla utlägg. Du måste alltid sätta ett värde i samtliga konteringsfält (utom konto) när du gör en kontering.
För varje meny finns ett frågetecken "?" med mer information.
Date from
Time
Date to
Time
Ändamål
Syfte
1 - Fyll i datum och klockslag när du lämnade Sverige och ankom till din destination.
(om du rest mellan flera olika länder klickar du på "Ny rad")
2 - Fyll i datum och klockslag när du lämnade din destination och ankom till Sverige igen. Alla tider anges i lokal tid.
Avresa landet
Klockslag
Land
Ankomst landet
Klockslag
New row
Ta bort rad
Date from
Time
Time
Ankomst Sverige
Klockslag
Kontering
?
Nattraktamente
?
Information
Help filling out the form.
- There are question marks (?) where you can click to get more information and explanations.
- If you move the pointer over a button, the explantory text up.
- For information on meal allowance, that is, whether it should be cost benefit or not, click on the question mark next to the meal button. More info on this can be found on the tax agency's website.
Remember to print the outlays Appendix, if you have receipts, before sending your case.

Why is UX Important?

- Efficiency, Time
- User satisfaction
- Sales, attracting and keeping customers
- Safety
- Sanity

What does UX include?

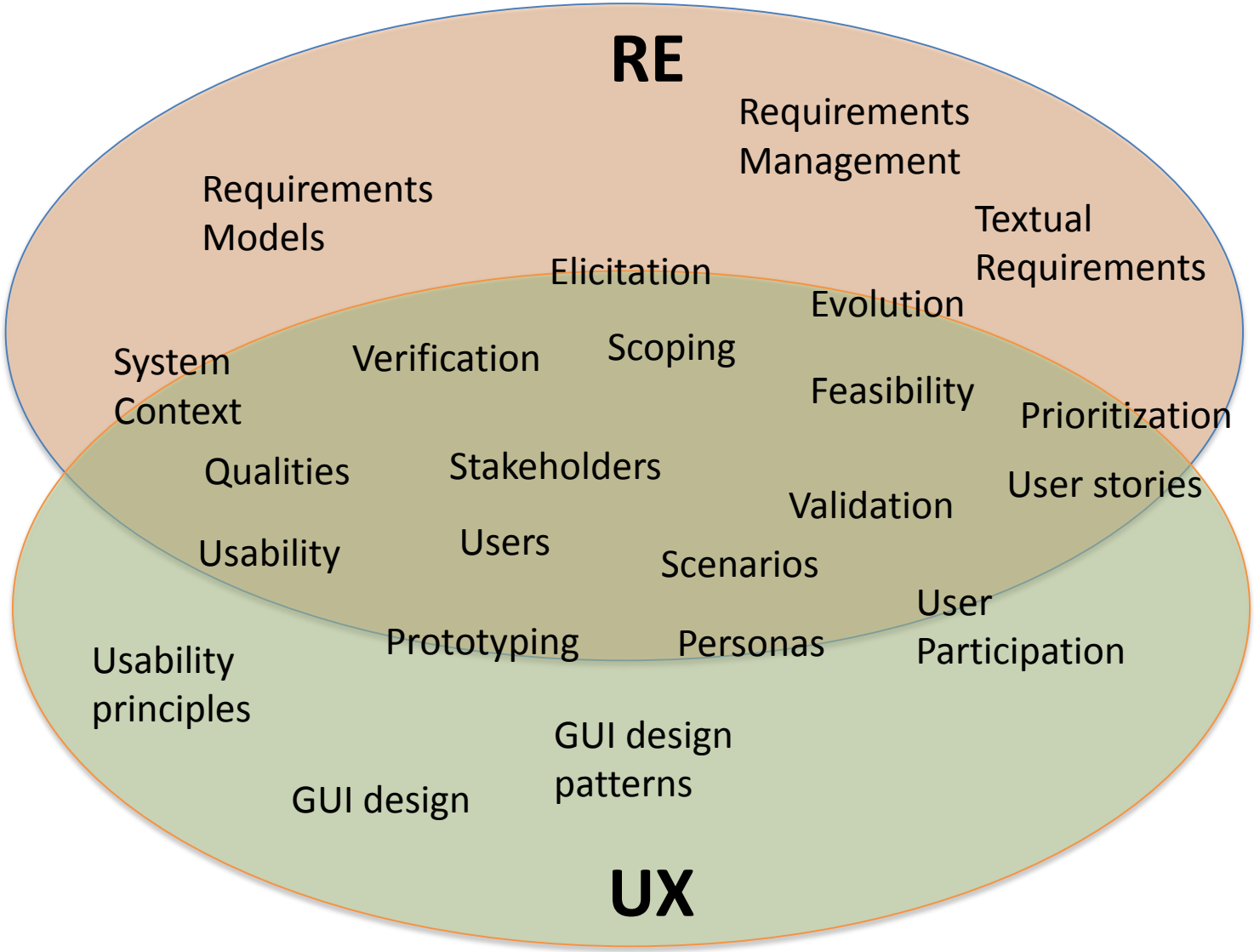
- One view (Tidwell, textbook)
  - User Research
    - Direct observation
    - Case studies
    - Surveys
    - Personas
  - User patterns & design

- Another (Garrett)
  - Product objectives and user needs
  - Functional specifications and content requirements
  - Interaction design and information architecture
  - Navigation
  - Sensory Design

Learning Objectives (UX subset)

- *Knowledge and understanding*
  - explain key techniques to account for usability in software products,
- *Skills and abilities*
  - apply techniques to identify personas, scenarios and user stories,
  - design and implement graphical user interfaces according to usability principles,
- *Judgement and approach*
  - choose an appropriate technique to evaluate the usability of a software product,
  - choose and motivate appropriate methods for involving users in the design process.

How do UX and RE relate?



Assignments

- "**Assignments** *(Inlämningsuppgifter)*, 3 higher education credits Grading scale: Pass (G) and Fail (U)"

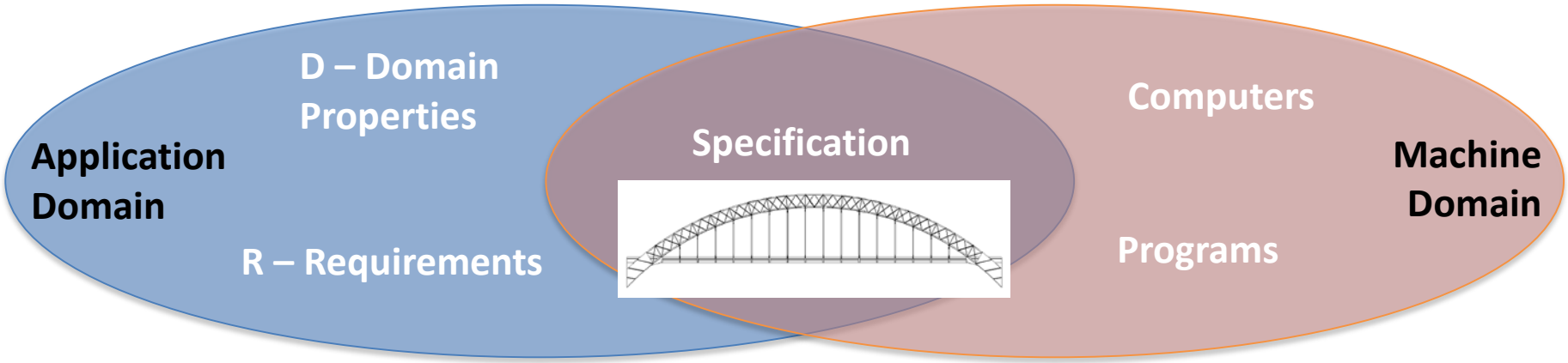
- Three assignments
- The assignment part of the course is worth 3 credits, so 1 credit per assignments.
- You will be given a grade on assignments in %
- Final 3-credit assignment course is pass/fail only ☹
- Will average the grade for all three assignments, each weighted equally.

Exam

- **"Written exam** *(Tentamen)*, 4.5 higher education credits Grading scale: Pass with Distinction (VG), Pass (G) and Fail (U)"

- Probably the 2nd week of January
  - Check with the schedule/student office!

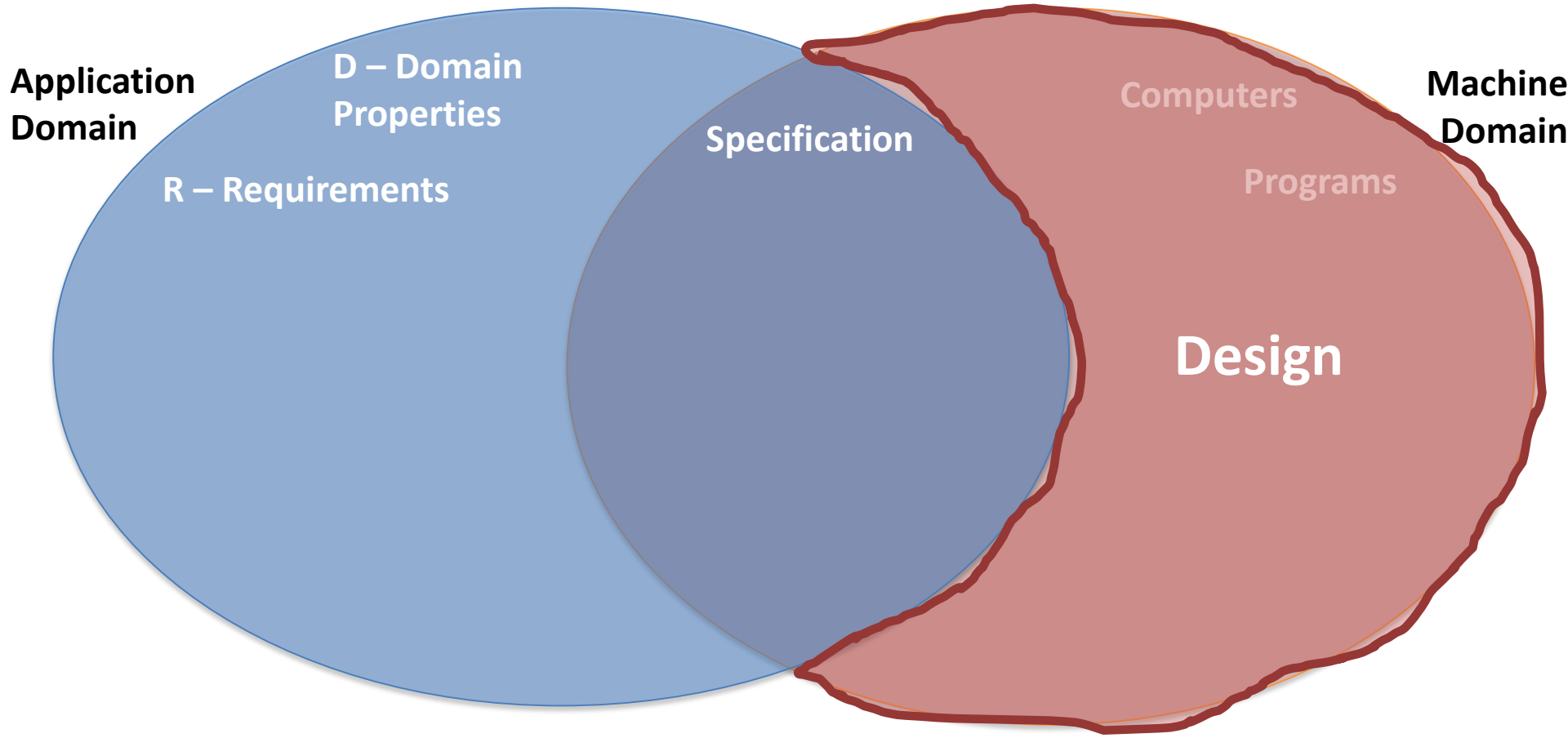
What is Requirements Engineering (RE)? (6)



- Domain Properties: things in the application domain that are true whether or not we build the system
- Requirements: things in the application domain we want to make true by building the system
- Specification: the behavior that a program needs in order to solve the problem
- S, D |- R (the specification along with domain assumptions entails (satisfies) Requirements)

(Zave & Jackson, Easterbrook)

Requirements vs. Design



- Don't include design information/unnecessary constraints in requirements

Functional vs. Non-Functional Requirements

- "Functional requirements are things the system must do" (Robertson^2)
- "Nonfunctional requirements are qualities the product must have" (Robertson^2)
- Functional Requirements (FRs):
  - I want to rent a bike
  - I want to return a bike
  - If the bike is returned before it is due, the system shall display a positive message to the user
  - The system shall allow the user to indicate how long they would like to rent the bike, choosing from 1, 3, or 5 hours.
- Non-Functional Requirements (NFRs):
  - It should be easy to rent a bike
  - The system should allow quick bike rentals
  - The system should be usable, it should be evaluated at 0.8 on a standard usability questionnaire
  - 80% of users should be able to rent a bike within 2 minutes

