CHALMERS |  UNIVERSITY OF GOTHENBURG

DIT045 H17 Requirements and User Experience

2. Requirements & Concepts

Jennifer Horkoff

What is Requirements Engineering (RE)? (1)

- Several complementary definitions:
  - "RE is the branch of **software engineeri**ng concerned with real-world goals for, functions of, and constraints on **software systems**. It is also concerned with the relationship of these factors to precise specifications of **software** behavior, and to their evolution over time and to their evolution over time and across **software families**." (Zave)
- Can be rewritten to be more general:
  - "RE is the branch of **engineering** concerned with the real-world goals for, functions of, and constraints on, **systems**. It is also concerned with the relationship of these factors to precise specifications of **system** behavior and to their evolution over time and across families of related **systems**." (Laplante)

- We focus on RE from a software perspective, but the methods and tools can apply to any system.
- But...

What is Requirements Engineering (RE)? (2)

- What make software unique?
  - Arbitrary complexity
    - Not limited to physical space and resources
  - Instant distribution
    - No need for factories or manufacturing
  - Off-the-shelf components
    - Many available solutions, don't re-invent the wheel

- -> intensity of competition
- -> must reduce "time to market"

(Dick et al.)

What is Requirements Engineering (RE)? (3)

- But… "time to market" is not sufficient, the real goal is time to market with the "right product".
- Also called "fitness for purpose".

- "A vital part of the systems engineering process, requirements engineering first defines thee problem scope and then links all subsequent development information to it. Only in this way can one expect to control and direct product activity; managing the development of a solution that is both appropriate and cost effective."
  - Planning and control!

(Dick et al.)

- So… RE can be applied to all types of systems, but is particularly important for Software Systems.

What is Requirements Engineering (RE)? (4)

- 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)

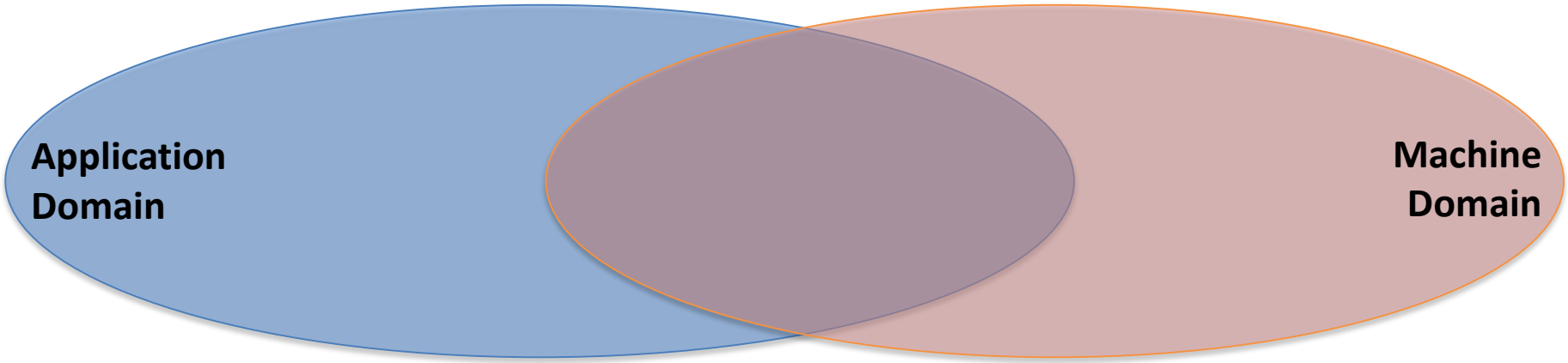
What is Requirements Engineering (RE)? (5)

- "Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies."

  (Easterbrook)

- RE is a bridge between technology and the world

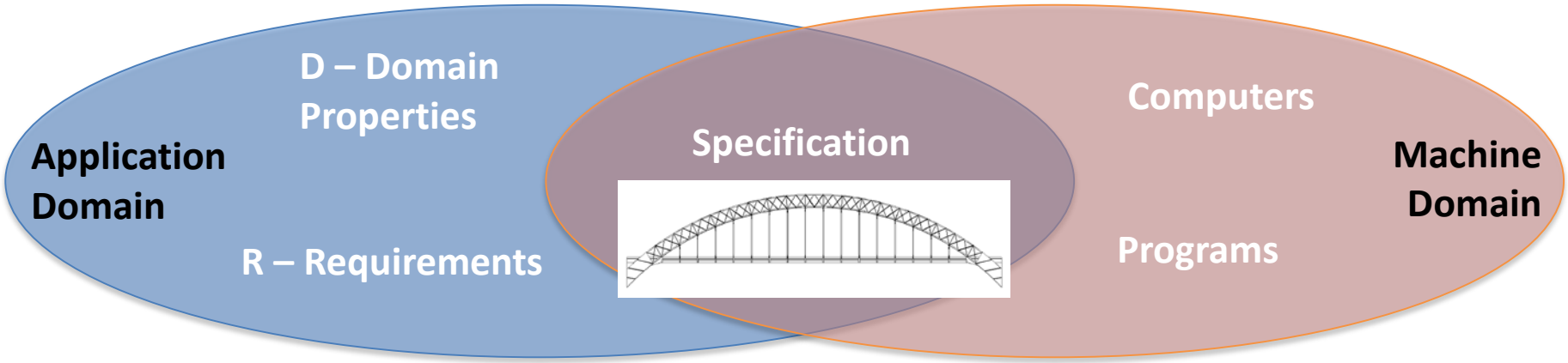
What is Requirements Engineering (RE)? (5)



- Application Domain: the world, where people, organizations and problems live.
  - Think of it as the domain where the (Software) application is applied
- "Machine" (Software, Computer, Program) Domain: the software + hardware that solves some problem, meets some need

Zave & Jackson, Easterbrook

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)

