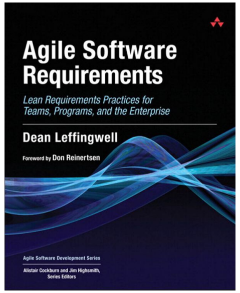
CE Department

Software Requirements Engineering

40688

These slides are designed to accompany Agile Software Requirements (2011) by Dean Leffingwell and support the university course Software Requirements Engineering, instructed by Mehran Rivadeh. Created and designed by Mahnaz Rasekhi.


Agile Software Requirements
Lean Requirements Practices for Teams, Programs, and the Enterprise
Dean Leffingwell
Foreword by Don Reinertsen
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors

Agile Software Requirements (2011)
Dean Leffingwell

Stakeholders, User Personas, And User Experiences

Chapter 7

Mehran Rivadeh
mrivadeh@sharif.edu
Software Requirements Engineering
October 2025 - Fall 1404 - SUT

Contents

Introduction

Introduction

So far we have studied that

- ➔ User story is the primary artifact for identifying system behaviors that deliver value to the customer.
- ➔ **Engaging** a **user** in a dialogue about **how they use** the **system**, and **what benefit** they **derive**, is a straightforward process.
- ➔ **This process** becomes significantly **easier** and more **effective** when your **solution** is **already deployed** and you can **directly engage with actual users**.

Introduction

But what if you are building a **new application** or **service**?

➔ Who are the **users**?

➔ What **key stakeholders** use the system result but don’t actually use the system themselves?

➔ **How** do we **know who they are**, and **how** do we **design for them**?

➔ If we design a system that works for our **direct** and **indirect users**, are there still other stakeholders whose needs must be met?

Introduction

But what if you are building a **new application** or **service**? (Cont.)

- ➔ What about the **project sponsors**?
- ➔ If we develop a system that meets the needs of those mentioned, are we on solid footing?
- ➔ What about these other **stakeholders** who **have** a **material interest in**:
    - ◆ How the system is developed
    - ◆ What it costs
    - ◆ How it is deployed
    - ◆ When it is deployed

Contents

1. Stakeholders

Stakeholders

- ➔ It is clear that **users** are indeed **key project stakeholders** and we must, in large part, **design for them**.
- ➔ But the issues are much deeper than that.
- ➔ **To** help assure **success**, we will have to take a much broader look at **who all** the **stakeholders** of our proposed system are.
- ➔ There are two different classes of stakeholders:
  - ◆ System Stakeholders
  - ◆ Project Stakeholders

System Stakeholders

- ➔ A system stakeholder is anyone who:
  - ◆ Directly uses the system
  - ◆ Works with the results of those who use the system
  - ◆ Will be impacted by the deployment and operation of a system

Example:

Users, operators, users of reports, data, signals, and other outputs of the system

Managers, purchasers, and administrators for those users

Support and help-desk staff

Developers working on other systems

Project Stakeholders

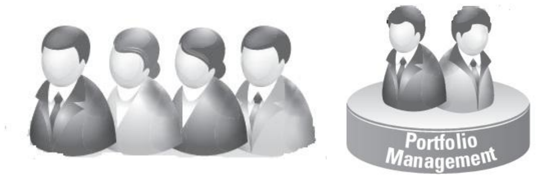
- A project stakeholder is anyone who:
  - Has a vested interest in the budget and schedule
  - Has a vested interest in understanding how the product/system/solution is developed
  - Will be involved in marketing, selling, installing, or maintaining the system

Example:

The project sponsors, project management

Portfolio management, executives

Financial governance staff and so on.



Contents

Voice of Stakeholders: Product Owner

- ➔ Each project stakeholder will have their own vision, requirements, and priorities.
- ➔ The **product owner**’s primary job is to **merge** these **diverse stakeholder voices** into a single prioritized backlog for the team.
- ➔ They can do this by **facilitating** or **leading**, or some appropriate mix of each.

Voice of Stakeholders: Product Owner

Facilitating

- ➔ Oftentimes, the product owner is the facilitator of a process intended to converge diverse opinions into a single product vision.
- ➔ There are often no clear-cut, black-and-white answers as to which potential solution is more valued or important than the other; each has pros and cons.
- ➔ The product owner works to help each stakeholder find common ground so they can accept a combined solution.