Requirements vs. Domain Properties

- Example domain assumptions:
  - We assume the bike rental system has access to a secure power source
    - But what if it doesn't? What happens if the power goes out?
  - We assume the city will provide sufficient law enforcement to prevent vandalism
  - We assume connection to the external payment system is available 24/7.
  - We assume the system will not be used during weather conditions in which the bikes and mechanical parts can freeze

Requirements vs. Constraints

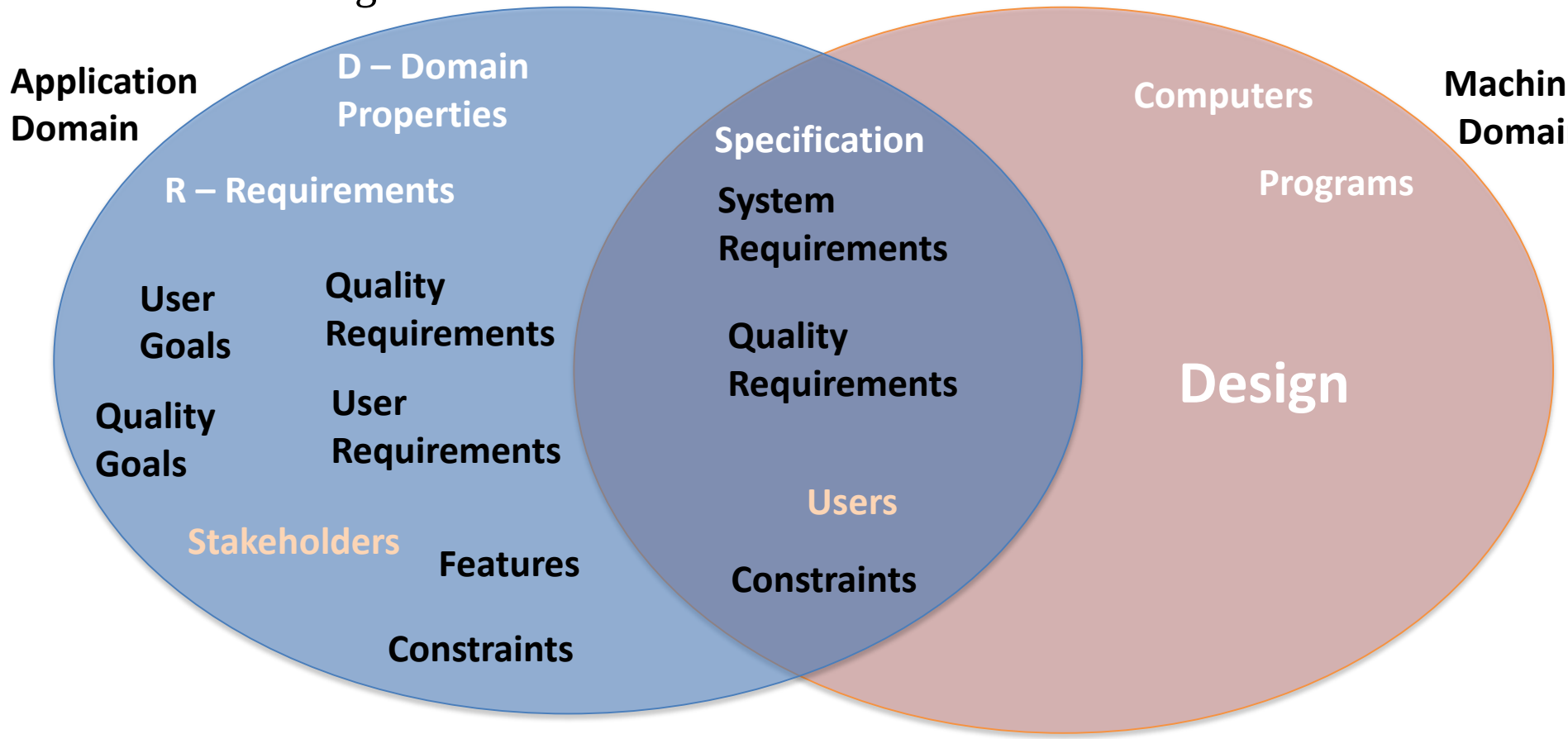
- "Constraints are global issues that shape the requirements" (Robertson^2)
  - The system shall be ready in fall of 2018
  - Wireless networking must follow the IEEE 802.11 standard

- Where do they come from?
  - The business (money, agreements)
  - Laws and regulations
  - Existing infrastructure
  - ....

- "Constraints are simply another type of requirement" (Robertson^2)
- They are a requirement that doesn't come from user needs

How does it all fit together?

- Where do goals live?



How to Capture/Document Requirements

- A few ways, we'll cover some of the main options

- Using Natural language (Part 1 of this lecture)
  - Traditional SRS form
  - User stories
  - Templates
  - Structured Natural Language

- Using models (Lecture 5)

SRS Requirements

- Examples for a lift (elevator):
  - A lift will only reverse direction when stopped at a floor
  - The system will cycle the lift doors every time that a lift stops at a floor
  - The lift must never be moved with the doors open
  - Each lift should be used an approximately equal amount

- Examples for a boat racing results program:
  - All input to the system is to be entered by the user
  - The user can enter and modify boat details
  - The user can enter and modify race details
  - When the user selects the option to modify the boat details, they are prompted to enter the boat's name
  - Subject to the constrains detailed below, the user can enter amend and delete details of: boat-class, boat, series, race, series-entry, race-entry
    - <details>

(Bray)

User Story Format

- The written description is often written in structured language:
- *As a < type of user >, I want < some goal > so that < some reason >*

- Example Epic: "As a user, I can backup my entire hard drive."
- Split into
  - "As a power user, I can specify files or folders to backup based on file size, date created and date modified.
  - As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved."

(Cohn, https://www.mountaingoatsoftware.com/agile/user-stories)

Volere Template

http://www.itib.net/Conform/Models/Volere/Release%2011/Volere%20Requirements%20Resources.htm

The Easy Approach to Requirements Syntax (EARS)*

The *Easy Approach to Requirements Syntax* (EARS) consists of a set of patterns for specific types of functional requirements and constraints

Pattern Name	Pattern
Ubiquitous	The <system|actor> shall <action> <object>.
Event-Driven	**When** <trigger> <optional precondition>, the <system|actor> shall <action> <object>
State-Driven	**While** <system state|actor state>, the <system|actor> shall <action> <object>
Unwanted Behavior	**If** <unwanted state|unwanted event>, THEN the <system|actor> shall <action> <object>
Optional Feature	**Where** <feature is included>, the <system|actor> shall <action> <object>
Compound	(combinations of the above patterns)

User Story vs. SRS Requirements Quality

User Story Quality

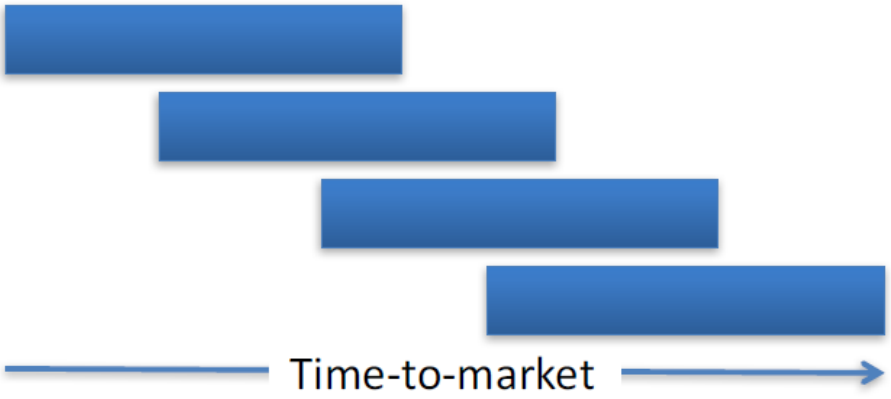
- Estimatable
- Testable
- Independent
- Valuable
- Small
- Negotiable

SRS Requirements Quality

- Feasible
- Prioritized
- Verifiable
- Modifiable
- Valid
- Traceable
- Concise
- Unambiguous
- Consistent
- Avoids Unnecessary Design Information
- Complete

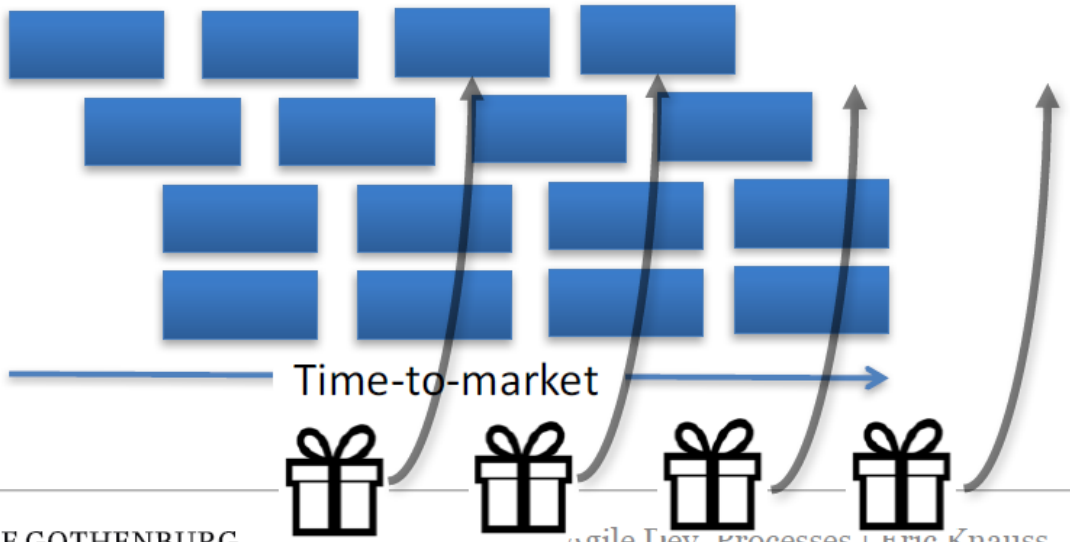
Towards Iterative/Incremental development

- Requirements
- Design
- Programming
- Test



Goal: Minimize risk, optimize learning from customer

- Requirements
- Design
- Programming
- Test



Scrum

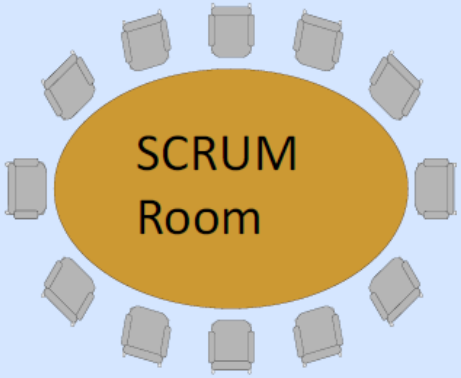
Product Owner

Product Backlog (prioritized)

SCRUM Master

Others

SCRUM team 7+/-2



SPRINT Backlog

SPRINT: 30 days

Increment

Daily SCRUM
15 min.

Do you need requirements engineering?

Context	Agile	RE
Startup	(-) MVP and Learning over process!	(+) No. 1 problem: No clear problem to solve
Small mobile/web app	(+) Ideal case	(-) Agile RE practices sufficient
Large-scale software dev	(+) But hard to implement, see SAFe and LESS framework	(+) Need to align work of several teams
Large-scale system dev	(?) Very hard to implement, but promises huge gains. Hardware, mechanics cannot be developed agile	(+) Need to align work of different domains, suppliers, etc.