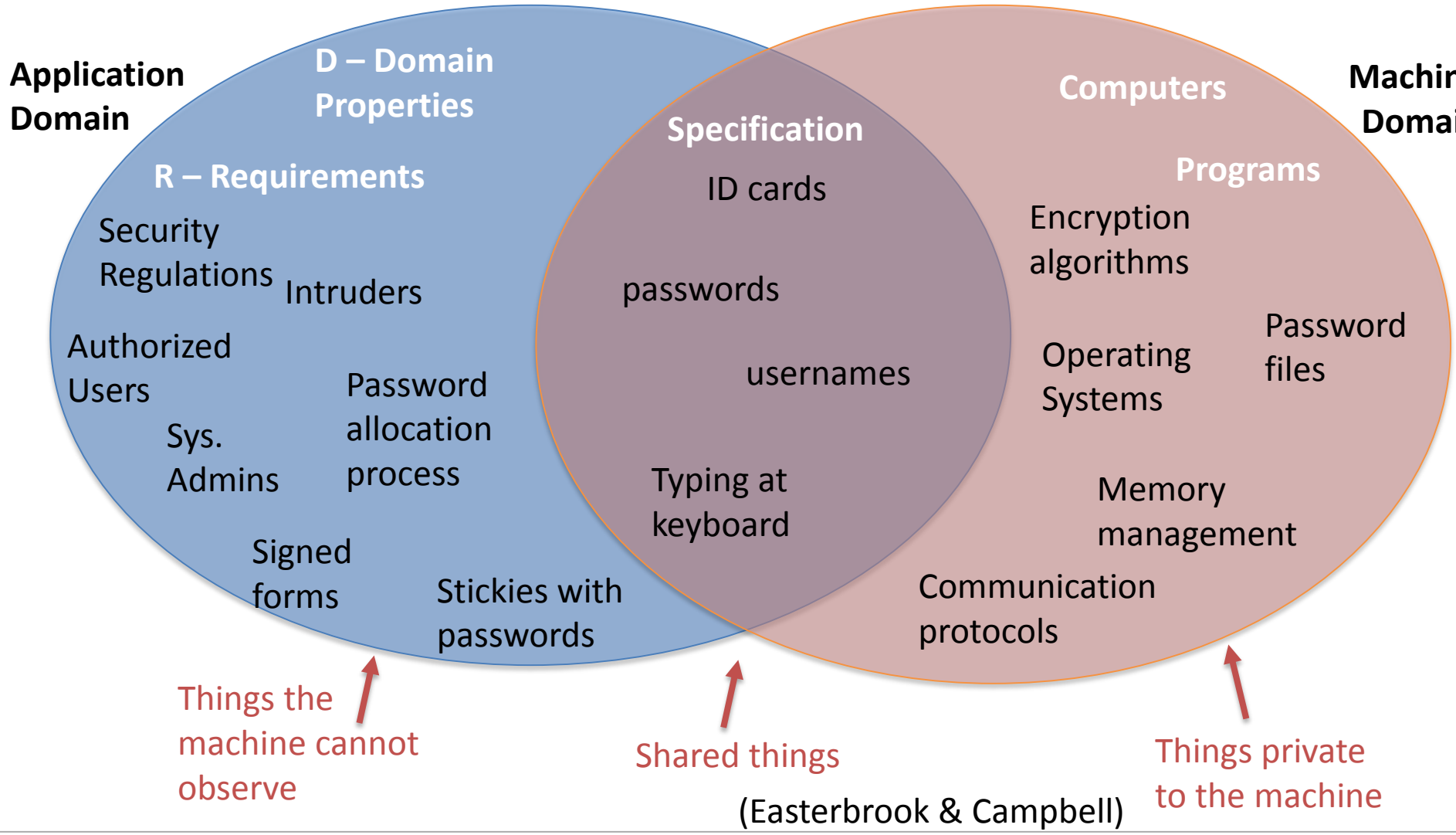
Recall RE Example 1: Network Security

- Example: Network Security
- Requirement R: "The network shall only be accessible by authorized personnel"
- Domain Properties D:
  - Authorized personnel have passwords
  - Passwords are never shared with non-authorized personnel
- Specification S:
  - Access to the network shall only be granted after the user types an authorized password

- Is the network secure?

(Easterbrook & Campbell)

Network Example



(Easterbrook & Campbell)

(Another) Running Example

- Bike share system
- Can rent pedal bikes in urban areas

- For such a system, what are the:
  - Requirements, goals, constraints, specification, assumptions...

What is a Requirement?

- "The effects that the client wishes to be brought about in the problem domain" (Bray)

- "A software capability needed by the user to solve a problem to achieve an objective" (Dorfman & Thayer)

- "A requirement is something the product must do or a quality it must have" (Robertson^2)

- " (1) A condition or capability needed by a user to solve a problem or achieve an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specificiation, or other formally imposed documents.
- (3) A documented representation of a condition or capability as in (1) or (2)." (IEEE Std 610.12-1990)

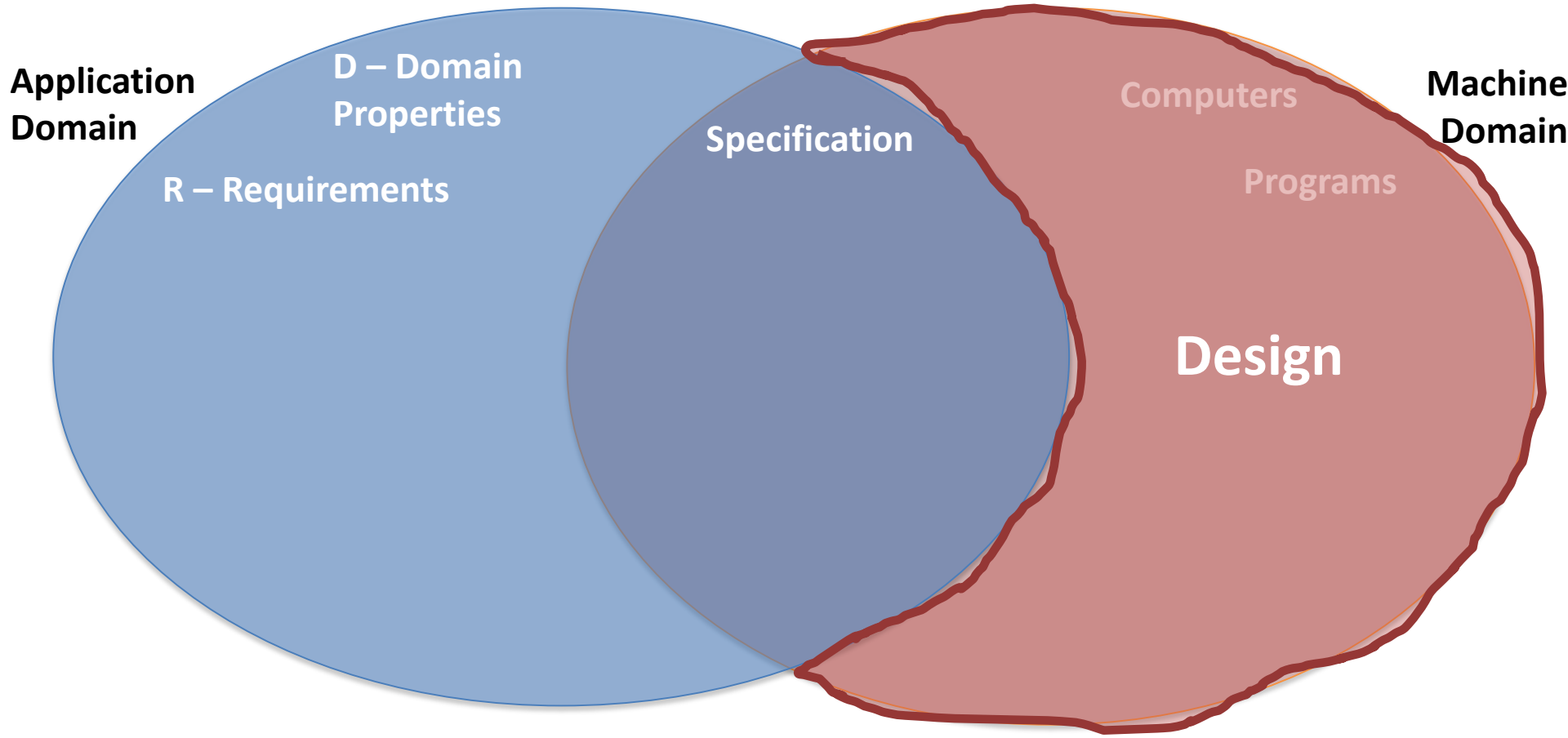
Requirements vs. Design

- Requirements describe ***what*** the system should do
  - The system shall allow users to rent a bicycle
  - The system should support payment by common credit and debit cards
  - The bike lock will not be released until the payment transaction has completed successfully

- Requirements do NOT describe ***how*** the system will do these things (this is Design)
  - The system will implement an event-based architecture to satisfy rental demands
  - The system will use a payment gateway which connects to a merchant account
  - The system will include a lock sensor to detect the presence of bikes

- Design: concerns the internal workings of the solution system. (Bray)

Requirements vs. Design



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

Requirements vs. Design

- "Requirements are expressed in a technically neutral way so as to avoid influencing the design of the solution" (Robertson^2)
- Why is the distinction between requirements and design important?
  - Design information in requirements become constraints for developers
    - Did this come from a user need or constraint?
    - Or did the person writing the requirement just think it should be done this way (in which case it can be changed)
  - Allow designers/architects/developers as much freedom as possible
    - To allow them to make the best (or a good) solution
  - While still making sure user needs are met

User vs. Stakeholder

- Users: people who directly use the system (any function of the system)
  - (Not developers or installers)

