

CHALMERS



UNIVERSITY OF GOTHENBURG

DIT045/DAT355

Requirements and User Experience

Lecture 12: Course Review, Exam Info & Practice Quiz

Jennifer Horkoff
jenho@chalmers.se

Lectures

1. Course Intro

RE Lectures

2. Creativity, Scoping, Modeling

3. Modeling

4. Concepts, Writing Requirements, Requirements Quality

5. Elicitation, Scenarios, Personas

UX Lectures

7. Intro to UX & Prototyping

8. UX Design Patterns

9. UX Design Patterns & Mobile Patterns

10. Usability & Usability Testing

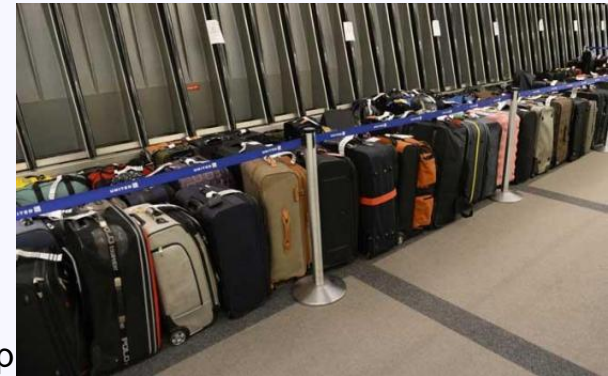
11. Usability Experiments & Videos

12. Guest Lecture

13. Verification and Validation, ~~RE & UX Research~~

RE Example: 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 million (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: prospect 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/>



(Laplane)

System Failures

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



Even this solution involves RE methods and tools!

Why is RE important?

- Requirements form the basis for:
 - Project planning
 - Risk management
 - Trade off
 - Acceptance testing
 - Change control
- Avoid system failures (see previous examples)
- Organizational memory
- Liability

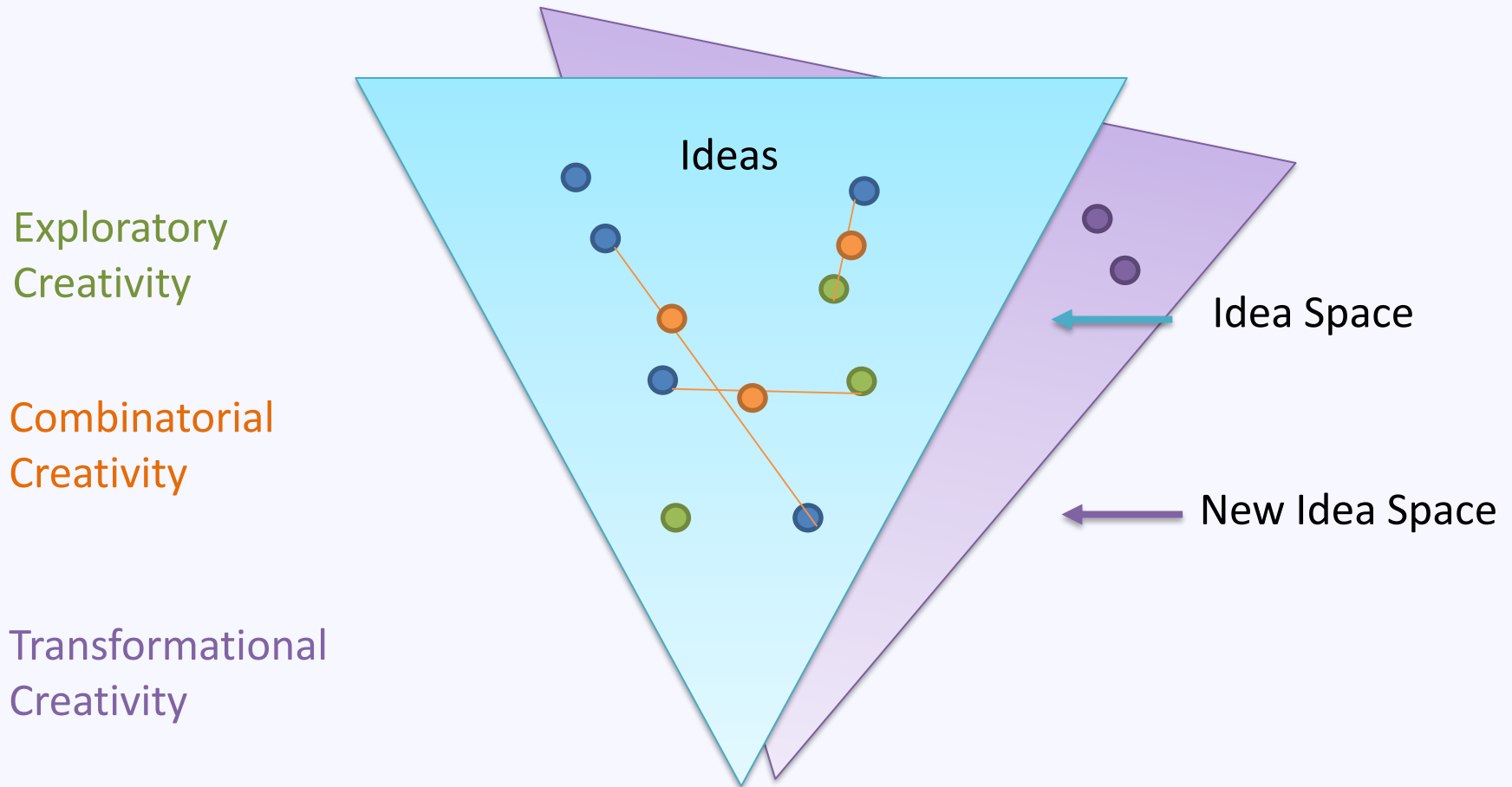
(Dick et al.)

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.

Exploratory, Combinatorial, Transformational Creativity

- (Presentation idea via N. Maiden)