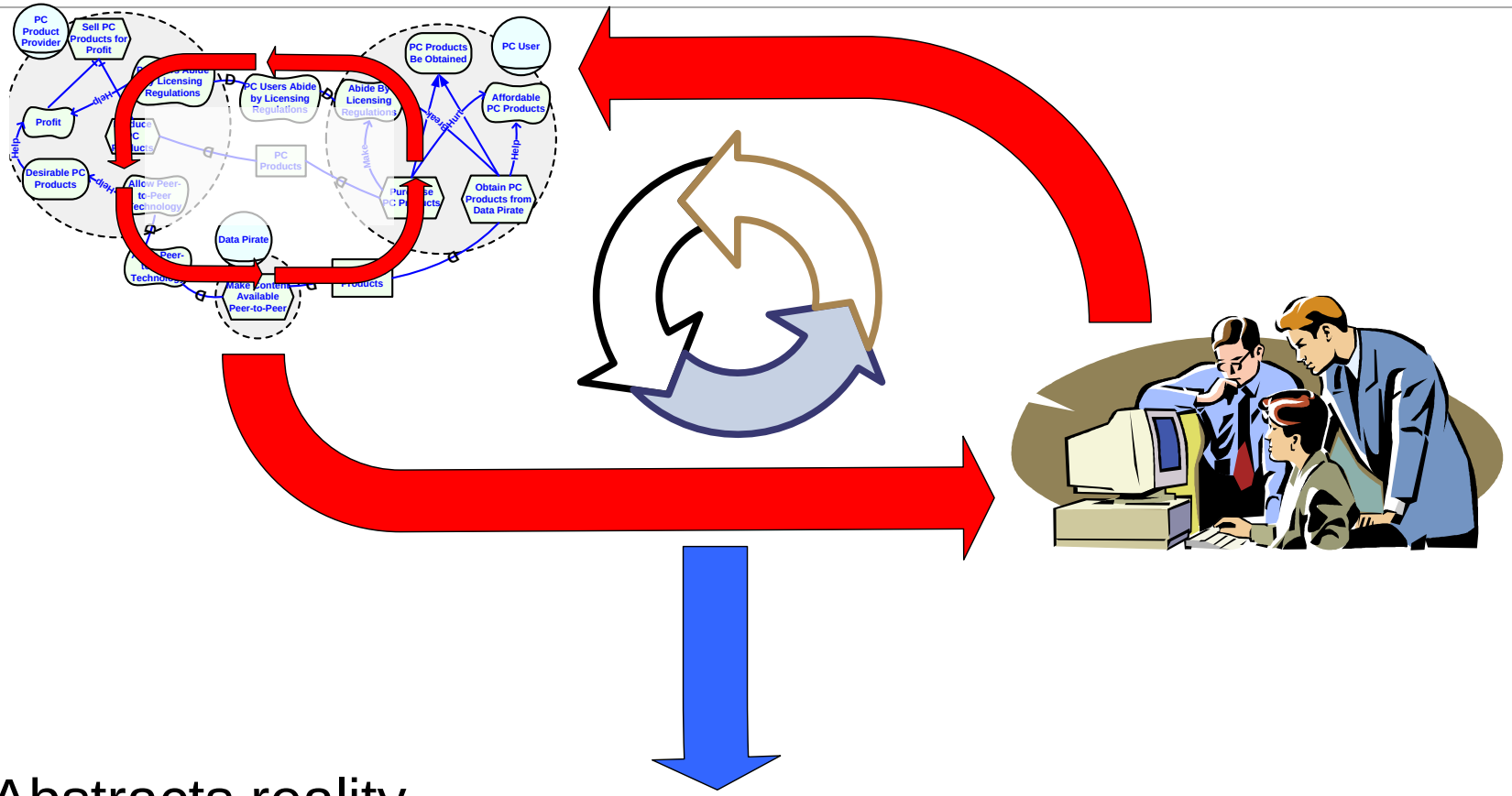
Requirements Eng (2006) 11: 1–3
DOI 10.1007/s00766-004-0206-4

VIEWPOINTS

Alan M. Davis · Didar Zowghi

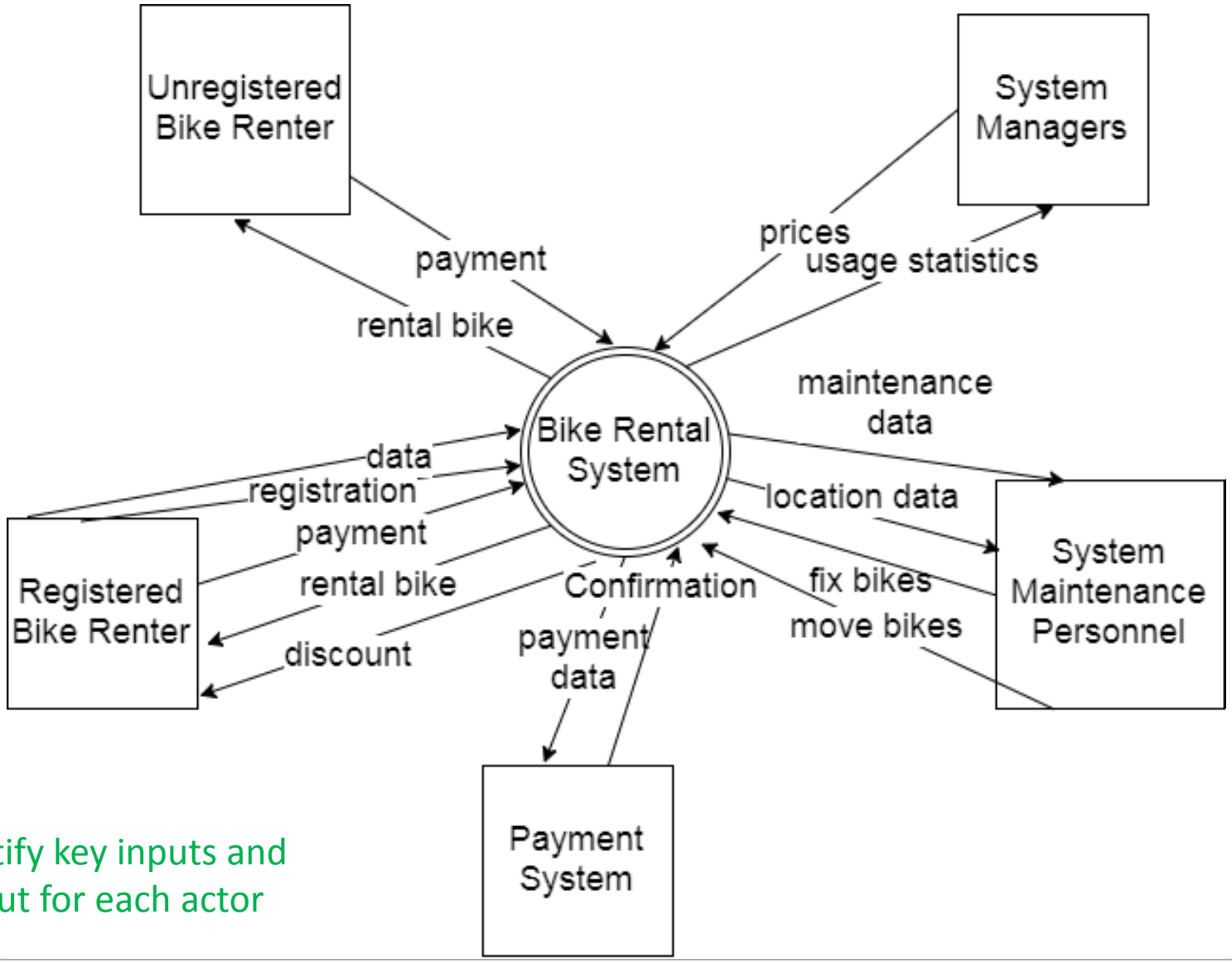
Good requirements practices are neither necessary nor sufficient

Why Use Requirements Models?



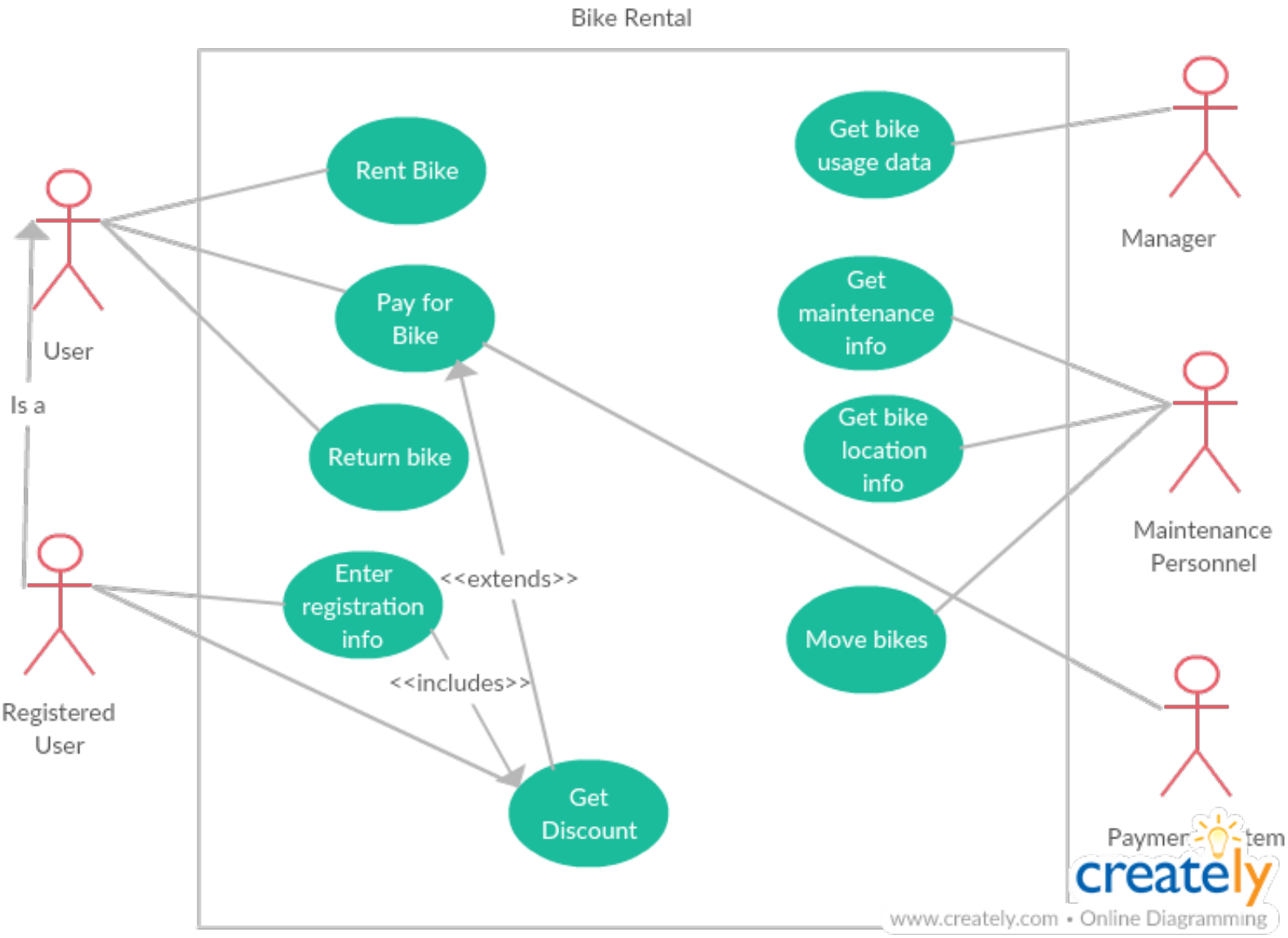
- Abstracts reality
- Facilitates communication and understanding
- Reveals gaps in knowledge
- Promotes domain analysis, “what if…?”
- Helps elicit requirements

Context Example: Bike Rental



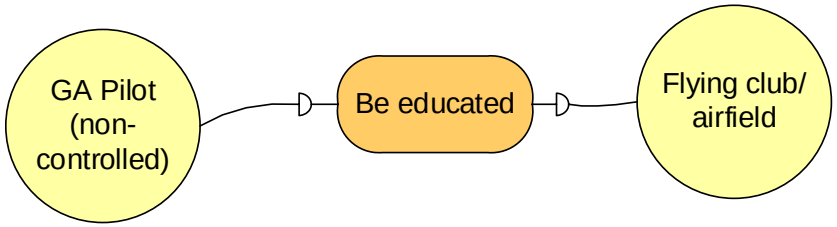
Identify key inputs and output for each actor

Bike Rental



i* Strategic Dependencies

- Goal Dependency: I want you to achieve my goal, I don't care how



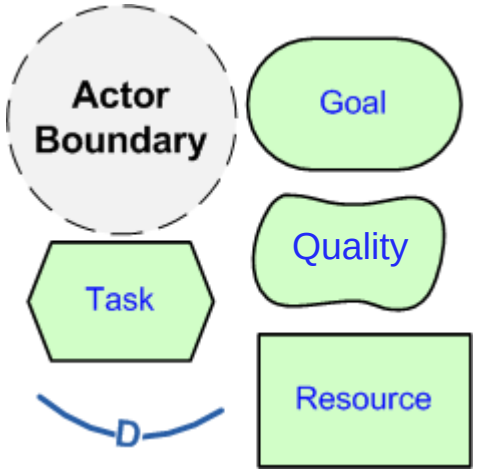
- Task Dependency: I want you to achieve this task, in an agreed upon way

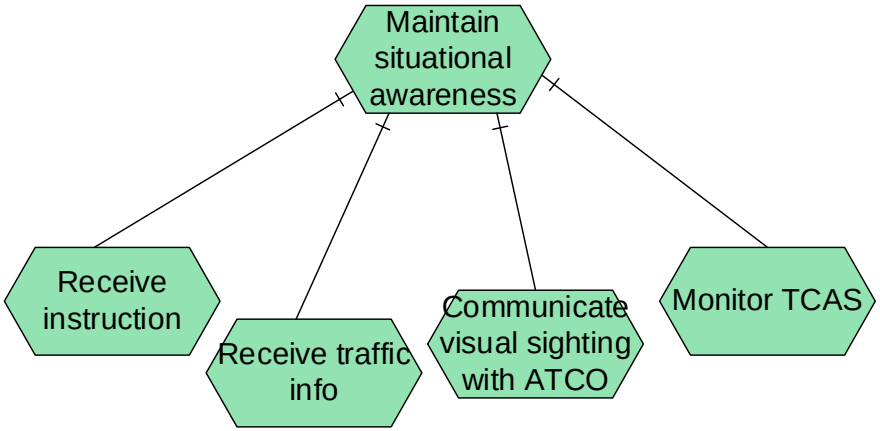
- Resource Dependency: I want you to provide this thing (entity)