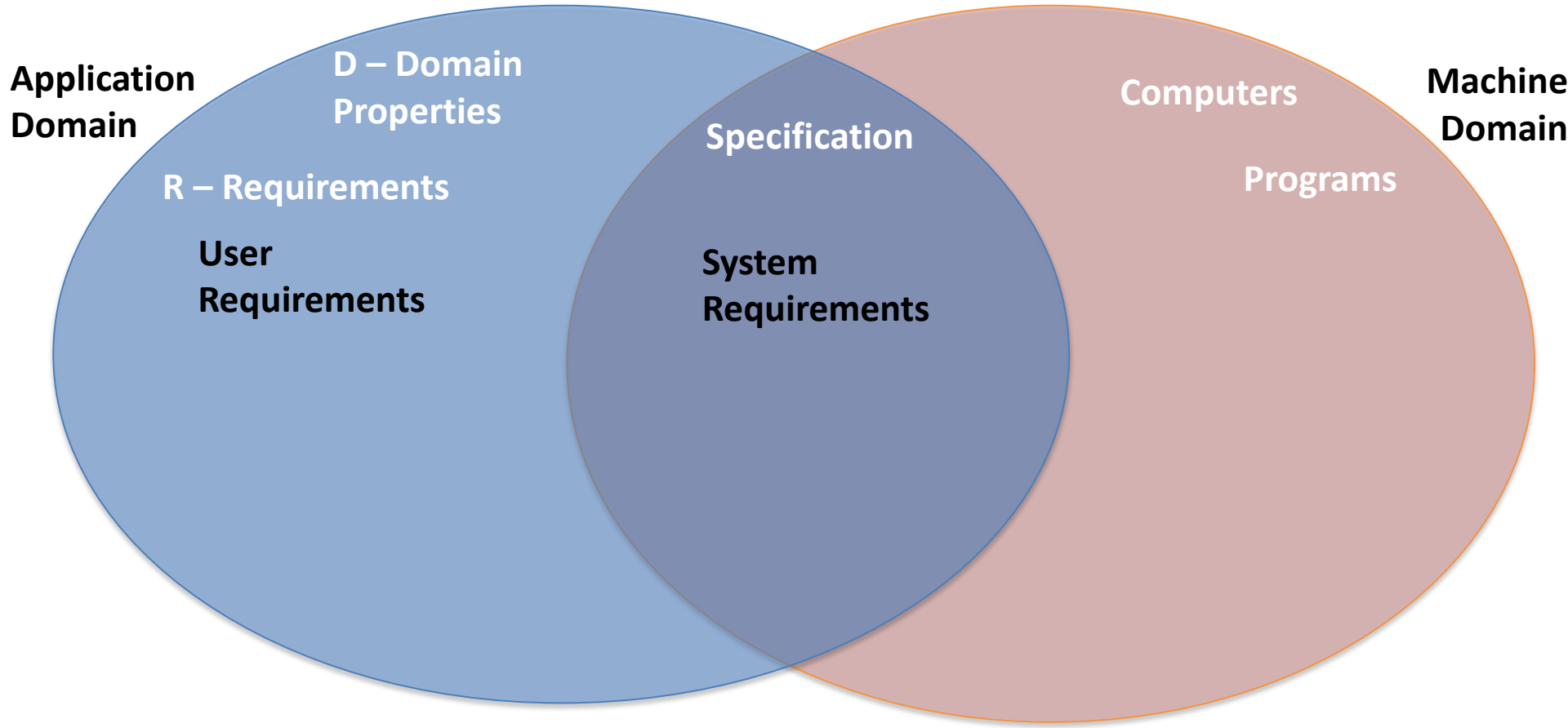
- Stakeholders: anyone who has a "stake" in the system
  - Anyone who can be positively or negatively affected

  - Users are stakeholders
    - Bike renters, maintenance staff (who use the system)
  - Managers (those who pay for the system)
    - City of Gothenburg or Bike Share Company
  - Users/managers of interacting systems
    - Whoever runs the payment system used, transportation planners
  - Anyone else affected
    - Neighbors to bike locations, city cleaners, ….

User Requirements vs. System Requirements

- User requirements:
  - Describe user needs
  - Come from the user
  - Are often high-level
- System requirements:
  - Describe what the system should do to meet user needs (without describing design)
  - Are typically derived from user requirements
  - Are usually low-level
- (Can also have business requirements, component requirements, sub-component requirements, ...)

User Requirements vs. System Requirements



User Requirements vs. System Requirements

- User requirements:
  - I want to rent a bike
  - I want to return a bike
  - I want to keep the bike for more than one hour

- System requirements:
  - The system shall allow users to indicate they would like to rent a particular bike in the rack, initiating payment processes (described furher in...)
  - The system shall allow users to return a rented bike
  - If the bike is returned before it is due, the system shall display a positive message to the user
  - If the bike is returned after it is due, the system will warn the user about late charges that will be charged to the card he/she previously provided
  - The system shall allow the user to indicate how long they would like to rent the bike, choosing from 1, 3, or 5 hours.

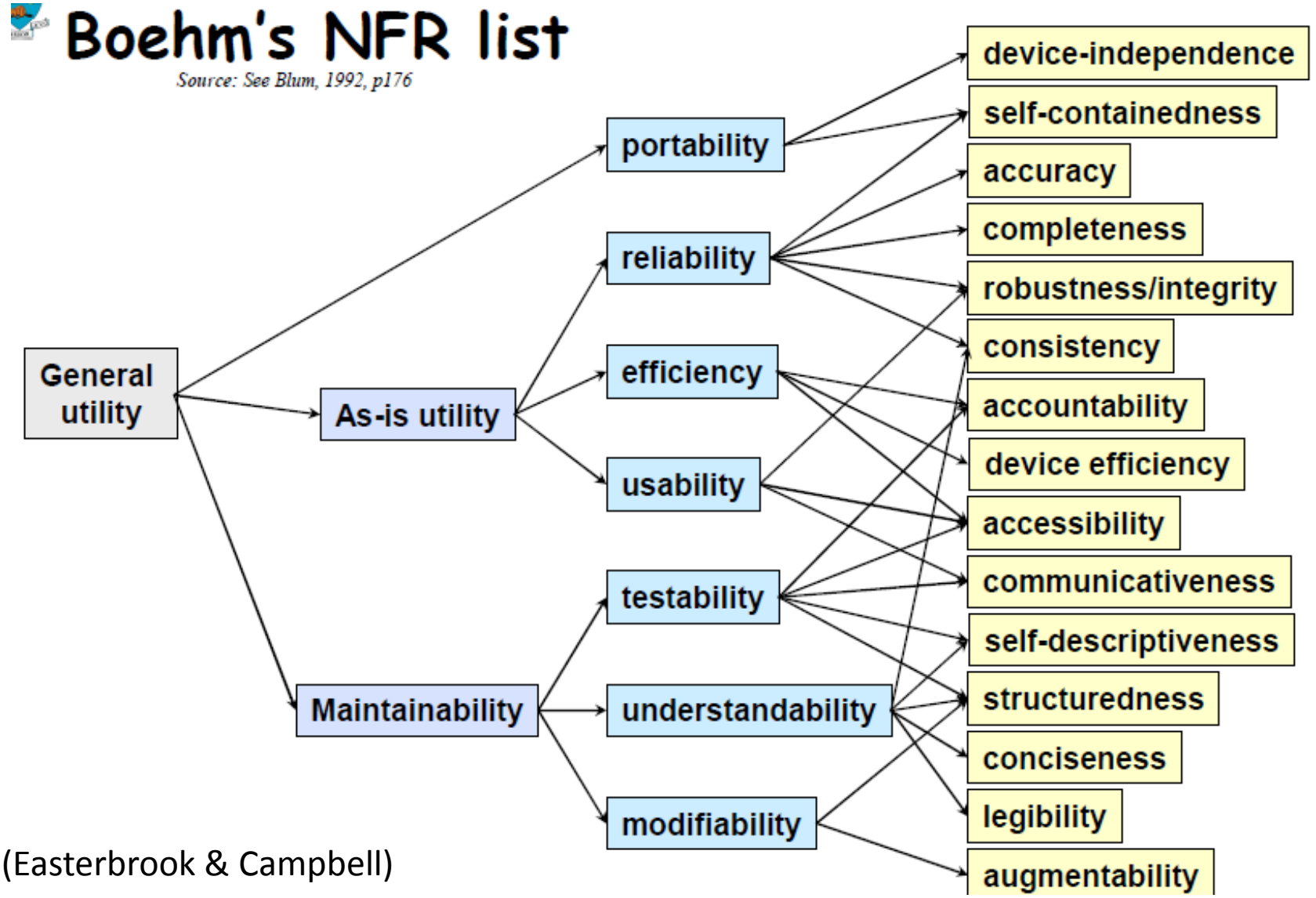
Functional vs. Non-Functional Requirements