Voice of Stakeholders: Product Owner

Leading

- ➔ Other times, the product owner makes decisions for stakeholders based on their personal, expert knowledge or experience in the industry.
- ➔ These decisions may not be advocated by any particular stakeholder but rather are driven from a move in markets, a change in competition, or other trends that the product owner feels are material to the solution requirements.

Contents

Levels of Stakeholder Involvement

- ➔ **Understanding** the **degree** of **stakeholder involvement** is one key to **building consensus**, while still making fast, agile, forward progress.
- ➔ When you know how involved each stakeholder is:
  - ◆ You can **include them just enough** to make them feel heard.
  - ◆ You **avoid unnecessary delays** by not involving everyone in every decision.
  - ◆ You **balance teamwork** with speed and keep things move quickly.

Example:

Imagine you're planning a school event. If you know the principal wants to be informed but not make decisions, and the teachers want to help choose activities, you can involve each person the right amount. That way, everyone feels respected, and you can still plan the event quickly.

Levels of Stakeholder Involvement

The following list applies to some stakeholders.

➔ **They should be kept informed.**

- Some stakeholders simply need to know the status of the project and be informed of decisions that impact it.
- They may or may not have input on those decisions but may influence those that do.

➔ **They should be consulted.**

- Some stakeholders have a specific area of expertise that aids in the definition or building of the product.
- They should be involved in decisions within their area of expertise.
- Such as subject-matter experts, marketing analysts, architects, and user-interface designers.

Levels of Stakeholder Involvement

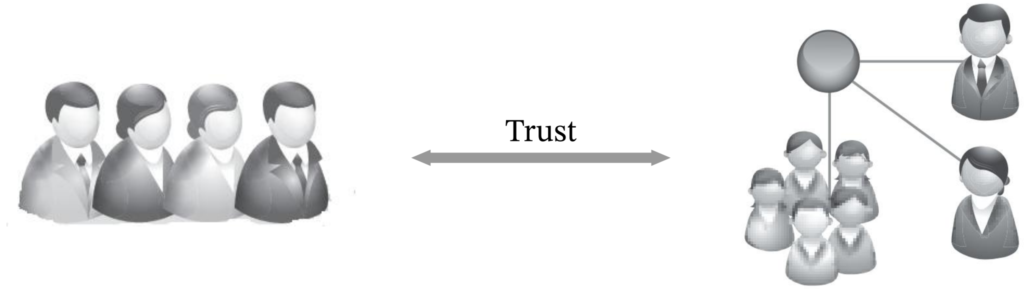
The following list applies to some stakeholders (Cont.).

- **They are partners in development.**
  - Some stakeholders are partners in the process.
  - Other product or business owners, other development teams, business or requirements analysts, and providers of solutions that the system interacts with.
- **They are in control of outcomes.**
  - Some stakeholders make final decisions for the solution.
  - These may include executives, release managers, business owners, and key customers who will be using the solution.

Contents

Building Stakeholder Trust

- ➔ Projects run a lot smoother when the stakeholders trust the product owners and the teams and when that trust is reciprocated.
- ➔ The easiest way to build trust is to communicate all aspects of the project, clearly and without prejudice, to all stakeholders.
- ➔ Visibility and transparency are key.



Building Stakeholder Trust

- The product owner:
  - Should describe the **rationale** behind any key decisions.
  - Should explain any **weightings** or **factors used to** arrive at a **decision**.
  - Should conduct all **activities in** an open and **honest manner**.
- No hidden agendas— just the current facts of the market, system, and development context.

Contents

Stakeholder Interactions

- ➔ To be successful, all project stakeholders must actively work with the team to achieve the team’s goals.
- ➔ This creates a number of expectations and implications:
  - ◆ Timely decisions are needed.
  - ◆ Active participation is needed.
  - ◆ Teams must take an enterprise view.
  - ◆ Production and support staff should be involved from the start.
  - ◆ Plan for system maintenance.

Contents

2. Identifying Stakeholders

a. Identifying Project Stakeholders

b. Identifying System Stakeholders

c. Classifying System Stakeholders

d. Understanding System Stakeholder Needs

Identifying Stakeholders

- The following questions can be used to help identify stakeholders:
  - Who needs to be consulted on the scope of this project?
  - Who has input to the budget and schedule?
  - Who ultimately manages the business relationship between the teams and the customer?
  - Who will determine how and when the system is released to customers?
  - Who can support or harm this project politically?
  - What partners are dependent on our system?
  - Who cares about the process we use to develop the system?

