HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Project Envisioning & PO Tools

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

Image by *stavos* from flickr: https://www.flickr.com/photos/stavos52093/14315443921/ (CC BY-NC-ND 2.0)

Agenda

1. **Project Envisioning & PO Tools**
   - Product Owner Goals
   - Wireframing
   - Working Backwards
   - Impact Mapping
   - Product Owner Workflow
   - MVP
2. Value-based Software Engineering

Product Owner Goals

PO: Represent product's stakeholders & voice of the customer

- Tools and concepts focused on communicating and analyzing **domain information & business value**
- Needs support to
  - **Synthesize information** gathered from customer
  - Create concept & understanding of what should be built
  - **Negotiate priorities** between stakeholder factions
  - **Provide communication** between teams and stakeholders
  - Find consensus within PO team

Product Owner Goals

The value of communicating business value

- What happens when teams **have autonomy** on how to realize a task but aren't informed of the business context?
- What happens when the **business value isn't sensibly defined**?

FYI: Roughly 20 tennis balls would fit into the can using a blender.

Product Owner Tools

Discover business value & develop a product vision

- **Interview** the stakeholders
  - Understand their challenges and domain problems
  - Pay attention to the time you and them talk
  - Distinguish **what was said & what was interpreted**
- Shadow and observe the stakeholders (if possible)
- Survey competitive products

- **Verify outcomes** with interviewees
  - Share structured transcripts and notes
  - Clarify uncertainties
  - Collect feedback

Product Owner Tools

Visualize and share a product vision

- **Visualizations** can help structure thoughts & share ideas
- Sketching may be the fastest way to share ideas

- **Wireframing (*page schematic / screen blueprint*)**
  - Graphic of application's **skeletal framework**, i.e. **"rough sketching"**
  - Expand your own understanding at **minimal cost** (i.e. no code)
  - Minimum necessary to communicate
  - Visualize **flow of application screens**

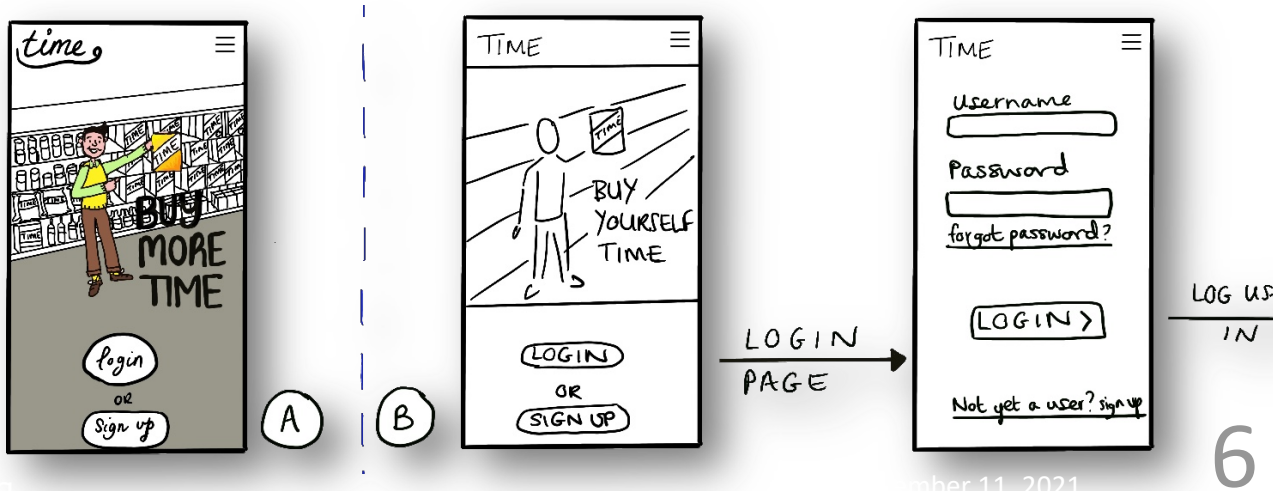
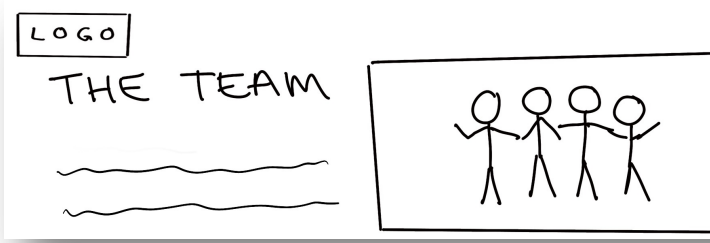