- "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

Functional vs. Non-Functional Requirements

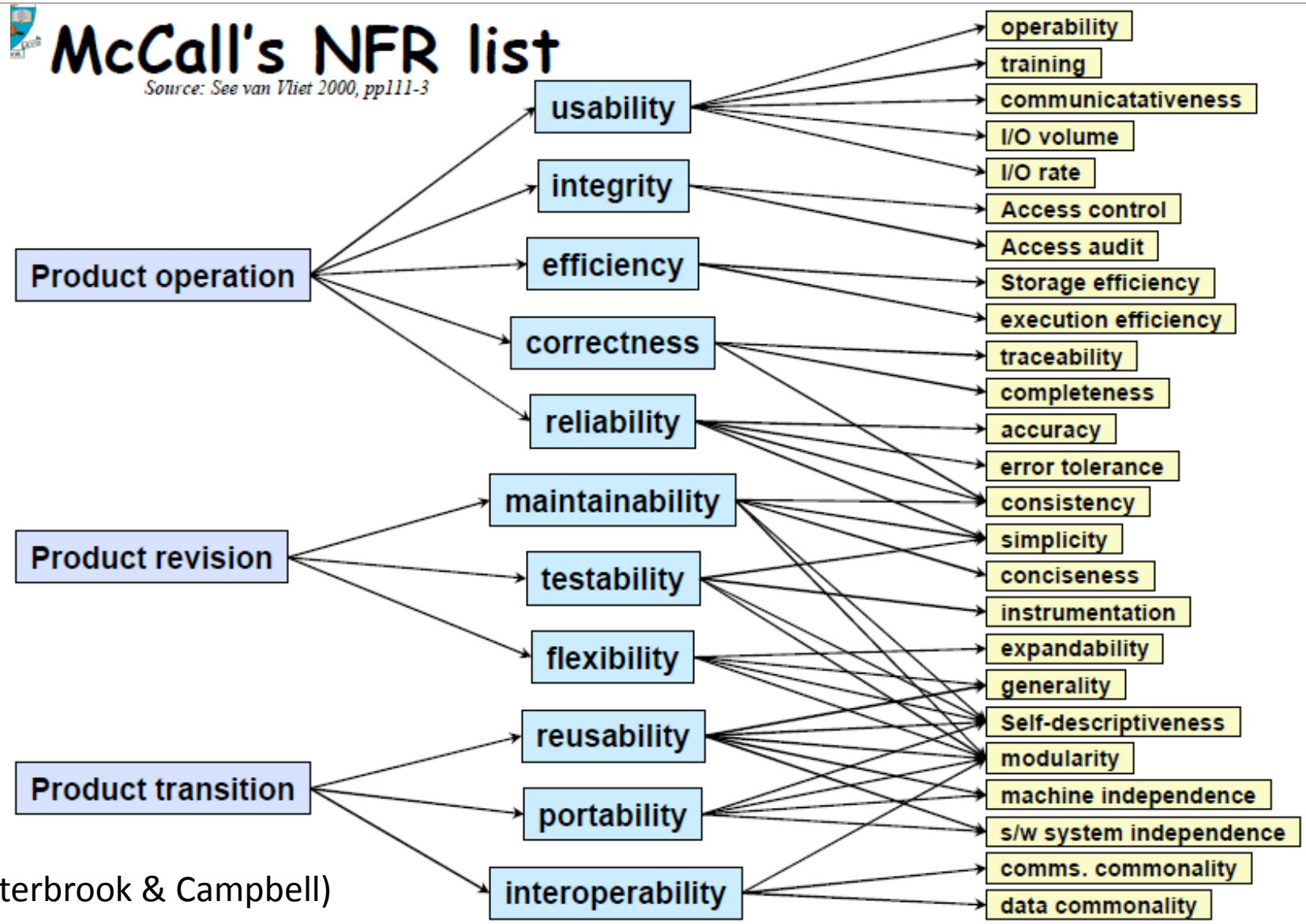
- Can have FRs and NFRs at both the user and system level
- We will learn later about what makes a requirement a good requirement
  - One property is verifiable (testable, measurable)
- In an ideal world all requirements would be precise and verifiable
- Realistically, user requirements are often high-level and vague
- System requirements should be verifiable
- User NFRs:
  - It should be easy to rent a bike
  - The system should allow quick bike rentals
- System NFRs:
  - 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

NFR Hierarchies



(Easterbrook & Campbell)

NFR Hierarchies



(Easterbrook & Campbell)

NFRs as Qualities

- Technically an NFR is any requirement that is not-functional
- This is not a very helpful definition (defining something in terms of what it is not)
- More useful to think about qualities
- When eliciting requirements, we not only need to understand ***What*** the system needs to do, but ***How Well*** it needs to do it

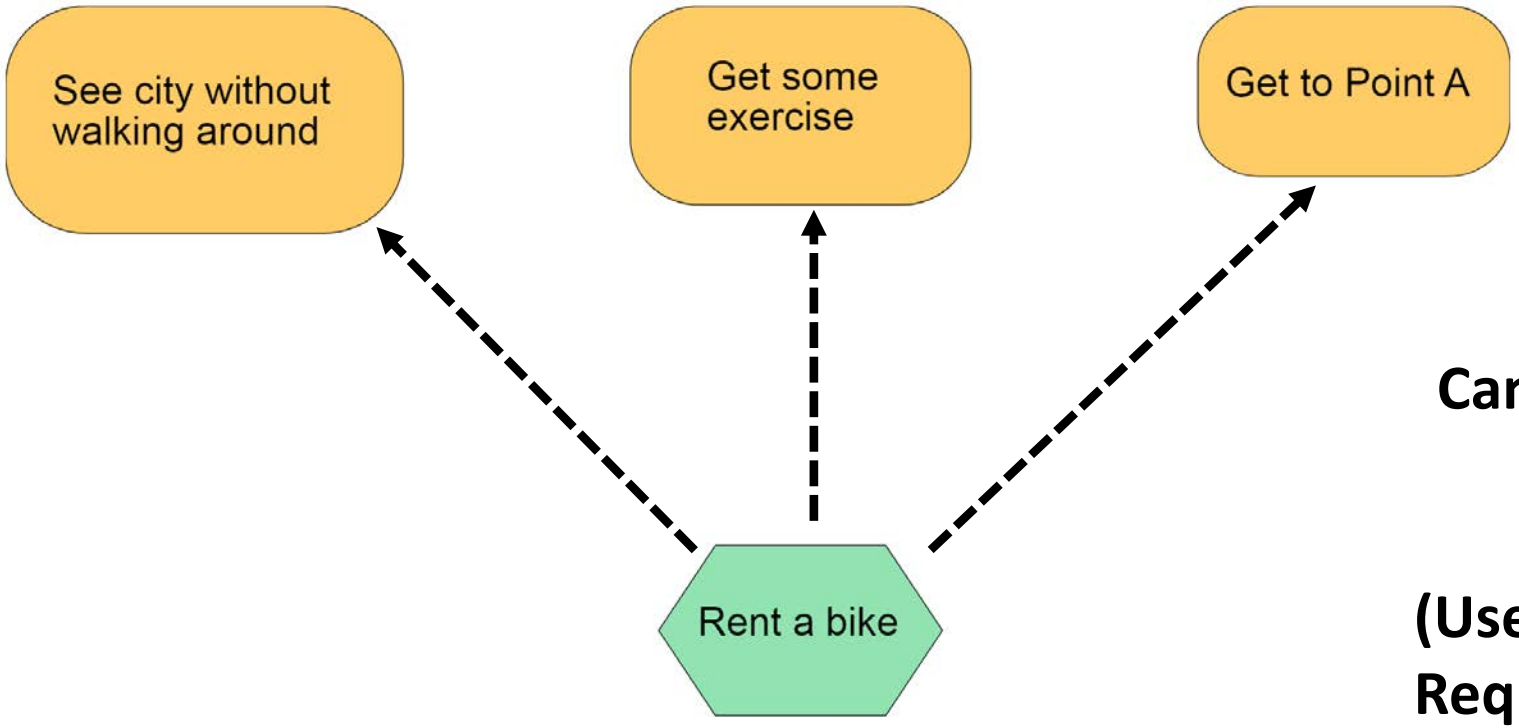
- A system could meet all it's FRs, but still fail due to poor quality
  - Poor performance
  - Poor usability
  - Poor availability

