Requirements

CSE 3311 / 5324
Christoph Csallner
University of Texas at Arlington (UTA)

Several slides in this set are copied almost verbatim (with permission) from David Kung (UTA)

Additional Sources

- [HVV] = Hans van Vliet: Software Engineering: Principles and Practice, 3rd edition, Wiley, 2008
- [PA] = Shari Lawrence Pfleeger and Joanne M. Atlee: Software Engineering: Theory and Practice, 4th edition, Prentice Hall, 2009

- IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications. 1998.

The Indispensable First Step to Getting the Things You Want Out of Life:

Decide what you want.

– Ben Stein

- As opposed to: Just start doing things, without first determining the goal of these actions
  - Will get you somewhere
  - But most likely you won’t end up where you want to
- In this sense life is just like a software engineering project ☺

Motivation

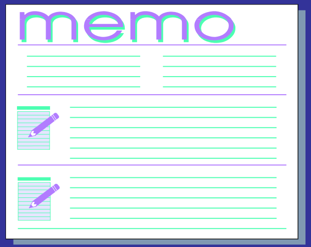
- Requirements = Capabilities and conditions the system must have/satisfy

- What is really needed?
  - Find it out
  - Remember it → write it down / document it
  - Communicate it
- “What is really needed” may change over time!
  - Because our world changes constantly
  - Cannot “freeze” the world for duration of our project
  - ➔ Cannot “freeze” the requirement documents

Step 1: Collect Information

Interview customer, users, domain experts

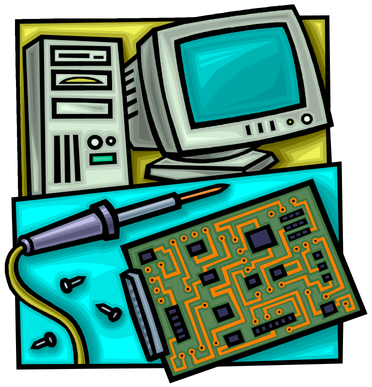
Study similar projects



Business forms & operating procedures

Regulations & standards

Study existing documentation



Rapid prototyping

Information to Collect (1/2) [DK 4.5]

- What is their business?
  - What is their current business situation?
- What is the business goal?
  - Why do they want to build the new system? E.g.:
    - Increase profit / market share
    - Improve productivity / quality of products and services
    - Reduce costs
- What are the existing business processes?
  - What are the inputs and outputs?
  - How do they interact?

Information to Collect (2/2) [DK 4.5]

- What are the problems of the current system? E.g.:
  - System is old / should be replaced
  - System is slow / difficult to use / costly to maintain

- What is the system’s environment and context?
- What are the resources and constraints?
  - Quality, Security, Performance, Timeline

- Who are the users?
- What do the various stakeholders want from the new system?

How to Collect [DK 4.5]

- Request and study documents
  - Relevant business documents, procedures, manuals, forms, industry/company standards, policies, regulations
- Attend training or tutorials by domain experts
- Study similar projects
  - Both for requirements & design
- Survey stakeholders: Interviews, questionnaire
  - Anonymous questionnaire may give better results
  - Questionnaire: Brief, focused, reviewed by customer

Presentation by Customer [DK 4.5] 1/2

- Can give a quick initial overview
- Guidelines:
  - Presenter should have relevant knowledge
    - Good: Manager / administrator responsible for daily operations
    - Avoid: New hire
  - Tell presenter in advance what should be covered
  - Ask for pointers to additional information
  - Review presentation slides before presentation to ensure it covers the requested information

Presentation by Customer [DK 4.5] 2/2

- Guidelines:
  - – During presentation, ask only questions to clarify presented material – do not get into implementation discussions
  - – Remain focused on information to be collected

  - – Obtain copy of presentation and share with team
  - – Keep presentation under 2 hours

Interview Stakeholders

- Hard: Each stakeholder has own agenda
- Success of our project may cost stakeholders
  - Influence on processes

- **After studying documents and survey results**
  - Explore open questions, questions raised by survey
- After preparing interview questions
  - Focused on important requirements
- Limited, e.g., to max 1 hour each
- Interviewee is the expert → Listen
  - Do not provide advice, even if it seems obvious to you

Business Domain Jargons Differ

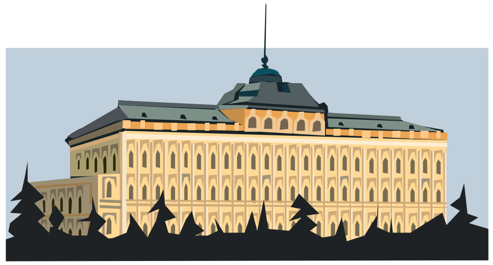
Investment
- premium
- security
- option
- margin
- ...