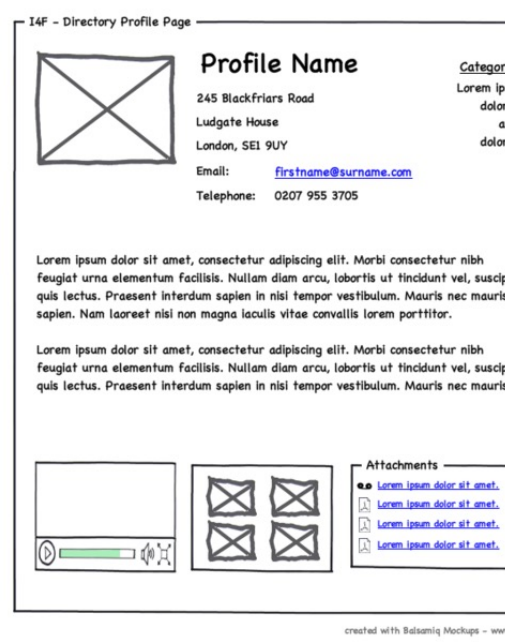
Image: https://github.com/dwyl/learn-wireframing

Product Owner Tools

Wireframing options

- **Paper-based sketches**
  - Fast to create, disposable
  - Can be automated/augmented by human actions
  - Really fast to change, even on demand in **customer meeting**
  - Share by taking & uploading images
- **Software sketching tools**
  - Choose whatever best allows collaboration
  - General purpose tools, e.g. Google Drawings/Slides or https://www.draw.io/
  - Specialized (online) tools, e.g. https://wireflow.co/, https://mydraft.cc/, https://wireframe.cc/





Product Owner Tools

"Work Backwards" to help define what needs building

- Start from the end goals to understand what is required at the start

- **1.** Write the **press release first!** (Before coding anything)
  - What does the product do, what are features and benefits
- **2.** Write **Frequently Asked Questions** (FAQ) document
  - Questions that came up when writing the press release
  - What are the product's limitations? What are potential issues?

Product Owner Tools

"Work Backwards" to help define what needs building

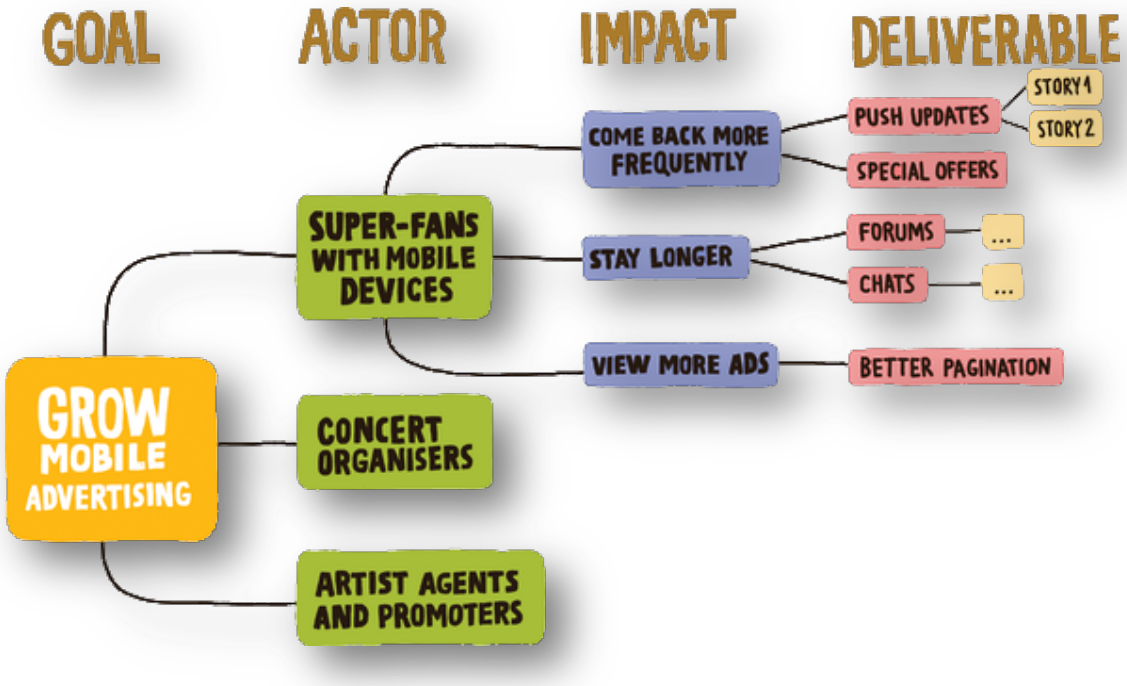
- **3.** Define the **customer experience**
  - Detail how a customer/user interacts with the product
  - Tell a story of someone solving a problem using the product.
- **4.** Write the **User Manual**
  - Typically three sections:
    - concepts, how-to, and reference (definition of key terms)
  - Everything users need to know
    - Multiple kinds of users → multiple manuals

