LECTURE 4: Requirements Engineering

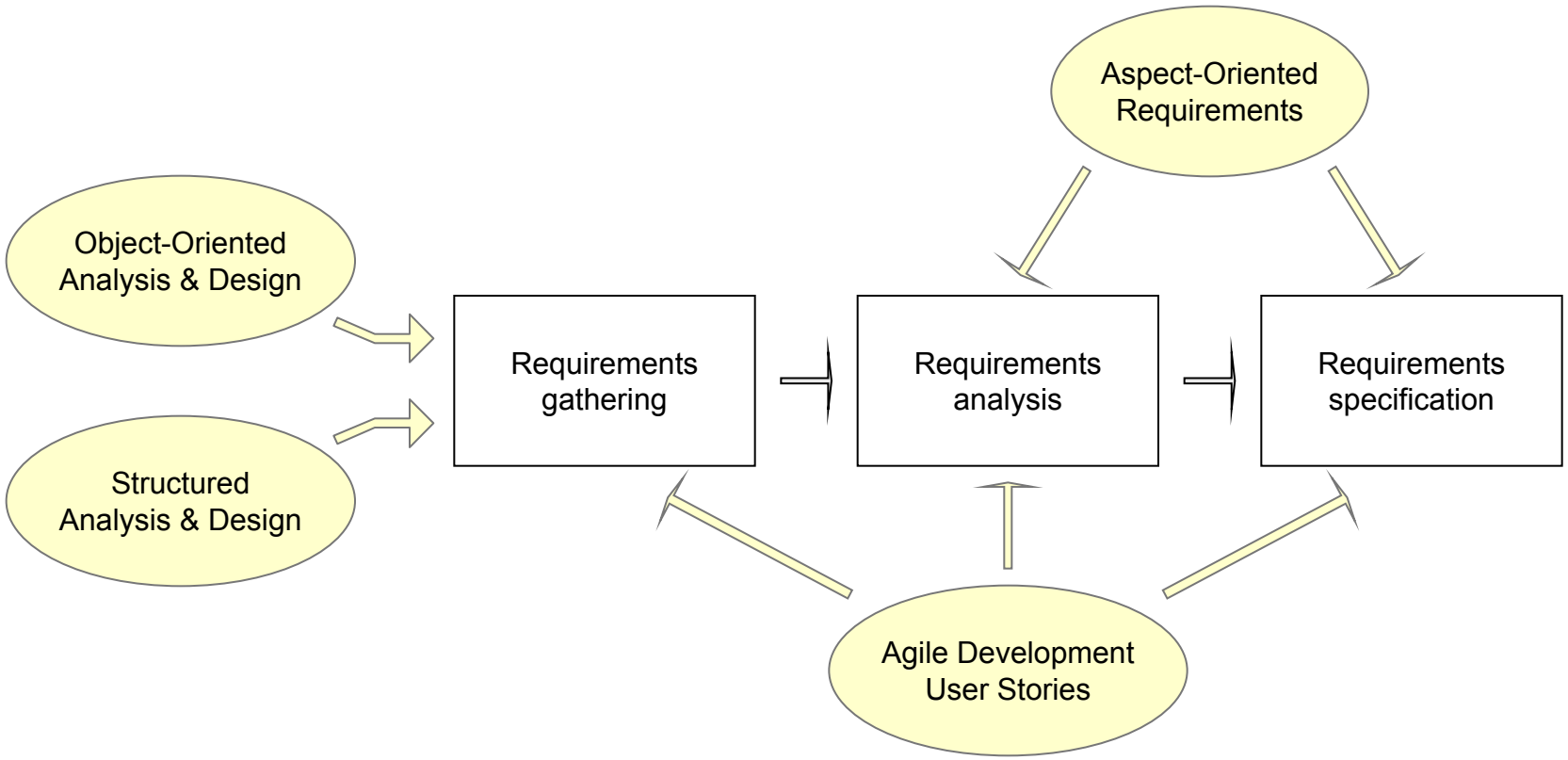
Topics

- Requirements Engineering Components
- Requirements and User Stories
- Types of Requirements
- Effort Estimation (Agile Methods)

