SAFe Architecture Approach

Kim Horn

Version 0.5 **1 July 2016**

Agenda

1. SAFe Architecture Material*
   a) SAFe Architecture Principles;
   b) Three kinds of Architects:
      - Enterprise;
      - Solution;
      - System.
   c) Content Providers and Architects;
   d) Knowledge Items & Terminology.
   e) Agile Architecture
2. Need for Matrix Communities - SAFe & Spotify
3. Appendix:
   - SAFe Requirements Overview
   - Krushen: Framework
   - Essence: Agile @ Scale.

* Majority of the text is taken directly from SAFe.

Principles

SAFe Principles of Agile Architecture

1. Design emerges. Architecture is a collaboration.
2. The bigger the system, the longer the runway.
3. Build the simplest architecture that can possibly work.
4. When in doubt, code or model it out.
5. They build it, they test it.
6. There is no monopoly on innovation.
7. Implement architectural flow.

These are discussed in the following slides.

Kinds of Architects

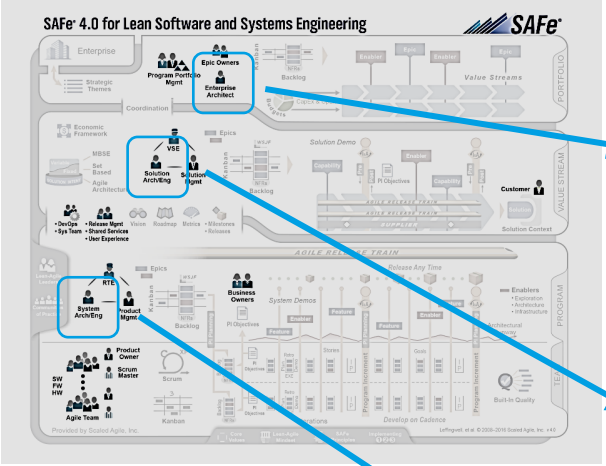
Role of the Architect and Engineer

The role of an ***Architect*** is common in software development and has been included in SAFe as one of the critical roles at the program and portfolio levels*. Also, the role of an Architect often extends beyond just the software domain to include responsibilities that enable value delivery in a technologically diverse and heterogeneous, multi-domain solution environment, so SAFe takes a fairly expansive view of that role.

Architect/Engineering align the Value Stream and Agile Release Trains to a common technological and architectural vision of the Solution under development. They participate in defining the system and subsystems, validate technology assumptions, and evaluate alternatives. They support solution development through providing, communicating, and evolving the larger technological and architectural view of the solution.

* Also at Value Stream level, which is optional

3 Kinds of Architects in SAFe



Enterprise Architect

Solution Architect

System Architect

1. Enterprise Architect at the Portfolio level
2. Solution Architect at the Value Stream level (when req'd)
3. System Architects at the Program/ Agile Release Train level

- This level is optional
- Required when multiple ARTs need to come together (or where e2e value chain is not fully described, help define and elaborate)

- Dedicated to one and only one ART

Enterprise Architecture Responsibilities

- Maintain a high-level, holistic vision of Enterprise Solutions and development initiatives;
- Help define key technical initiatives that support Budgets via Enabler Epics;
- Participate in the strategy for building and maintaining the **enterprise Architectural Runway***;
- Understand and communicate strategic themes and other key business drivers for architecture to System Architects and nontechnical stakeholders;
- Drive architectural initiatives in the Portfolio Kanban system and participate in Epic analysis where applicable;
- Influence common modeling, design, and coding practices;
- Collect, generate, and analyze innovative ideas and technologies that are applicable across the enterprise;
- Facilitate the reuse of ideas, components, and proven patterns;
- Synchronize the following across solutions whenever applicable:
  - System and data security and quality;
  - Production infrastructure;
  - Solution User Experience governance;
  - Scalability, performance, and other NFRs.

