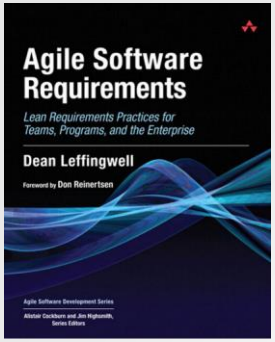


Agile Software Requirements

Software Requirements Engineering – 40688

Computer Engineering department

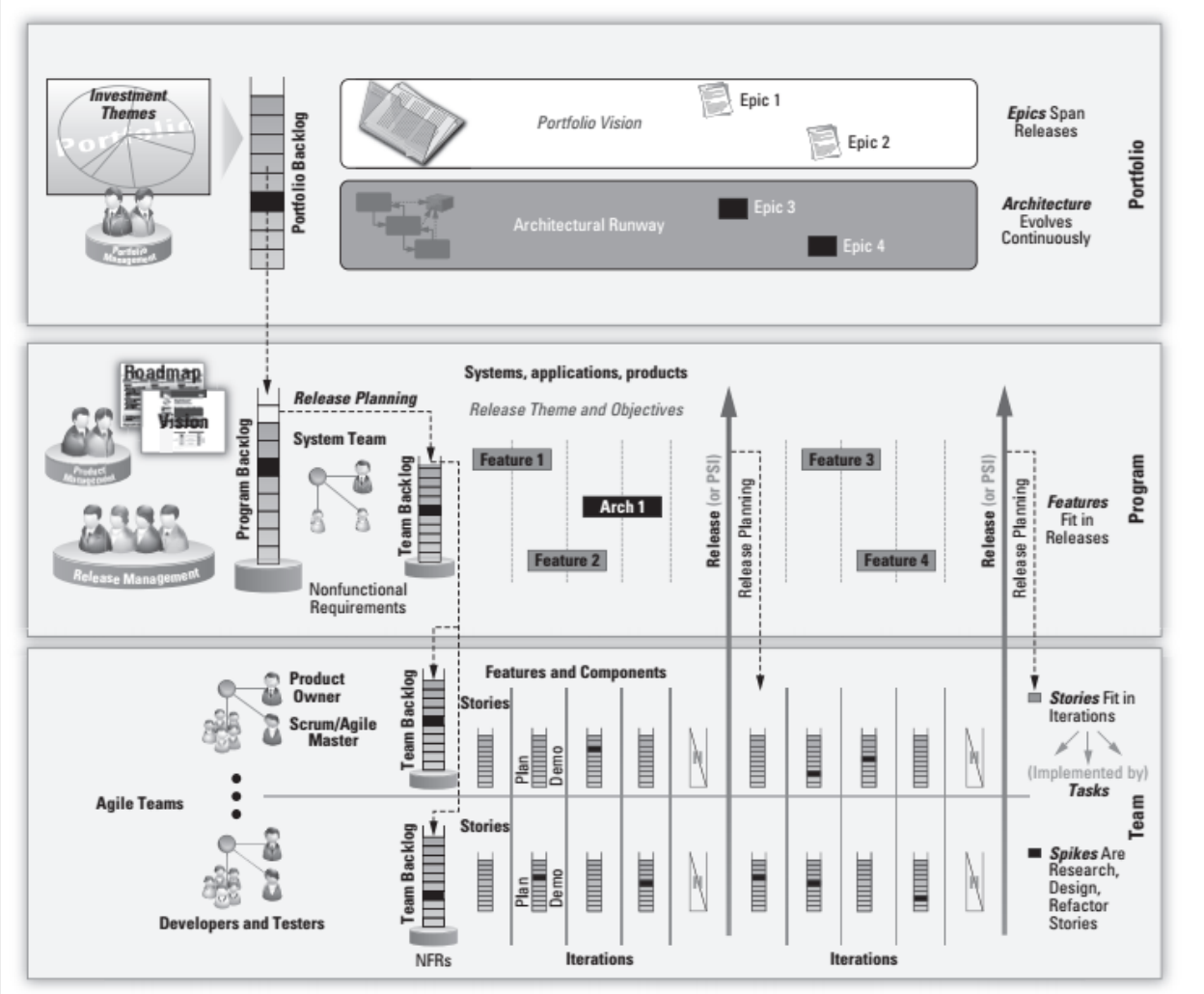
Sharif university of technology

Fall 402

Chapter 2:

The Big Picture of Agile Requirements

The Agile Enterprise Big Picture



Big-Picture Highlights

- **The Team Level**

  At the Team level, agile teams of 7±2 team members define, build, and test user stories in a series of iterations and releases. In the smallest enterprise, there may be only a few such teams. In larger enterprises, groups, or pods, of agile teams work together to support building up larger functionality into complete products, features, architectural components, subsystems, and so on. The responsibility for managing the backlog of user stories and other things the team needs to do belongs to the team's **product owner.**

Big-Picture Highlights

The Team Level

- **7±2** team members define, build, and test **user stories** in a series of **iterations** and **releases**.
- In the **smallest enterprise**, there may be only **a few such teams.**
- In **larger enterprises**, ***groups*, or *pods*, of agile teams** work together to support building up larger functionality into complete products, features, architectural components, subsystems, and so on.
- The **responsibility for managing the *backlog* of user stories** and other things the team needs to do belongs to the team's **product owner.**

Big-Picture Highlights

- **The Program Level**

  At the Program level, the development of larger-scale systems functionality is accomplished via multiple teams in a synchronized **Agile Release Train (ART).** The ART is a standard cadence of timeboxed iterations and milestones that are date- and quality-fixed, but scope is variable (no iron triangle). The ART produces releases or potentially shippable increments (PSIs) at frequent, typically fixed, 60- to 120-day time boundaries. These evaluable increments can be released to the customer, or not, depending on the customer's capacity to absorb new product as well as external events that can drive timing. We'll use the generic **product manager** label as the title for those who are responsible for defining the features of the system at this level, though we'll also see that many other titles can be applied to this role.

Big-Picture Highlights

The Program Level

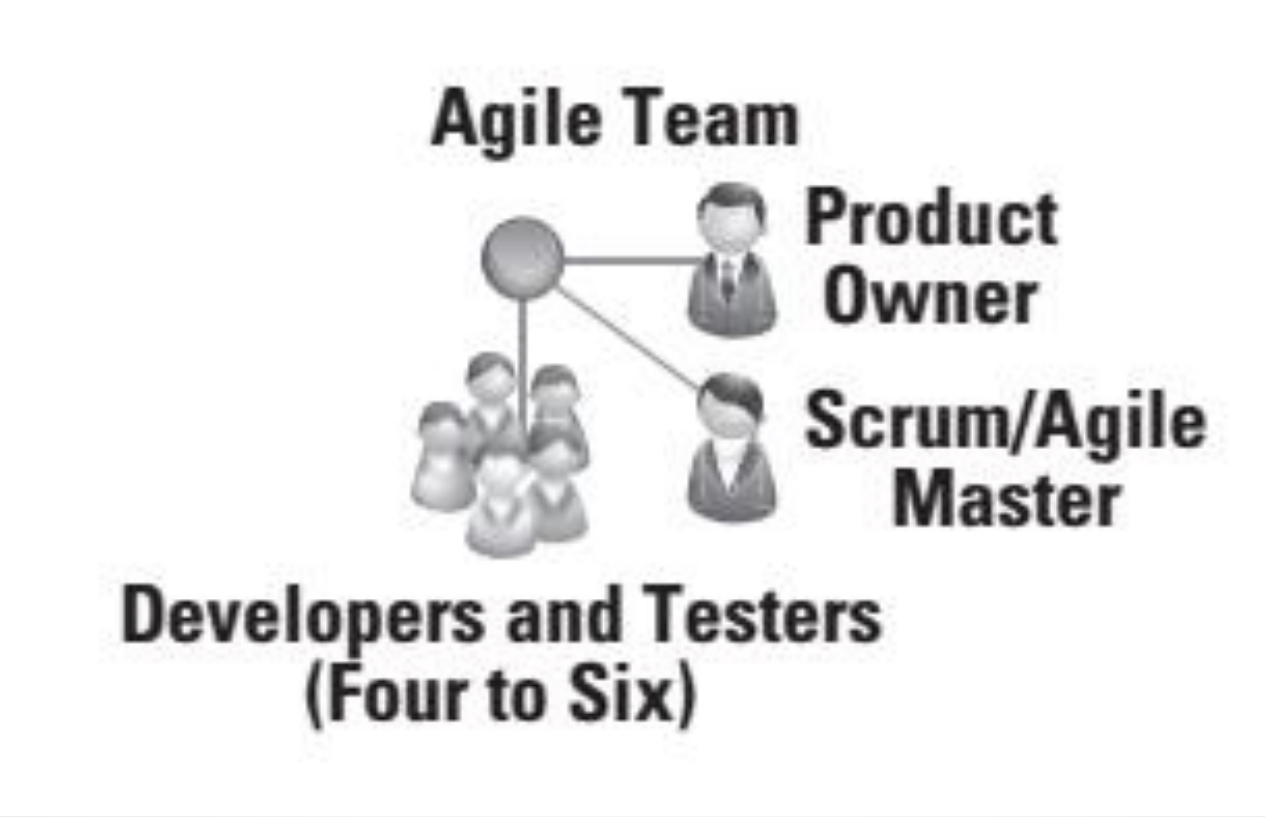
- The development of larger-scale systems functionality is accomplished via multiple teams in a synchronized **Agile Release Train (ART).**
- The **ART** is a standard cadence of timeboxed iterations and milestones that are **date-** and **quality-fixed**, but **scope** is **variable** (no iron triangle).
- The **ART** produces releases or potentially shippable increments (**PSIs**) at frequent, typically fixed**, 60- to 120-day time boundaries**.
- These evaluable increments **can be released to the customer, or not**, depending on the customer's capacity to absorb new product as well as external events that can drive timing.
- **Product manager** label as the title for those who are responsible for **defining** the **features** of the system at this level