Software Requirements

- A **requirement** specifies the business functions that the user will be able to perform using the system-to-be in different "situations" or "contexts", and the kind of experience the user will have during this work
  - Other concerns, such as how the system will manage the resources (computing, network, …), how the system will manage and protect user's data, etc.
- User requirements will often be high-level, vague and incomplete. They are more like high-level goals, or business goals, rather than software requirements needed by the developer
- When trying to achieve a given high-level goal, we will need to consider what matters, what are the important parameters, so that we can derive the detailed technical requirements
- Only based on deeper understanding of detailed issues, we can identify important "scenarios" or "situations" and identify what parameters should be considered in each situation
- Then using these parameters, we decide what the system should do, or how to respond to this situation (i.e., inputs)

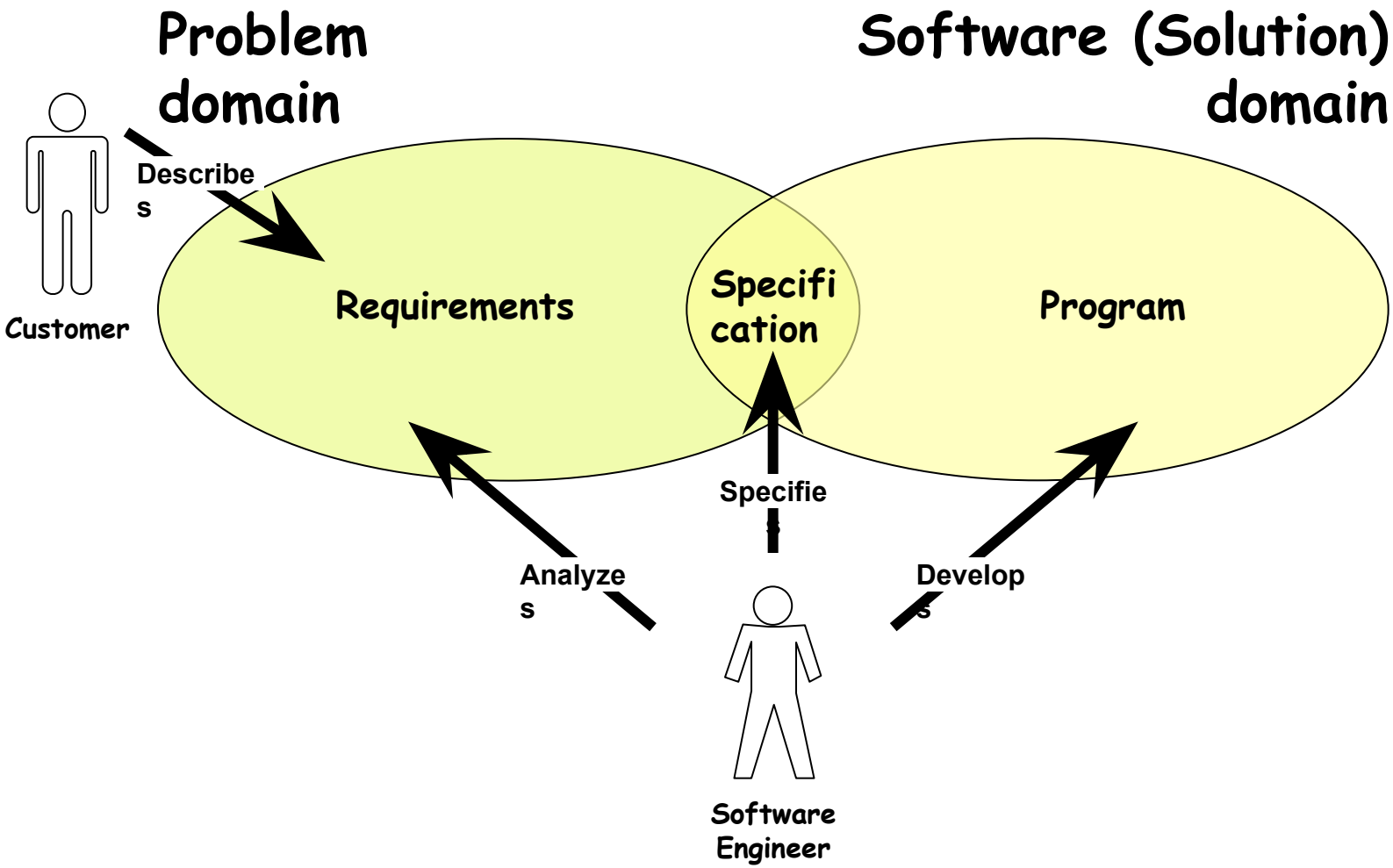
Requirements Process



Requirements Engineering Components

- Requirements gathering
  - (a.k.a. “requirements elicitation”) helps the customer to define what is required: what is to be accomplished, how the system will fit into the needs of the business, and how the system will be used on a day-to-day basis
- Requirements analysis
  - refining and modifying the gathered requirements
- Requirements specification
  - documenting the system requirements in a semiformal or formal manner to ensure clarity, consistency, and completeness