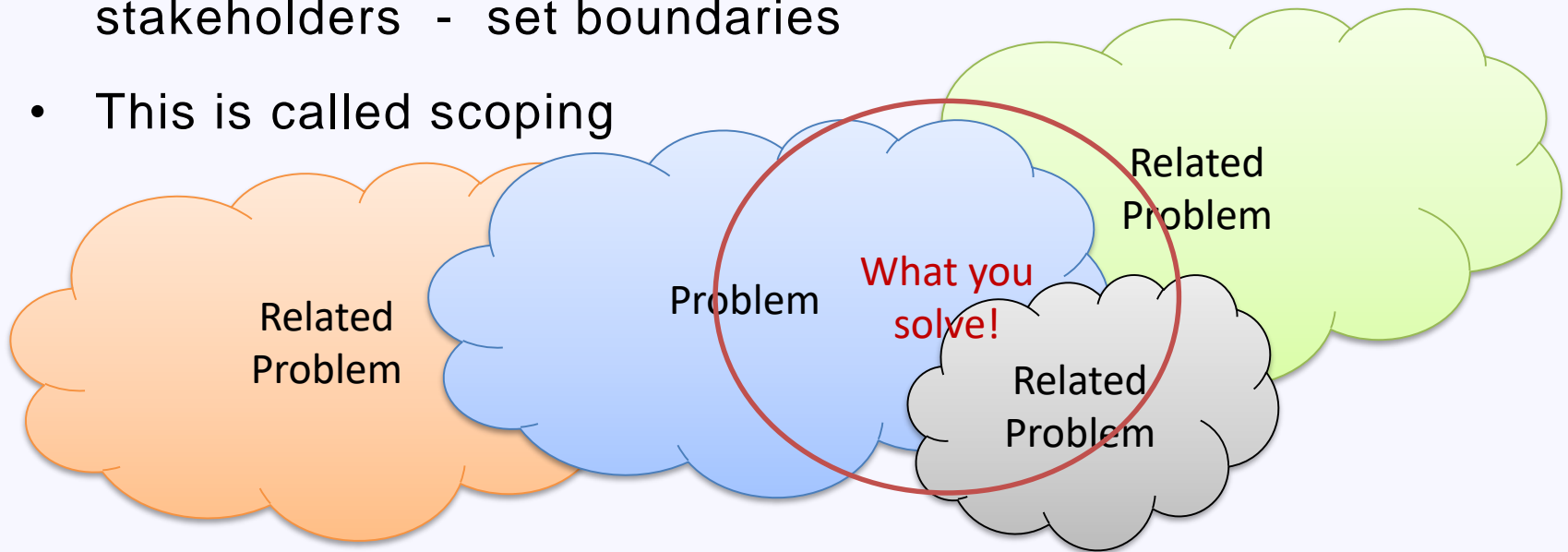


Creativity Activities

- (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/>

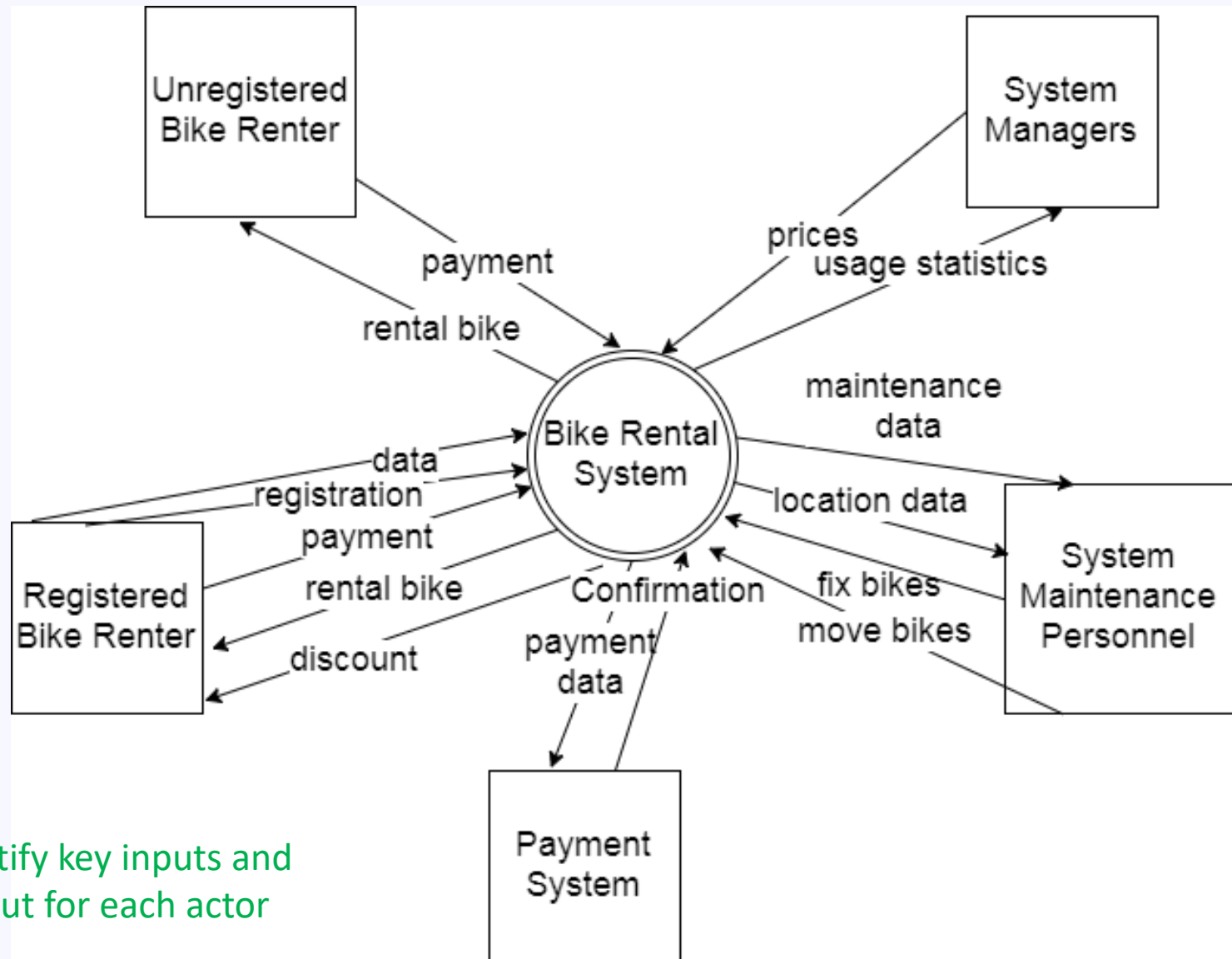
RE Scoping

- Problems are usually boundless
 - Can always find more related stakeholders with more problems
- Need to focus on a reasonable part of the problem and set of stakeholders - set boundaries
- This is called scoping

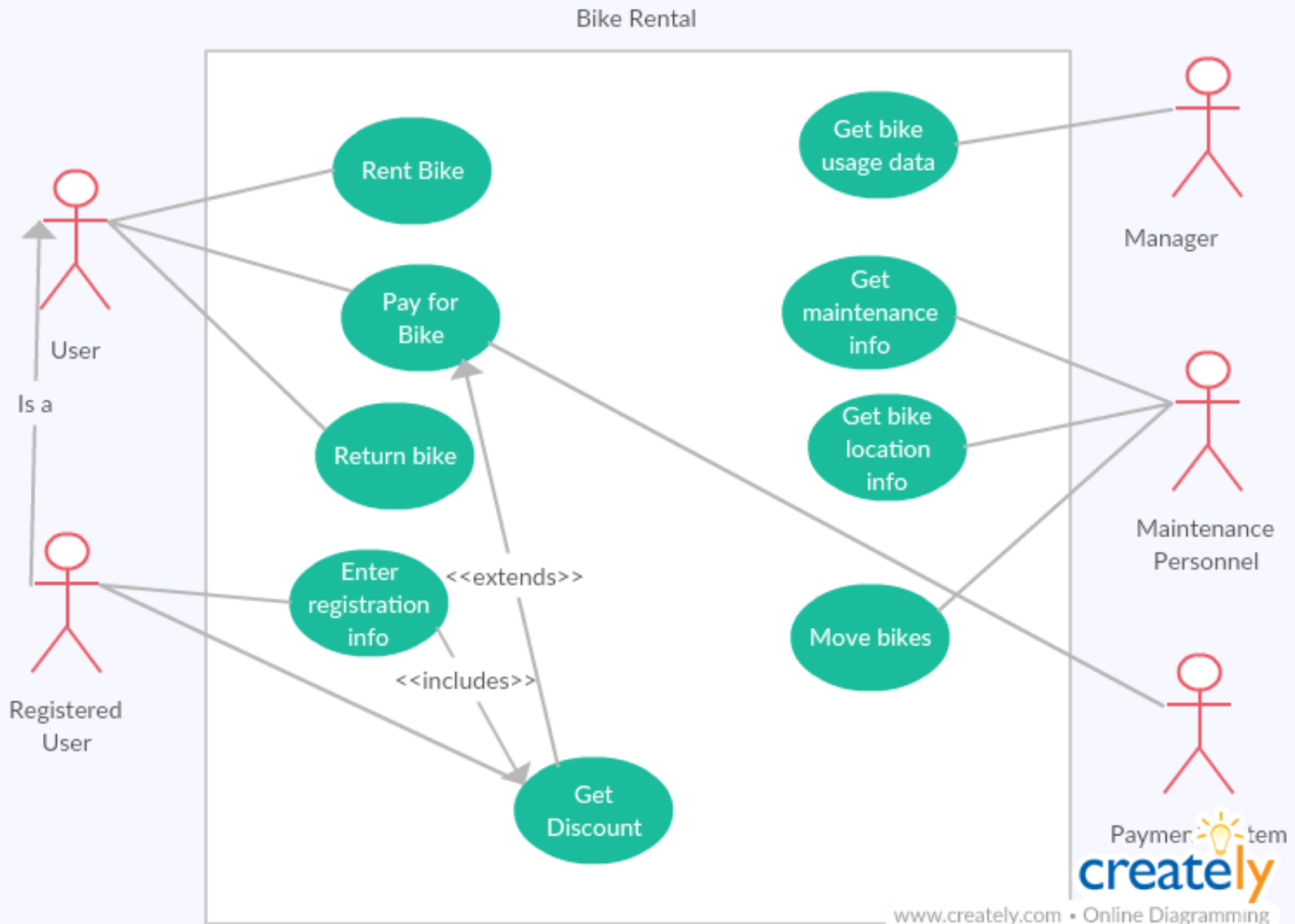


- Further material: Easterbrook/Mylopoulos scoping slides

Context Example: Bike Rental

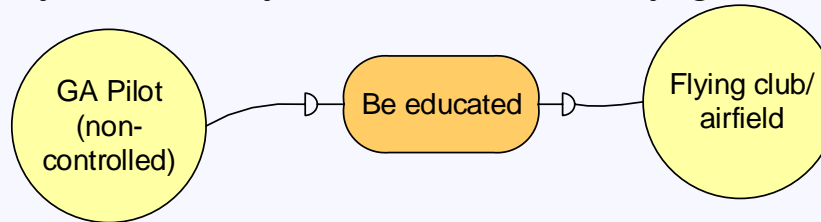


Bike Rental

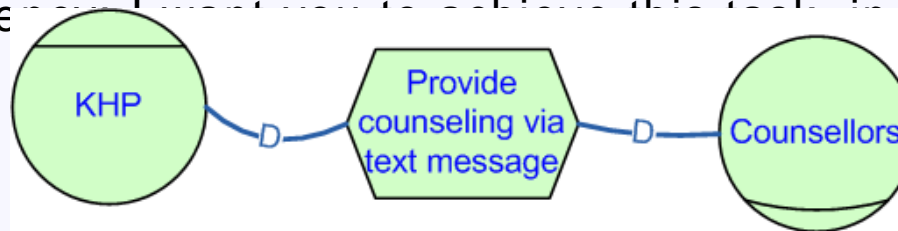


i* Strategic Dependencies

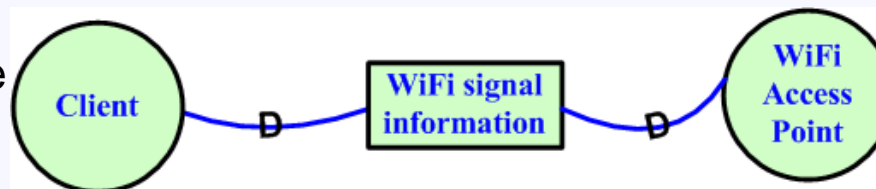
- Goal Dependency: I want you to achieve my goal, I don't care how