Contents

2. Identifying Stakeholders

Identifying Project Stakeholders

Partner

Product Expectations:

- ➔ The product interfaces will remain the same or as agreed to
- ➔ A stable product integration
- ➔ Backward compatibility

Project Expectations:

- ➔ Wants input in the prioritization of features
- ➔ To be notified with changes in the project schedule or prioritization
- ➔ Changes that impact their product

Identifying Project Stakeholders

Sales / Marketing

Product Expectations:

- ➔ New features to be available as promised
- ➔ To have many new checks to select on request for proposal (RFP) responses
- ➔ Wants an understanding of the roadmap
- ➔ Wants input into when and how the system will be released
- ➔ Wants to be able to intelligently articulate product benefits

Identifying Project Stakeholders

Sales / Marketing (Cont.)

Project Expectations:

- ➔ Wants input in the prioritization of features
- ➔ Expects to be notified when prioritization changes, especially when it impacts their customers
- ➔ Wants to know whether there any delays in the project schedule

Identifying Project Stakeholders

Operation

Product Expectations:

- ➔ Expects detailed documentation about how to install the product and dependencies on the product
- ➔ Has clear expectations for system reliability and performance

Project Expectations:

- ➔ Wants to be informed about the project status
- ➔ Wants to be more involved toward the end of the project as details of the installation and dependencies are known

Identifying Project Stakeholders

Support

Product Expectations:

- ➔ Expects a high-quality product that can be easily supported
- ➔ Wants the error management system to help the customer resolve their own problem
- ➔ Wants to be able to resolve issues quickly when there is a problem

Project Expectations:

- ➔ Wants to be informed about project status, especially when it relates to support
- ➔ Expects to have input in prioritization of support issues

Identifying Project Stakeholders

Sponser

Product Expectations:

- ➔ Wants to understand the team has effective processes for understanding requirements

Project Expectations:

- ➔ Wants to be continually apprised of schedule, budget, and status
- ➔ Wants to know development is in accordance with governance

Identifying Project Stakeholders

Development Management

Product Expectations:

- ➔ Wants to know system will be fit for intended purpose

Project Expectations:

- ➔ Expects system to be developed in accordance with budget and schedule guidelines
- ➔ Wants to know resources stay in line with strategic investment themes

Identifying Project Stakeholders

Security

Product Expectations:

- ➔ Expects the system to be secure through following security coding and testing practices
- ➔ Expects all relevant security standards to be adhered to

Project Expectations:

- ➔ Consulted about security issues Reviews designs for security flaws Expects to have input in security prioritization

Contents

Identifying System Stakeholders

- The following questions can be used to help identify system stakeholders:
  - Who will be directly using the system?
  - Who will be using the results of those who use the system?
  - Who will be responsible for supporting the system?
  - What other systems will our system interact with?
  - What interfaces must it provide?
  - Who can provide guidance on the functionality and system qualities for the system?
    - Like: Usability, reliability, performance, and supportability

Contents

2. Identifying Stakeholders

Classifying System Stakeholders

- It can also be helpful to categorize system stakeholders in one of three categories:
  - **First degree:** Direct users of the system
  - **Second degree:** People who work with the results of those who work with the system
  - **Third degree:** People who install, deploy, or support the system

Classifying System Stakeholders

Example: **Online Shop Platform**

First-degree stakeholders (Direct users):

- Store owners who manage products, prices, and orders.
- Customers who browse and buy items.
- Admins who handle shipping, inventory, and customer service.

Second-degree stakeholders (Work with the results):

- Marketing teams who use sales data to run campaigns.
- Accountants who analyze transaction reports for financial planning.
- Suppliers who respond to order trends and restock inventory.

Classifying System Stakeholders

Example: **Online Shop Platform** (cont.)