Writing an entire manual might not be necessary for all applications.

Product Owner Tools

Impact Mapping

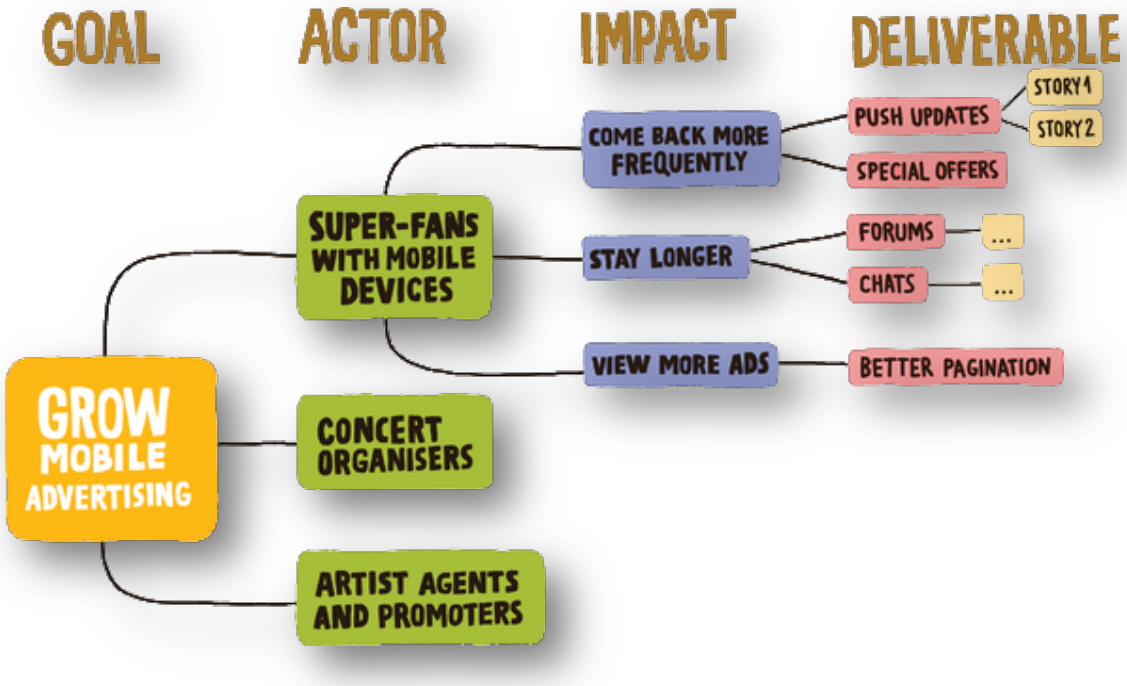
- **Hierarchical mind map** of goal & steps to deliver on it
- Find & show relationships between
  - Business goals (**why** are we doing this?)
  - Users/stakeholders (**who** is involved?)
  - Impact (**how** is behavior changed?)
  - Team deliverables (**what** is the scope?)