Requirements vs. Goals

- Goals:
  - What users would (really) like to achieve
  - Usually at a high level of abstraction
  - Usually does not directly reference the system
  - Captures users overall motivations and intentions

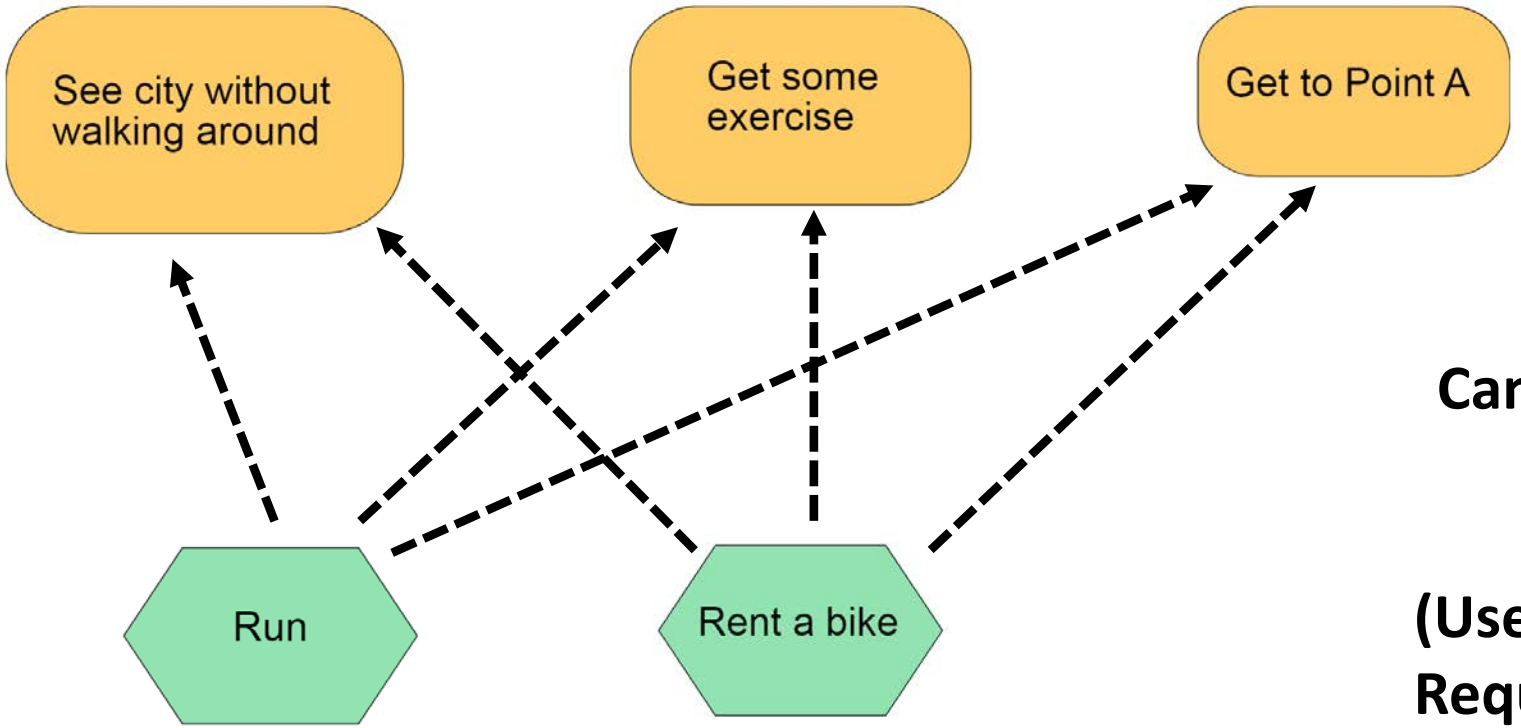
- Goal: I want to rent a bike
  - Is this a user goal?
  - WHY do you want to rent a bike?
    - I want to see the city without walking around
    - I want to get some exercise
    - I want to get to point A

Requirements vs. Goals



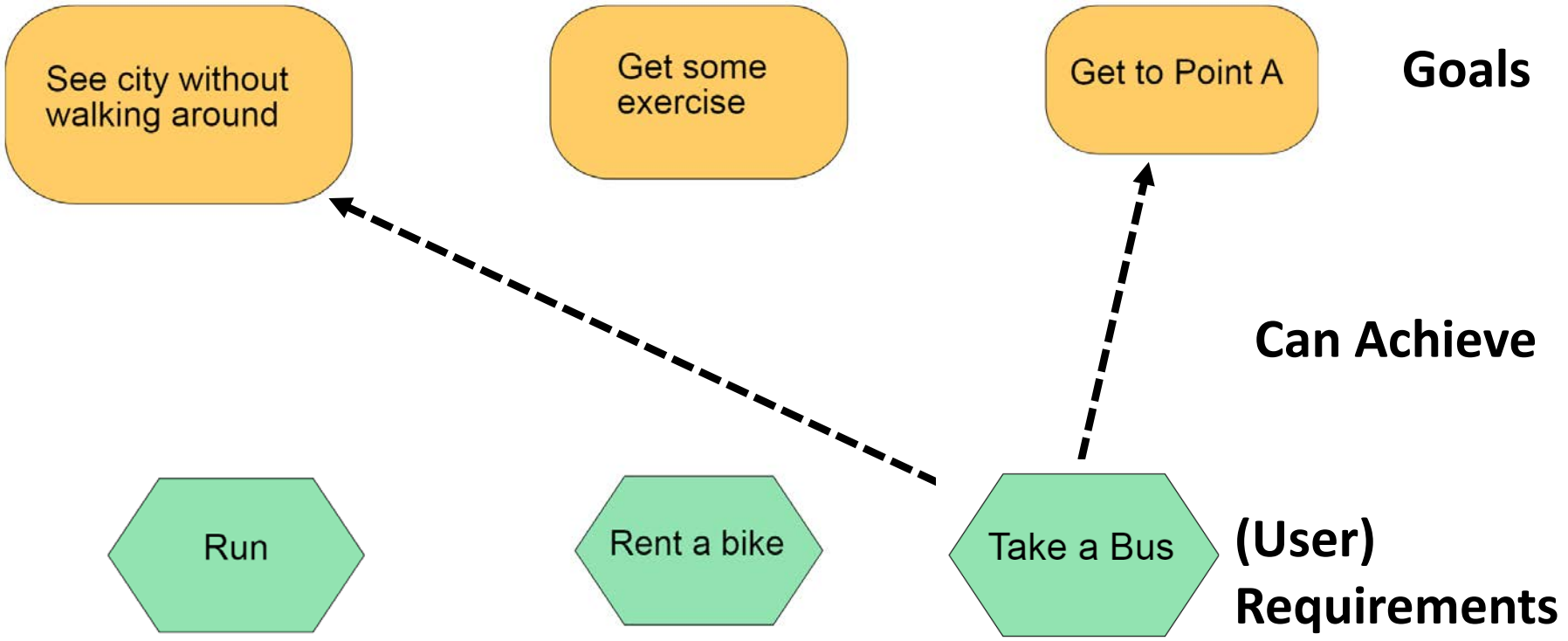
- But is renting a bike the best solution?