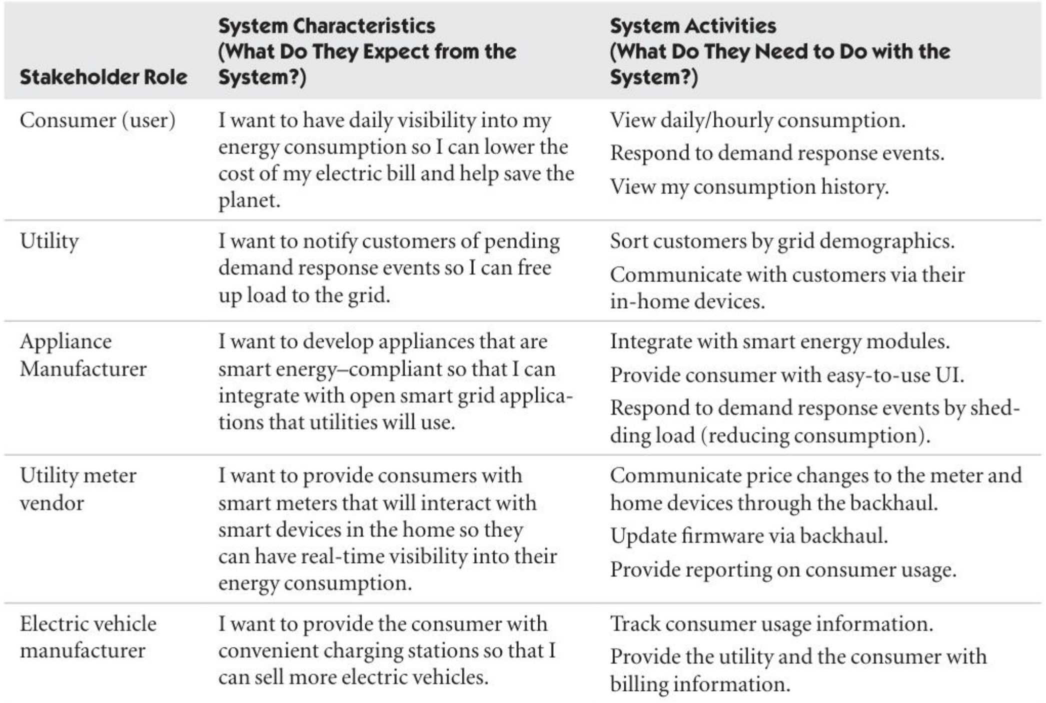
Third-degree stakeholders (Install, deploy, support):

- Developers who build or customize the platform.
- IT support teams who maintain servers and fix technical issues.
- Hosting providers who ensure the site stays online and secure.

Contents

2. Identifying Stakeholders

Understanding System Stakeholders Needs



Stakeholder Role	System Characteristics (What Do They Expect from the System?)	System Activities (What Do They Need to Do with the System?)
Consumer (user)	I want to have daily visibility into my energy consumption so I can lower the cost of my electric bill and help save the planet.	View daily/hourly consumption. Respond to demand response events. View my consumption history.
Utility	I want to notify customers of pending demand response events so I can free up load to the grid.	Sort customers by grid demographics. Communicate with customers via their in-home devices.
Appliance Manufacturer	I want to develop appliances that are smart energy–compliant so that I can integrate with open smart grid applications that utilities will use.	Integrate with smart energy modules. Provide consumer with easy-to-use UI. Respond to demand response events by shedding load (reducing consumption).
Utility meter vendor	I want to provide consumers with smart meters that will interact with smart devices in the home so they can have real-time visibility into their energy consumption.	Communicate price changes to the meter and home devices through the backhaul. Update firmware via backhaul. Provide reporting on consumer usage.
Electric vehicle manufacturer	I want to provide the consumer with convenient charging stations so that I can sell more electric vehicles.	Track consumer usage information. Provide the utility and the consumer with billing information.

➔ Once the stakeholders have been identified and classified, we need to establish the high-level expectations for the system and to start to identify some of the activities they will be doing with the system.

Contents

3. User Personas

a. Primary And Secondary User Personas

b. Finding Personas With User Story Role Modeling

User Personas

- Designing systems that make simple things simple and complex things possible is an important skill for any teams whose systems have significant interaction with users.
- In addition to thinking of the user generically (“As a user, I can...”), user personas provide a means of further refining the approach to the user to make sure that the needs of different types of users are met.

Primary And Secondary User Personas

Alan Cooper:

➔ **Primary Personas** represent users with specific needs that can be satisfied only with a user interface designed specifically for them.

➔ **Secondary Personas** are people who also use the system but can use an interface that was designed for a primary persona.

Identifying personas helps development teams create user experiences that are optimum for a certain class of user.