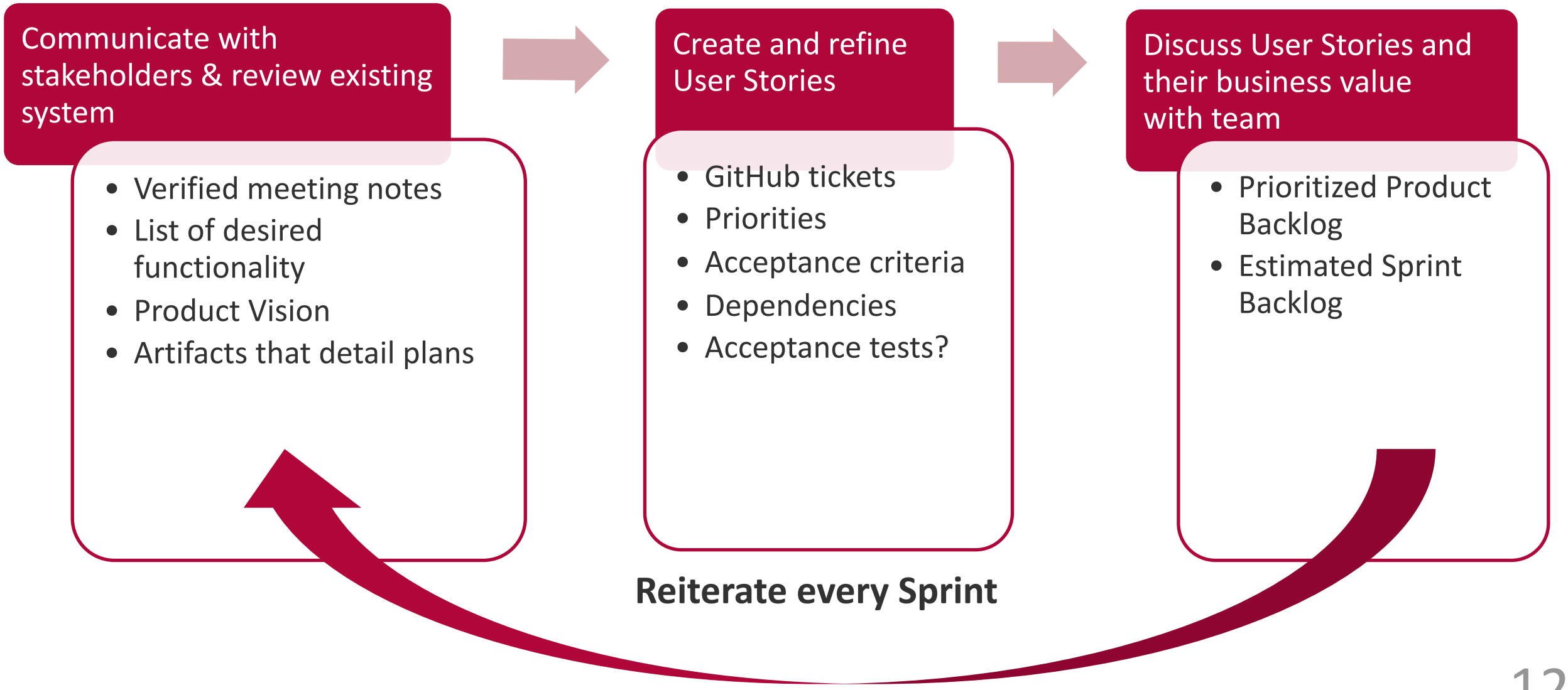
Product Owner Tools

Impact Mapping

- Concentrate on **most promising category**
  - No need to fill everything
- **Limitations**
  - Inspiration mainly drawn from previous hierarchical step
  - No focus on how behavior change affects other goals