Insurance
- premium
- option
- coverage
- liability
- ...

Retailing
- sales
- inventory
- markup
- profit
- ...

Ask the customer/users to explain terms and document them in the data dictionary

What Are the Constraints?



Budgetary

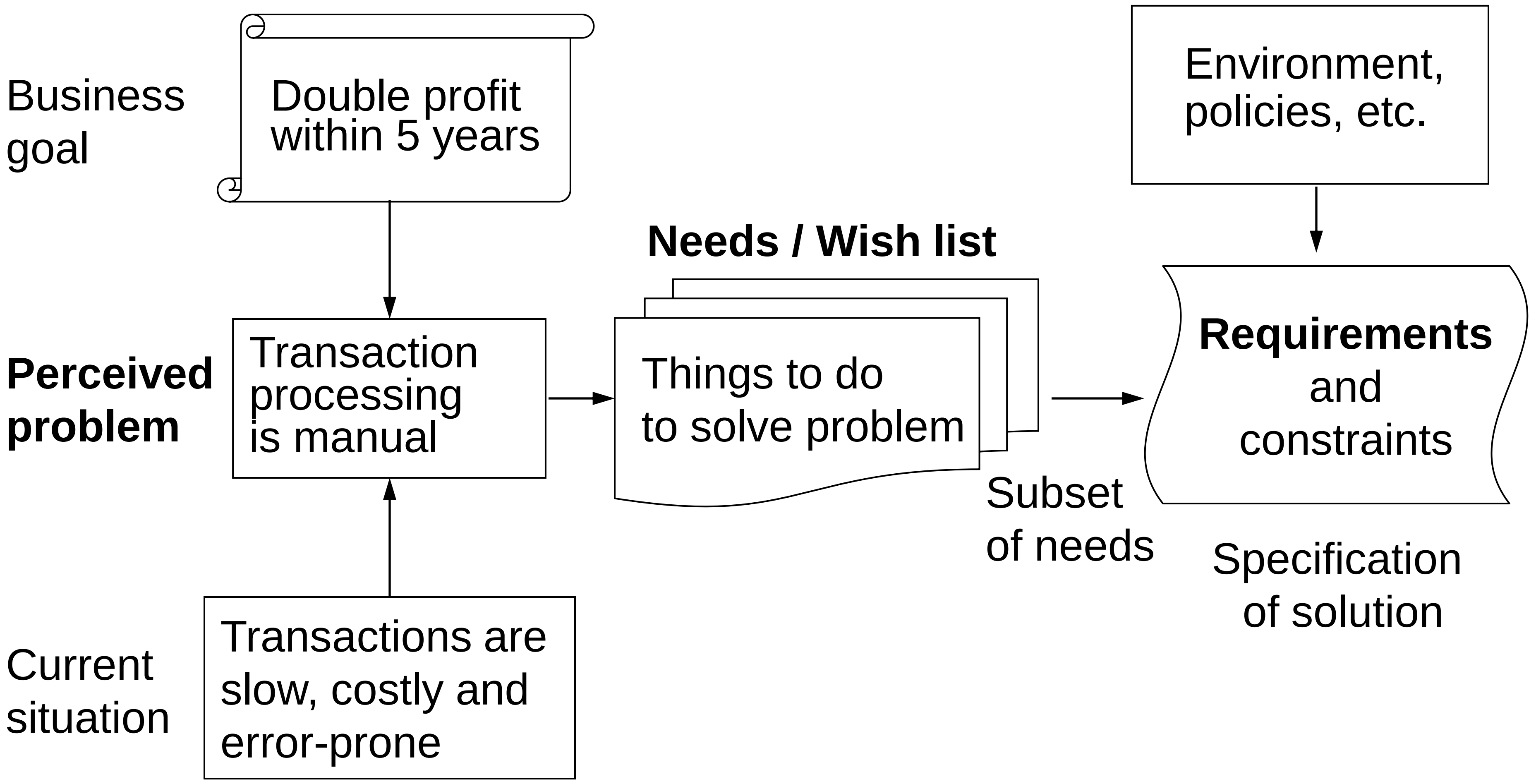
Environment

Political

Technology

Cultural

Identify Requirements Example



Business goal
Double profit within 5 years
Perceived problem
Transaction processing is manual
Current situation
Transactions are slow, costly and error-prone
Needs / Wish list
Things to do to solve problem
Subset of needs
Environment, policies, etc.
Requirements and constraints
Specification of solution