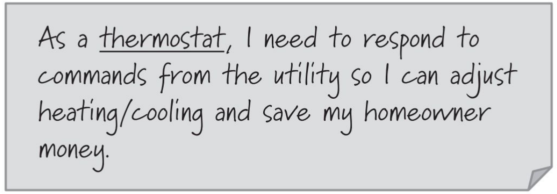
Primary And Secondary User Personas

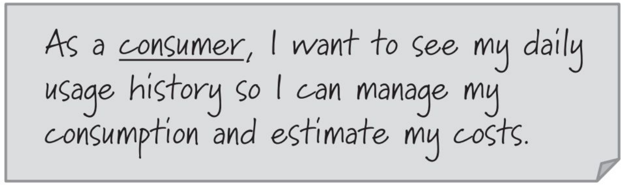
As a <role>, I can <activity> so that <business value>.

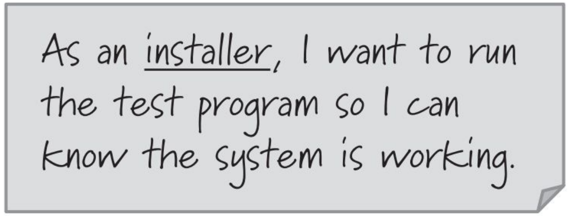
- ➔ The **<role>** element identifies the role the user is playing as they interact with the system.
- ➔ In some systems, there is only one role and the role element of this form doesn’t add much value.
- ➔ In other systems, many user roles can interact with a system, or a subsystem of a larger system.
- ➔ Developing a set of user stories helps us identify roles, which in turn helps identify personas.

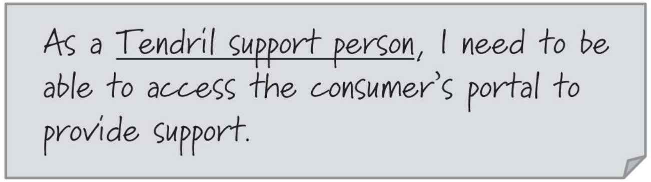
Primary And Secondary User Personas

➔ Example: our case study system has a plethora of user roles:


As a thermostat, I need to respond to commands from the utility so I can adjust heating/cooling and save my homeowner money.


As a consumer, I want to see my daily usage history so I can manage my consumption and estimate my costs.


As an installer, I want to run the test program so I can know the system is working.


As a Tendril support person, I need to be able to access the consumer's portal to provide support.

Primary And Secondary User Personas

Cooper provides a number of additional guidelines for thinking about personas:

- Don’t “make up” personas out of thin air.

  Discover them as a byproduct of your requirements discovery and user story writing process.

- Develop a specific, individual persona.

  An actual person who you can interview, interact with, and come to understand.

  Understand their abilities, background, environment, and usage of the system.

- Identify the persona’s goals.

  you can see what your system needs to do and not do.

Primary And Secondary User Personas

Cooper provides a number of additional guidelines for thinking about personas (Cont.):

- Secondary personas are just that, secondary.

  You do not have to design specifically for them.

  They will bend and stretch to use the system.

  Even then, however, the goal should be to develop software that bends and stretches to them.

  But you must do it in such a way as to not make the system harder for the primary persona to use.

Primary And Secondary User Personas

Cooper provides a number of additional guidelines for thinking about personas (Cont.):

- There shouldn’t be a large number of personas.

  The goal is to narrow down the people you are designing the system for.