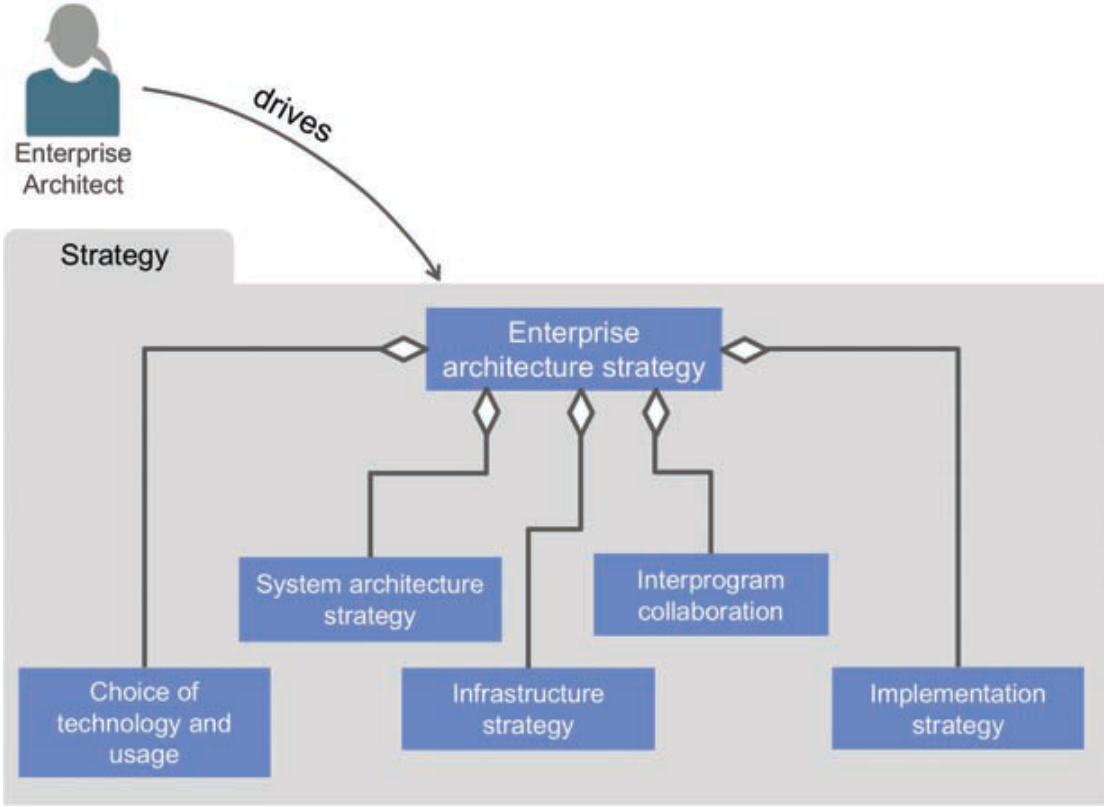
* this runway is not specifically described as distinct to the program level runway.

Principal #7 – Implement Architectural Flow

Enterprise-scale architectural initiatives require coordination across Agile Release Trains and Value Streams. Effective flow of these architectural initiatives is made visible via the Portfolio Kanban. There, program and value stream enabler features and capabilities follow a common work flow pattern of exploration, refinement, analysis, prioritization, and implementation. In addition, the “pull” nature of the Kanban system allows programs to establish capacity management based on WIP limits. This helps avoid overloading the system.

Together with the portfolio Kanban, the Program and Value Stream Kanban systems provide a SAFe enterprise content governance model. This instantiates portions of the Economic Framework and helps Decentralize Decision-Making, both of which are vital to a fast, sustainable flow of value.

Enterprise Architecture helps drive Strategy



Elements of Enterprise Architecture

1. **Choice of Technology –** to best support current Budgets.
   - Supporting activities include research and prototyping, understanding applicability and scope, and assessing maturity of innovative new technologies.
2. **System/Solution Architecture Strategy –** The EAs works closely with Solution and System Architects to ensure that individual ***program and product strategies*** align with enterprise objectives.
   - Emergent solutions to local problems should be consistent with the strategy. When that is not the case, the decision should be made explicitly, and the contraindicator may well influence enterprise strategy.
3. **Development and Deployment Infrastructure Strategy –** This infrastructure is unnoticed when it fulfills its function properly. However, the strategy for building and maintaining this infrastructure is a key challenge, one that overlaps with System Architect responsibilities, that include: the reuse of configuration patterns, common physical infrastructure, knowledge sharing between value streams, ARTs and System Teams.
   - Provide direction to manage overlap between this infrastructure and that of internal IT systems.
4. **Inter Program Collaboration –** Various and differing aspects of architecture work happen in different teams and programs. Thus it is helpful to ensure that common technology, common design practices, and common infrastructure are used where applicable.
5. **Implementation Strategy –** Building the technical foundation for Business Epics into the architectural runway must be an incremental process based on continuous technical learning and fast feedback, such that *architecture and business functionality* grow synchronously over time.