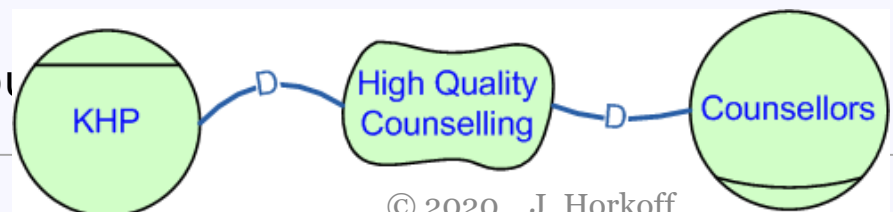
- Task Dependency: I want you to achieve this task in an agreed upon way



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

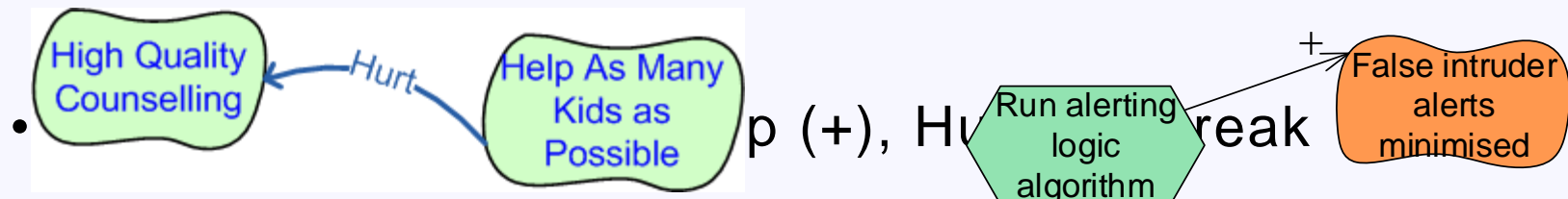
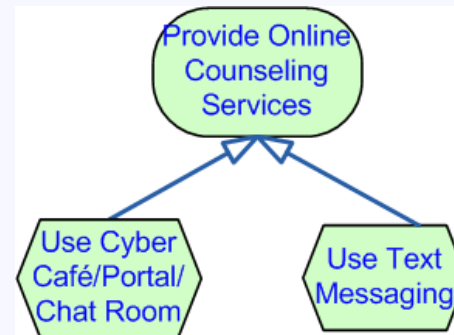
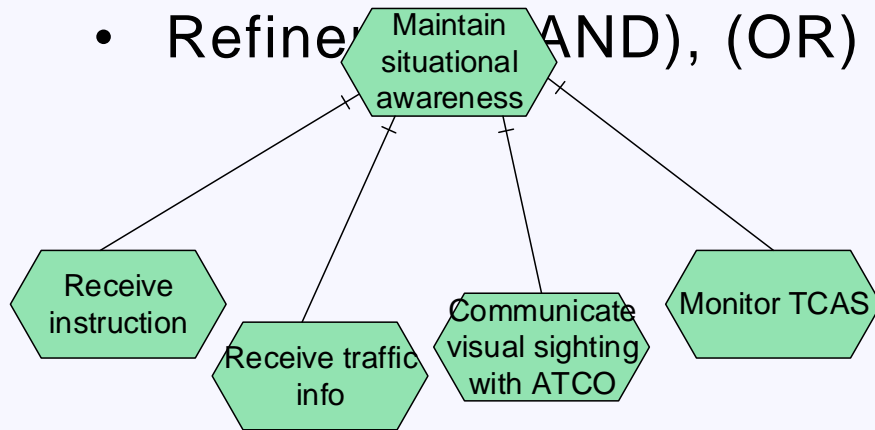
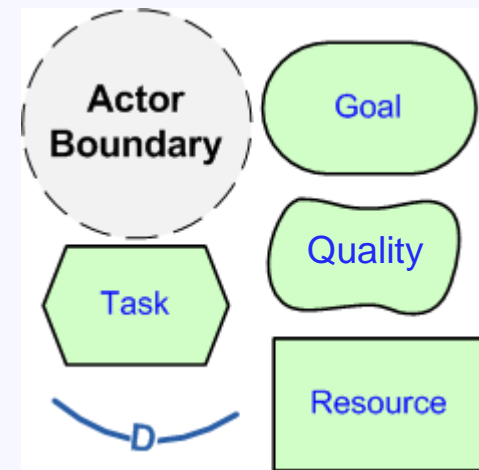