MoSCoW Requirement Classification

- Distinguish four types of requirements

  -- HVV, page 237

- **M**ust have
  - Top priority, definitely must be realized
- **S**hould have
  - Not strictly mandatory, but highly desirable
- **C**ould have
  - Will be realized if time allows. In practice, they won’t
- **W**on’t have
  - Recorded, but will not be realized

IN-CLASS EXERCISE: PRIORITIZE REQUIREMENTS

In-Class Exercise: Prioritizing

- Get together with your team (breakout room)
- Prioritize your current project requirements into:
  - **M**ust have: Top priority, must definitely realize
  - **S**hould have: Not strictly mandatory, but highly desirable
  - **C**ould have: Will be realized if time allows.
  - **W**on’t have: Recorded, but will not be realized
- Now re-prioritize your iteration plan
  - Make sure you deal with Must-Have before the others
- Post your prioritized requirements & updated iteration plan in the Teams chat

Expressing Requirements [DK 4.4]

- State as declarative sentence in active voice:
  - Use one of must / shall / will:
  - **<Subject> (must | shall | will) …**

- Examples:
  - “CRS **shall** allow a potential customer to inquire about the availability of rental cars [..]”
  - “Potential customers **shall** be able to inquire about the availability of rental cars [..]”

Making Requirements Testable

- In practice: **What gets measured gets done, e.g.:**
  - Have a JUnit test case for it → Will notice if it breaks
  - On the rubric → Good chance students will do it
  - Rewarded in annual employee evaluation → Gets done

- Once a requirement is stated, we should be able to check if a solution satisfies the requirement.

-- PA, pages 151 - 152

Technique 1: Quantify Requirements

- **Quantify the extent** to which each requirement must be met

- For which formulation is it easy to decide if system meets the requirement?

- Example:
  - Requirement formulation 1:
    Water quality information must be accessible immediately
  - Requirement formulation 2:
    System X must retrieve water quality records **within five seconds** of a request

IN-CLASS EXERCISE: QUANTIFY REQUIREMENTS

Quantify Your Requirements

- Get together with your team (breakout room)

- Summarize each requirement as one sentence
  - Use active voice: Who should do what
  - Use one of: must / shall / will
- Quantify each requirement (if possible)
  - Use a metric that is easy to measure (time, money, etc.)

- Post updated requirements in Teams chat

Additional Techniques

- Replace each pronoun with a specific name
- Define each noun in exactly one place in the requirements document

- **For each adverb and adjective:**
  - Specify a quantitative description
  - So that the meaning is clear and unambiguous
  - E.g.: “immediately” → “within 5 seconds of a request”

Making Subjective Req. Testable

- What about “soft” requirements?
  - It should be “easy” to use the system
  - It should be “easy” to maintain the system in the future
  - …

- Measure how **representative sample of target population** behaves, e.g.:
  - **75% of the representative user group** shall judge the new system to be **as usable as the existing system**
  - **After training, 90% of the representative user group** shall be able to process a new account **within 5 minutes**

IN-CLASS EXERCISE: MAKE REQUIREMENTS TESTABLE

In-Class Exercise: Being Testable

- Get together with your team (breakout room)
- Identify your current requirement R that is hardest to test
- **Use the techniques of the previous slides to make R as testable as you can, e.g.:**
  - Replace each pronoun with a specific name of an entity
  - Give measureable quantities
  - Measure representative sample of users

- Post original & improved version of R in Teams chat

FURPS+ Requirement Classification

- **F:** Functional (everything else = “non-functional”)
  – Features, capabilities, security
- **U:** Usability
  – Human factors, help, documentation
- **R:** Reliability
  – Frequency of failure, recoverability, predictability
- **P:** Performance
  – Response time, throughput, accuracy, availability, resource usage
- **S:** Supportability
  – Adaptability, maintainability, internationalization, configurability