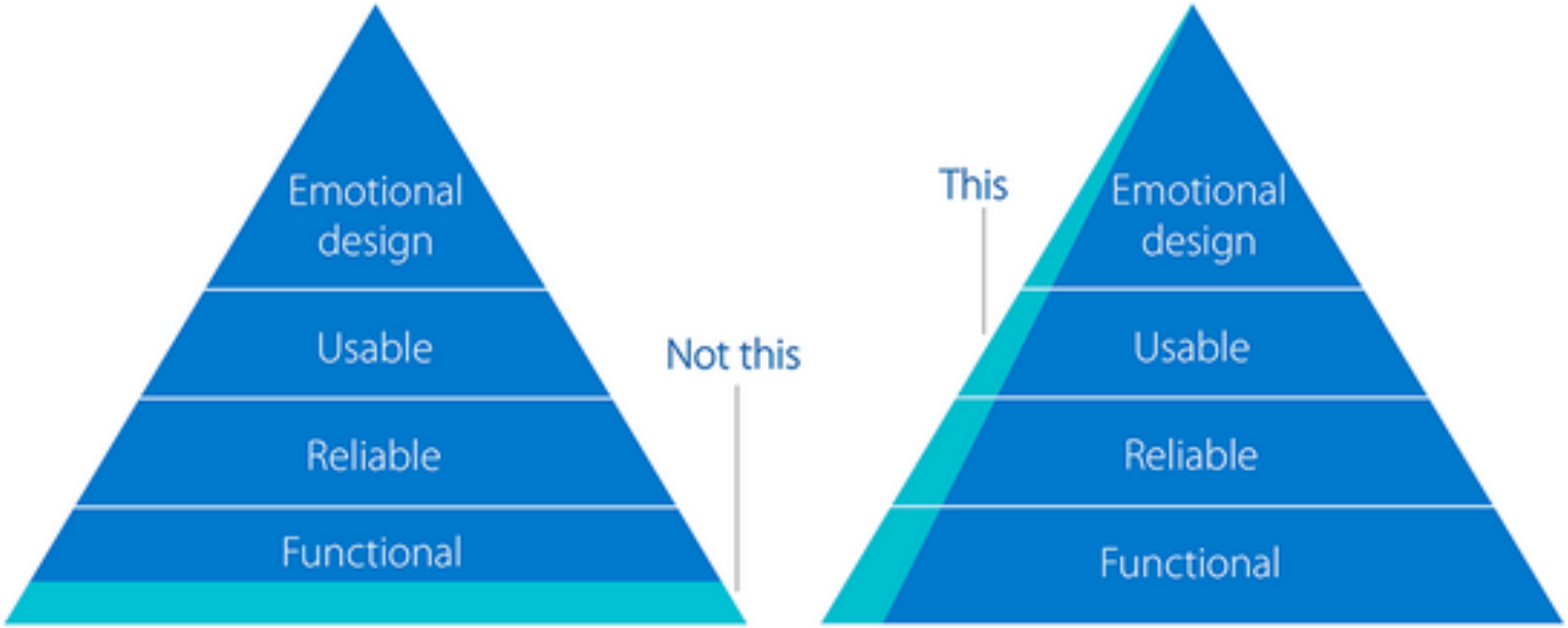


Product Owner Workflow



Minimum Viable Product (MVP)

Product with **just enough features** to satisfy early customers, and to **provide feedback** for further development.

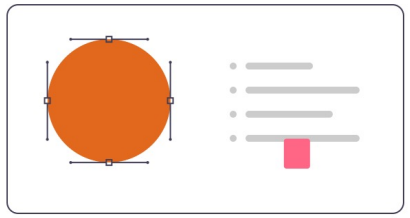


Author/Copyright holder: Jussi Pasanen. With acknowledgements to Aarron Walter, Ben Tollady, Ben Rowe, Lexi Thorn and Senthil Kugalur. Copyright terms and license: All rights reserved.
Source: https://www.interaction-design.org/literature/article/minimum-viable-product-mvp-and-design-balancing-risk-to-gain-reward

MVP (Dis-)Advantages

Advantages

- Early **user feedback**
  - Test initial understanding of user needs, test product hypothesis
  - Limited resources spent on MVP
- Move into **production early**
  - Software is developed for a reason, solve a problem!
  - Generate revenue
  - Entering a market first can be a competitive advantage



Challenges

- Definition of **minimally viable** *(why?)*
  - *Smallest possible* way to *meet the market need* with a *useful output*
  - Requires smart requirements management
- Requires early focus on usability, deployment, support, marketing

MVP Variants & Contexts

"Minimum Viable Product" used in many contexts. Some variants:

- Marketing MVP
  - Product to test the market that is being targeted
  - **Check demand assumptions**
- Technical Demonstration MVP
  - Prototype or **proof-of-concept**
  - Explore software designs
  - Prove that it will work using the technology
- "Must-Haves" MVP
  - Product with **only "the most important features"**
  - Might not be truly minimal in terms of effort
  - *Smaller version of full software? Is the main goal feedback collection?*

Allan Kelly, Allan Kelly Associates, https://techbeacon.com/mvp-broken-its-time-restore-minimum-viable-product

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Value-based Software Engineering

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

Image by *ripster55* from flickr: https://flickr.com/photos/ripster55/4738836471 (CC BY 2.0)

Agenda

1. Project Envisioning & PO Tools
2. **Value-based Software Engineering**
   - Definition
   - Requirements Prioritization
   - Basic Prioritization Schemes
   - MoSCoW
   - Prioritization Matrices

Value-based Software Engineering

> "Requirements are often analyzed in a value-neutral environment"
> — Barry Boehm [1]

- Tracking of project cost and schedule, not **stakeholder value**
- Developers write code that fulfills requirements (exactly?)