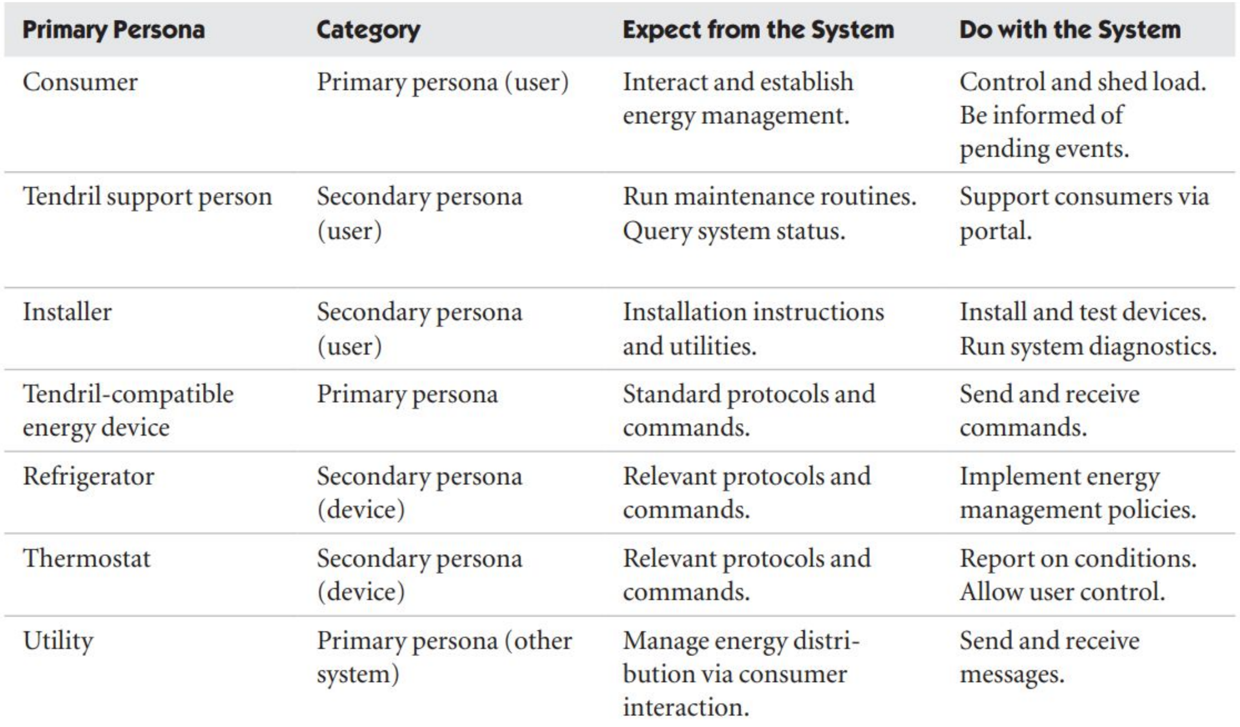
  If you identify more than three primary personas, the scope of your system is likely too large.

  If that is the case, then break the system into subsystems, and identify personas from there.

Primary And Secondary User Personas

- ➔ And finally, after you’ve identified each persona, attempt to first understand both:
  - ◆ What they expect from the system.
  - ◆ What they need to do with the system.
- ➔ With these primary and user personas identified and some expectations and actions identified, the team is ready to proceed with the story writing and any additional requirements discovery work.

Primary And Secondary User Personas



Primary Persona	Category	Expect from the System	Do with the System
Consumer	Primary persona (user)	Interact and establish energy management.	Control and shed load. Be informed of pending events.
Tendril support person	Secondary persona (user)	Run maintenance routines. Query system status.	Support consumers via portal.
Installer	Secondary persona (user)	Installation instructions and utilities.	Install and test devices. Run system diagnostics.
Tendril-compatible energy device	Primary persona	Standard protocols and commands.	Send and receive commands.
Refrigerator	Secondary persona (device)	Relevant protocols and commands.	Implement energy management policies.
Thermostat	Secondary persona (device)	Relevant protocols and commands.	Report on conditions. Allow user control.
Utility	Primary persona (other system)	Manage energy distri-bution via consumer interaction.	Send and receive messages.

Contents

4. Agile And UX Development

Agile And UX Development

- ➔ A common problem in agile is:
  - ◆ How to **incorporate** the **visual design** of the **product into** the **rapid iteration** structure.
- ➔ When teams attempt to resolve complex user interactions while trying to code and test that system at the same time, they can often end up “**churning**” through many iterations.
- ➔ Teams feel that this creates waste—thrashing through many design alternatives but doing so in code.
- ➔ It’s OK to iterate and refactor, but why do it more than might otherwise be necessary?
  - ◆ It’s perfectly fine—and even expected—to improve your design or code over time. But if you keep doing it too much, especially when it could’ve been avoided, it becomes wasteful.

Contents

The UX Problem

- User experience (UX) design is **further complicated when user experience testing** is **required**.
- The **scheduling** and **running** of **multiple experience tests** typically **cannot occur** within an **iteration** that is also attempting to complete the story.
- The teams should assume that all **usability tests** will **have** an **impact** on **design** and **implementation**; otherwise, there's really no reason to do them.
- However, the **result** of this process is **delayed feedback loops** that **complicate productivity** and introduce **delays** in **delivering** the **value**.

The UX Problem

- This is not just a user interface or prototyping problem.
- This is a **meta problem** with the **design** and **architecture** in agile—a result of the elimination of big up-front design (BUFD).
- Fortunately, teams have found a number of practices practical for addressing these design elements.
- The common key is to **leverage** the **iterations** to **drive out uncertainty** and **risk through fast feedback**.