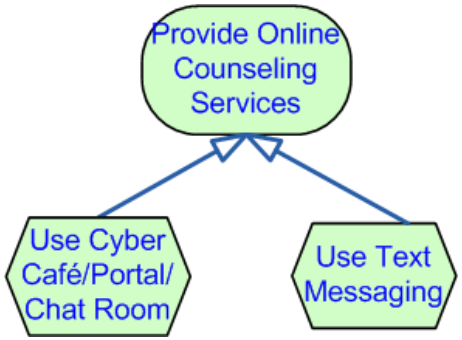
- Quality Dependency: I want you to achieve my quality

i* Strategic Rationale (SR) Diagrams

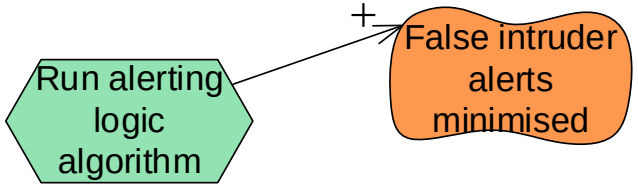
- Actor boundaries
- Goals, Qualities, Tasks, Resources
- Dependencies (as before)
- Refinement (AND), (OR)



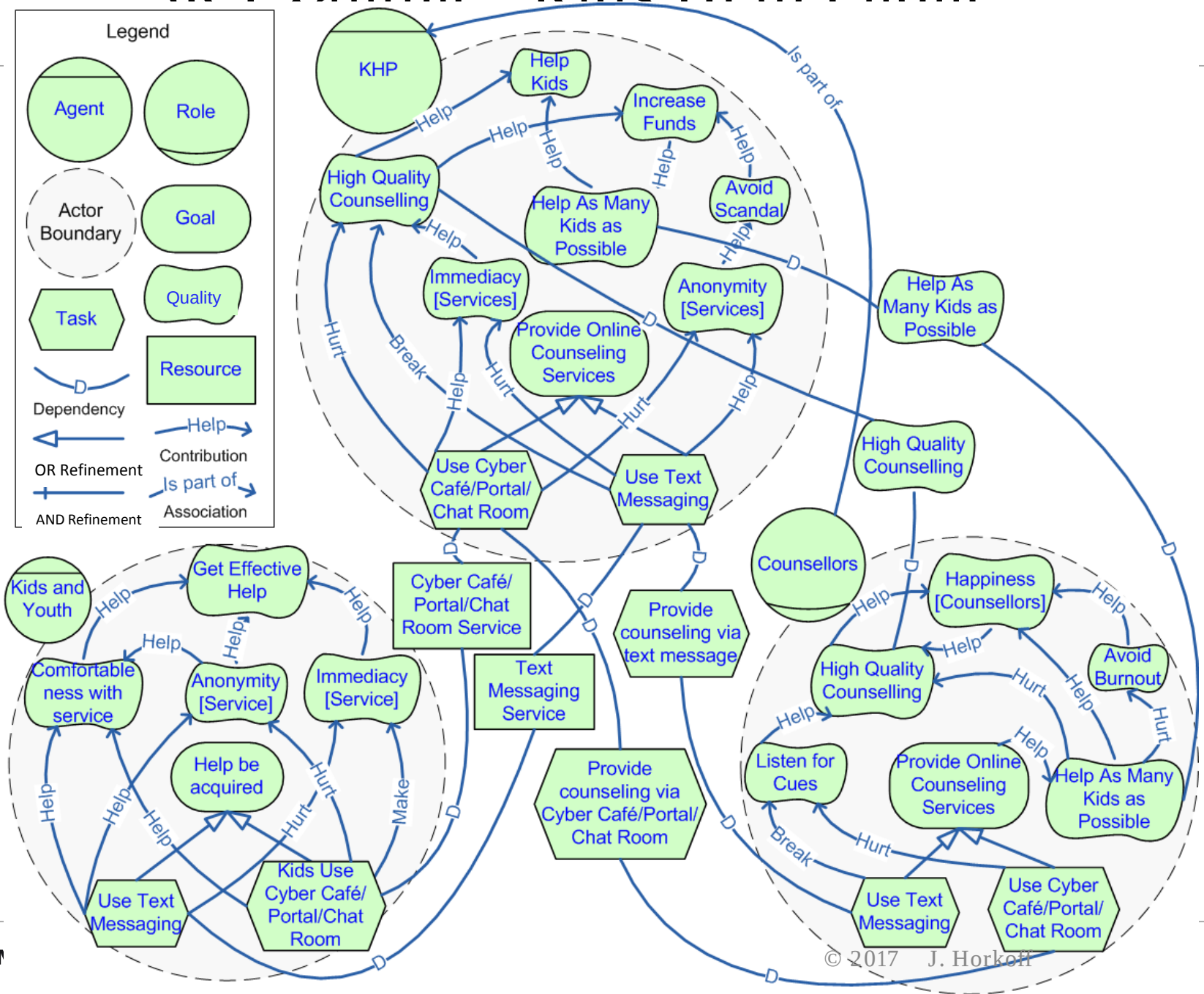




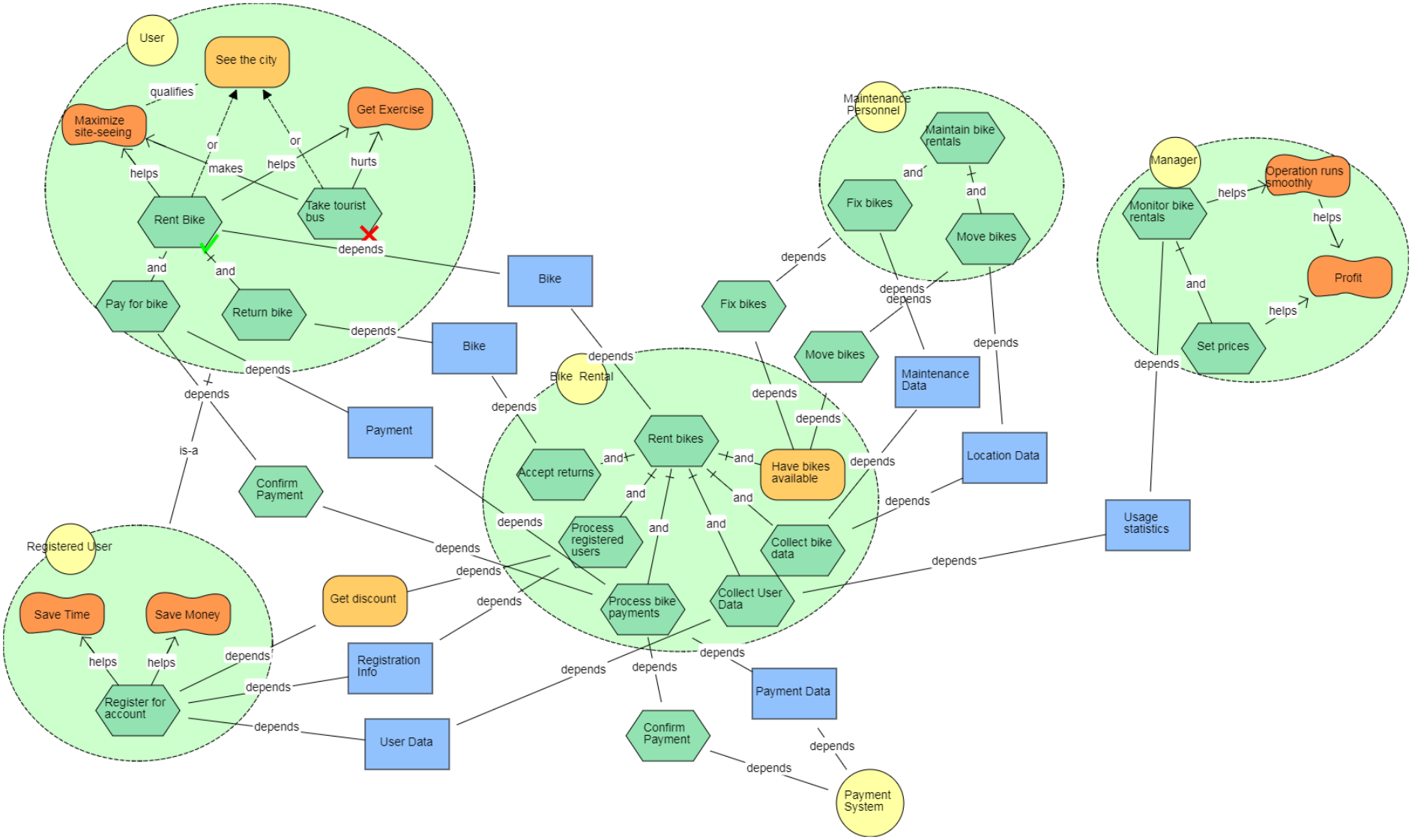
- Contribution: Make, Help (+), Hurt (-), Break



SR Example: Kids Help Phone



Bike Rental: All



Approximate Model Mappings

Context Diagram	Use Cases	Goal Models
System actor	System boundary	System actor
Other actors	Actors	Actors
Inputs/Outputs	Roughly map to use cases	Dependencies
	Use Case	(Usually) Task
		Qualities
		And/Or Refinement
		Contribution

Why is Elicitation Difficult

- Thin spread of domain knowledge (Easterbrook)
  - It is rarely available in an explicit form (i.e. not written down)
  - ...distributed across many sources
  - ...with conflicts between knowledge from different sources
- Tacit knowledge (The “say-do” problem)
  - People find it hard to describe knowledge they regularly use
  - Limited Observability
  - The problem owners might be too busy coping with the current system
  - Presence of an observer may change the problem
- Bias
  - People may not be free to tell you what you need to know
  - People may not want to tell you what you need to know
    - The outcome will affect them, so they may try to influence you (hidden agendas)

(a selection of) Elicitation Techniques

(Easterbrook)

- Traditional Techniques
  - Documentation
  - Data Sampling
  - Interviews
  - Surveys/Questionnaires
- Collaborative Techniques
  - Focus groups
  - Prototyping
- Contextual (social) approaches
  - Participant Observation
  - ...
- Cognitive techniques
  - Think aloud protocol
  - ...

Example Persona

USDA Senior Manager Gatekeeper

From www.usability.gov

Photo:

Fictional name: Matthew Johnson

Job title/ major responsibilities: Program Staff Director, USDA

Demographics: 51 years old, Married

Father of three children, Grandfather of one child

Has a Ph.D. in Agricultural Economics.

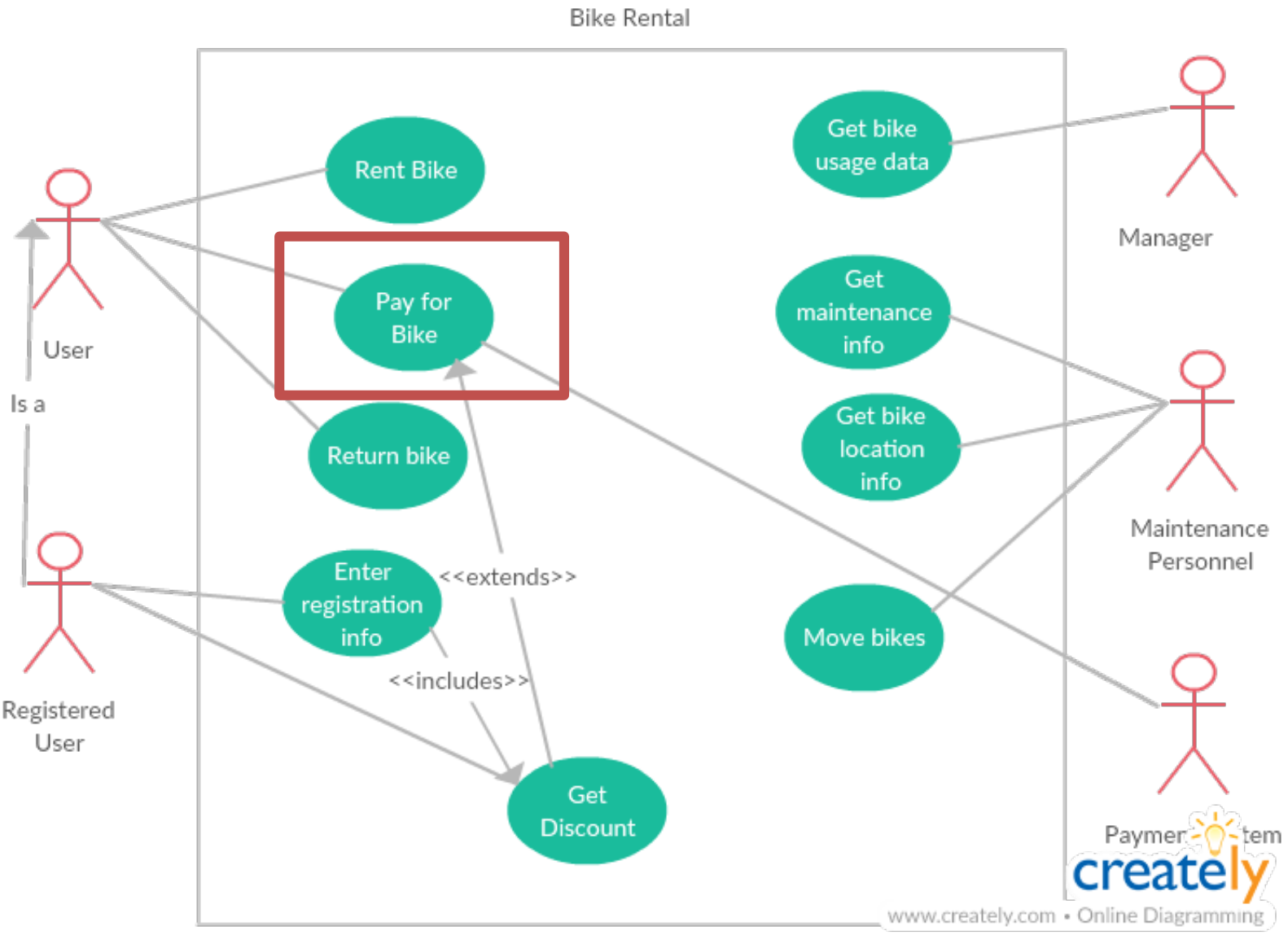
Goals and tasks: He is focused, goal-oriented within a strong leadership role. One of his concerns is maintaining quality across all output of programs.