- Quality Dependency: I want you

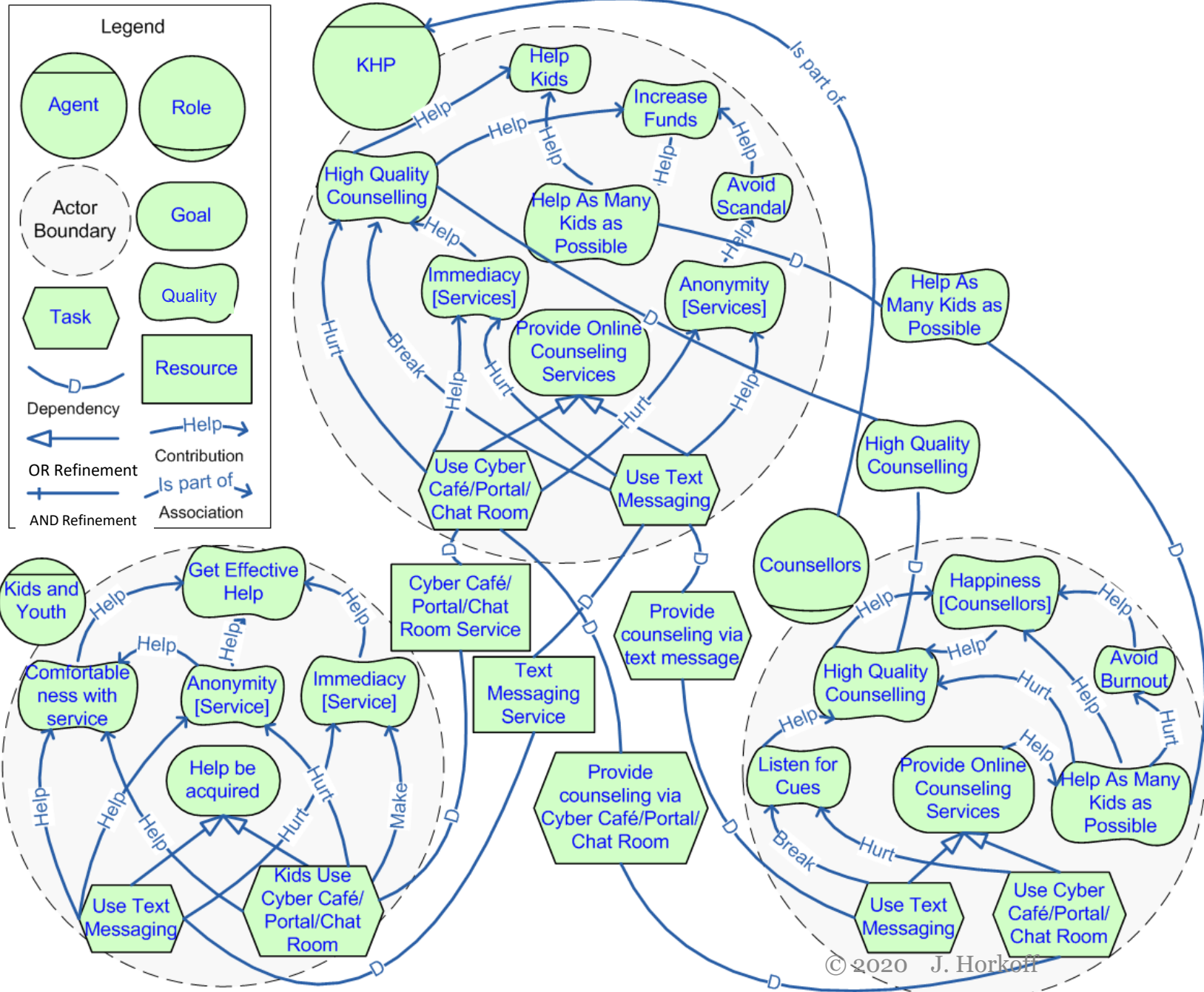


i* Strategic Rationale (SR) Diagrams

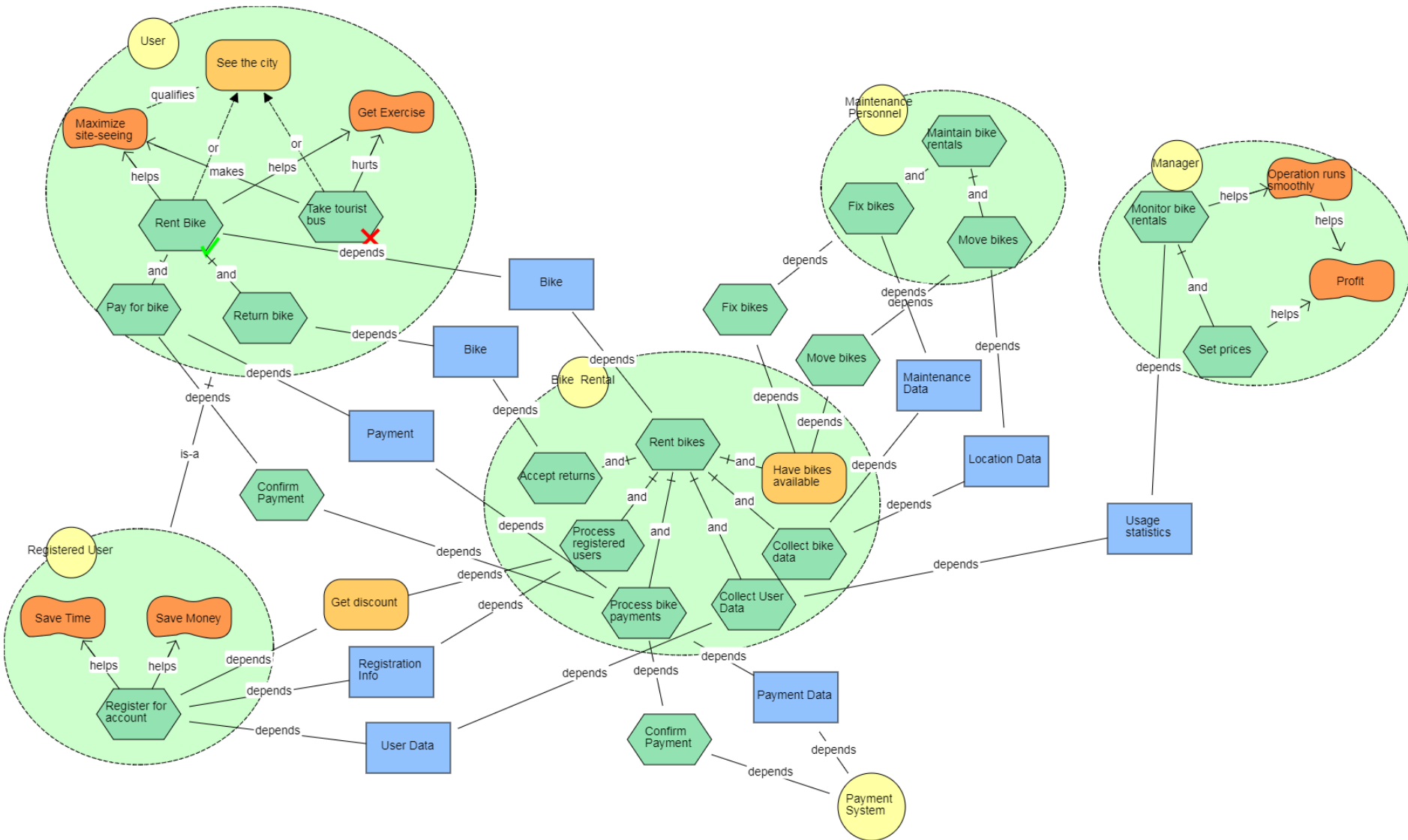
- Actor boundaries
- Goals, Qualities, Tasks, Resources
- Dependencies (as before)
- Refinement (AND), (OR)



SR Example: Kids Help Phone



Bike Rental: All



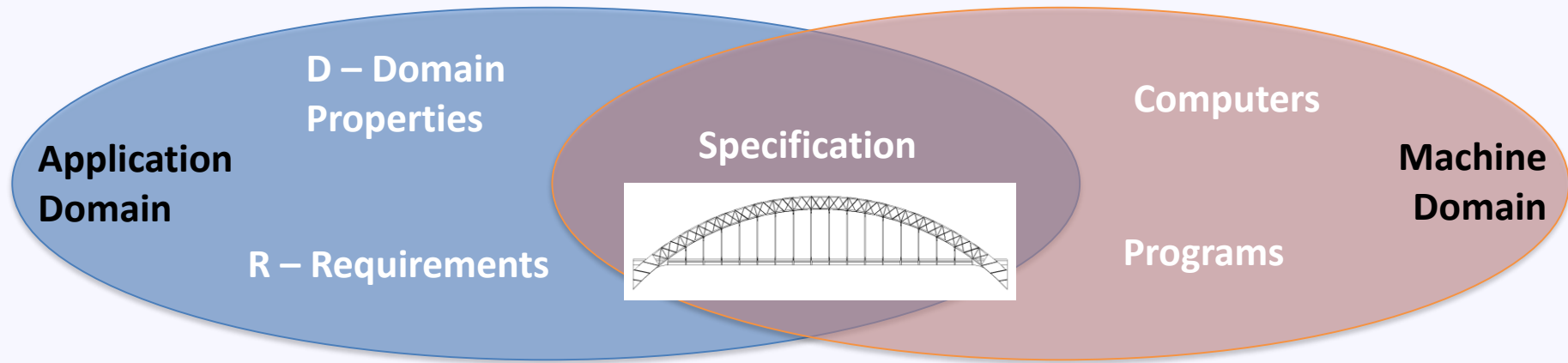
Approximate Model Mappings

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

Bike Rental

	Acquisition				Riding		Return		Feedba ck
	Bike Select ion	Select Option s	Payme nt	Confir mation & Info	Bike Riding	Remin ders	Return Bike	Confir mation	Feedba ck
Bike Rack	★								
Rental Display		★							
Payment Panel			★				★		
User Device				★		★		★	
Bike					★				
Email									★

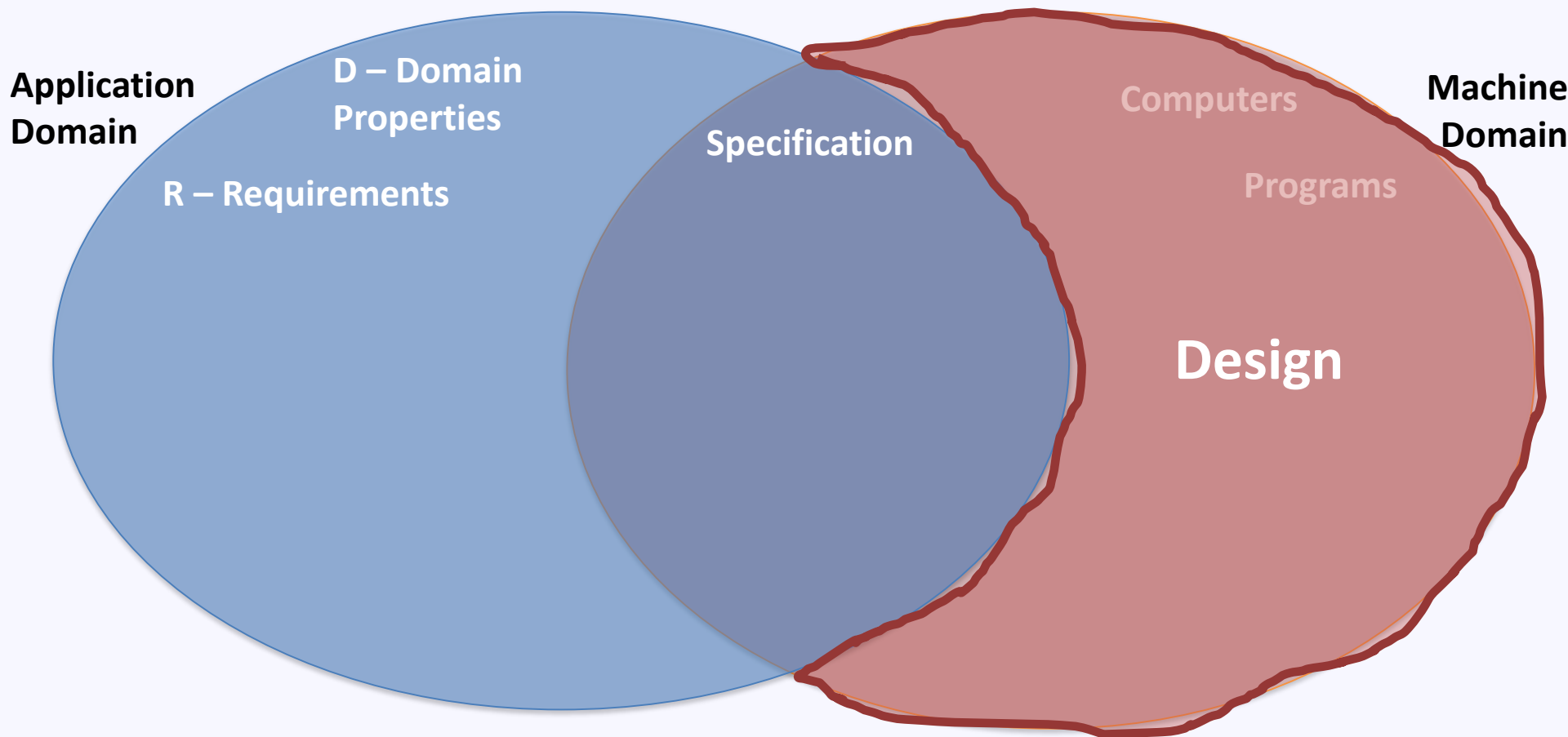
What is Requirements Engineering (RE)? (6)