Requirements and Specification



Requirements Derivation

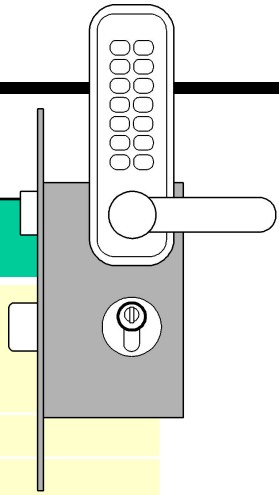
Requirements are determined by:

- Judgment about customer's business needs
- Conditions imposed by real-world constrains:
  - Physical
  - Social/Cultural
  - Legal
  - Financial
  - …
- Threats created by adversaries

 Requirements are **not** just desires!

- Requirements are desires *adjusted to real-world constraints and threats*

Example System Requirements



Identifier	Priority	Requirement
REQ1	5	The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).
REQ2	2	The system shall lock the door when commanded by pressing a dedicated button.
REQ3	5	The system shall, given a valid key code, unlock the door and activate other devices.
REQ4	4	The system should allow mistakes while entering the key code. However, to resist "dictionary attacks," the number of allowed failed attempts shall be small, say three, after which the system will block and the alarm bell shall be sounded.
REQ5	2	The system shall maintain a history log of all attempted accesses for later review.
REQ6	2	The system should allow adding new authorized persons at runtime or removing existing ones.
REQ7	2	The system shall allow configuring the preferences for device activation when the user provides a valid key code, as well as when a burglary attempt is detected.
REQ8	1	The system should allow searching the history log by specifying one or more of these parameters: the time frame, the actor role, the door location, or the event type (unlock, lock, power failure, etc.). This function shall be available over the Web by pointing a browser to a specified URL.
REQ9	1	The system should allow filing inquiries about "suspicious" accesses. This function shall be available over the Web.

- Problem: Requirements prioritization.
- See how solved in **agile methods**.

From Requirements to Business Policies

Explicit identification of **business policies** is important for two reasons:

1. Making the need for BP explicit allows involving other stakeholders, particularly the customer, in decision making about the BP solutions to adopt
2. Helps to anticipate potential future changes in the policies, so mechanisms can be implemented in the code that localize the future changes and allow quick substitution of implemented business policies

Acceptance Tests

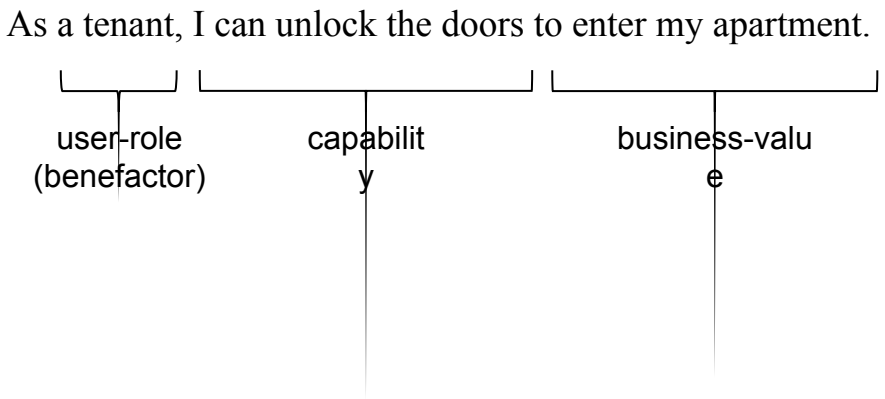
- Each *requirement* describes for a given "situation" (i.e., system *inputs*), the *output* or behavior the system will produce
- An **acceptance test** specifies a set of scenarios for determining whether the finished system meets the customer requirements
- An **acceptance test *case*** specifies, for a given "situation" or "context" (defined by current system inputs), the output or behavior the system will produce in response

- [See examples in Appendix G of the lecture notes]

Problem: Requirements Prioritization

- When prioritizing requirements, “important” and “urgent” aspects can be confused
- It is also difficult to assign a numeric value of priority to each requirement
  - It requires more mental effort than just rank-ordering the requirements in a linear sequence

User Stories



- Similar to system requirements, but focus on the user benefits, instead on system features.
- Preferred tool in **agile methods**.