– CL, page 57

The Many “+” of FURPS+

- **+:** Implementation
  - Resource limitations, languages, tools, hardware, etc.
- **+:** Interface
  - External systems
- **+:** Operations
  - System management
- **+:** Packaging
  - Shrink-wrapped, etc.
- **+:** Legal
  - License

Programming Language Used

- Often dictated by customer’s existing systems or expertise

- Using language X becomes a requirement
  – Example: Bank X uses .Net based systems, requires new software system to use a managed .Net language

Eliciting and Structuring Requirements

- “Goal-driven requirements engineering”

  -- HVV, pages 234 - 235

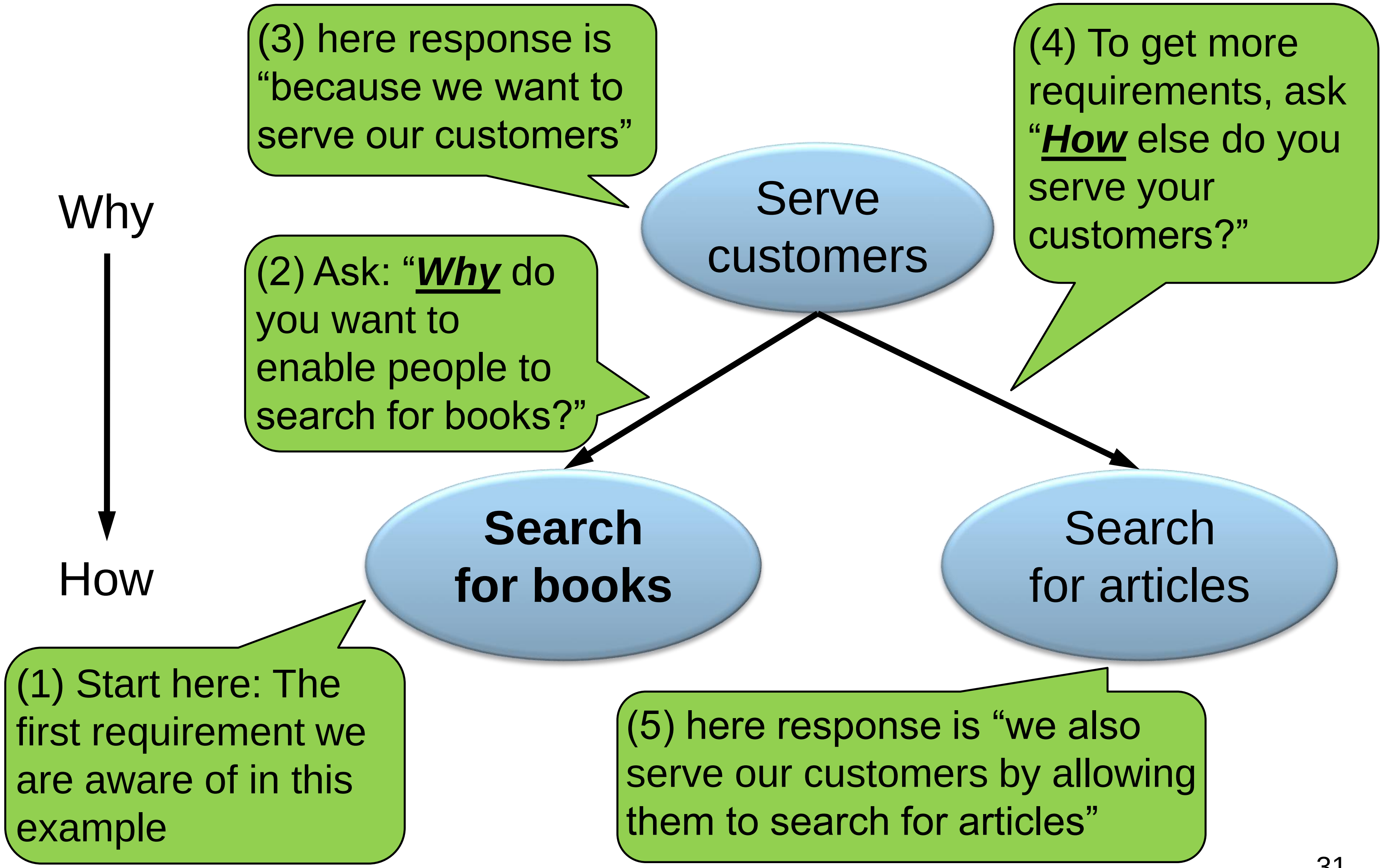
- Assume you have already elicited one requirement
  - **Example: “Search for books”**