Requirements vs. Goals



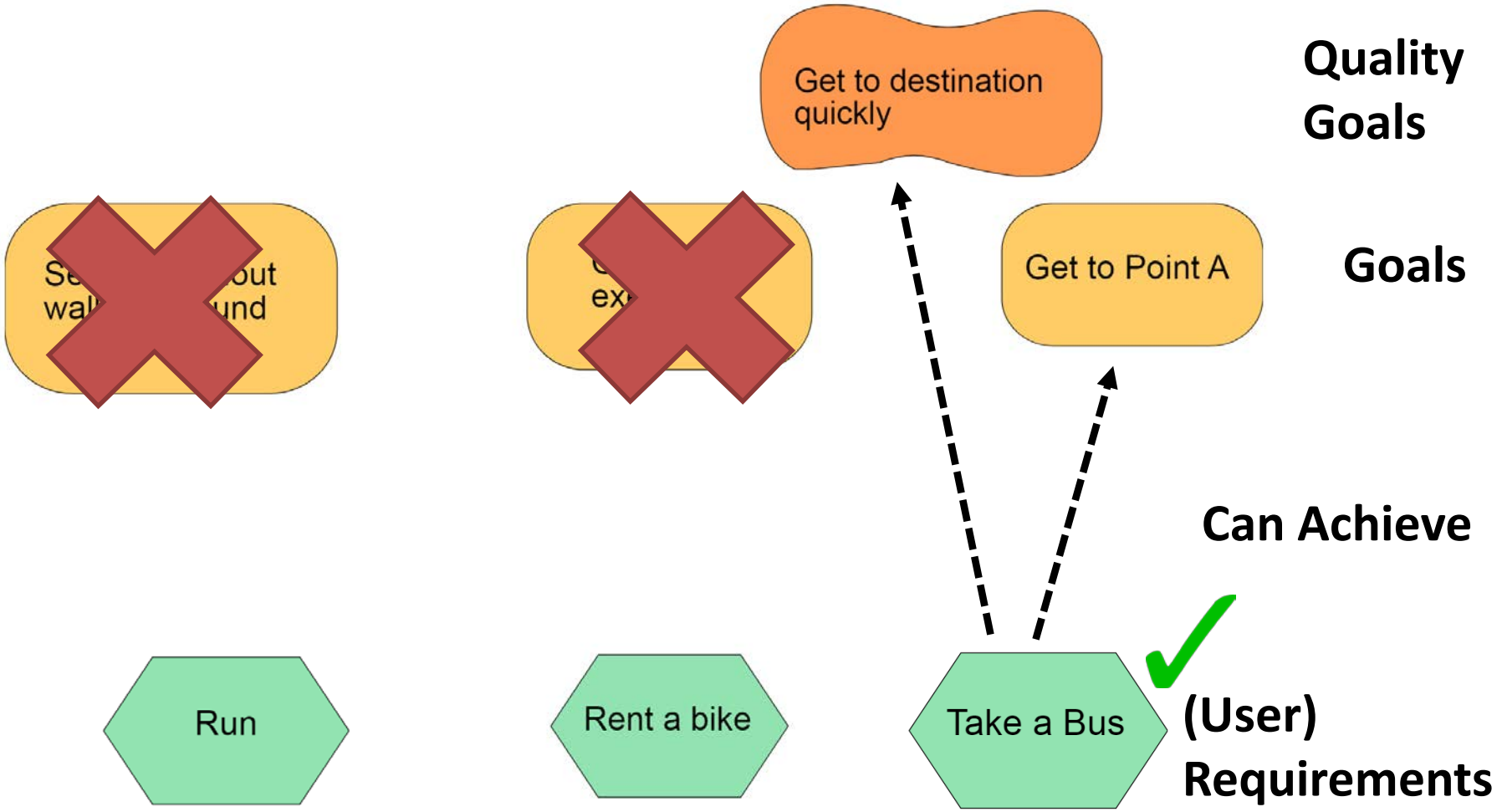
- But is renting a bike the best solution?

Requirements vs. Goals



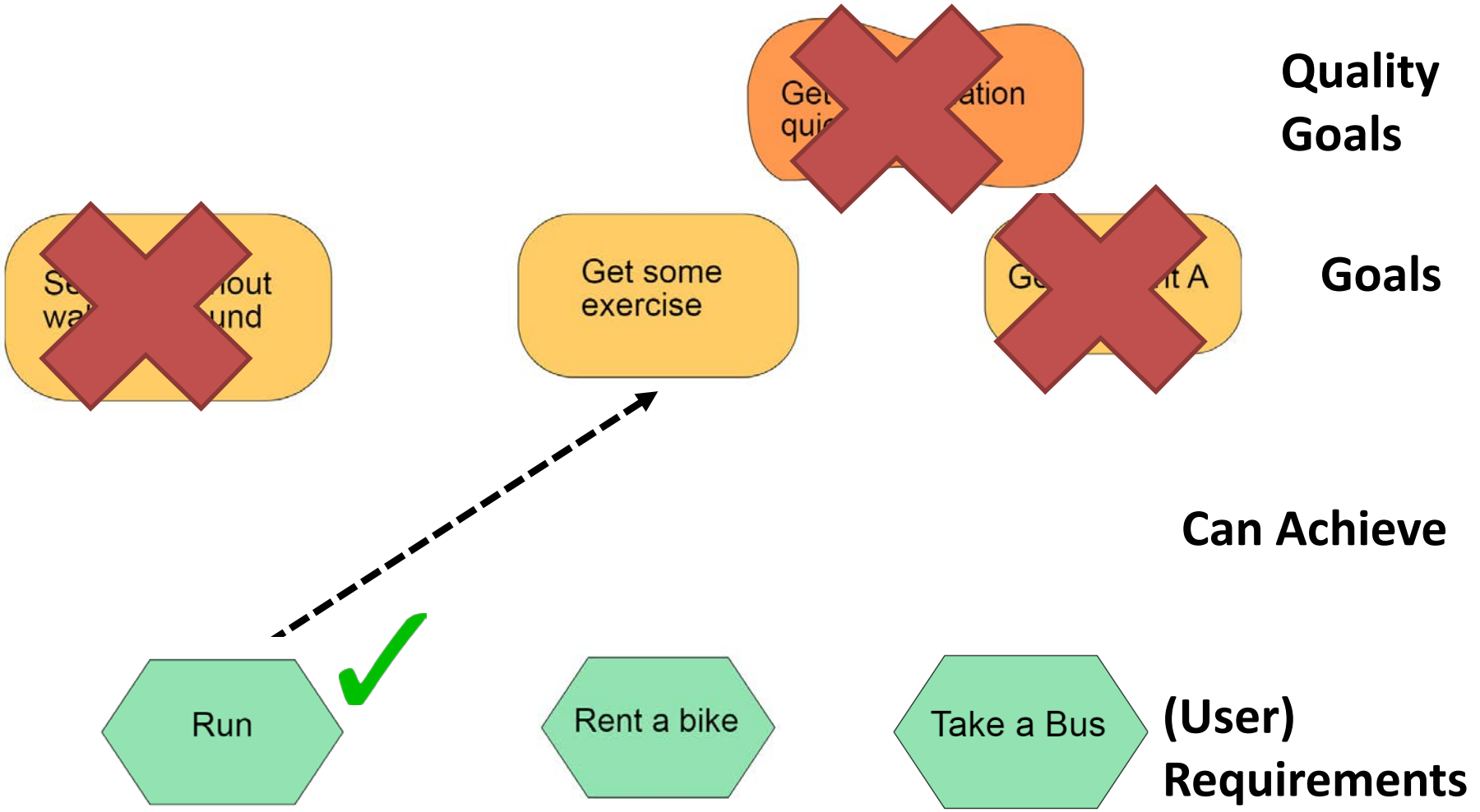
- But is renting a bike the best solution?