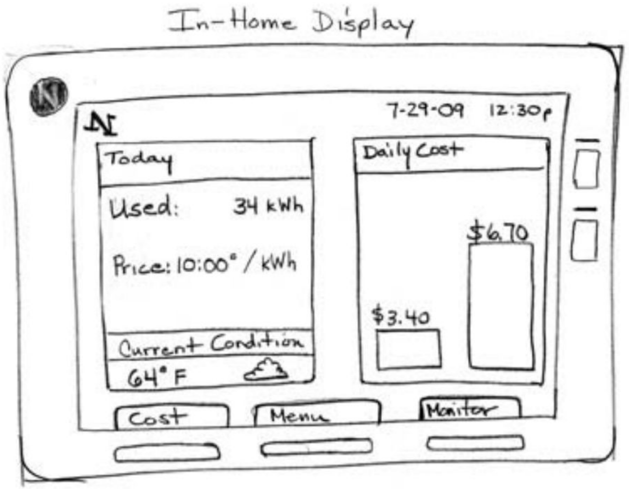
Contents

Low Fidelity Options For UI Development

- Feedback does not have to come only from writing code.
- Teams must also consider simpler alternatives that can generate feedback such as:
  - Low-fidelity paper prototypes
  - Simple HTML
  - PowerPoint prototypes
  - Wireframes, mock objects
  - Coded skeletons
  - Stubs

Low Fidelity Options For UI Development

- ➔ Often the easiest solution is the simplest—sketch ideas on paper, and work with your users and stakeholders to determine how it should look.
- ➔ This can be done by working just-in-time ahead of the iteration and then by using the mock-ups as reference for story acceptance criteria in the implementation iteration.



Contents

4. Agile And UX Development

UX Story Spikes

- ➔ User experience story spikes can be used to **develop alternative user interfaces** and **test** them through **actual** or **representative** users.
- ➔ These stories are put in the backlog, sized, developed, and tested as any other story.
- ➔ The only difference is that **they may not end with working software**.
- ➔ The act of **scheduling them in** the **iteration** has the **effect** of **deferring** the **code development until after** the **tests** are **complete**.
- ➔ This can improve efficiency immensely.

Contents

Centralized UX Development

Fully distributing UX development to the team can actually be quite **problematic**:

- **Velocity** may seem high at first, because the teams are able to move quickly through the initial iterations to value delivery.
- Later, however, problems start to occur as users use different parts of the system for similar purposes but may be presented with different style, presentation, and interaction selection paradigms.
- The net result is a system that has disconcertingly different implementations of like functions.
- User confusion and dissatisfaction are the likely results.

Centralized UX Development

- **Repairing** this work **is problematic too**, because many teams will then have to refactor their code to some new, common standard.
- Adopting common style, presentation, and user action standards can help immensely, but the devil is in the details, and that alone may still not create a comprehensive and holistic solution.
- This is not a very productive process.