(Business) value-based Requirements

- 80% of the productivity/value can be achieved by doing 20% of the tasks (**Pareto principle**) [2]
- ➡ Focus on delivered **stakeholder value**

[1] Barry Boehm. 2003. *Value-based software engineering: reinventing*. SIGSOFT Software Engineering Notes 28, 2. DOI: 10.1145/638750.638775

[2] Koch, Richard. 1998. *The 80/20 Principle : the Secret of Achieving More with Less*. New York: Doubleday. ISBN 9780385491747.

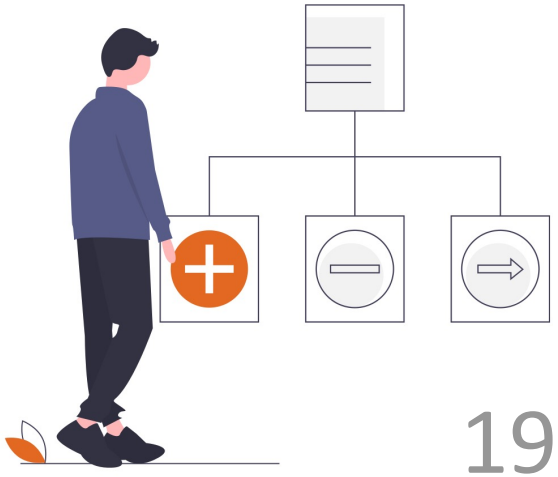
Requirements Prioritization

Focus on stakeholder/business value

- Identify the system's success-critical stakeholders
- Obtain their value propositions & understand the (technical) details
- **Estimate/investigate value** of specific requirements
- **Prioritization**: determine order, separate what must be done now and what later

Factors influencing prioritization

- **Value**
- **Risk or uncertainty**
- **Dependencies**



Basic Prioritization Schemes

Relative Ranking

- Numbered list of features

Monopoly Money / 100-Point Method

- Provide project budget as banknotes/points
- Stakeholders distribute the budget among features
- Use feature level: Tasks w/o immediate value (e.g. documentation) might be omitted otherwise
- One and only priority -> everything on one feature

Dot voting / Multi-voting

- Everyone given a limited number of dots (e.g. 20% of the number of all options)
- Place votes on features (optional: max. 1 dot/feature/participant)

Introducing scarcity makes prioritizing more clear

MoSCoW Prioritization

Reach **common understanding** with stakeholders on the importance of each requirement

Is a simple scheme like 'prio 1,2,3' more suitable for the beginning?

MoSCoW: Must have, Should have, Could have, and Won't have

- Categories with **semantics** instead of *high*, *med* and *low*
- Get customers to better understand the impact of setting a priority
- Try to deliver all the *Must haves*, *Should haves* and *Could haves*
- *Should haves* and *Could haves* will be removed first if plan for delivery is threatened

Dai Clegg, Richard Barker. 1994.
Case Method Fast-Track: A RAD Approach. Addison-Wesley. ISBN 978-0-201-62432-8.

MoSCoW Prioritization

Must have

- Critical for success of delivery
- If only a single ***Must have* is missing, project delivery is considered a failure**

Could have

- Desirable, but not necessary, **included if time and resources permit**
- Could improve customer satisfaction for little development cost

Should have

- Important, but **not necessary in the next iteration**
- Can be as critical as *Must haves*, maybe not as time-sensitive or workaround exist

Won't have (this time)

- Lowest-payback items, **outside of current scope**
- Either dropped or reconsidered for inclusion in a later timebox

Dai Clegg, Richard Barker. 1994.
Case Method Fast-Track: A RAD Approach. Addison-Wesley. ISBN 978-0-201-62432-8.