- 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
- $S, D \models R$ (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” (Robertson²)
 - 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” (Robertson²)
- They are a requirement that doesn't come from user needs

How to Capture/Document Requirements

- A few ways, we'll cover some of the main options
- Using Natural language
 - Traditional SRS form
 - User stories
 - Templates
 - Structured Natural Language
- Using models

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 constraints 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>)

User Story vs. SRS Requirements Quality

User Story Quality

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

SRS Requirements Quality

- Necessary
- Implementation Free
- Unambiguous
- Consistent
- Complete
- Singular
- Feasible
- Traceable
- Verifiable
- Affordable
- Bounded

(a selection of) Elicitation Techniques

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

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)

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

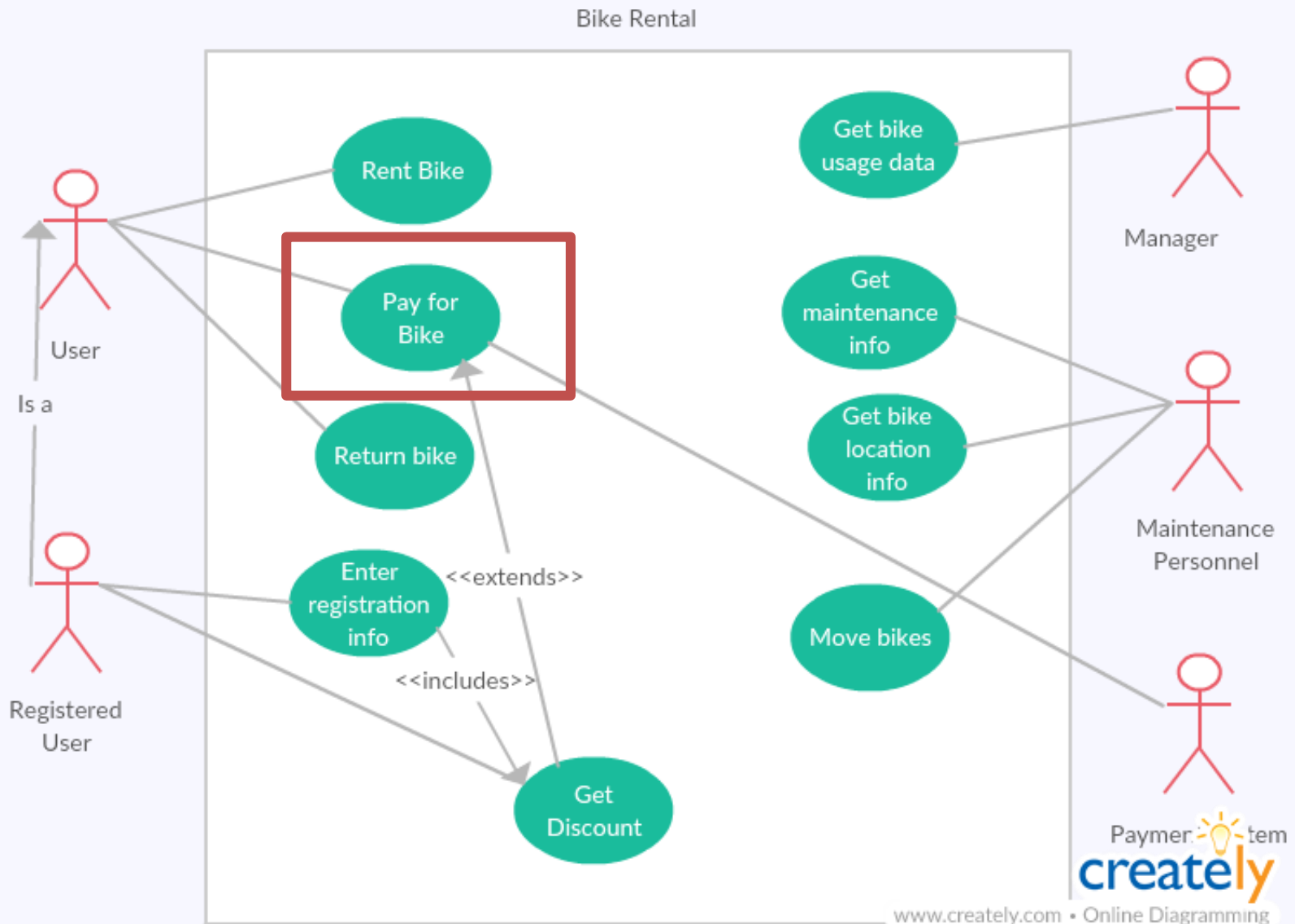
Father of three children, Grandfather of one child

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.

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

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

UX Example 5

- Your bad day...
 - Wake up to full sunlight, clock says 3:43 am... you have 10 minutes to get to school!
 - Turn on the coffee maker... no coffee ☹️
 - Drive to school... car needs gas! Gas station pump takes credit cards, but won't take yours. Must wait in line at the cashier, takes forever!
 - Driving detour due to accident...
 - Late for school! And no coffee..
- What does this have to do with UX? It's just bad luck?
(Garrett)

UX Example 5 (cont.)

- What does this have to do with UX? It's just bad luck?
 - Accident: the driver took his eyes off the road to turn the radio down, it was impossible to identify the volume button from touch alone
 - Register: the line moved slowly because the cash register was complex and confusing. The clerk would make a mistake and have to start all over again
 - Pump: You turned the card the wrong way to swipe it, but nothing on the pump indicated this, and you didn't notice
 - Coffee: you didn't push the power button all the way! No lights to tell you whether or not it has been turned on.
 - Clock: your cat stepped on the clock in the middle of the night and reset the time, a slightly different button configuration would have made the alarm cat proof!

(Garrett)

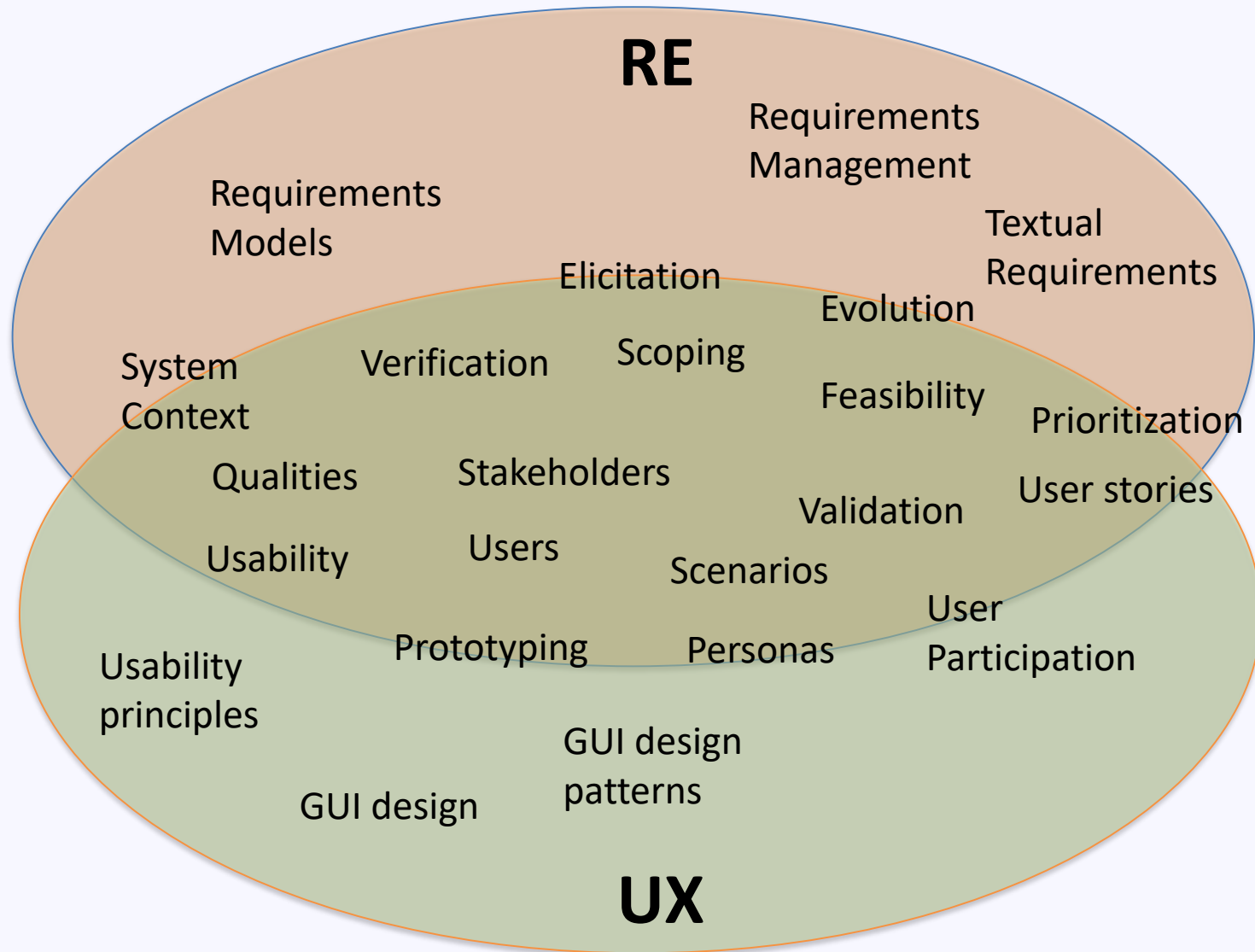
Why is UX Important?