Spends his work time: Requesting and reviewing research reports, preparing memos and briefs for agency heads, and supervising staff efforts in food safety and inspection.

Environment: He is comfortable using a computer and refers to himself as an intermediate Internet user. He is connected via a T1 connection at work and dial-up at home. He uses email extensively and uses the web about 1.5 hours during his work day.

Quote: "Can you get me that staff analysis by Tuesday?"

Example: Bike Rental



Example: Pay for Bike (part 1)

Use Case: #2 pay for Bike

CHARACTERISTIC INFORMATION

Goal in Context: To give payment in order to rent a bike

Scope: For the bike rental system, paying only for bike rentals

Level: Primary Task

Preconditions: User has indicated they would like to rent bike, has selected a bike.

For registered users, user has logged in.

Success End Condition: payment transaction successful

Failed End Condition: payment failed, payment aborted

Primary Actor: User (bike renter)

Trigger: The user tries to rent a bike

MAIN SUCCESS SCENARIO

1. User is prompted to pay for bike rental, showing payment amount
2. User Oks amount
3. User chooses form of electronic payment (Debit, Credit, Touch payment)
4. User uses card to pay for bike
5. User enters card PIN
6. Payment confirmation is received from Payment System, confirmation shown to user

Example: Pay for Bike (part 2)

EXTENSIONS

1. <registered users> Payment amount shows registration discount

SUB-VARIATIONS

2. User does not OK amount, bike rental cancelled
3. User does not choose form of electronic payment, bike rental cancelled
4. User does not provide card after 60 seconds, user notified, bike rental cancelled
5. User does not provide correct PIN, bike rental cancelled
6. Payment confirmation is not received from Payment system, user notified, bike rental cancelled

RELATED INFORMATION (optional)

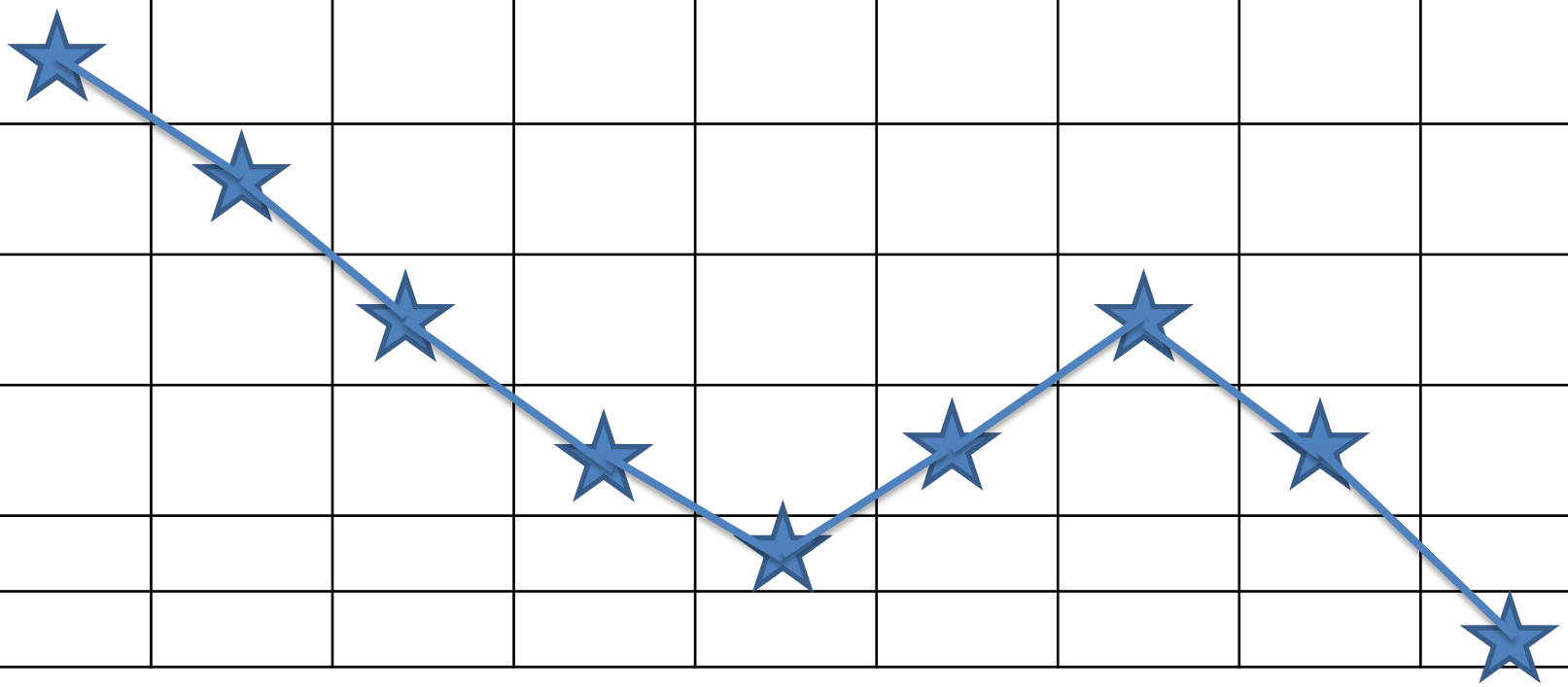
Priority: Coming later in the lecture (could be high/medium/low, or a number)
Performance Target: less than 30 seconds
Frequency: often, once every 5 minutes during peak periods
Superordinate Use Case: N/A
Subordinate Use Cases: N/A
Channel to primary actor: interactive
Secondary Actors: Payment System
Channel to Secondary Actors: network transaction

SCHEDULE

Due Date: Before System deployment, Spring, 2018

Bike Rental

	Acquisition				Riding		Return		Feedback
	Bike Selection	Select Options	Payment	Confirmation & Info	Bike Riding	Reminders	Return Bike	Confirmation	Feedback
Bike Rack									
Rental Display									
Payment Panel									
User Device									
Bike									
Email									



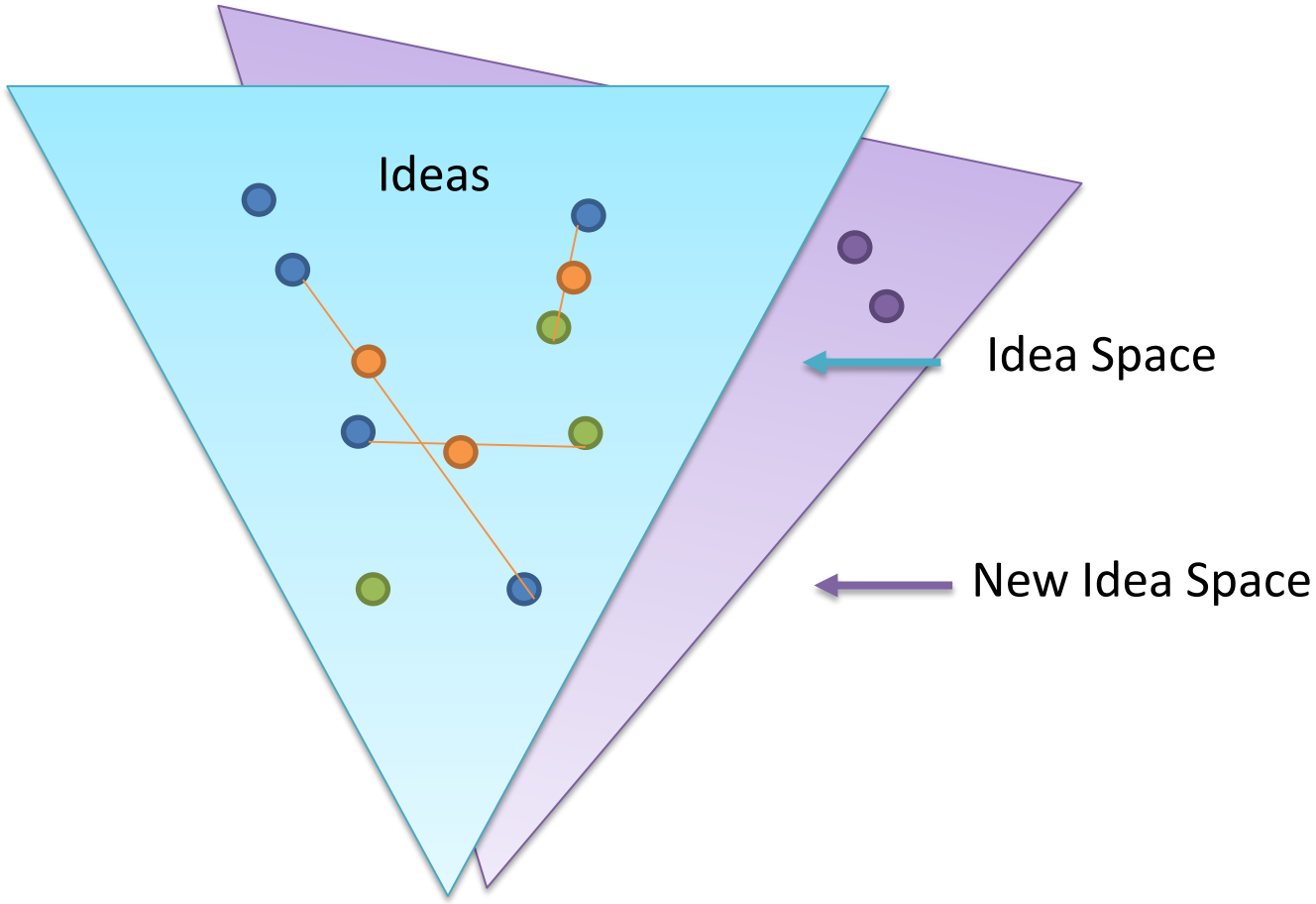
Exploratory, Combinatorial, Transformational Creativity

- (Presentation idea via N. Maiden)