Requirements vs. Goals



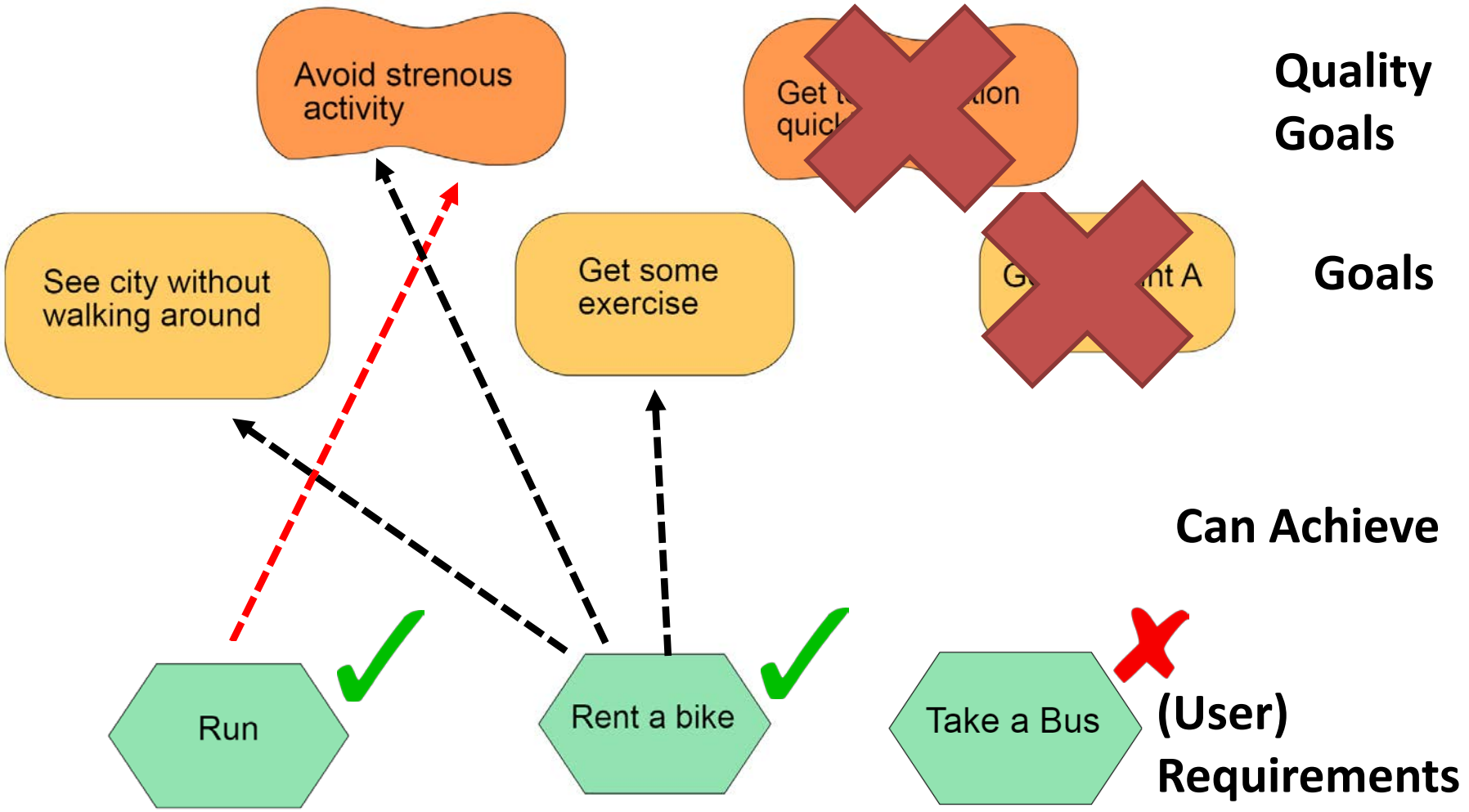
- Qualities?

Requirements vs. Goals



- Qualities?

Requirements vs. Goals



- Qualities?

Requirements vs. Goals

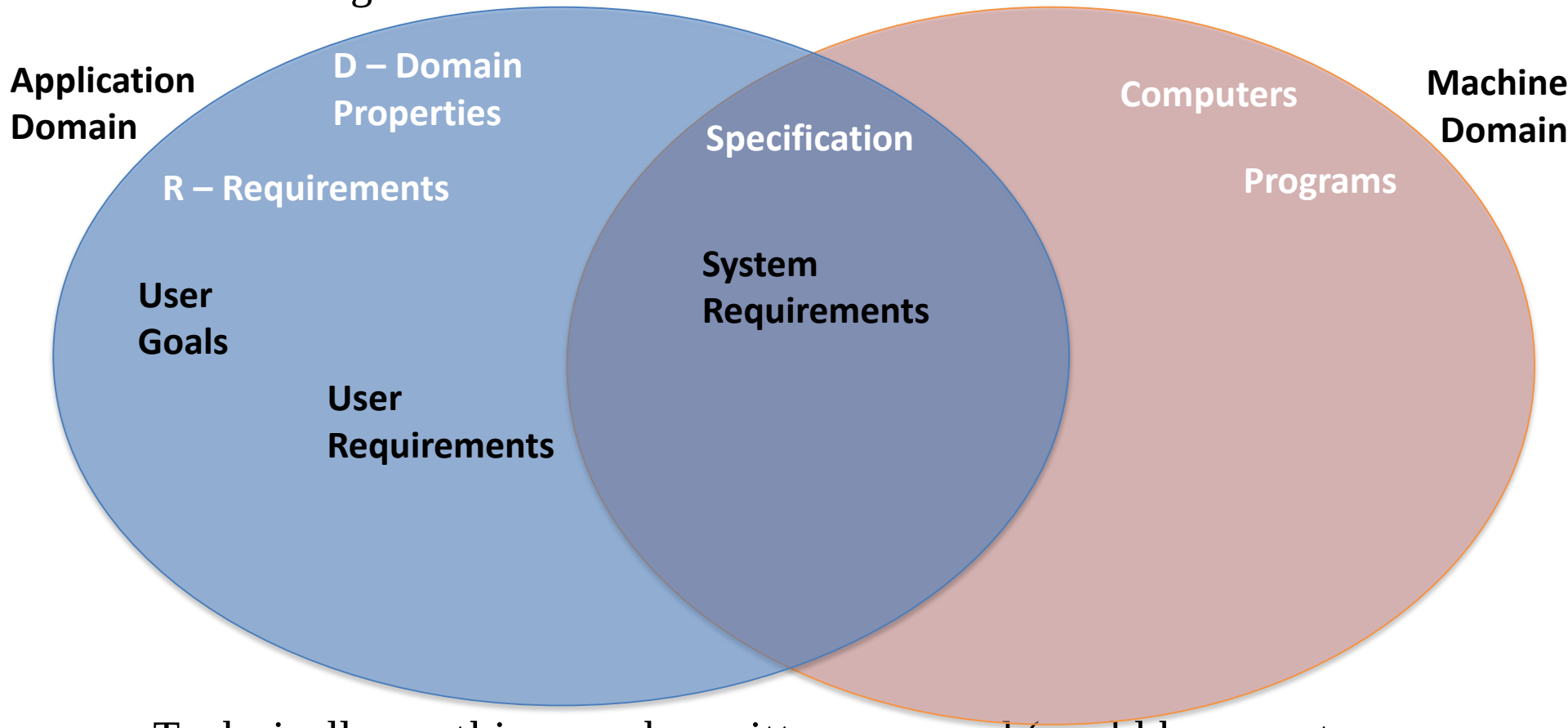
- Requirements achieve goals, they are a means to an end
- A requirements can achieve more than one goal
- A goal can also be achieved by more than one requirement

- -> Choice, Alternatives

- There isn't necessary one set of correct requirements, but a number of possible requirements, which may achieve user goals to different degrees
- Eliciting goals help us to make decisions in the requirements space
- More on this in lecture 5

Requirements vs. Goals

- Where do goals live?