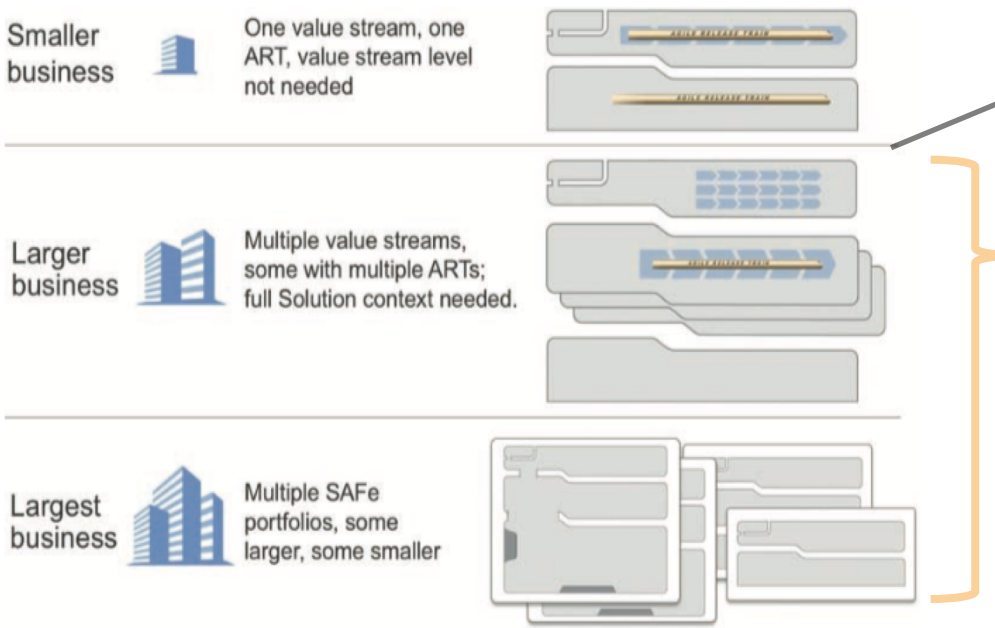
Solution/Systems Architecture Responsibilities

- Participate in planning, definition, and high-level design of the **solution** and explore solution alternatives
- Define subsystems and their interfaces; allocate responsibilities to **subsystems**; understand solution deployment, and communicate requirements for interactions with solution context;
- Work with customers, stakeholders, and suppliers to establish **high-level Solution Intent**; help establish the solution intent information models and documentation requirements;
- Establish critical **Nonfunctional Requirements** at the solution level; participate in the definition of others;
- Operate within the Economic Framework to validate the economic impact of design decisions;
- Work with portfolio stakeholders, particularly the Enterprise Architect, to develop, analyze, split, and realize the implementation of **Enabler Epics**;
- Participate in PI Planning and Pre- and Post-PI Planning, System and Solution Demos, and Inspect and Adapt events;
- Define, explore, and support the implementation of **value stream and program enablers** to evolve solution intent; work directly with Agile Teams to implement, explore, orsupport them;
- Plan and develop the **Architectural Runway** in support of upcoming business Features/Capabilities;
- Work with Product and Solution Management to determine capacity allocation for enablement work;
- Support technology/engineering aspects of Program and Value Stream Kanbans;
- Supervise and foster Built-in Quality;

Architects: System and Solution

- **Arch/Eng align the Value Stream and Agile Release Trains** to a common technological and architectural vision of the Solution under development. They participate in defining the system and subsystems, validate technology assumptions, and evaluate alternatives. They support solution development through providing, communicating, and evolving the larger technological and architectural view of the solution.
  - **System Arch/Eng** operates mainly in the context of the Agile Release Train, where they work with Agile Teams and provide technical enablement with respect to subsystems and capability areas under the purview of the ART.
  - **Solution Arch/Eng** teams provide technical leadership for evolving architectural capabilities of the entire solution.
  - **Both** involve tight collaboration with business stakeholders, teams, Customers, Suppliers, and third-party stakeholders in defining the technology infrastructure, decomposition into components and subsystems, and the definition of interfaces between subsystems and between the solution and Solution Context.

Solution Architect Required with Multiple ARTS



Also one ART, multiple value Streams

Signals the use of the 4 layer SAFe model. Difficulty now of multiple ARTS that synchronously build solutions incrementally.

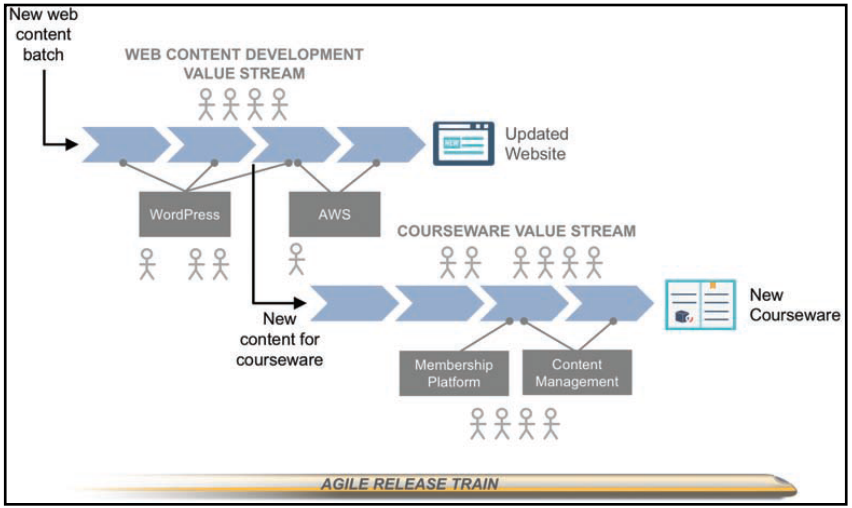
However, ARTS are meant, ideally, to be self managing and self organising.