Big-Picture Highlights

The Portfolio Level

- At the Portfolio level, we'll talk about a mix of **investment themes** that are used to drive the **investment priorities** for the **enterprise**.
- We'll use that construct to assure that the work being performed is the work necessary for the enterprise to deliver on its chosen business strategy.
- **Investment themes** drive the **portfolio vision**, which will be expressed in as a series of larger, epic-scale initiatives, which will be allocated to various release trains over time.

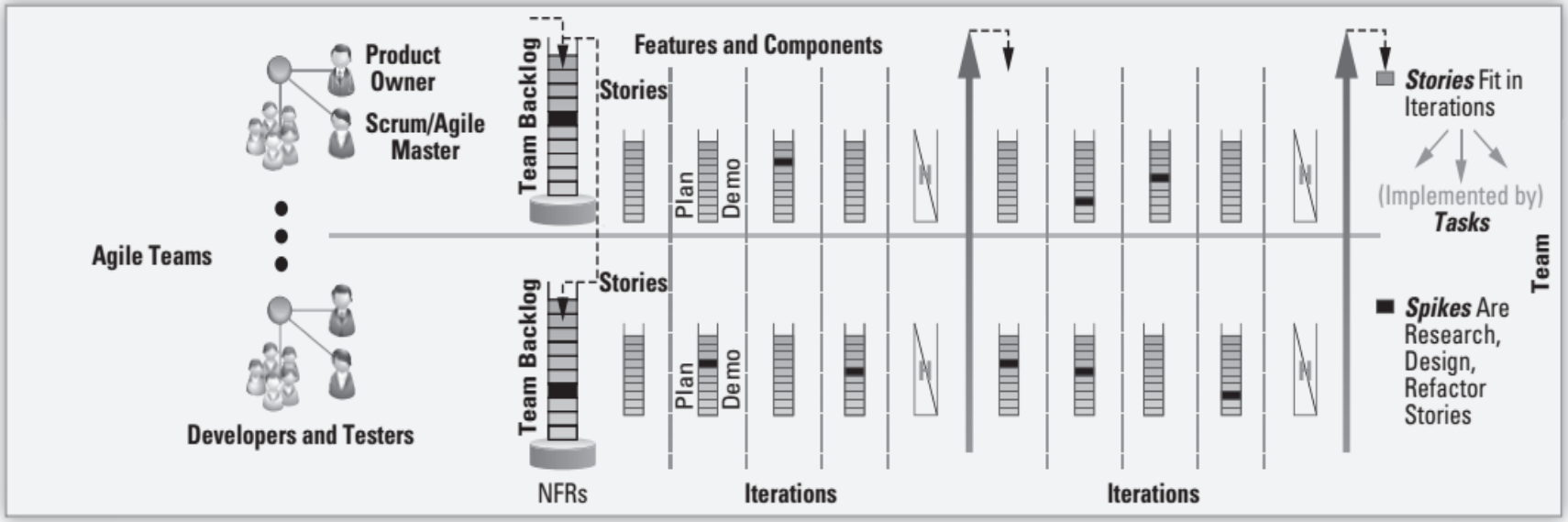
Big Picture: Team level



Big Picture: Team level

- In its daily work, the team is supported by architects, external QA resources, documentation specialists, database specialists, source code management(SCM) / build / infrastructure support personnel, internal IT, and whoever else it takes such that the core team is fully capable of **defining**, **developing**, **testing**, and **delivering working and tested** software into the system baseline.