Example User Stories

Identifier	User Story	Size
ST-1	As an authorized person (tenant or landlord), I can keep the doors locked at all times.	**4 points**
ST-2	As an authorized person (tenant or landlord), I can lock the doors on demand.	**3 pts**
ST-3	The lock should be automatically locked after a defined period of time.	**6 pts**
ST-4	As an authorized person (tenant or landlord), I can unlock the doors. (Test: Allow a small number of mistakes, say three.)	**9 points**
ST-5	As a landlord, I can at runtime manage authorized persons.	**10 pts**
ST-6	As an authorized person (tenant or landlord), I can view past accesses.	**6 pts**
ST-7	As a tenant, I can configure the preferences for activation of various devices.	**6 pts**
ST-8	As a tenant, I can file complaint about “suspicious” accesses.	**6 pts**

- Note no priorities for user stories.
- Story priority is given by its order of appearance on the work backlog (described next)
- Size points (last column) will be described later

Requirements Analysis Activities

- Not only refinement of customer requirements, but also feasibility and how realistic
- Needs to identify the points where **business policies** need to be applied.
- Explicit identification of business policies (BP) is important for two reasons:
  1. Making the need for BP explicit allows involving other stakeholders in decision about the solutions to adopt—Helps to involve others, particularly the customer, in decision making about each policy to adopt
  2. Helps to anticipate potential future changes in the policies, so mechanisms can be implemented in the code that localize the future changes and allow quick substitution of business policies

Types of Requirements

- Functional Requirements
- Non-functional requirements (or quality requirements)
  - FURPS+
  - Functionality (security), Usability, Reliability, Performance , Supportability
- On-screen appearance requirements

On-screen Appearance Requirements

- Do not waste your time and your customer's time by creating elaborate screen shots with many embellishments, coloring, shading, etc., that serves only to distract attention from most important aspects of the interface
- Hand-drawing the proposed interface forces you to ***economize*** and focus on the most important features
- Only when there is a consensus that a good design is reached, invest effort to prototype the interface

https://arstechnica.com/gadgets/2018/01/with-ink-to-code-microsoft-is-turning-back-of-napkin-sketches-into-software/?amp=1

Tools for Requirements Eng.

- Tools, such as user stories and use cases, used for:
  - Determining what exactly the user needs ("requirements analysis")
  - Writing a description of what system will do ("requirements specification")
- Difficult to use the same tool for different tasks (analysis vs. specification)

Acceptance Tests

- Means of assessing that the requirements are met as expected
- Conducted by the customer
- An acceptance test describes whether the system will pass or fail the test, given specific input values
- Cannot ever guarantee 100% coverage of all usage scenarios, but *systematic approach* can increase the degree of coverage

Project Estimation using User Story Points

- Recall "hedge pruning points" from the first lecture
- Size points assigned to each user story
- Total work size estimate:
  - Total size = $\sum$ (points-for-story i), $i = 1..N$
- Velocity (= productivity) estimated from experience
- Estimate the work duration