Complex cross cutting dependencies:

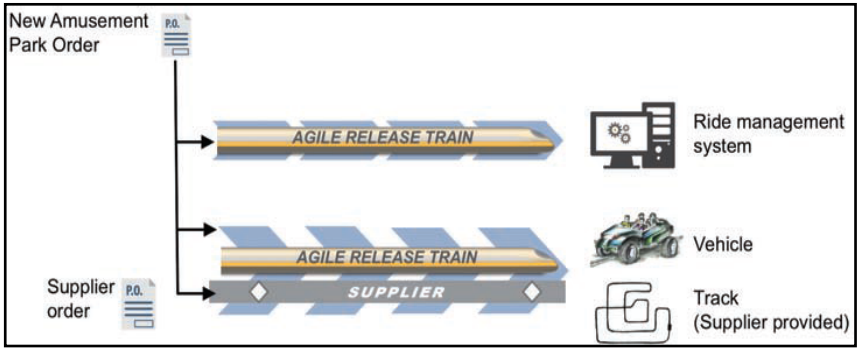
- Requires the Value Stream Backlog to organise.
- Need Solution Architects
- DevOps/Systems teams are now part of the trains, as well as part of the solution team at the Value Stream Level.
- DevOps, systems, infrastructure are part of the Solution Context

Examples

One ART – Multiple Value Streams



2 ARTs – One Value Streams + Supplier

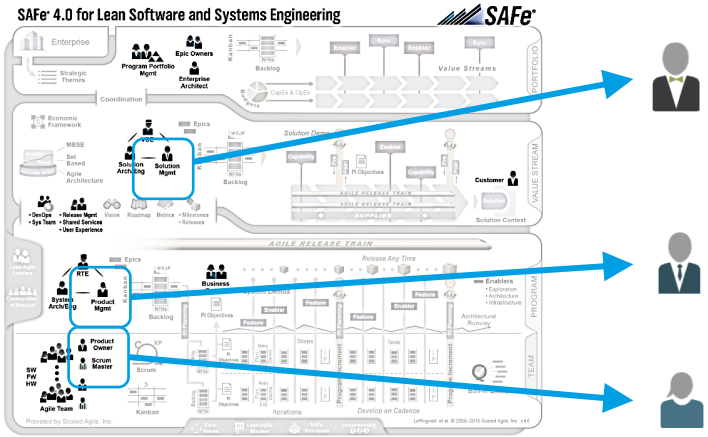


Content Providers

3 Level Content Chain

Top Down Alignment by clear line of content authority.

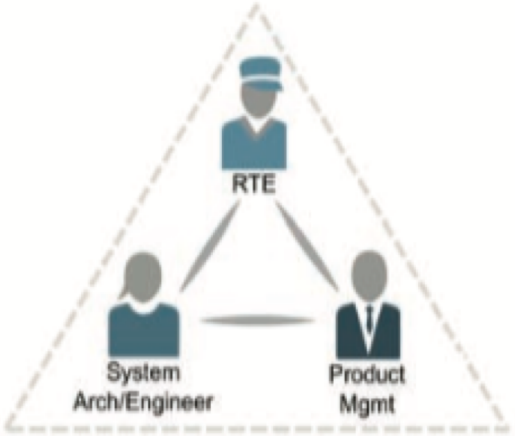
- Solution Management and Product Management are the main content authorities in SAFe, guiding the Value Stream and Program Levels respectively.



- *Solution Management* is responsible for guiding the solutions that constitute the Value Stream. Represents Customers' needs as well as the strategic themes of the portfolio Vision.
- *Product Management* is responsible for Program Vision and Backlog. They are the internal VOC.
- *Product Owners* make fast, local content decisions on behalf of the Agile Team.

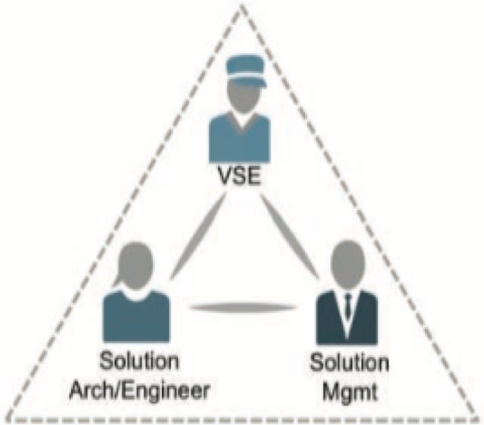
Architects work closely with these Content Authorities.