- Often the testers are logically part of the QA organization but are physically assigned and dedicated to an agile team.
- Pods of Agile Teams

Big Picture: Team level



Iterations

In agile development, new functionality is built in short time boxed events called **iterations** (**sprints** in Scrum).

Number of Iterations per "Release":

- A series of iterations is used to aggregate larger, system wide, functionality for release (or potential release) to the external users.
- In the Big Picture, we've illustrated four **development** iterations (indicated by a full iteration backlog) followed by one **hardening** (or stabilization) iteration (indicated by an empty backlog) prior to each release increment.
- This pattern is arbitrary, and there is no fixed rule for how many times a team iterates prior to a **potentially shippable increment** (PSI).

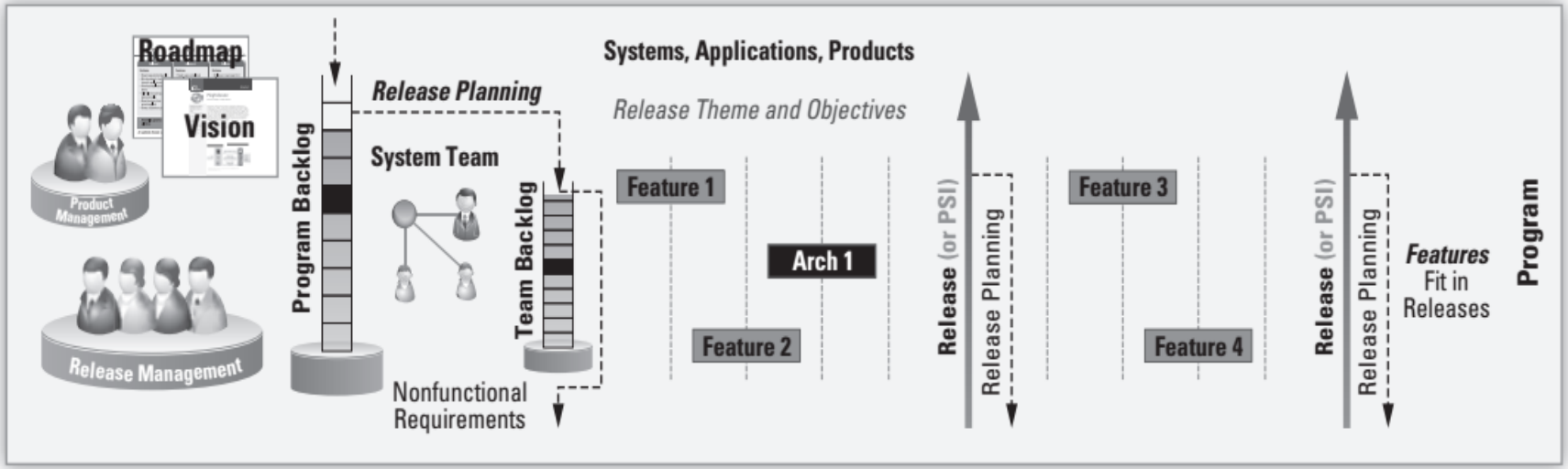
User Stories and the Team Backlog (1)