Exploratory Creativity

Combinatorial Creativity

Transformational Creativity



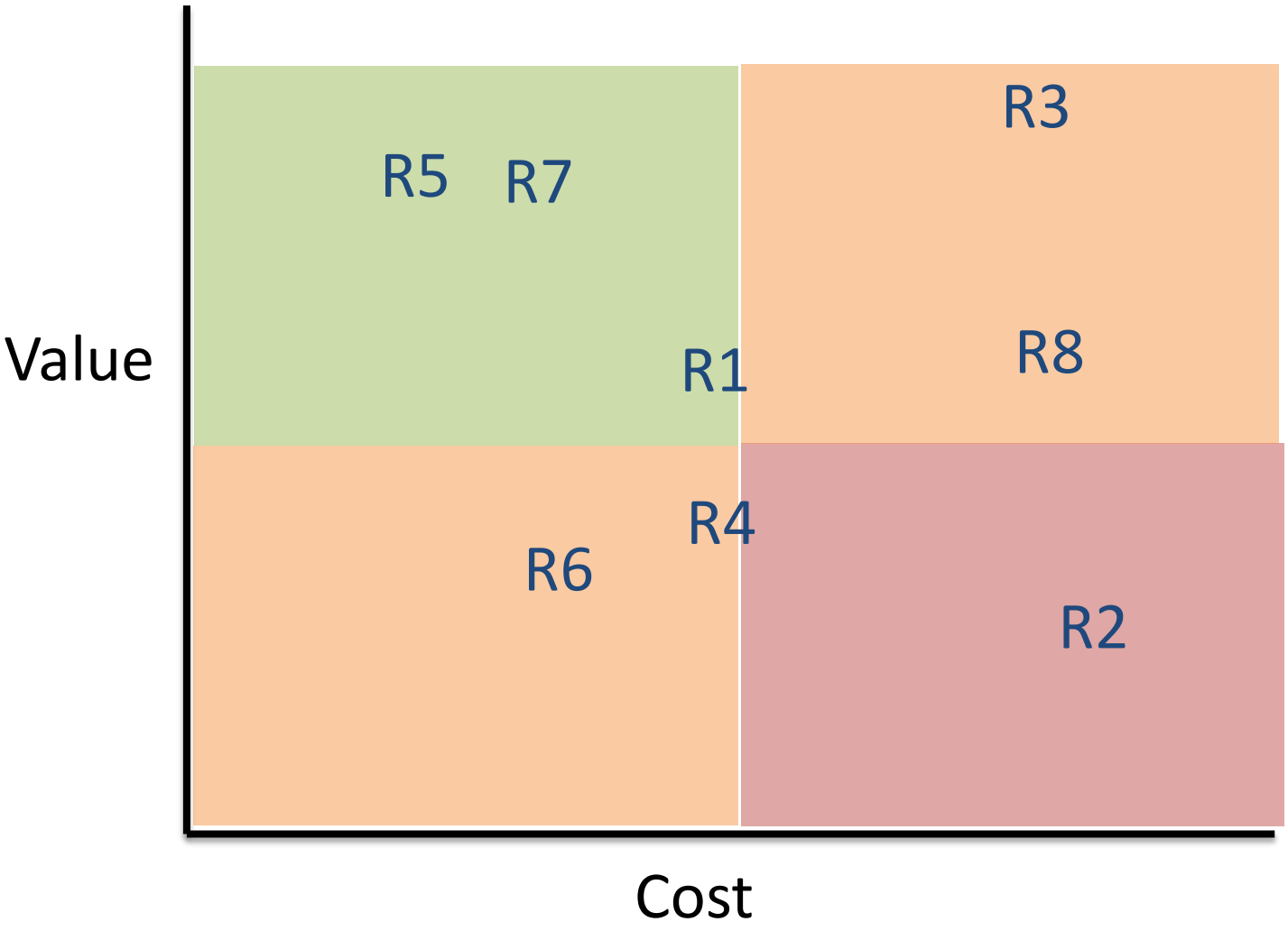
Creativity Activities

- (Somewhat) structured techniques in order to generate ideas
- E.g., Brainstorming, Hall of Fame/Bright Sparks, Creative Search, Pairwise Comparison, Creativity Triggers, Assumption Busting, Roleplaying, ...

- Can be performed manually, or can be supported by tools
- Usually conducted on groups of people

- Long list found at:
  - http://becreative.city.ac.uk/

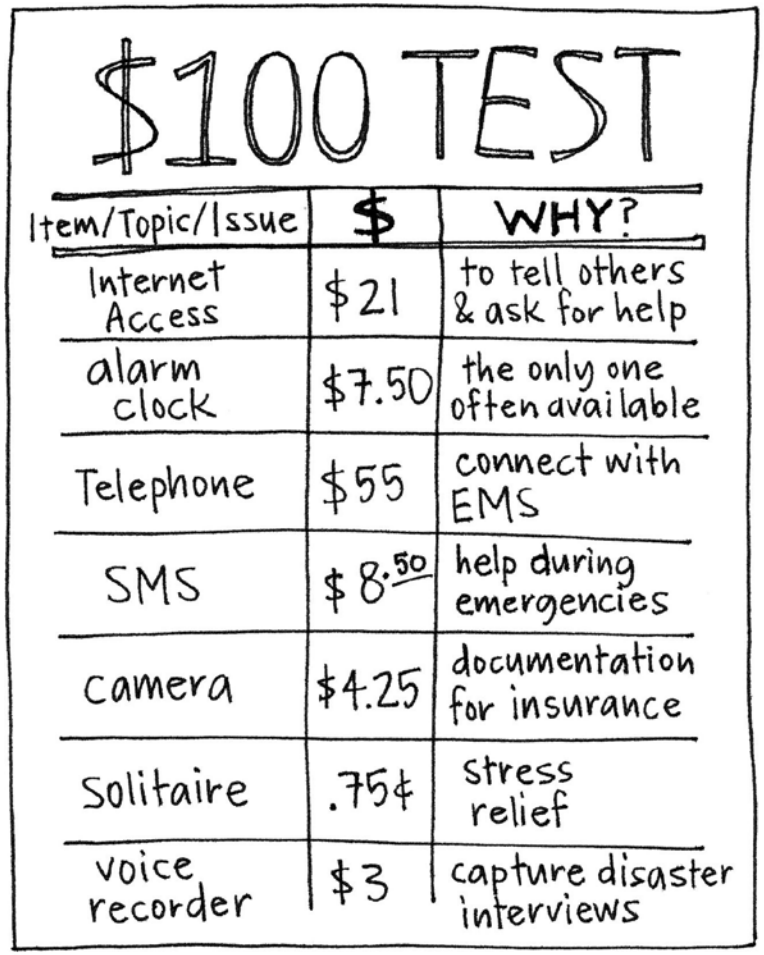
Value vs. Cost



R5,
R7,
R1,
{R6, R3, R8},
R4,
R2

$100 Method

- You have $100 to spend (or whatever currency)
- You have a certain number of requirements (not too many)
- Ask a group of people to pay money for the requirements
- How much would you pay for each?
- Capture “why?”
- The ordering of spending is the relative priority



http://gamestorming.com/100-test/

Terminology (Part 1)

- Usability: The ability of a user to use a thing to carry out a task successfully
- UX (User Experience): a broader view, looking at the individual's entire interaction with the thing, as well as thoughts, feelings and perceptions
- UI (User Interface): the part of the system that the users sees and interacts with (screens, buttons, etc.)

(Tullis & Albert)

UCD

Research>	Concept>	Design>	Development>	Implementation>
Stakeholder interviews	Card Sorts	Wireframes	Usability Testing	Usability Testing
Metrics	Information Architecture	Paper Prototypes	Bug Testing	Bug Testing
Competitive Analysis	Flowchart	Interactive Prototypes	Content Creation	Post-Launch Surveys
Surveys	User Stories	Desirability Testing		Metrics to Demonstrate Improvements/ Success
Interviews	User Interaction Flowchart	Usability Testing		Continued User Research
Focus Groups	System Interaction Flowchart			
Ethnographic studies	Content Inventory			
Field Studies	Content Maps			
Task Analysis	Accessibility Planning			
Diary Studies				
Desirability Testing				
Usability testing				

Process is iterative!

https://blogs.uoregon.edu/uxuo/2013/09/20/ucd-ux-usability-so-whats-the-difference/

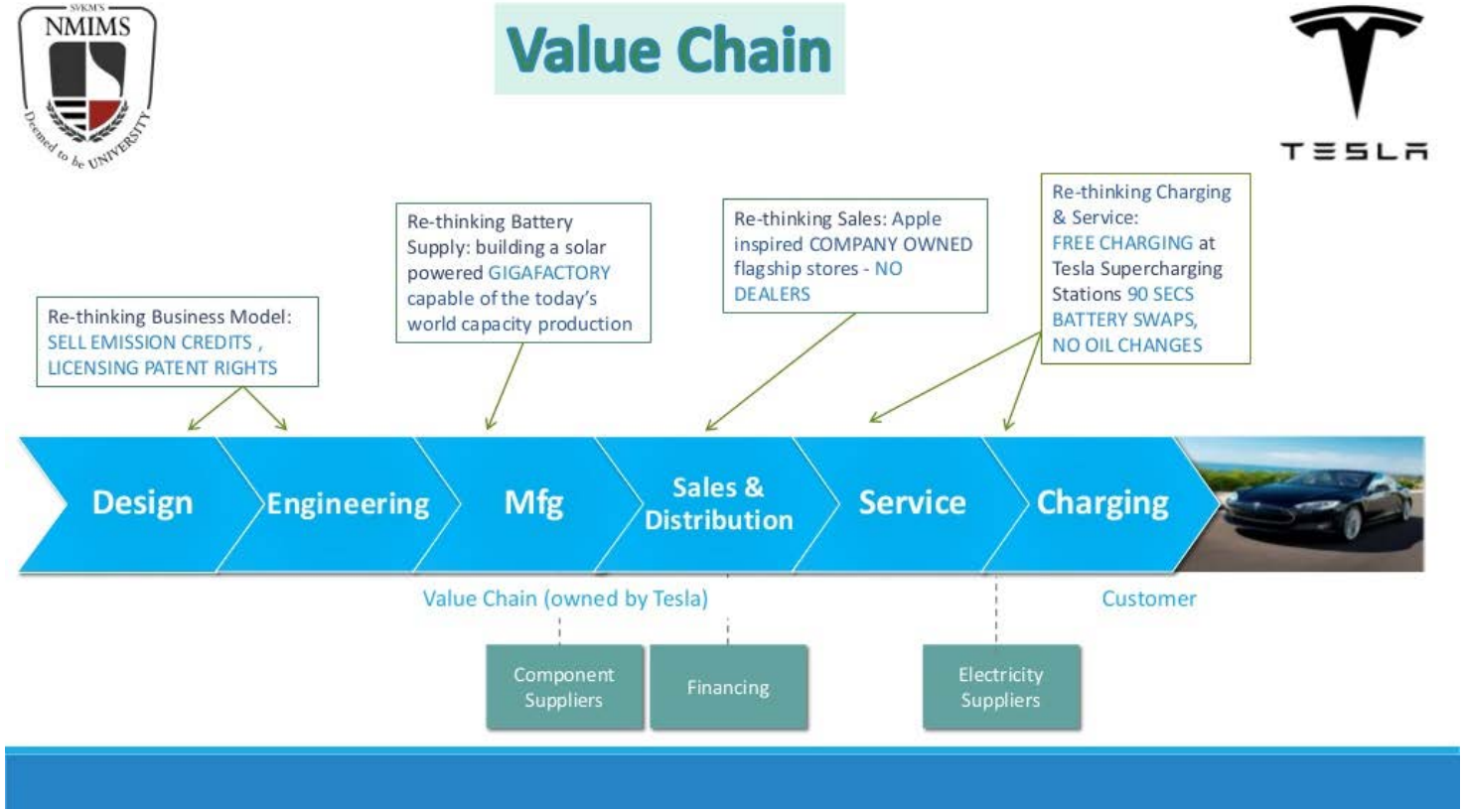
Business Strategy

The Business Model Canvas (Osterwalder, 2009)

Key Partners	Key Activities	Value Proposition	Customer Relationships	Customer Segments
	Key Resources		Channels	
Cost Structure		Revenue Streams		

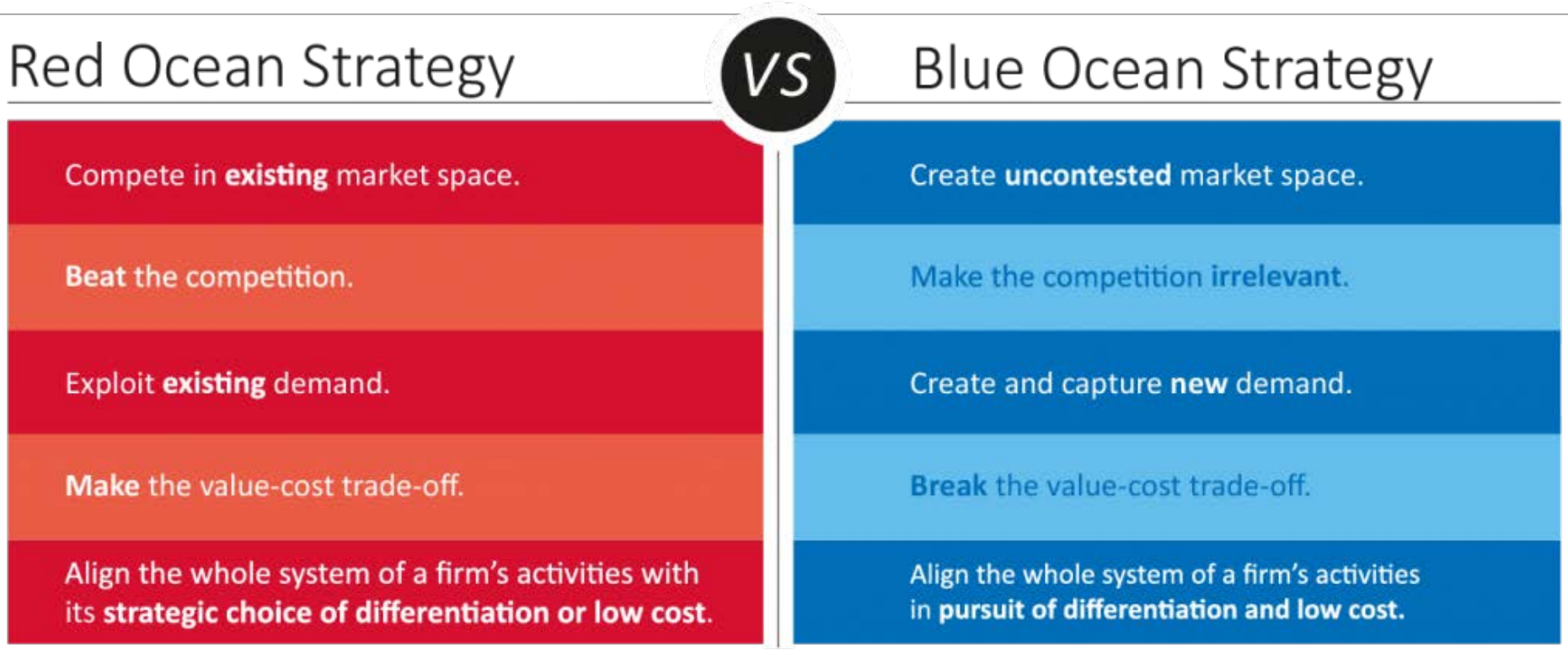
Value Innovation

- Product must deliver value to be useful (Levy, Chapter 1)
- Value chain (Porter), the chain of activities that allows an organization to provide value



https://www.slideshare.net/chetanpalta/tesla-value-chain-presentation

Blue Ocean Strategy



- Example: Airbnb

https://successfulculture.com/your-experience-is-your-blue-ocean-strategy/

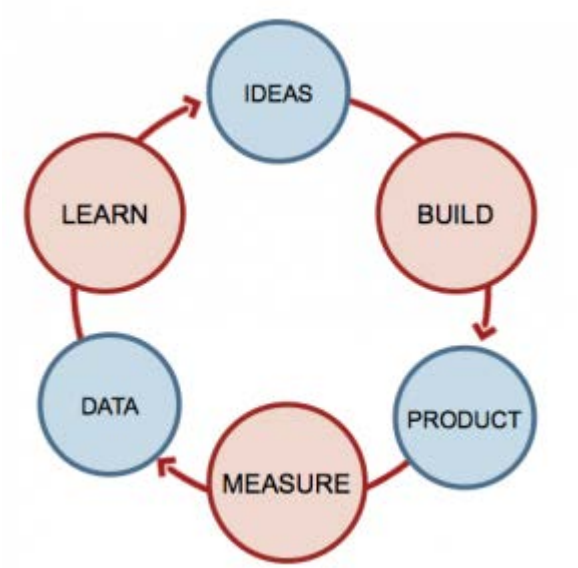
https://www.blueoceanstrategy.com/what-is-blue-ocean-strategy/