The Architects and the Three Troicas



Program Level

These three primary functions help ensure successful execution of the vision and roadmap initiatives at the program level



Value Stream Level (Optional)

Much like the “troika” of the program level, the value stream level has its own troika of three critical roles necessary to coordinate and advance the value stream.

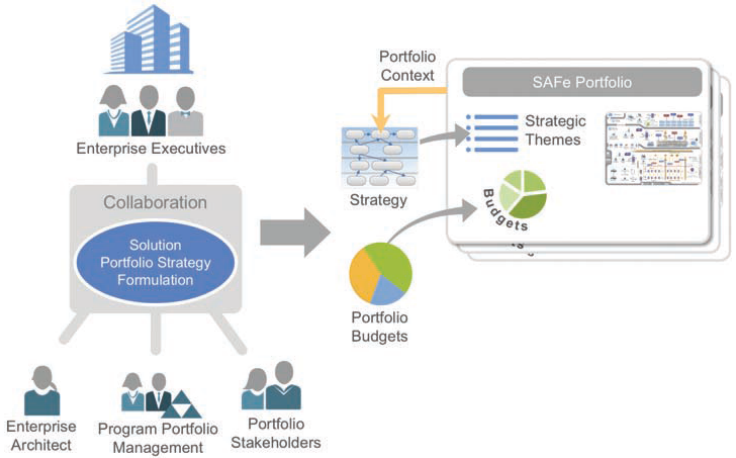
Portfolio Level

This “troika” provides three primary functions necessary to help ensure successful execution of portfolio initiatives.

Portfolio Level

EAs have significant portfolio duties and have the requisite knowledge to work across value streams and programs and help provide strategic technical direction that can optimize portfolio outcomes.

- Can include recommendations for epic enablers that harmonize development and delivery technology stacks, interoperability of solutions, APIs, and hosting strategies.
- Enterprise Architect may act as an **Epic Owner for Enabler Epics.**
- Participate in the strategy for building and maintaining the **enterprise Architectural Runway**;
- ***Strategic Themes*** are business objectives that provide strategic differentiators that highlight changes to the Enterprise strategy that affect a particular Solution portfolio. They connect the Portfolio to the Enterprise Business Strategy.



Value Streams Level and ARTS

Value Streams are represented at the Portfolio Level:

- **the most fundamental construct in SAFe**;
  - a long-lived series of system definition, development, and deployment steps to deliver value;
- enable fiscal governance; are funded (they get a budget) rather than projects;
- organized to be largely independent;
- contain the people who do the work, the systems they develop or operate, and the flow of information and materials;
- may move across multiple departments and organizations and across many deployed systems. In such cases, *finding the value stream* is an important analytical and business context-based activity.

Each SAFe portfolio contains a set of Value Streams and the additional constructs necessary to provide funding and governance for the products, services, and Solutions that the *Enterprise* needs to fulfill some element of the business strategy.

Two Types of Value Streams:

- **Operational** – show the steps for providing goods or services to customer;
- **Developmental** - show the steps used to develop new products, systems, or services capabilities.

Program Level

Product Management continuously develops and communicates the vision to the development teams and defines the features of the system.

- **In collaboration with System and Solution Architect/ Engineering**, they also define and maintain the Nonfunctional Requirements (NFRs), to help ensure that the solution meets relevant standards and other system quality requirements.
- **Work with System Architect/Engineering to understand Enabler work** – While Product Management is not expected to drive technological decisions, they are expected to understand the scope of the upcoming enabler work and to work with System and Solution Architect/Engineering to assist with decision-making and sequencing of the key technological infrastructures that will host the new business functionality. is can o en best be accomplished by establishing a capacity allocation, as described in the “Program Backlog” article.

Knowledge

Specific Knowledge Items & Terminology

- Initiatives.
- Systems.
- Capabilities.
- Enablers
- Solution:
  - Solution Context
  - Solution Intent
  - Solution Vision
- Architectural Runway
- Architectural Initiatives
- ART/Team organisation

Initiatives

The responsibility of the portfolio level is the discovery, definition, and administration of major **initiatives** that are required by the business but are crosscutting in nature.

- **Epics** are containers for significant initiatives that are large and typically crosscutting over multiple Value Streams and ARTs. They are investment intensive and far ranging in impact.
  - **Business Epics** capture and reflect the new business capabilities that can only be provided by cooperation among value streams.
  - **Epic Enablers** reflect the **architectural and other technical initiatives** that are necessary to enable the new Capabilities.