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

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.

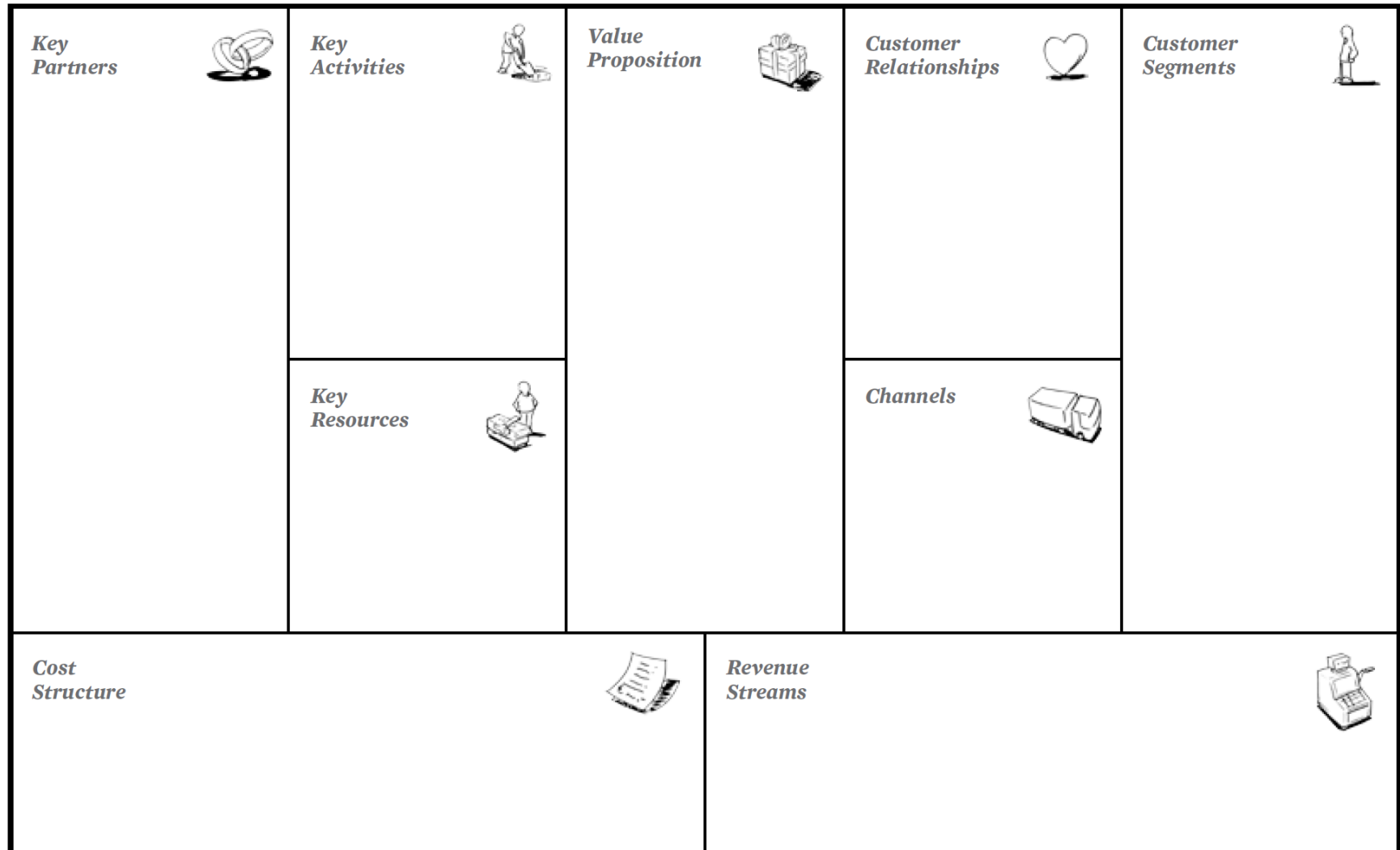
How do UX and RE relate?



Business Strategy

The Business Model Canvas

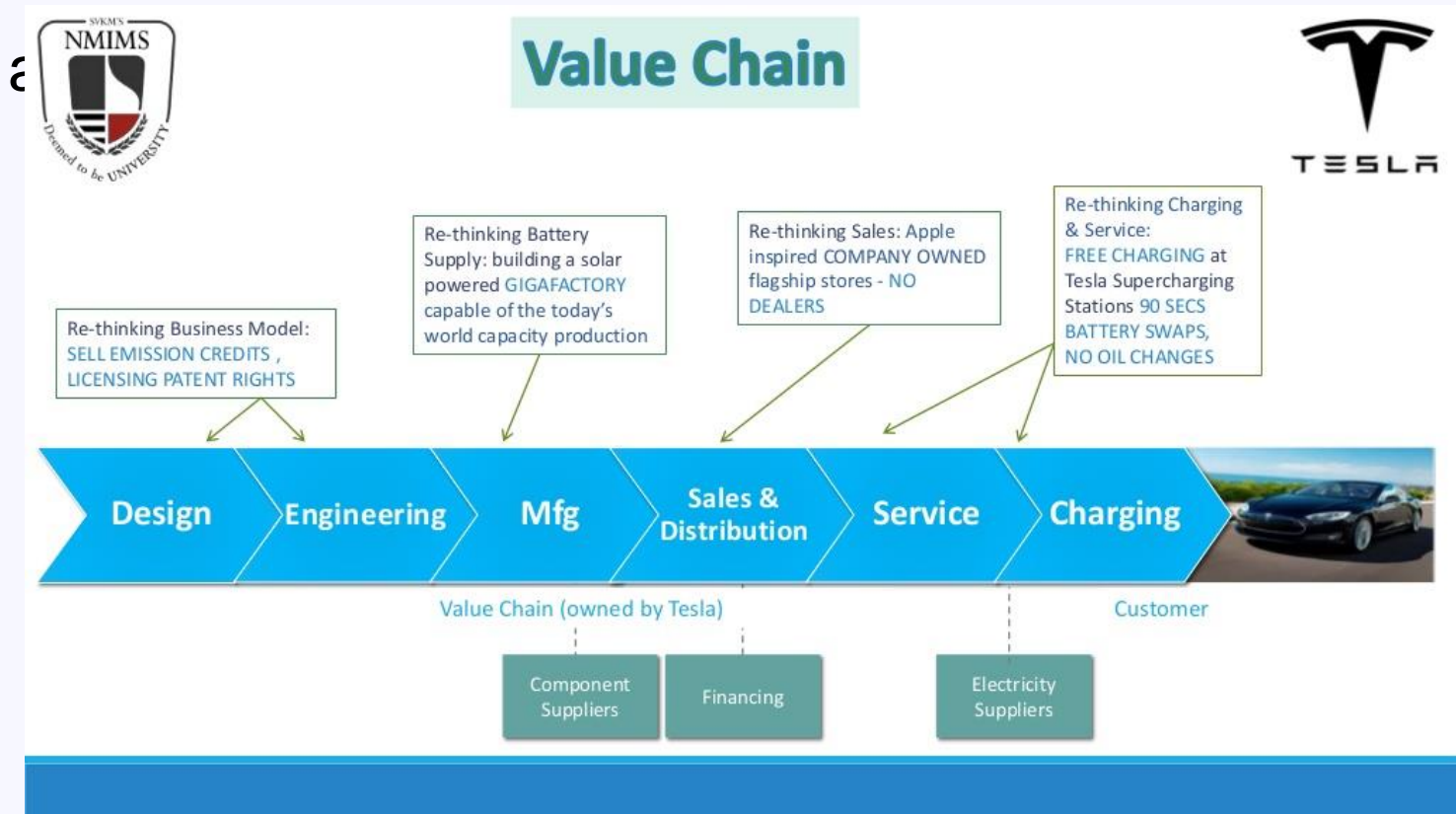
(Osterwalder, 2009)



Value Innovation

(Levy, Chapter 1)

- Product must deliver value to be useful
- Value chain (Porter), the chain of activities that



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

Blue Ocean Strategy

Red Ocean Strategy

VS

Blue Ocean Strategy

Compete in **existing** market space.