- Technically anything can be written as a goal (could have system goals), but goals are most useful when considering high-level user needs and requirements, including alternatives

Requirements vs. Domain Properties

- Domain Properties/Assumptions:
  - Characteristics of the problem domain that are assumed to hold
  - Characteristics which are relevant for the functionality expressed by the requirements
  - Also called domain assumptions, as you assume they are true
  - They are not requirements, because they don't have to be achieved by the system
  - But they should be stated and remembered, as the system may break if they later do not hold

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

Requirements vs. Features

- A Feature is a “service that the system provides to fulfill one or more stakeholder needs”
- Features can be broken down into requirements
- Features can be groupings of requirements into common functionality
- Requirements tend to describe what the system will do (or what a user will do with a system)
- Features tend to be noun phrases, not actions
  - Card payment for bike rentals
  - Late bike returns
  - City navigation
  - Bike damage detection
- To avoid confusion we won’t use features in the course, but it’s a common term you will run into in practice

Specifications

- Two schools of thoughts
- As in the diagram

- "Concerns the interaction between the problem domain and the solution system" (Bray)
- "Specification: the behavior that a program needs in order to solve the problem" (Easterbrook)
- ≈ System Requirements

Software Requirement Specification (SRS)

- As in a collection of requirements in a document
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society
- Focuses on System Requirements, but can include user requirements, and other useful information
- Usually long
- Rather formal, acts as a contract between the requirements analyst/business/users and the architect/developers
- (Usually very boring)
- Not as common now (although they are still made and used)
- Agile backlogs of user stories + other agile processes are meant to replace this

SRS Sections

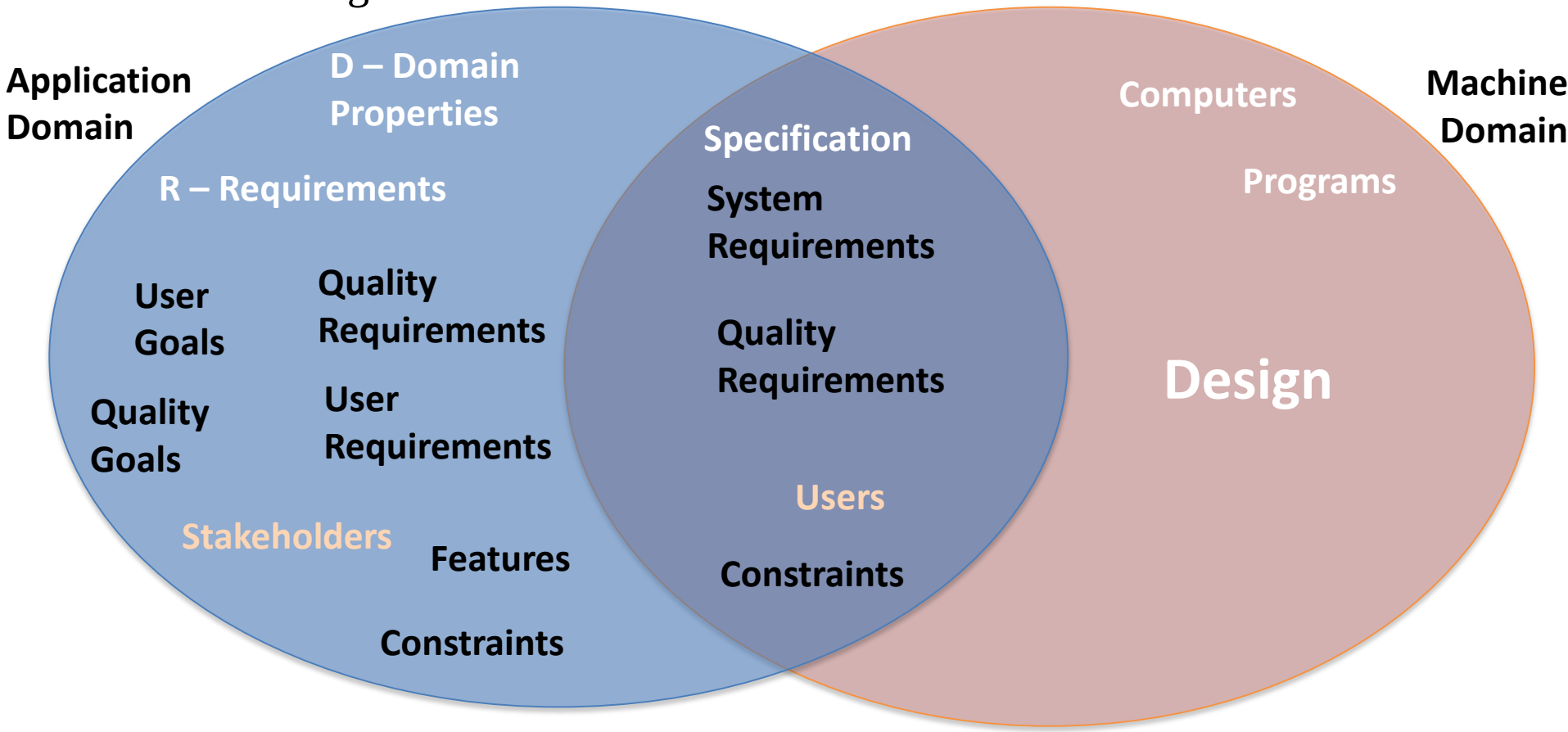
Title Page

Software Requirements Specification
for
<Project>
Version 1.0 approved
Prepared by <author>
<organization>
<date created>

SRS Sections

How does it all fit together?

- Where do goals live?



Questions?

Lecture Sources

- IREB (International Requirements Engineering Board)
  - https://www.ireb.org/en/downloads/
- Requirements Engineering (CSC340) S. Easterbrook, J. Campbell
  - http://www.cs.toronto.edu/~sme/CSC340F/
- Requirements Engineering for Software and Systems, Second Edition, By Phillip A. Laplante Kilicay-Ergin, Nil, and Phillip A. Laplante. "An online graduate requirements engineering course." *IEEE Transactions on Education* 56.2 (2013): 208-216.
- Dick, Jeremy, Elizabeth Hull, and Ken Jackson. *Requirements engineering*. Springer, 2017.
- "What is Requirements Engineering?" the draft chapter 1 and "What are Requirements?" the draft chapter 2 of Fundamentals of Requirements Engineering (FoRE), S. Easterbrook, 2004.
- Bray, Ian K. *An introduction to requirements engineering*. Pearson Education, 2002.

Lecture Sources

- Wilson, Tom. "Software failure: management failure. Amazing stories and cautionary tales: S. Flowers Wiley, Chichester, New York,(1996) 197 pp£ 19.99 ISBN 0 171-95113-7." *International Journal of Information Management* 17.5 (1997): 387.
- Macaulay, Linda A. *Requirements engineering*. Springer Science & Business Media, 2012.
- Pohl K (1993) The three dimensions of requirements engineering. CAiSE'93, Paris, France
- Jackson, Michael, and Pamela Zave. "Deriving specifications from requirements: an example." *Software Engineering, 1995. ICSE 1995. 17th International Conference on*. IEEE, 1995.