(W. Chan Kim and Renée Mauborgne)

Validated User Research

- Can use traditional means:
  - Field studies, focus groups, eye-tracking, personas, etc.
  - i.e., very similar to Elicitation, as we've covered
- Levy recommends:
  - Lean Startup: Confront your users early and often
    - Build-measure-learn feedback loop

http://thestartupmag.com/lean-start-up/



  - Minimum Viable Product: create the minimum product you can release
    - This is very similar to an agile way to thinking

UX Research vs. RE Elicitation

Requirements Elicitation

- Documentation
- Data Sampling
- Interviews
- Surveys/Questionnaires
- Focus groups
- Prototyping
- Participant Observation
- Think aloud protocol
- Models
- Scenarios
- Personas
- Social Media

UX Research

- System analytics
  - User flows
  - User analytics
- Surveys
- Tree Jacking
  - Tests navigation
- Eye tracking
- A/B Testing (Nunally & Farkas)
- Card sorting
- Customer feedback
- Landscape analysis
  - Look at competitors
- Usability heuristics
- Contextual inquiry
  - Observations
- Product testing and validation

Approaches to Prototyping

- Prototyping "A software prototype is a partial implementation constructed primarily to enable customers, users, or developers to learn more about a problem or its solution." [Davis 1990]
- "Prototyping is the process of building a working model of the system" [Agresti 1986]
- Approaches to prototyping (Easterbrook)
  - Presentation Prototypes
    - used for proof of concept; explaining design features; etc.
    - explain, demonstrate and inform – then throw away
  - Exploratory Prototypes
    - used to determine problems, elicit needs, clarify goals, compare design options
    - informal, unstructured and thrown away.
  - Breadboards or Experimental Prototypes
    - explore technical feasibility; test suitability of a technology
    - Typically no user/customer involvement
  - Evolutionary (e.g. "operational prototypes", "pilot systems"):
    - development seen as continuous process of adapting the system
    - "prototype" is an early deliverable, to be continually improved.

Types of Prototypes

- Paper
  - Advantages: fast, inexpensive, (can be fun), concrete
  - Disadvantages: unrealistic, distracting, awkward
- Digital
  - Advantages: fast, more realistic, flexible (can change)
  - Disadvantages: have to learn software, doesn't transition to code
- Web-Based
  - Advantages: can test on any device, technical foundation, shortcut
  - Disadvantages: hard to make (need to code), inhibits creativity

https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/

Book Outline

- Organizing the Content: Information Architecture and Application Structure
- Getting Around: Navigation, Signposts, and Wayfinding
- Organizing the Page: Layout of Page Elements
- Lists of Things
- Doing Things: Actions and Commands
- Showing Complex Data: Trees, Charts, and Other Information Graphics
- Getting Input from Users: Forms and Controls
- Using Social Media
- Going Mobile
- Making It Look Good: Visual Style and Aesthetics

Won't go through all patterns!

This would take a long time.

Will pick a few examples for each section for illustration.

The rest you should look at on your own and consider using in your assignments.

Information Patterns

1. Feature, Search, and Browse
2. News Stream
3. Picture Manager
4. Dashboard
5. Canvas Plus Palette
6. Wizard
7. Settings Editor
8. Alternative Views
9. Many Workspaces
10. Multi-Level Help

Navigation Patterns

1. Clear Entry Points
2. Menu Page
3. Pyramid
4. Modal Panel
5. Deep-linked State
6. Escape Hatch
7. Fat Menus
8. Sitemap Footer
9. Sign-in Tools
10. Sequence Map
11. Breadcrumbs
12. Annotated Scrollbar
13. Animated Transition

Layout Patterns

1. Visual Framework
2. Center Stage
3. Grid of Equals
4. Titled Sections
5. Module Tabs
6. Accordion
7. Collapsible Panels
8. Movable Panels
9. Right/Left Alignment
10. Diagonal Balance
11. Responsive Disclosure
12. Responsive Enabling
13. Liquid Layout

Actions Patterns

1. Button Groups
2. Hover Tools
3. Action Panel
4. Prominent “Done” Button
5. Smart Menu Items
6. Preview
7. Progress Indicator
8. Cancelability
9. Multi-Level Undo
10. Command History
11. Macros

Information Graphic Patterns

1. Overview Plus Detail
2. Datatips
3. Data Spotlight
4. Dynamic Queries
5. Data Brushing
6. Local Zooming
7. Sortable Table
8. Radial Table
9. Multi-Y Graph
10. Small Multiples
11. Treemap

Patterns

- Similar choices for entering numbers and dates
- See the textbook for a list of options, consider the pros and cons
- 1. Forgiving Format
- 2. Structured Format
- 3. Fill-in-the-Blanks
- 4. Input Hints
- 5. Input Prompt
- 6. Password Strength Meter
- 7. Autocompletion
- 8. Dropdown Chooser
- 9. List Builder
- 10. Good Defaults
- 11. Same-Page Error Messages

Social Media Patterns

1. Editorial Mix
2. Personal Voices
3. Repost and Comment
4. Conversation Starters
5. Inverted Nano-pyramid
6. Timing Strategy
7. Specialized Streams
8. Social Links
9. Sharing Widget
10. News Box
11. Content Leaderboard
12. Recent Chatter

Mobile Patterns

1. Vertical Stack
2. Filmstrip
3. Touch Tools
4. Bottom Navigation
5. Thumbnail-and-Text List
6. Infinite List
7. Generous Borders
8. Text Clear Button
9. Loading Indicators
10. Richly Connected Apps
11. Streamlined Branding

Style Patterns

1. Deep Background
2. Few Hues, Many Values
3. Corner Treatments
4. Borders That Echo Fonts
5. Hairlines
6. Contrasting Font Weights
7. Skins and Themes

- Can check these out in the book

Usability Factors

- Fit for use (functionality): The system can support the tasks the user has in real life
- 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?
- Ease of use (User friendliness): combination of all factors but the first

(Laueson, Chapter 1)

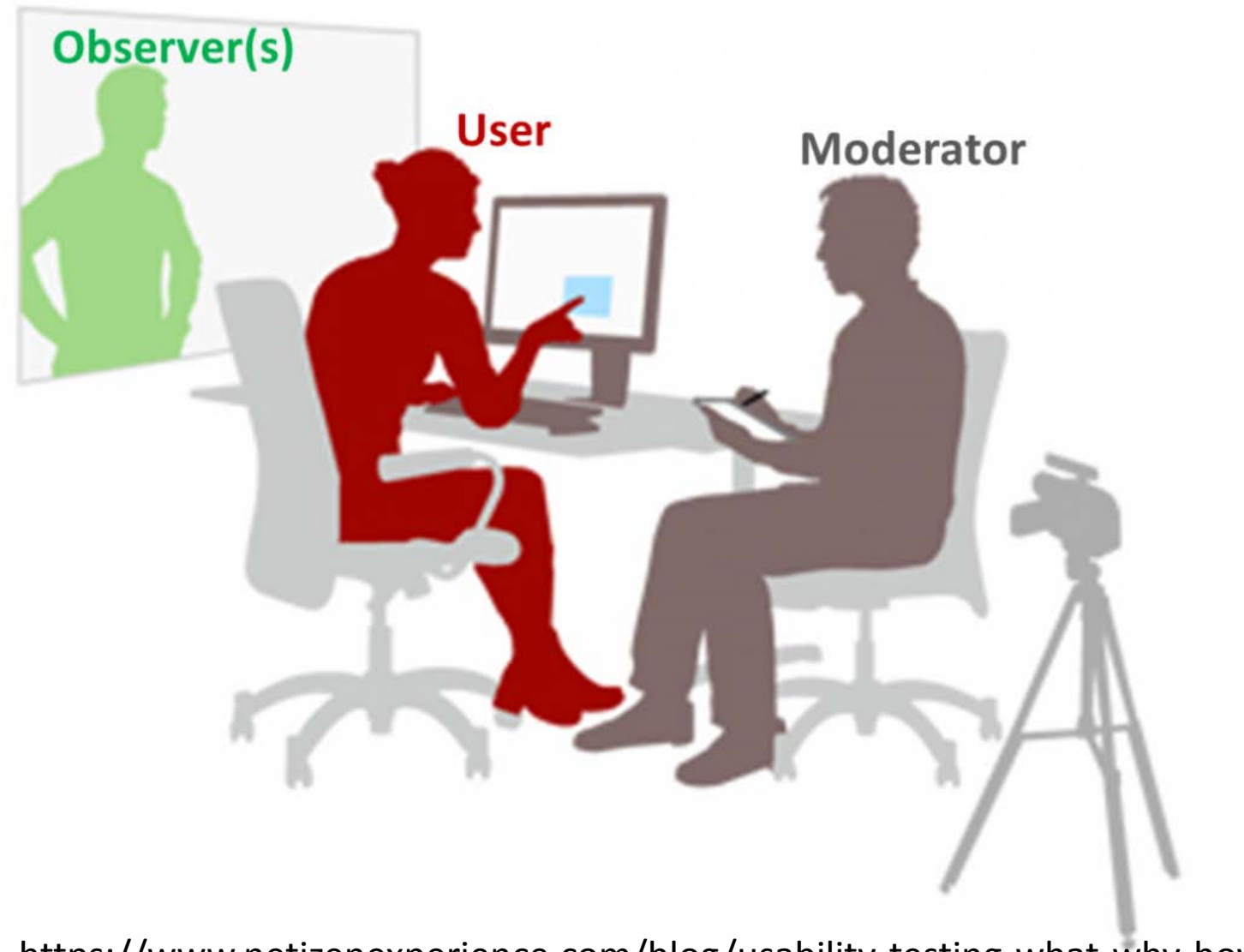
Program Defect Types

- 1. **Missing functionality or bug:** function the user wants is not there, or crashes, doesn't work
- 2. **Task Failure:** user cannot figure out how to do a task in a fixed amount of time
- 3. **Annoying:** user can do a task in a reasonable amount of time, but is annoyed
- 4. **Medium problem:** user succeeds in task after a long time
- 5. **Minor problem:** user succeeds in task after a short amount of time (but it still took longer than expected)

- **Critical problems**

(Laueson, Chapter 1)

Usability Tests



https://www.netizenexperience.com/blog/usability-testing-what-why-how/

Usability Test Tasks

- Find tasks that the users should attempt in order to test usability
- Something they would do in a real situation
- Hint: One usability task per use case makes sense
- Usually involves several steps, clicks, screens
- Examples:
  - Wrong: Find a bus connection around 11 pm from route 6, stop 12 to route 8, stop 23.
  - Better: You are planning to go to a party tomorrow at 20 Brickwood Street, Brighton. You would like not driving home to 55 Westbank Terrance, Richmond. IS there any public transportation that can help you? How late? And what would it cost?
  - Make it realistic
- Some examples for the online marking tool uploaded to GUL

(Laueson, Chapter 1)

Metrics vs. Usability Factors

Factor	Time	Problem Count	Keystroke	Polls
Fit for use	Hard to measure all tasks	Hard to measure all tasks	Hard to measure all tasks	Well-suited
Ease of learning	Well-suited	Well-suited	Not suited	Some indication
Task efficiency	Well-suited (experienced users)	Indirectly	Some indication	Some indication
Ease of remembering	Difficult, need to run twice	Difficult, need to run twice	Not suited	Some indication
Subjective satisfaction	Not suited	Some indication	Not suited	Some indication
Understandability	Not suited	Some indication	Not suited	Some indication