- **User stories** (stories for short) are the general-purpose agile substitute for what traditionally has been referred to as **software requirements**.
- Originally developed within the constructs of XP, user stories are now endemic to agile development in general and are typically taught in Scrum, XP, and most other agile implementations.
- **As a <user role>, I can <activity> so that <business value>**

User Stories and the Team Backlog (2)

- The team's backlog (typically called a *project* or *product* backlog) consists of all the user stories the team has identified for implementation.
- For more detailed tracking of the activities involved in delivering stories, teams typically decompose stories into *tasks* that must be accomplished by individual team members in order to complete the story.
- However, the iteration tracking focus should be at the story level, because this keeps the team focused on business value, rather than individual tasks.

Big Picture: Program level



Big Picture: Program level

- Moreover, there are legitimate business reasons why not every increment should be shipped to the customer.
- Potential interference with a customer's licensing and service agreements.
- Potential for customer overhead and business disruption for installation, user training, and so on.
- Potential for disrupting customer's existing operations with minor regressions or defect.

Vision, Features, and the Program Backlog

- Within the enterprise, the product management (or possibly program management or business analyst) function is primarily responsible for maintaining the Vision of the products, systems, or application in their domain of influence.
- The Vision answers the big questions for the system, application, or product, including the following.
- What problem does this particular solution solve?
- What features and benefits does it provide?
- For whom does it provide it?
- What performance, reliability, and so on, does it deliver?
- What platforms, standards, applications, and so on, will it support?

Content of the Vision Is a Set of Features

- A Vision may be maintained in a document,
- in a backlog repository,
- or even in a simple briefing or presentation form.
- But no matter the form, the prime content of the Vision document is a prioritized set of *features* intended to deliver *benefit's* to the users.
- The Vision must also contain the various nonfunctional requirements,
- such as reliability, accuracy, performance, quality, compatibility standards, and so
- That are necessary for the system to meet its objectives.

Undelivered Features Fill the Program Backlog

- In a manner similar to the team's backlog, which contains primarily *stories*, the program (or *release*) backlog contains the set of desired and prioritized *features* that have not yet been implemented.
- Any estimates at this scale are coarse-grained and imprecise, which prevents any temptation to over invest in inventory of too early feature elaboration and estimation.