$$\text{Project duration} = \frac{\text{Path size}}{\text{Travel velocity}}$$

Example User Stories

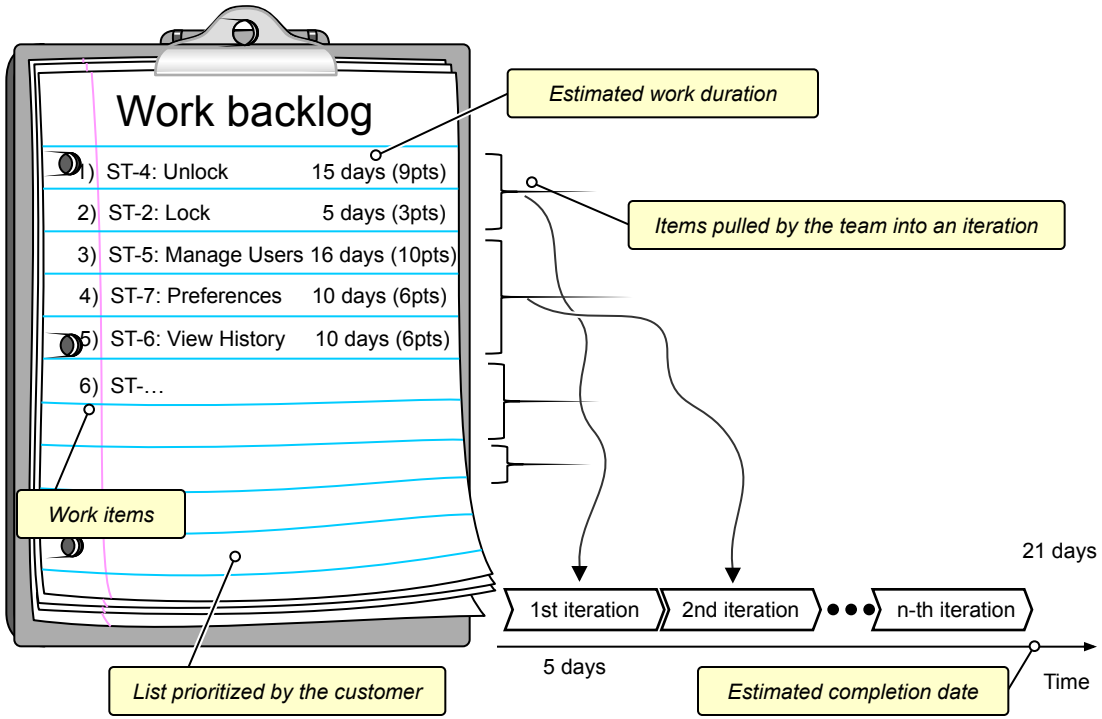
Identifier	User Story	Size
ST-1	As an authorized person (tenant or landlord), I can keep the doors locked at all times.	**4 points**
ST-2	As an authorized person (tenant or landlord), I can lock the doors on demand.	**3 pts**
ST-3	The lock should be automatically locked after a defined period of time.	**6 pts**
ST-4	As an authorized person (tenant or landlord), I can unlock the doors. (Test: Allow a small number of mistakes, say three.)	**9 points**
ST-5	As a landlord, I can at runtime manage authorized persons.	**10 pts**
ST-6	As an authorized person (tenant or landlord), I can view past accesses.	**6 pts**
ST-7	As a tenant, I can configure the preferences for activation of various devices.	**6 pts**
ST-8	As a tenant, I can file complaint about “suspicious” accesses.	**6 pts**

Agile Estimation of Project Effort

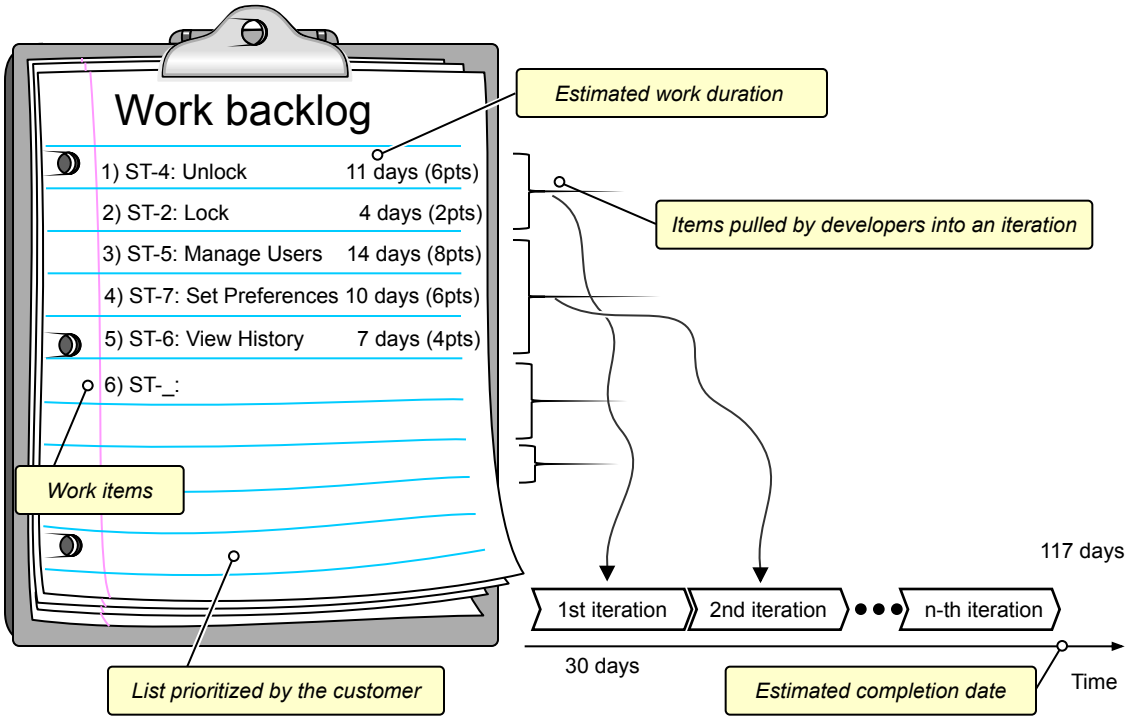
1) Prune Section 6 1 day (2pts)
2) Prune Section 5 2 days (4pts)
3) Prune Section 7 2 days (4pts)
4) Prune Section 4 1.5 days (3p)
5) Prune Section 8 3.5 days (7p)