In larger Enterprises, there will be multiple SAFe portfolios, each helping to manage **a set of initiatives**, typically at the business unit or departmental level.

- **Portfolio Kanban:**
  - Make the strategic business initiative backlog fully visible
  - Bring structure and visibility to the analysis and decision-making that moves these initiatives into implementation.
- **Program and Value Stream Kanbans:**
  - Manage Large initiatives, such as program and **Value Stream Epics** (called Solution Epics – these are broken down into capabilities)
  - Result in features and capabilities intended to advance the Solution.

Systems

SAFe synchronizes alignment, collaboration, and delivery for large numbers of Agile Teams. It supports both **software solutions** and complex **cyber-physical systems** that require thousands of people to create and maintain.

- **Lean-Agile Principle** #5 – Base milestones on objective evaluation of working systems..
- **System Demo:** deliver a working system increment every 2 weeks;
- Each ART delivers system level solutions.

Enterprises that build **systems** that are largely independent, may not need the 'value stream level of SAFe.

- ***The value stream level helps enterprises that face the largest systems challenges: those building large scale, multidisciplinary software and cyber-physical systems.***

A ***feature*** **is a service** provided by the system that addresses one or more user needs.

Each feature reflects a **service** provided by the system that fulfills some stakeholder need.

System Integration – Systems collaborate

Different components and subsystems—software, firmware, hardware, and everything else—must collaborate to provide effective solution-level behaviors. Practices that support solution-level quality include:

- Frequent system and solution-level integration;
- Solution-level testing of functional and Nonfunctional Requirements;
- System and Solution Demos.

Solution

The one purpose of SAFe is to deliver *Solutions* that provide value to the Customer.

- A solution is either a final product delivered to the ultimate economic buyer or, alternately, **a set of systems** that enables an operational value stream within the organization.

- The primary role of a **SAFe Portfolio** is to finance and nurture a set of Solutions development activities (a *development value stream*) that either deliver end user value directly or support other business (operational) value streams.

Solution

Portfolio level:

- Manages multiple solutions.
- Portfolios contains multiple value streams. Some may have a number of cross cutting solution concerns and dependencies, that need to be managed.

Program Level

- The solution appears at the Program Level in 3-level SAFe.
- Solution behavior and decisions are managed in Solution Intent.
- Every Solution is delivered by a value stream.

Value Stream Level (optional)

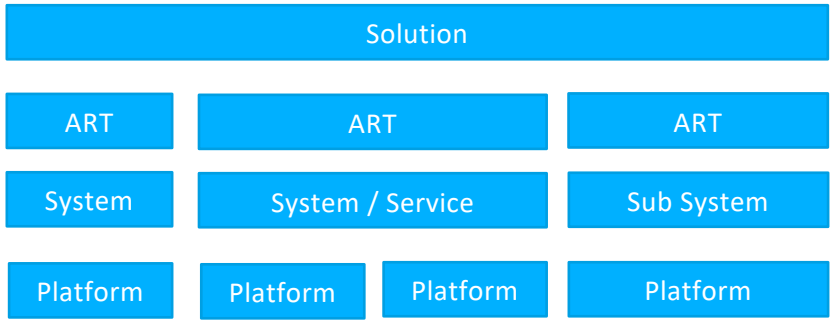
- The main construct of the Value Stream Level is the **solution**, this level adds additional practices and details to the level 3 model. These may be used in 3 level model too.
- Solution development is the entire subject of the Value Stream Level in SAFe.
- The Solution Architect/Engineering is a team in charge of defining the overarching architecture that connects the solution across ARTs. They also work with the System Architect/ Engineering team to help guide the architecture developed by the trains.

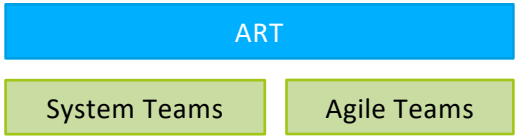
Platforms, System and Solution

"An architectural runway exists when the **Enterprise's platforms** have sufficient existing technological infrastructure to support the implementation of the highest-priority Features in a near term Program Increment without excessive, delay-inducing redesign."

- By definition an ART focuses on a system, subsystem, or service.
- A solution may involve multiple ARTs.
- An ART can work across Platforms.
- A new system, service or feature may require work across Platforms.
- May allocate teams in the ART to each Platform.