Beat the competition.

Exploit **existing** demand.

Make the value-cost trade-off.

Align the whole system of a firm's activities with its **strategic choice of differentiation or low cost.**

Create **uncontested** market space.

Make the competition **irrelevant.**

Create and capture **new** demand.

Break the value-cost trade-off.

Align the whole system of a firm's activities in **pursuit of differentiation and low cost.**

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

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

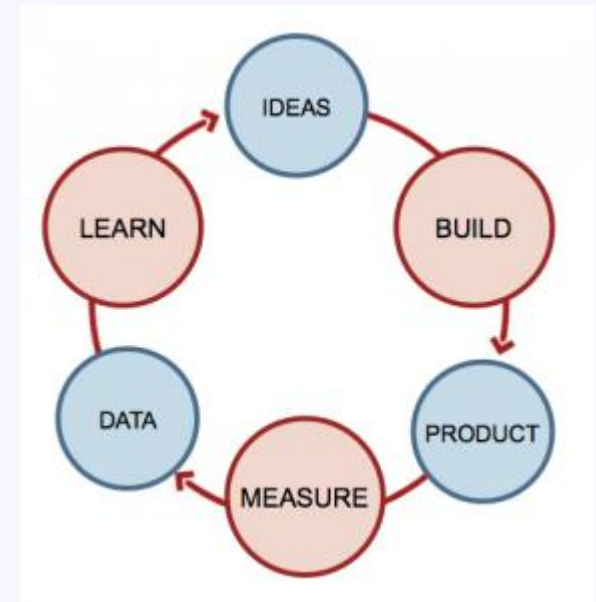
- **Example: Airbnb**

(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

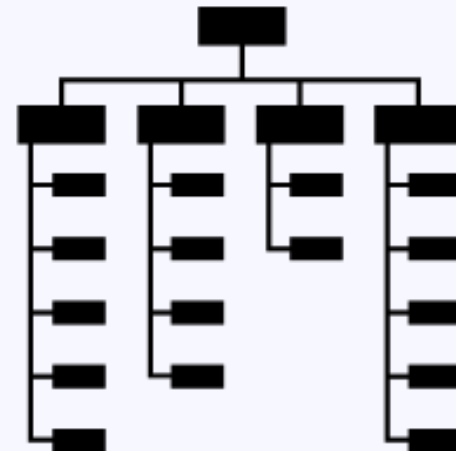
Types of Prototypes

- 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

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

Sitemap/UI Planning

- High-level view of the whole site/system
- Hierarchical, shows navigation
- Part of planning before prototyping
- Could include screens in different platforms
 - Mobile
 - Website
 - Kiosk
 - Other?



<https://uxdesign.cc/how-to-create-a-ux-sitemap-a-simple-guideline-8786c16f85c1>

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

2nd Edition

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

3rd Edition

1. -- (same)
2. **Mobile Direct Access**
3. Streams and Feeds
4. Media Browser
5. --
6. --
7. --
8. --
9. --
10. --
11. Help Systems
12. **Tags**

Navigation Patterns

2nd Edition

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

3rd Edition

1. --
2. --
3. --
4. --
5. Deep Links
6. --
7. --
8. --
9. --
10. Progress Indicator
11. --
12. --
13. --

Navigation Patterns

2nd Edition

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

3rd Edition

1. --
2. --
3. --
4. --
5. Deep Links
6. --
7. --
8. --
9. --
10. Progress Indicator
11. --
12. --
13. --

List of Things

2nd Edition

1. Two-Panel Selector
2. One-Window Drilldown
3. List Inlay
4. Thumbnail Grid
5. Carousel
6. Row striping
7. Pagination
8. Jump to Item
9. Alpha/Numeric Scroller
10. Cascading Lists
11. Tree Table
12. New-Item Row

3rd Edition

1. Two-Panel Selector or Split View
2. One-Window Drilldown
3. List Inlay
4. Cards
5. Thumbnail Grid
6. Carousel
7. Out
8. Pagination
9. Jump to Item
10. Alpha/Numeric Scroller
11. Out
12. Out
13. New-Item Row

Actions Patterns

2nd Edition

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

3rd Edition

1. --
2. Hover or Pop-Up Tools
3. --
4. ...or assumed Next Step
5. --
6. --
7. Spinners and Loading indicators
8. --
9. --
10. --
11. --

Information Graphic Patterns

2nd Edition

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

3rd Edition

1. Out
2. --
3. --
4. --
5. --
6. Out
7. Out
8. Out
9. --
10. --
11. The power of Data Visualization

Input Patterns

2nd Edition

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

3rd Edition

1. --
2. --
3. --
4. --
5. --
6. --
7. --
8. --
9. --
10. ... and smart prefills
11. 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

Style Patterns

2nd Edition

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

3rd Edition

- Described, but are no longer patterns

Mobile Patterns

2nd Edition

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

3rd Edition

1. --
2. --
3. --
4. --
5. Collection and Cards
6. --
7. --
8. Out
9. Loading or progress indicators
10. --
11. Out

Anti-Patterns

- Novel notions
- Needless complexity
- Metaphor mismatch
 - Control mismatch
 - Icon mismatch
 - Gesture mismatch
 - Mental model mismatch
 - Idiot boxes
 - Chart Junk
 - Oceans of buttons
 - Square peg, round hole

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)

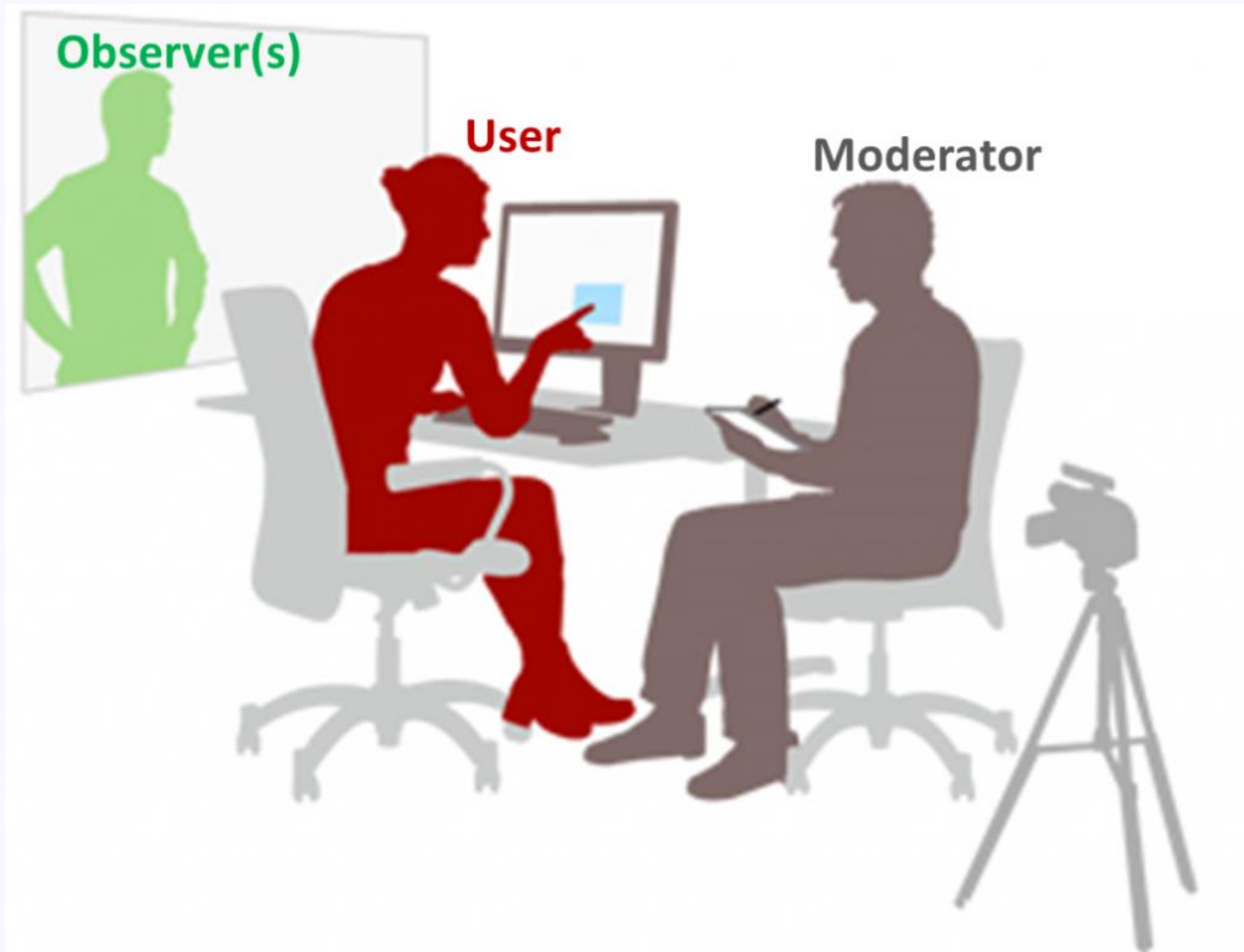
Program Defect Types

- 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)

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.