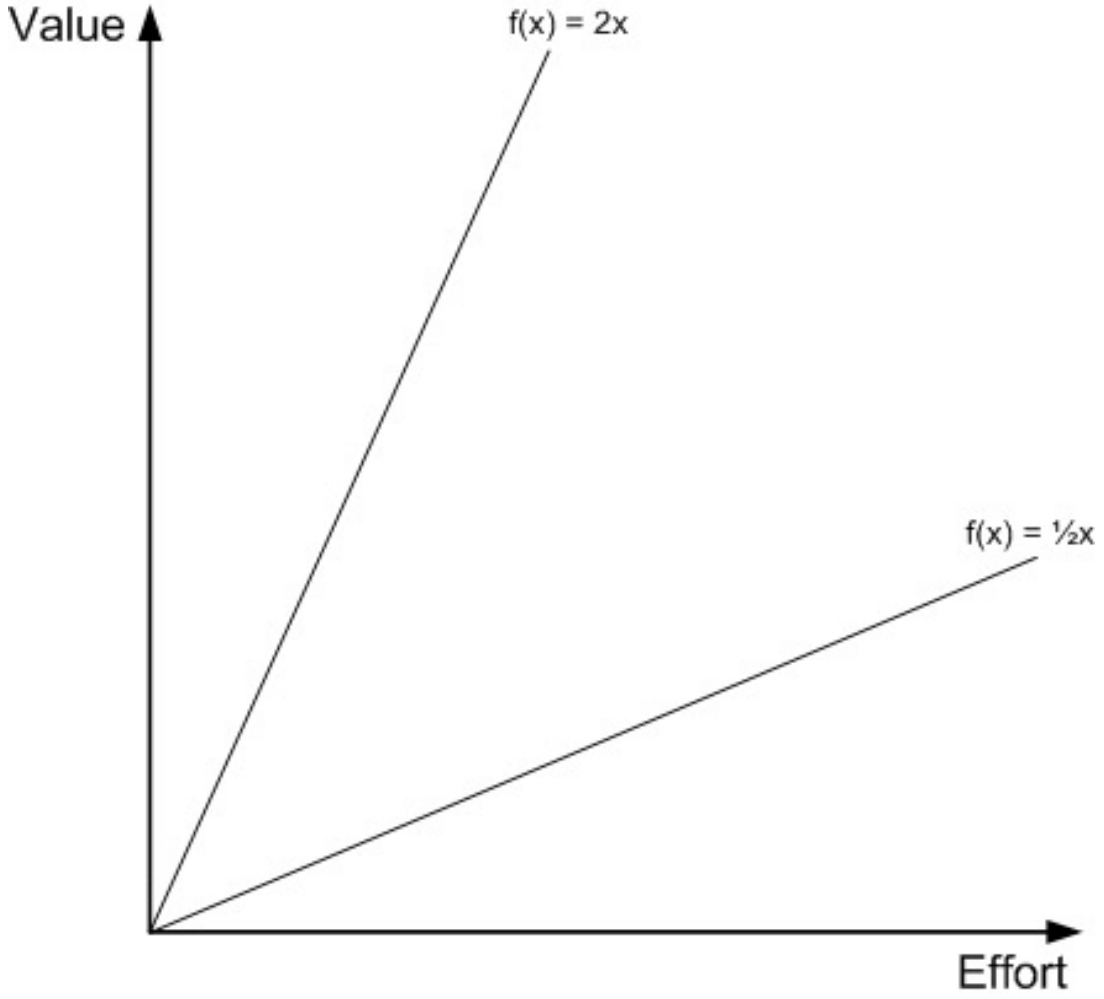
MoSCoW Prioritization

Challenges and pitfalls

- **Lack of documented rationale**/structured process for deriving priorities
  - What aspect makes a requirement a *Must have* and not *Should have*?
  - How was it decided that something was *Won't have?*
- Lack of specific **time information**
  - Are *Won't have* requirements not in the next delivery or never?
- No guidance/process for **technical aspects**
  - Handling refactoring and its priorities
  - Assigning status to bug fixes

Wiegers, Karl; Beatty, Joy (2013).
Software Requirements. Washington, USA: Microsoft Press. pp. 320–321.