- Elicit higher-level requirements by asking: **why?**
  - Example: “Serve customers”
- High-level requirements are called goals
  - Example: “Serve customers” is a goal

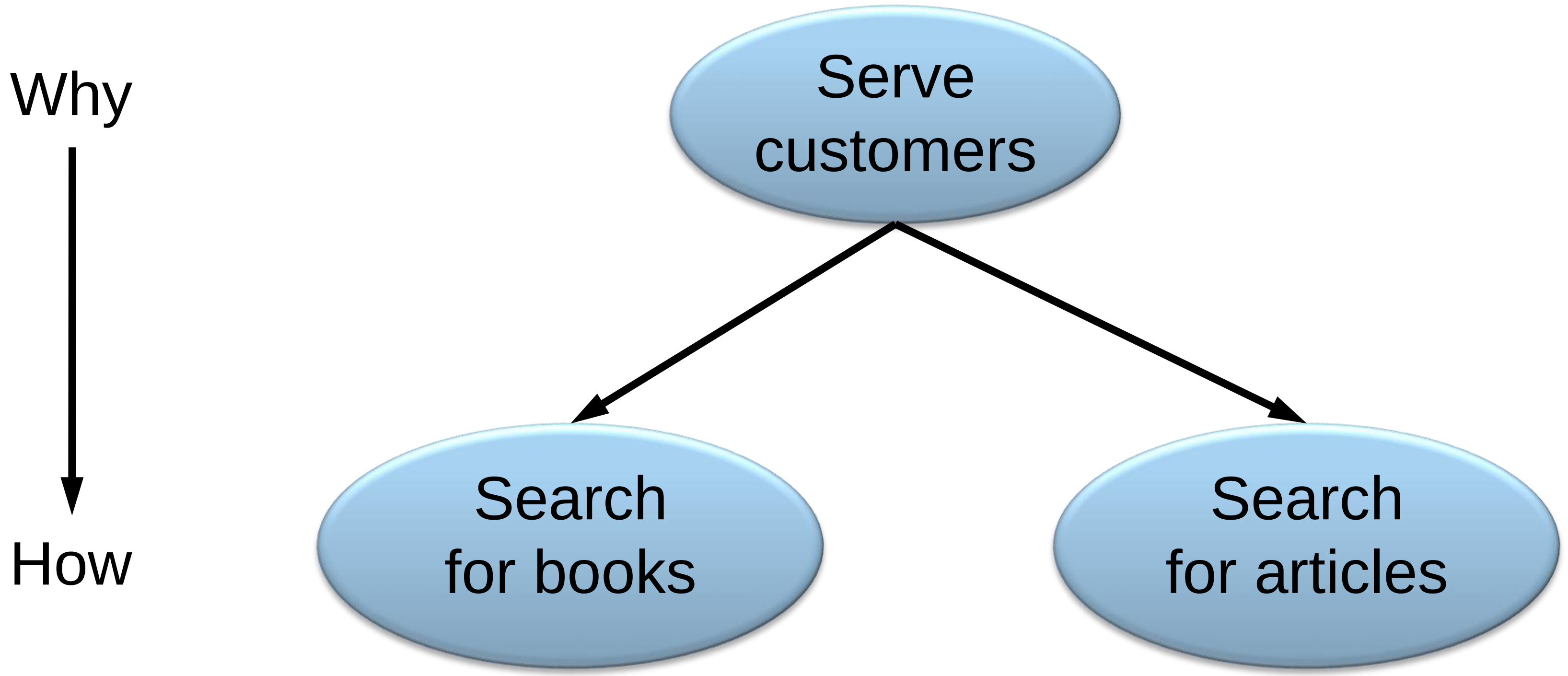
Goal-Driven Requirements Engineering

- Elicit lower-level requirements by asking: **how?**
  - Example: “Search for books” OR “search for articles”
- Refinement: Decompose requirement R into sub-requirements S1, .., Sn
  - Sub-requirements S1, .., Sn together satisfy parent requirement R
- Requirements + connections form a graph
  - Requirements on same level can be connected via AND, OR, etc.