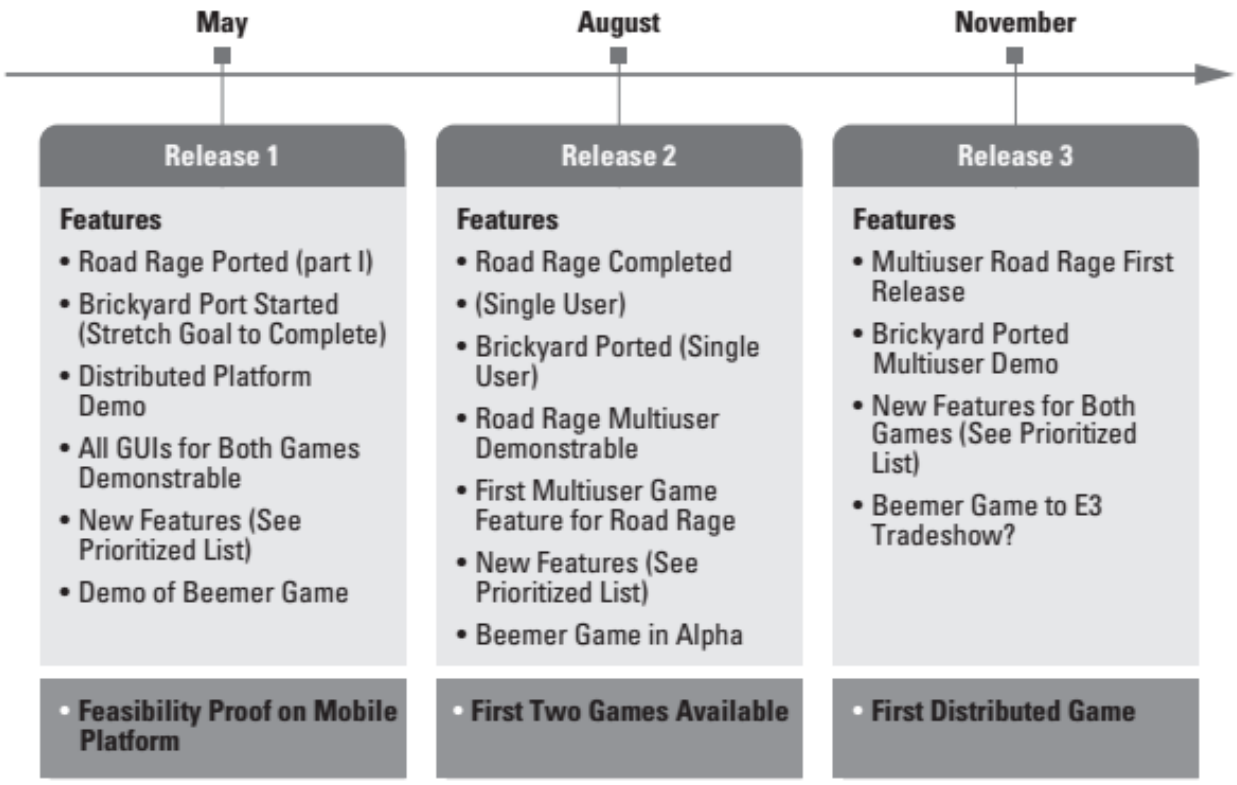
Release Planning

- In accordance with emerging agile enterprise practices, each release increment timebox has a kickoff release planning session that the enterprise uses to set the company context and to align the teams to common business objectives for the release.
- The input to the release planning session is the current Vision, along with a set of objectives and a desired, prioritized feature set for the upcoming release.
- By breaking the features into stories and applying the agreed-to iteration cadence and knowledge of their velocity, the teams plan the release, typically in a group setting.
- During this process, the teams work out their interdependencies and design the release by laying stories into the iterations available within the PSI time box.
- Thereafter, the teams endeavor to meet their commitment by satisfying the primary objectives of the release, even if it turns out that not every feature makes the deadline.

The Roadmap

- **The results of release planning** are used **to update** the (product or solution) Roadmap, which provides a sense of how the enterprise hopes to deliver increasing value over time.
- The **Roadmap** consists of a **series of planned release dates**, each of which has a theme, a set of objectives, and a prioritized feature set.
- The "next" release on the Roadmap is **committed to the enterprise**, based on the work done in the most recent release planning session.
- Releases beyond the next one are not committed, and their scope is fuzzy at best.
- The Roadmap, then, represents the enterprise's current "plan of intent" for the next and future releases.
- However, it is subject to change—as development facts, business priorities, and customers need change.

The Roadmap



An Updated, Themed, and Prioritized "Plan of Intent"

Product owner vs Product Manager

- In agile, there can be a **challenge** with the apparently overlapping responsibilities of the **product manager** and the **product owner**.
- For example, in **Scrum**, the **product owner** is **responsible** for the following:
  - Representing the interests of everyone with a stake in the resulting project . . . achieves initial and ongoing funding by creating the initial requirements, return on investment objectives, and release plans.

Responsibilities of the Agile Product Manager in the Enterprise

- Own the Vision and program (release) backlog
- Manage release content
- Maintain the product Roadmap
- Build an effective product manager/product owner team

Big-Picture elements: Portfolio level

- At the top of the Big Picture, we find the **portfolio management function**, Which includes those individuals, teams, and organizations dedicated to managing the investments of the enterprise in accordance with the enterprise business strategy.
- We also find two new artifact types, **Investment themes** and **epics**, which together create the portfolio vision.