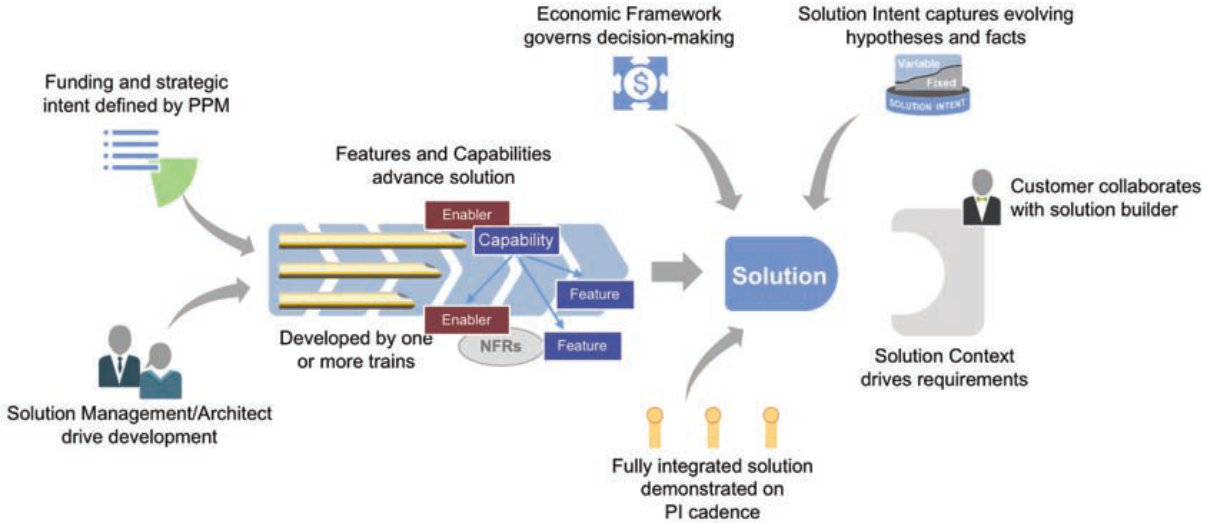
Concept of ***Application*** is not discussed in SAFe





Where: **Platform** = underlying software/hardware infrastructure Implementing the system, not business or strategic meaning

Solution Development



Principle #2 – *Apply systems thinking* guides the organization to adopt a systems view and to apply scalable and emergent practices around value definition, architecture, development practices, and process improvement.

Solution Capabilities, Enablers, and Context

Capabilities are the end-to-end solution services that support the achievement of user goals. Capabilities are implemented via vertical, end-to-end slices of value, which enable incremental solution development.

- Alternate: “A capability is a high-level solution behavior that typically spans multiple ARTs. They are sized and split as necessary within a single PI.”

Enablers provide for exploration of new capabilities, contribute to solution infrastructure and architecture, and enhance NFRs. They arise from the backlog and occur at all levels of the framework where **they are described as enabler portfolio epics, enabler capabilities, enabler features or enabler stories.**

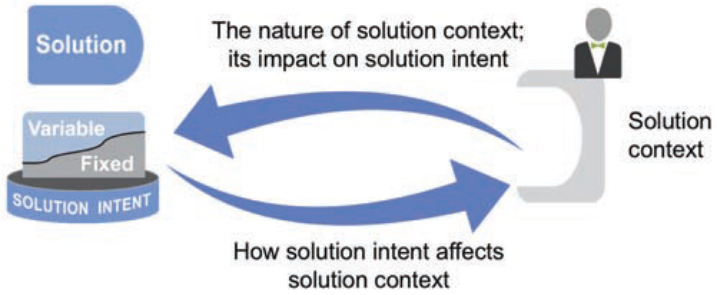
The Solution Context identifies critical aspects of the target Solution environment and its impact on usage, installation, operating, support, and even marketing, packaging, and selling. Context could be a system of systems, product suite, or IT deployment environments. Contexts could be Fixed and/or Evolving.

- Taking a systems view helps ensure that the solution builder understands the solution context, the broader ecosystem in which the solution operates. Solution context provides the additional pieces that determine operational requirements and constraints.
- Customers are part of the value stream. They participate in defining the **solution intent** and **solution context**, and they help validate assumptions and fitness for use.
- The **Solution** and the **Context** are coupled, and the degree represents an architectural and business challenge.

Solution Intent*

- Is a critical repository to store, manage, and communicate knowledge of the current and intended solution; *what* **systems** builders are building and *how* they are going to build it.
- It serves many purposes:
- Provides a single source of truth as to the intended and actual behavior of the solution
- Records and communicates requirements, design, and system architecture decisions
- Facilitates further exploration and analysis activities
- Aligns the Customer, the systems builders, and Suppliers to a common purpose
- Supports compliance and contractual obligations.