Goal-Driven Requirements Engineering: Example


Why
How
(3) here response is “because we want to serve our customers”
Serve customers
(4) To get more requirements, ask “***How*** else do you serve your customers?”
(2) Ask: “***Why*** do you want to enable people to search for books?”
Search for books
Search for articles
(1) Start here: The first requirement we are aware of in this example
(5) here response is “we also serve our customers by allowing them to search for articles”

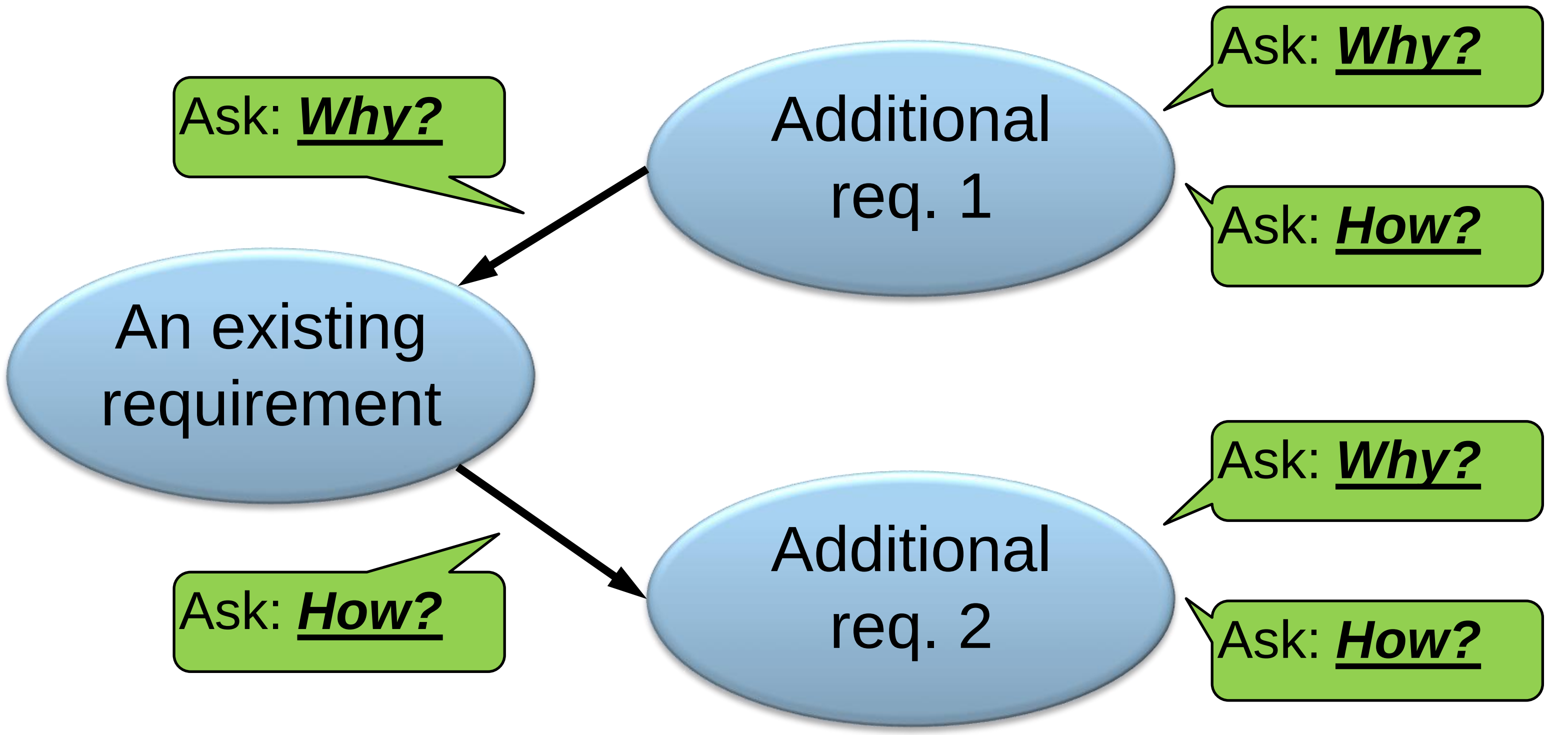
Resulting Graph



IN-CLASS EXERCISE: DISCOVER ADDITIONAL REQUIREMENTS

In-Class Exercise: Requirements

- Get together with your team (breakout room)
- Formulate ***why*** and ***how*** questions as conversation starters for discovering additional requirements :