2 points per day

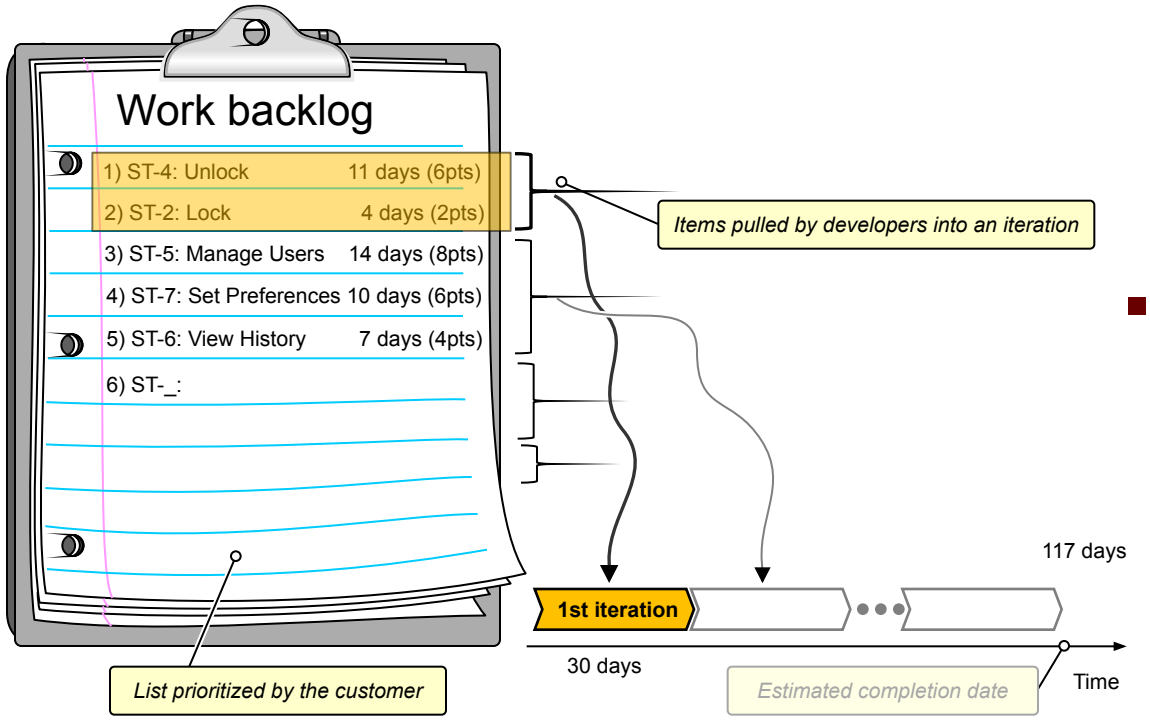
1 = 4 pts (2 days)
2 = 7 pts (3.5 days)
3 = 10 pts (5 days)
4 = 3 pts (1.5 days)
5 = 4 pts (2 days)
6 = 2 pts (1 day)
7 = 4 pts (2 days)
8 = 7 pts (3.5 days)