(Laueson, Chapter 1)

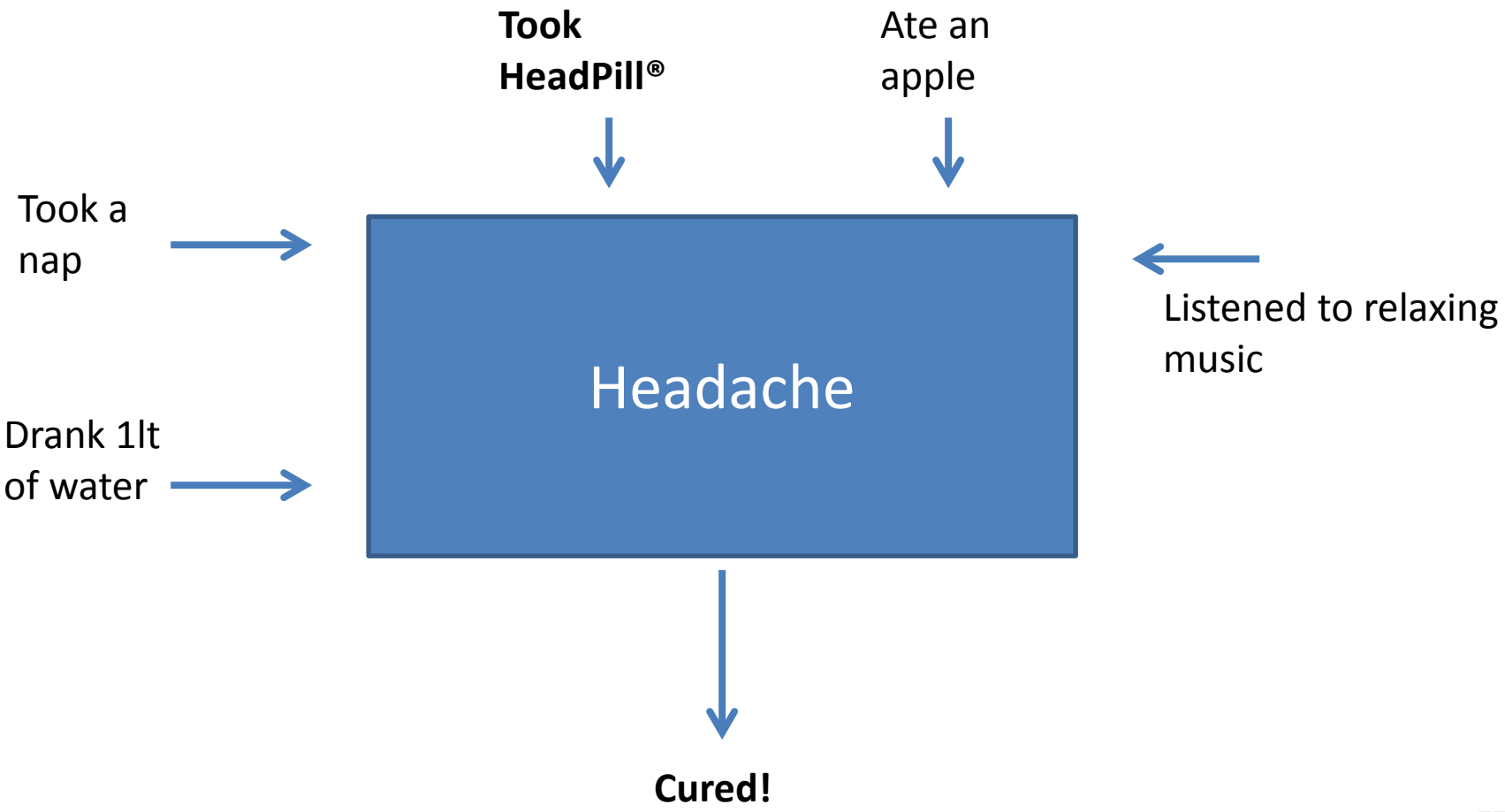
The System Usability Scale (SUS)

1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need to support of a technical person to be able to use this system.
5	I found that the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
8	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

Usability Experiments

- When you have to compare two candidate interfaces.
  - E.g. an old and a new one.
- There is a need for generalization for a larger class of systems (e.g. interested in statistical significance)
- Larger and representative sample sizes.
- More rigorous control of other influencing factors.

Experiments - Example



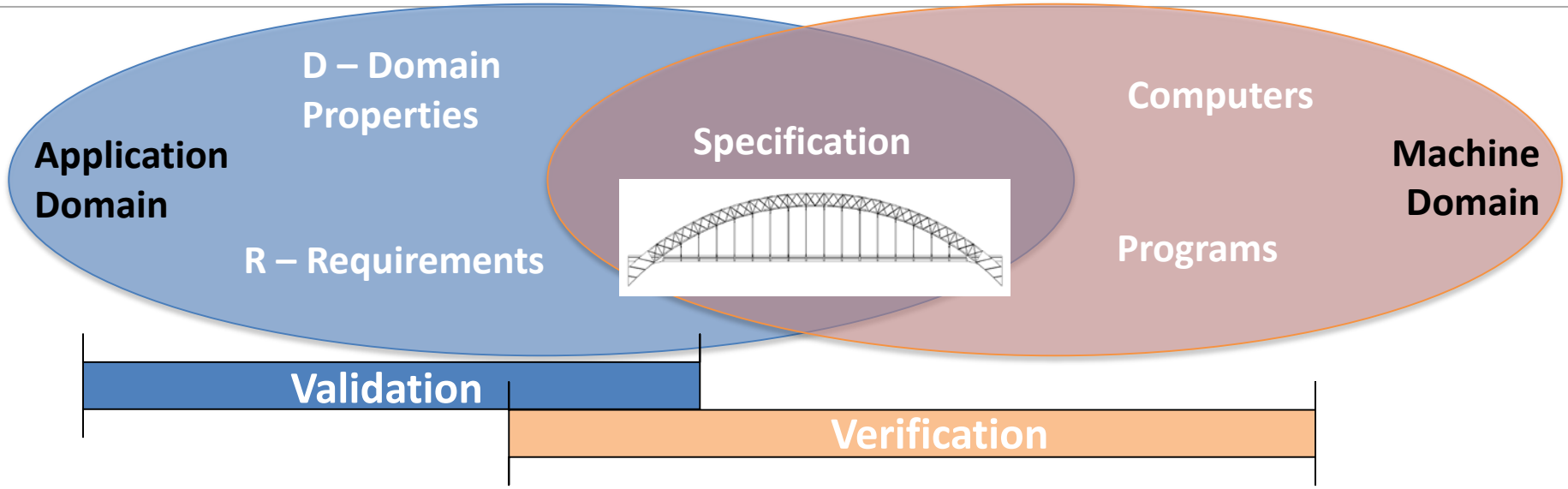
Experiments - Terminology

- **Independent Variable.**
  - The influencing factor you are tweaking in order to see if it has an effect to the **dependent variable**.
  - It is up to you to tweak it, that's why it is "independent".
  - **Example:** whether or not to take the pill.
- **Dependent Variable.**
  - The response/effect you are wondering if it should be attributed to the independent variable.
  - **Example:** whether headache goes away.
- **Confounding factors.**
  - Other influencing factors that, if left uncontrolled, they won't let you prove the connection.
  - **Example:** eating the apples, drinking the water, sleeping, etc.
- **Experimental Condition.**
  - The state of the independent variable at a given phase of the experiment.

Different, Same, Matched Participant Design

Design	Advantages	Disadvantages
Different	No order effects	Many subjects & individual differences a problem
Same	Few individuals, no individual differences	Counter-balancing needed because of ordering effects
Matched	Same as different participants but individual differences reduced	Cannot be sure of perfect matching on all differences

Validation & Verification

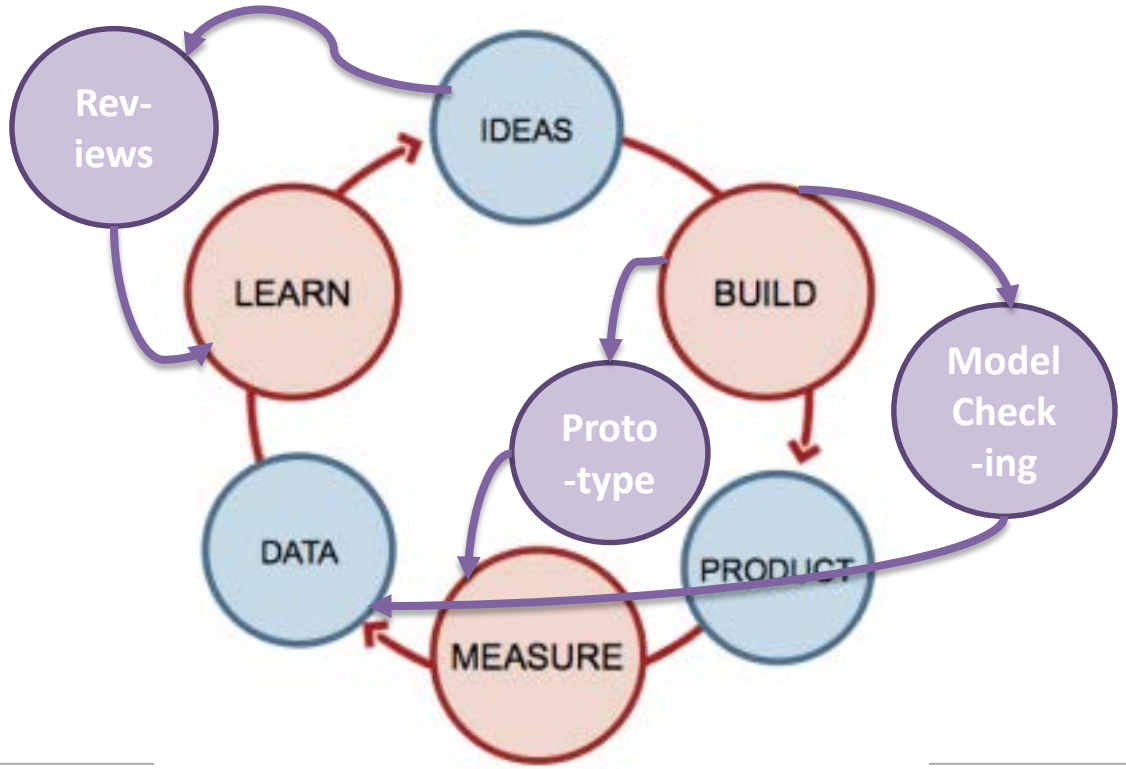


- Validation: Are we building the right system?
  - Does our solution solve the real problem?
  - Did we account for the most important stakeholder needs?
- Verification: Are we building the system right?
  - Does our design meet the requirements?
  - Does the system do what we say it would do?
  - Are our requirements representations consistent?

(Zave & Jackson, Easterbrook)

Shortcuts in the inquiry cycle

- Emulate the solution:
  - Prototypes (see Lecture 8)
  - Model checking
  - Reviews, walkthroughs, inspections



(Easterbrook & Campbell)

Exam Format

- Multiple Choice Questions (~10%)
- Short Answer Questions (~20%)
- Domain Analysis (~80%)
- You will be given a domain description (like the ones we've seen)
  - Then asked to create some artefacts from the following list:
    - SRS requirements
    - User stories
    - Requirements templates (template given)
    - Context diagram
    - Use case diagram
    - Goal model (legend given)
    - Customer journey map
    - Personas
    - Scenarios (template given)
    - Prioritization
    - User interface design & Pattern application (book allowed)
    - User testing design

Example Multiple Choice Question 1

- 1.1 Which of the following are correct statements concerning validation and verification. There may be more than one correct answer. For full marks, list all the correct answers. Each correct answer listed is +1 point, each incorrect answer is -1 point.
  a) Verification asks: are we building the right system?
  b) Verification asks: are we building the system right?
  c) Validation asks: are we building the right system?
  d) Validation asks: are we building the system right?

- You write:
  – 1.1. b, c

Example Multiple Choice Question 2

- 1.2 Which of the following are characteristics of UX design patterns. There many be more than one correct answer.
  a) General, not concrete
  b) Valid across different platforms and systems
  c) Products, not processes
  d) Requirements, not suggestions

- You write:
  – 1.2. b, c

Example Short Answer Question 1

- 2.1 What is the application domain and the machine domain according to Zave & Jackson? Define each. You can draw a picture, but must give a written explanation of each concept. (4 marks)

- You write:
  - 2.1 Application Domain: the world, where people, organizations and problems live.
  - Machine" (Software, Computer, Program) Domain: the software + hardware that solves some problem, meets some need

Example Short Answer Question 2

- 2.2 What is the difference between functional requirements and non-functional requirements (qualities)? Define both.

- You write:
  - 2.2. Functional requirements are things the system must do
  - Nonfunctional requirements are qualities the product must have

Questions?