- Share your graph (e.g., via Teams Whiteboard)

Managing Conflicting Viewpoints

- Different stakeholders may have viewpoints that conflict with each other
- Goal: Recognize and manage such conflicts in requirements engineering

-- HVV, pages 236 - 237

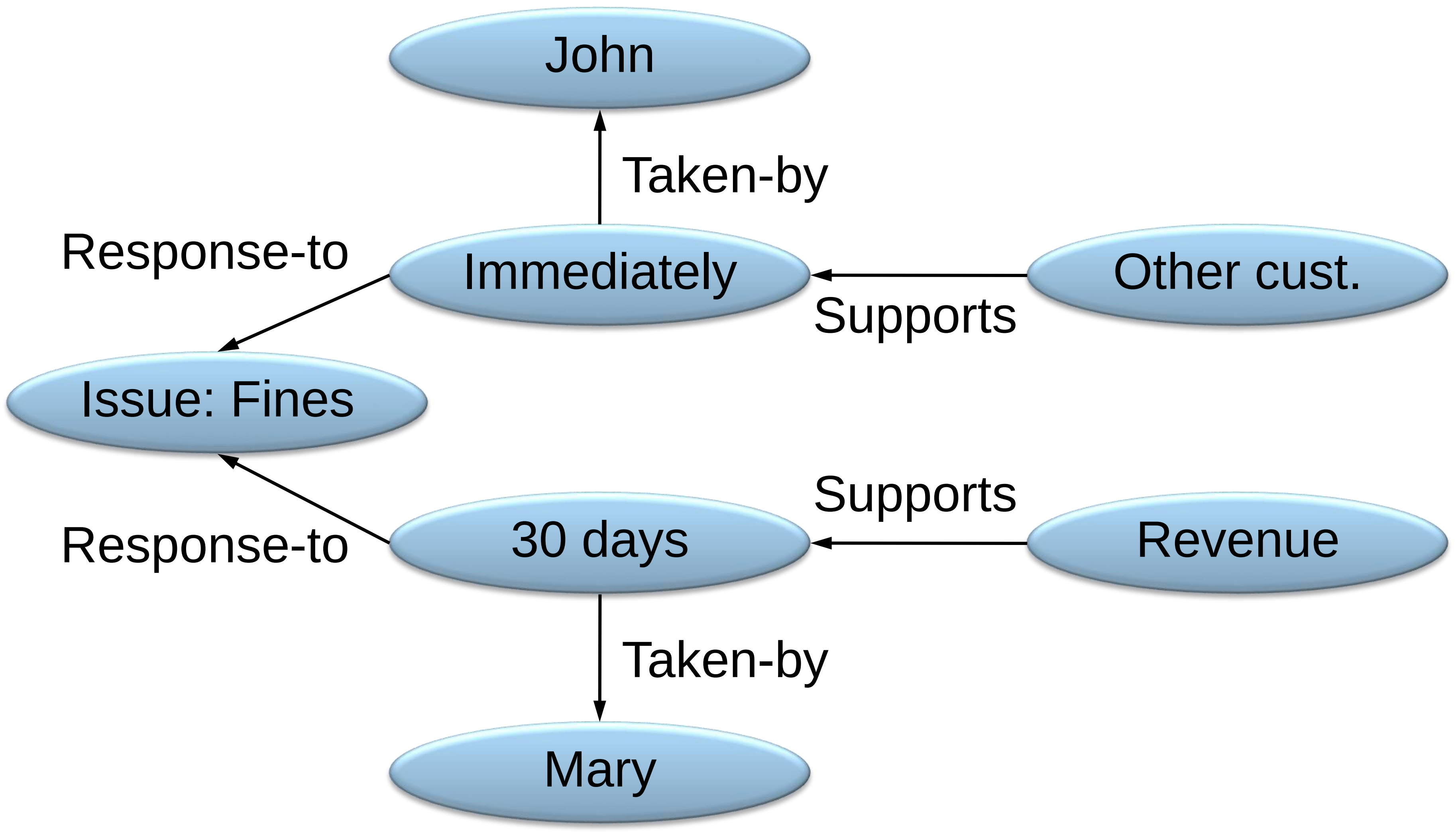
- Example: In library system, when warn a customer that an item is overdue (and incurs overdue fines)?
  - John: Immediately, else other customer cannot get item
  - Mary (manager): Only after 30 days, need to maximize revenue from fines