Agile Estimation of Project Effort

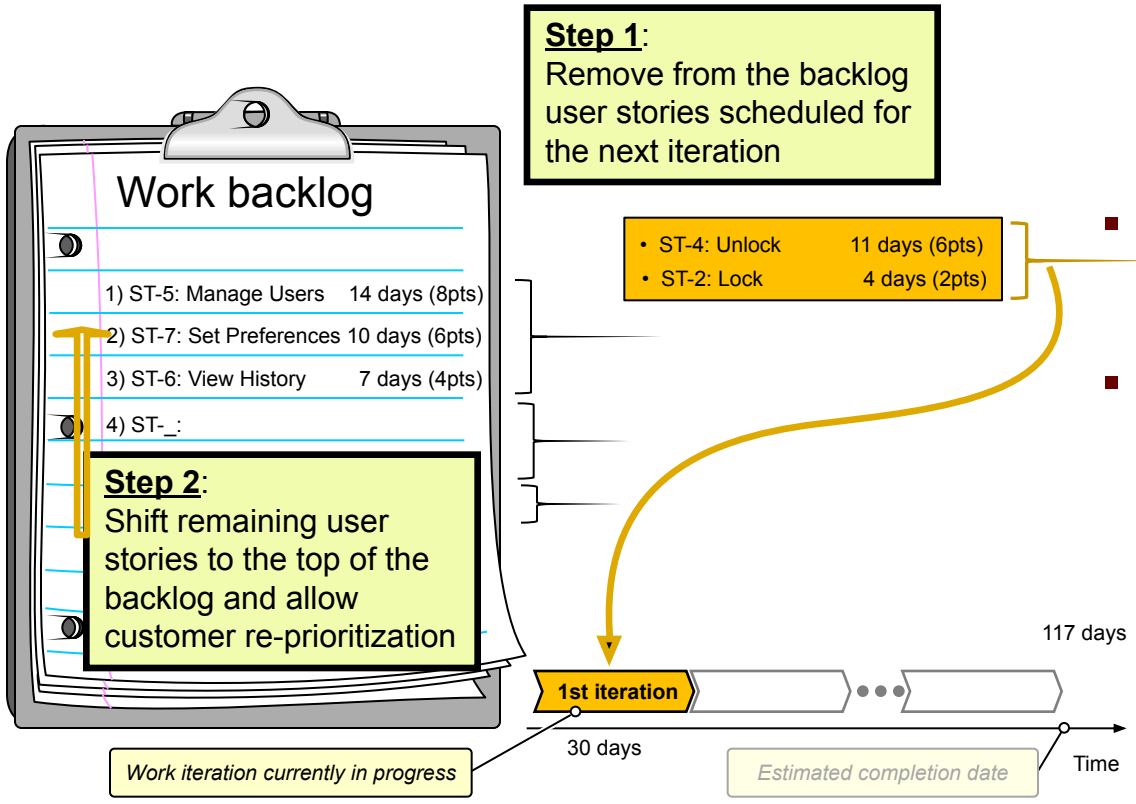


Agile Prioritization of Work



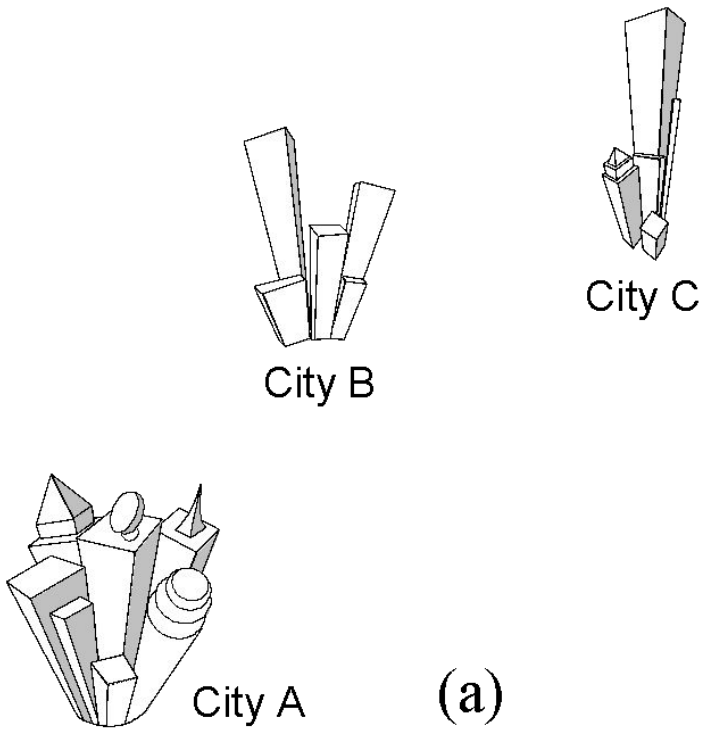
- Instead of assigning priorities, the customer creates an ordered list of user stories
- Developers simply remove the top list items and work on them in the next iteration