Prioritization Matrices



Value vs. Cost Matrix ("Bang for the Buck")

- Score requirements on **value** & **implementation cost**
- Common cost functions: engineering effort or complexity
- Implement: Above 2x
- Skip: Below 1/2x
- In-between: **Review**

Challenges

- Whole truth?
- Beware of dependencies!
- Keep in sync

https://foldingburritos.com/product-prioritization-techniques/

Prioritization Matrices

Lean Startup 2x2 Matrix

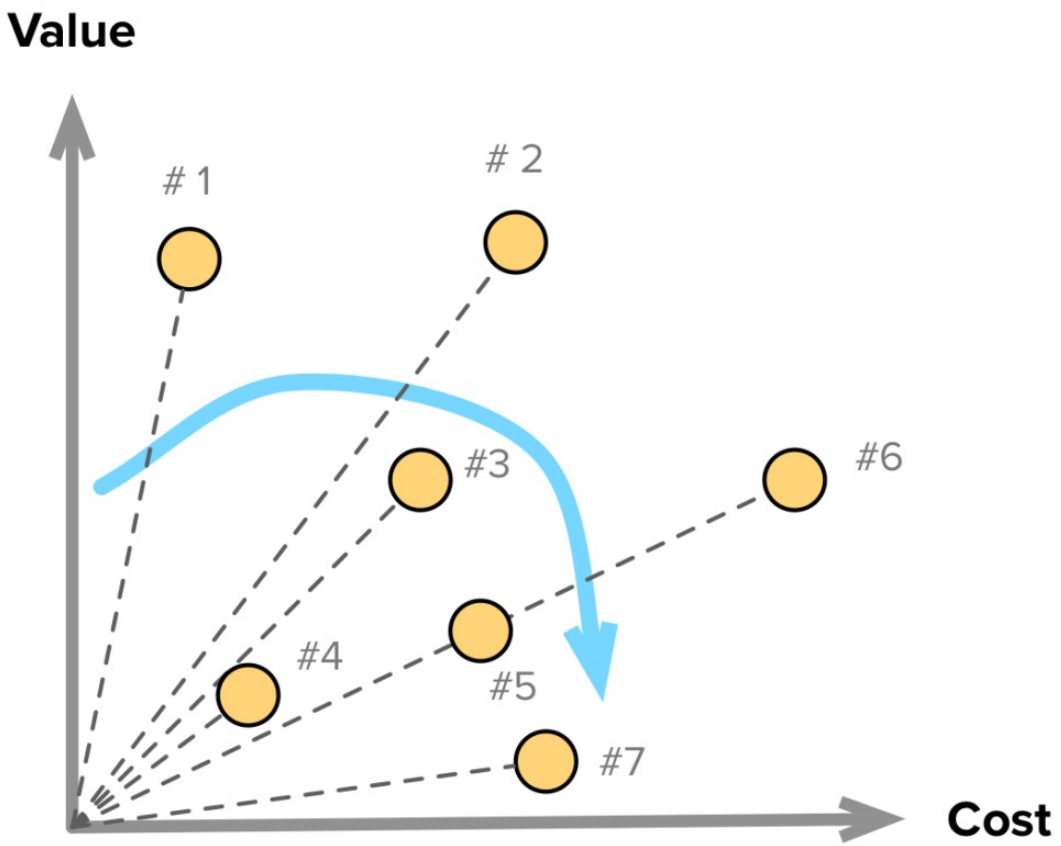
- **Do first**: Quick Wins
- **Do second**: Big Bets
- Think about Maybes
- Try to **avoid Time Sinks**

(Eisenhower Matrix)

- These are not new ideas!
- Eisenhower: 1954
- Criteria **(un)important** & **urgent/not urgent**

Pavel Kukhnavets. 24.04.2018. *Value/Effort Matrix: Lean Prioritization for Product Managers.* https://hygger.io/blog/lean-prioritization-approach-ongoing-pm-issues/

Prioritization Matrices



Prioritization challenges

- Steeper slope -> higher priority
- Low-value and low-cost items should be **balanced**

> "If you use time-to-build to prioritize [...], you'll end up with a product full of easy solutions."
> — Teresa Torres

Image: https://foldingburritos.com/product-prioritization-techniques/, https://www.producttalk.org/2014/02/the-simplest-and-most-important-question-you-face-as-a-product-leader/

Summary

Project Envisioning & PO Tools

- Product Owner Goals
- Wireframing
- Working Backwards
- Impact Mapping
- Product Owner Workflow
- MVP

Value-based Software Engineering

- Definition & meaning
- Basic Prioritization Schemes
  - Monopoly Money
- Requirements Prioritization
- MoSCoW
- Prioritization Matrices

We presented some (fairly basic) tools, explore the space!