Viewpoints

- Capture conflicts in a graph:
- Node type: Examples
  - **Issue:** “When warn a customer that an item is overdue?”
  - **Position:** “Immediately”, “after 30 days”
  - **Argument:** “Better for other customers”, “max revenue”
  - **Stakeholder:** John, Mary

- Edge types (directed):
  - **Response-to:** Position x Issue
  - **Supports:** Argument x Position
  - **Taken-by:** Position x Stakeholder

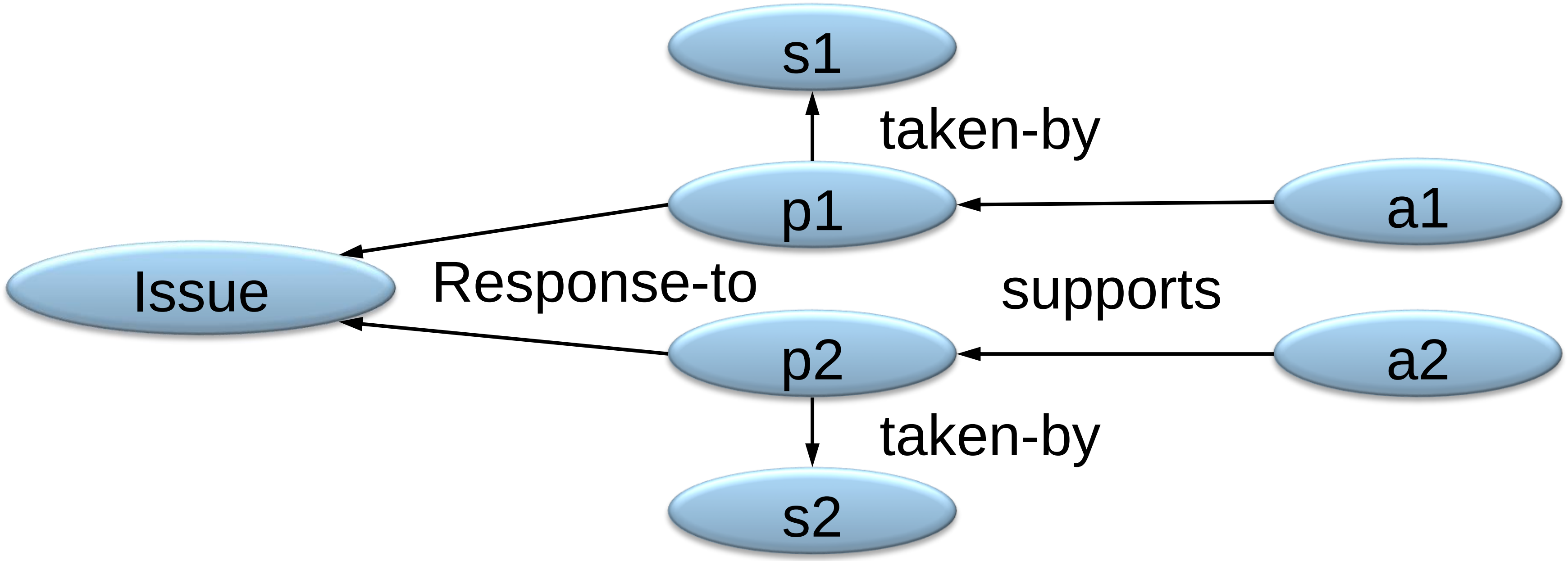
Viewpoints Example



IN-CLASS EXERCISE: MANAGE CONFLICTING VIEWPOINTS

In-Class Exercise: Viewpoints

- Get together with your team (breakout room)
- Pick **one requirement** of your class project on which stakeholders hold **different viewpoints**
- Document viewpoints (following is for 2 viewpoints):



- Share your results (e.g., via Teams Whiteboard)

In-Class Exercise, Posting in Chat

- Pick one requirement (issue) of your class project on which 2+ stakeholders hold different viewpoints
- You may post your results in the chat using the following template:
  - Issue: …
  - Stakeholder 1: …
  - Position 1: …
  - Argument 1: …
  - Stakeholder 2: …
  - Position 2: …
  - Argument 2: …