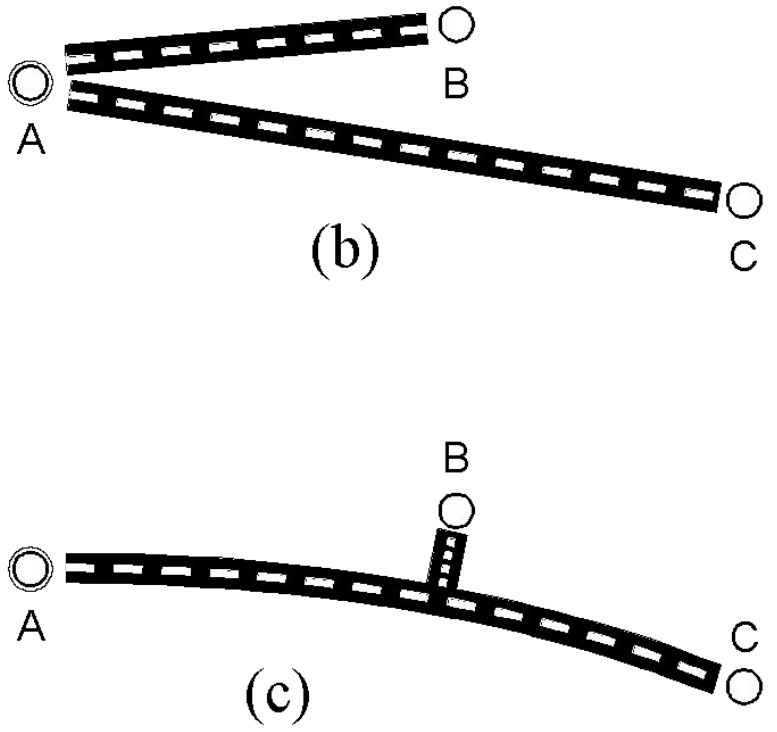
Tradeoff between Customer Flexibility and Developer Stability



- Items pulled by developers into an iteration are not subject to further customer prioritization
- Developers have a **steady goal** until the end of the current iteration
- Customer has **flexibility** to change priorities in response to changing market forces

How To Combine the Part Sizes?



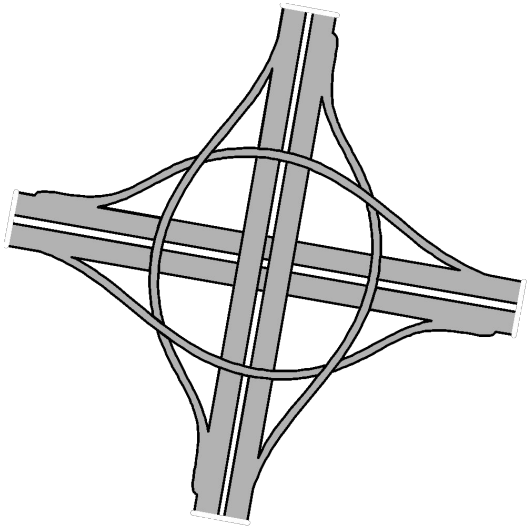


Costs are not always additive

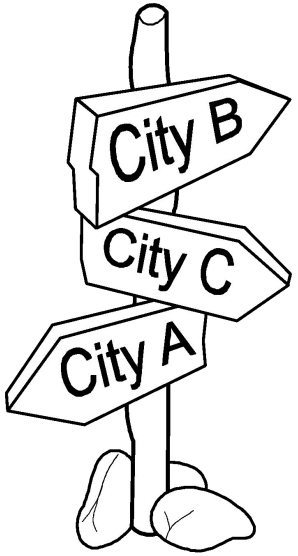
But, solution (c) is not necessarily “cheaper” than (b)

…

Additional Costs



Highway traffic-circle interchange



Traffic signs