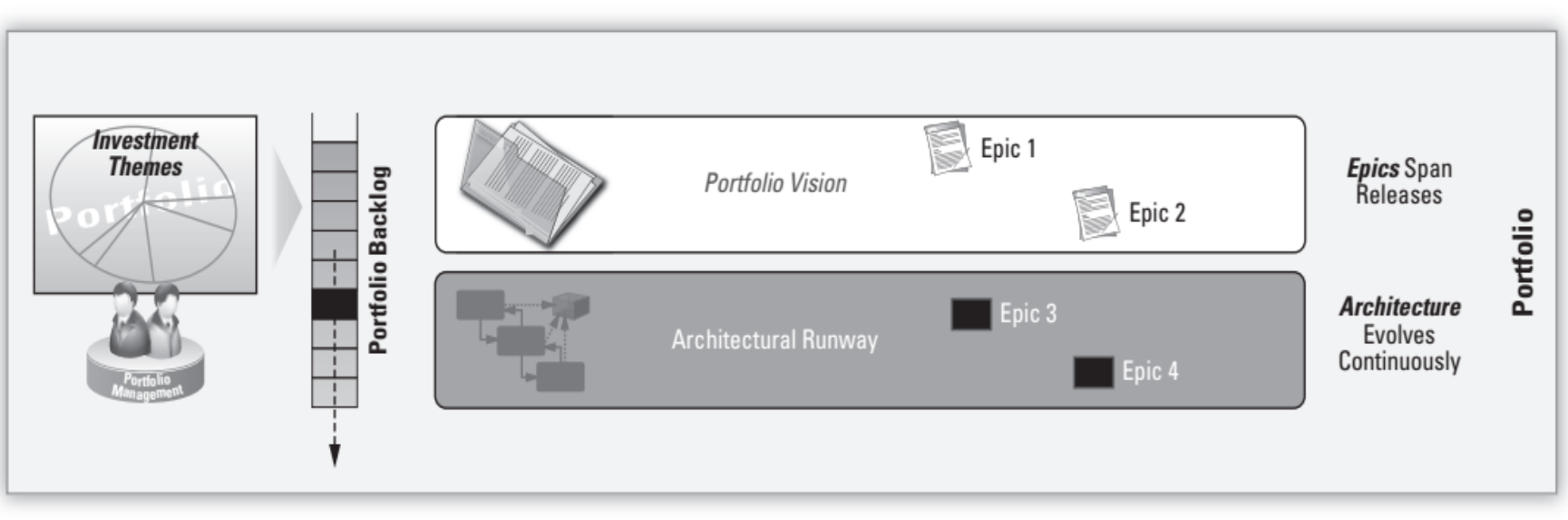
Investment Themes

- A set of **investment themes** establishes the relative investment objectives for the enterprise or business unit.
- These **themes** drive the vision for all programs, and new epics are derived from these themes.
- **The derivation of these decisions** is the responsibility of the portfolio managers, either line-of-business owners, product councils, or others who have fiduciary responsibilities to their stakeholders.
- **The result of the decision process** is a set of themes—key product value propositions that provide marketplace differentiation and competitive advantage. Themes have a much longer life span than epics, and a set of themes may be largely unchanged for up to a year or more.

Epics and the Portfolio Backlog

- **Epics** represent the highest-level expression of a customer need.
- **Epics** are development initiatives that are intended to deliver the value of an investment theme and are identified, prioritized, estimated, and maintained in the **portfolio backlog**.
- Prior to release planning, **epics** are decomposed into specific features, which in turn are converted into more detailed stories for implementation.
- **Epics** may be expressed in bullet form, in user-voice story form, as a sentence or two, in video, in a prototype, or indeed in ***any form*** **of expression suitable** to express the intent of the product initiative.
- With **epics**, clearly, the objective is strategic intent, not specificity. In other words, the epic need only be described in detail sufficient to *initiate a further discussion* about what types of features an epic implies.

Big-Picture elements: Portfolio level



Architectural Runway

- Design(architecture) and requirements are simply two sides of the same coin.
- However, even though course focuses on requirements, we can't ignore architecture, because experience tells us that teams that build some amount of **architectural runway**, which is the ability to implement new features without excessive refactoring, will eventually emerge as the winners in the marketplace.
- Therefore, system architecture is a first-class citizen of the Big Picture and is a routine portfolio investment consideration for the agile enterprise.