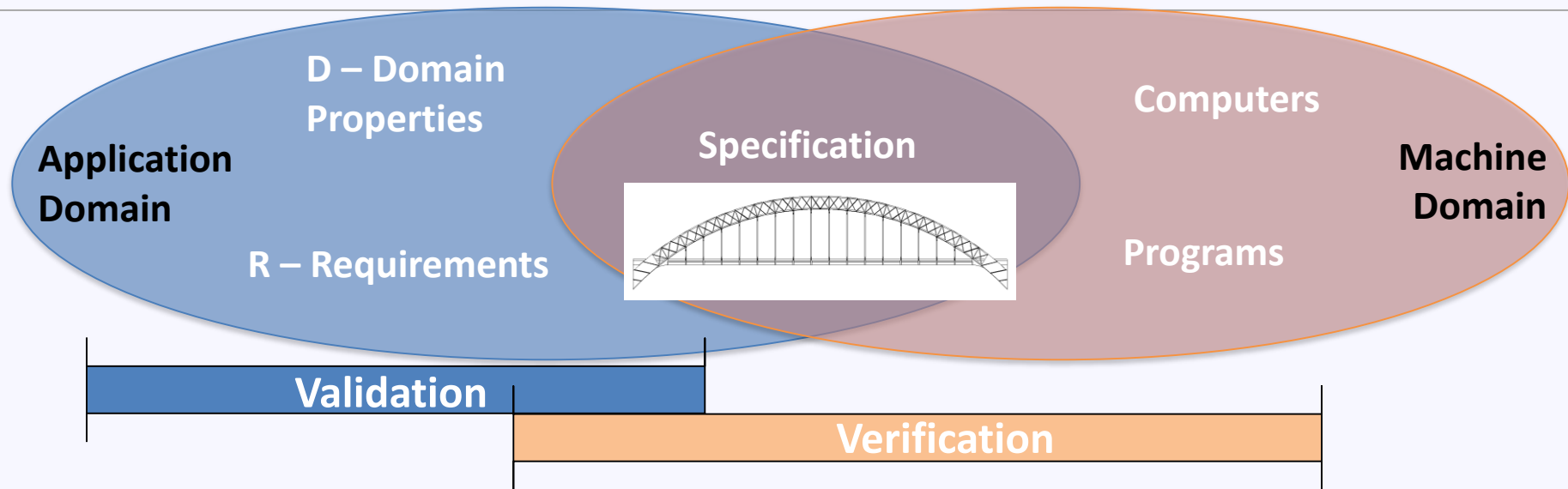
Experiments - Terminology

- **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.

Vision Videos for RE

- **VisionVideo**
 - **Video about a vision; here: about a software-based, future product**
 - *Video = format or representation*
 - *Vision = content*
- **Reasons to create Vision Videos (expectations)**
 - **Fast, easy, and low-effort to watch** – *as opposed to text or models*
 - **Short and concrete** – *not abstract and vague*
 - **Comprehensible** *without training, introduction or learning a notation*
- ... but
 - **it takes time to design and to create**
 - **it requires creativity to do right**
 - **many requirements/software people are hesitant to use**
- **An example Vision Video can tell more than text ...**

Validation & Verification

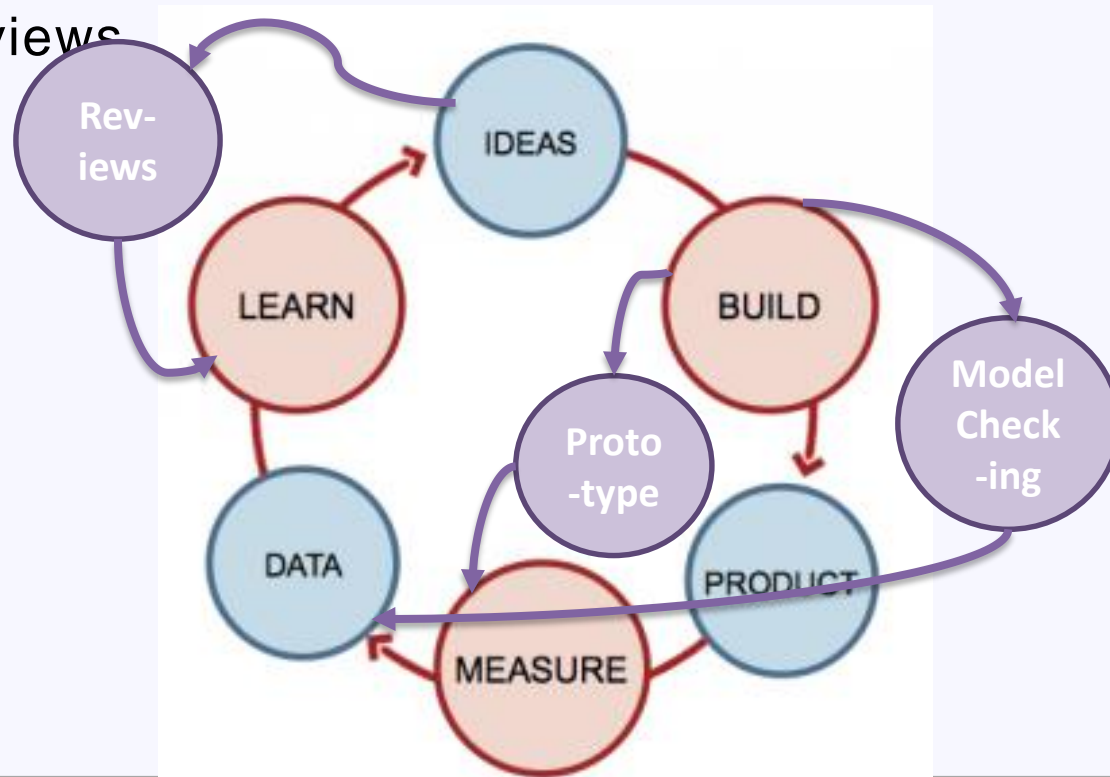


- Validation: Are we building the right system?
 - Does our solution solve the real problem?
 - Did we account for the most important stakeholder needs?
- Verification: Are we building the system right?
 - Does our design meet the requirements?
 - Does the system do what we say it would do?
 - Are our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

(Zave & Jackson, Easterbrook)

Shortcuts in the inquiry cycle

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



(Easterbrook & Campbell)

Exam in COVID Times

- A special canvas course is created by the student office
- If you have registered for the exam, you are added here
 - This may be done the week before, if you are not in this page ~4 or so days before the exam, email the student office (I'm not allowed to add you)
- At 8:30 am I post a PDF of the exam for you
- Do not communicate with each other
- Open book, access any book or slides, past assignments, exams.
- Stop writing at 12:30
- You get an extra 30 minutes to create a PDF
 - Scan, convert, could be pictures
- Submit ONE PDF by 13:00
 - Make sure the PDF is not corrupt (e.g., open in a few platforms)
- We have 15 working days to certify the grade in Ladok

Exam Format

- Multiple Choice Questions (~10%)
 - Short Answer Questions (~20%)
 - Domain Analysis (~70%)
 - You will be given a domain description (like the ones we've seen)
 - Then asked to create some artefacts from the following list (6 questions)
 - Context diagram
 - Use case diagram
 - Goal model (legend given)
 - Customer journey map
 - Personas
 - Scenarios (template given)
 - User testing targets
 - User testing tasks
 - **SRS requirements, User stories**
 - **User interface design & Pattern application**
- You will get 4/8 of these, randomly selected
- These are always there

Exam in COVID Times: Cheating

- During my last Canvas/COVID exam, I saw too many similar solutions.
- Solution from colleagues: I will make 3 exams of equal length and approximately equal difficulty.
- You will each get a random and equal selection from this exam, with the final makeup the same as the previous slide
 - Many questions will be the same, but the case may change, ordering, etc.
- You will be given a PDF for your student name, e.g., horkoje
 - Do this exam
- How do we know the exams are equal?
 - Each year I have to make 3 exams for this course, and you can take the exam in January, April or August.
 - I work to make these exams as equal as possible.
 - Your random selection will be as equal as these exams are normally
 - Probably even more equal as I'm making them all at the same time
 - You aren't getting 1 whole exam out of 3 exams, but a semi-random selection of the exam adding up to 1/3 (so there are many possible exam instances ~20?)

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 may 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

Live Exam

- Thursday Dec 17th
- 13-15
- Link in Zoom
- I'll pick an old exam and go through it question by question
- Will put recording in Canvas

Questions?