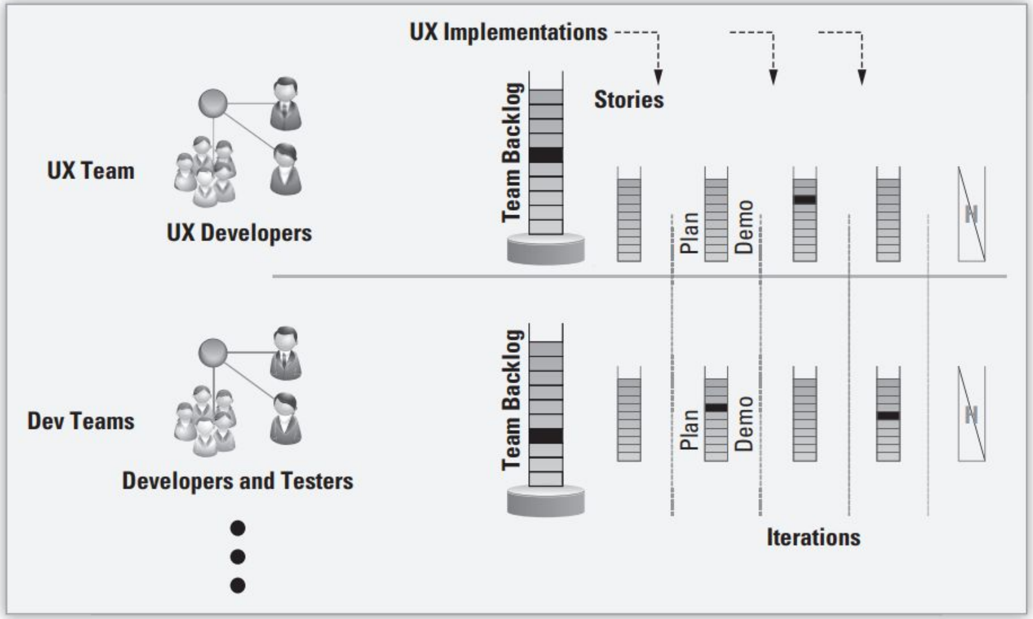
Centralized UX Development

- **To address this**, some organizations create a **central user interface design team** that iterates somewhat independently from the development teams.
- They **run** a common cadence and **iteration** model, but their backlog will contain user experience story spikes, user experience testing, prototyping, and implementation activities that are used to define a common user experience.
- They typically **work one** or **two iterations ahead** to **discover upcoming functionality** and define how it should be implemented.

Centralized UX Development

- The central team implements these designs across many features and system components throughout the course of the release.
- While the central team has interdependencies with the feature and component teams, centralizing the activity has the advantage of increasing core competence in the UX domain and prevents “wagging” the teams as designs inevitably change.

Centralized UX Development

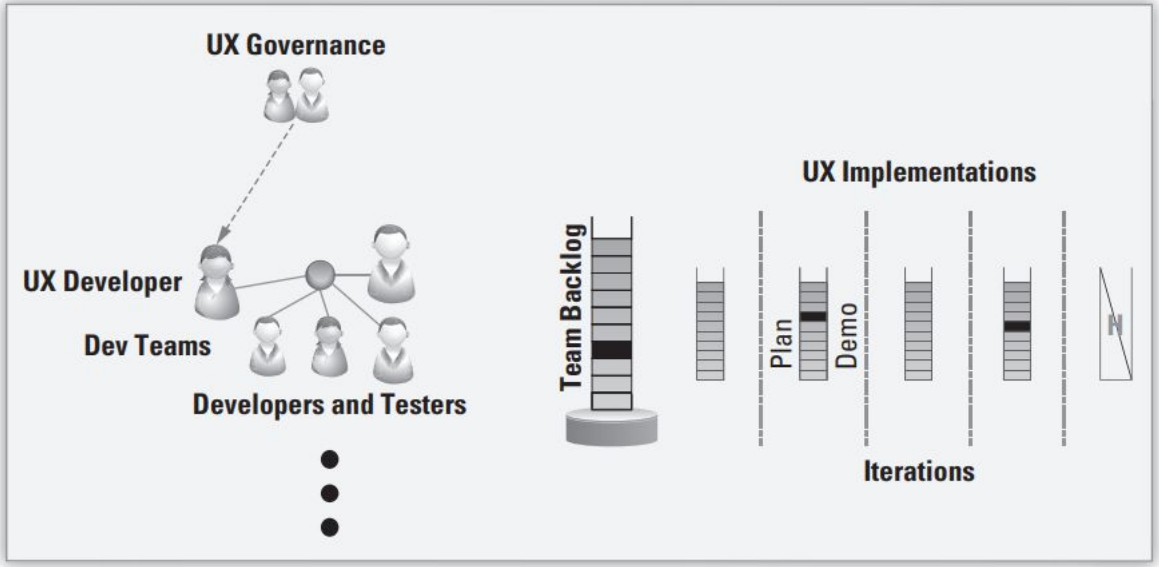


Contents

Distributed, Governed UX Development Model

- ➔ A **hybrid model** effectively applied for **larger systems**.
- ➔ In the “**distributed but governed**” model, there is a small, centralized UX design authority who provides the basic design standards and preliminary mock-ups for each UI, but the teams have team-based UX implementation experts for the implementation.
- ➔ In this case, the UX experts are distributed among the teams, but the centralized authority provides HTML designs, CSS style sheets, brand control, mock-ups, usability guidelines, and other artifacts that provide **conceptual integrity** of the **UX across** the **entire solution**.
- ➔ The **central team** also typically **attends iteration** and **PSI/release demos** to see how the overall system design is progressing.

Distributed, Governed UX Development Model



End of Chapter 7

Contributions

- Author of Reference Book: **Dean Leffingwell**
- Course Instructor: **Mehran Rivadeh**
- Slide Creator: **Mahnaz Rasekhi**
  - These slides are primarily based on Agile Software Requirements by Dean Leffingwell, with occasional adaptations to enhance clarity and engagement.