The **Solution Context** drives the **Solution Intent**

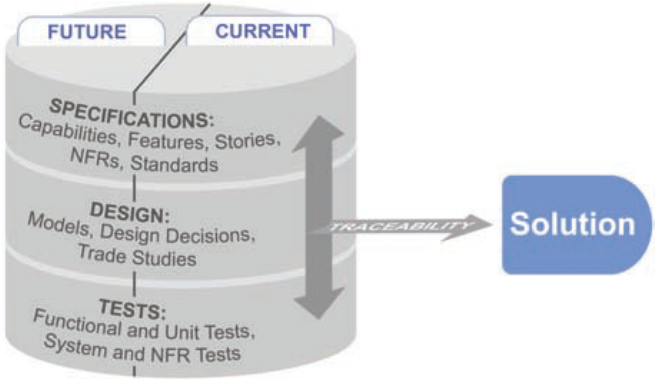


*See Appendix: Krushen for background on Intent

Solution Intent

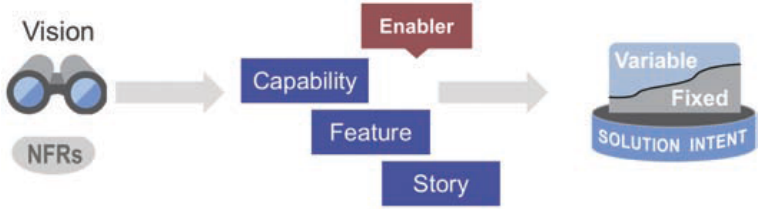
The single source of truth about the solution, includes:

- NFRs
- Functional requirements
- Traceability links between:
  - NFRs, requirements they impact,
  - and tests used to verify them.



- Some of the requirements of the solution are likely to be well understood and **fixed** from the beginning, while others are **variable** and can only be understood during the development process.
- Models (domain, use case), technical knowledge, and even various design decisions are recorded to provide a central point of communication.
- Also used to record innovative ideas.
- **A system's solution intent** does not necessarily stand alone. Many solutions are *systems* that participate in a higher-level system of systems.

Developing Solution Intent



- Solution intent begins with a Vision that describes, at a high level, the purpose and key capabilities of the intended solution, along with the critical nonfunctional requirements.
  This knowledge, along with an emerging Roadmap and critical Milestones, can provide sufficient guidance to the teams for initial PI Planning and execution. Features, capabilities, Stories, and Enablers are used to further define and realize the solution behavior.
- In more complex and/or regulated environments, substantially more investment in solution intent documentation is required. Compliance needs may mandate the creation of standards-based or other technical specifications.

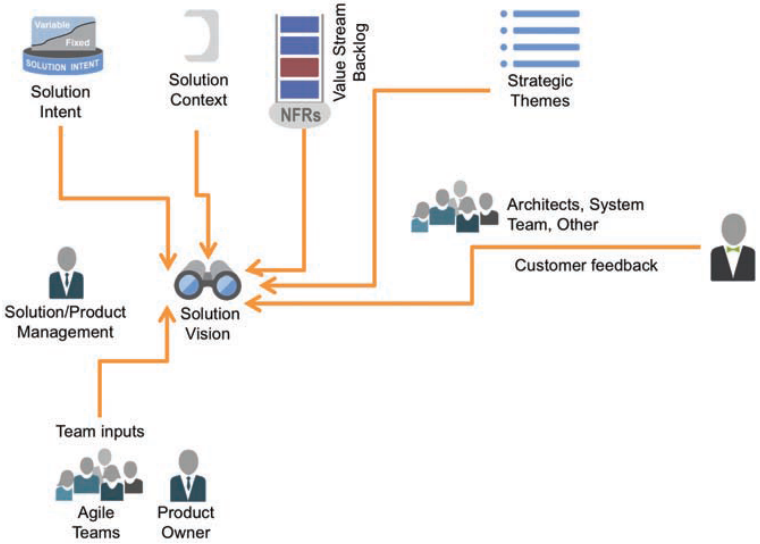
Solution Vision

Given that longer-term view, Product and Solution Management have the responsibility for converting the portfolio vision to a solution vision indicating the reason and direction behind the chosen solution.

- In order to do so, specific questions must be asked and answered:
- • What will this new solution do?
- • What problem will it solve?
- • What Features and benefits will it provide?
- • For whom will it provide them?
- • What NFRs will it deliver?

Program Vision

- In 4-level SAFe, in addition to the solution vision, each ART will likely have its own vision, detailing the direction of the specific capabilities or subsystems that it produces.
- This vision should be tightly coupled to the solution vision it supports.



Agile Architecture

Agile Architecture

Agile Architecture is a set of values and practices that support the active evolution of the design and architecture of a system, *concurrent with* the implementation of new business functionality. Enables incremental value delivery by balancing between two end points:

- ***Emergent Design*** provides the technical basis for a fully evolutionary, incremental implementation approach, and it helps the design respond to immediate user needs. The design *emerges* as the system is built and deployed.

- ***Intentional Architecture*** provides guidance and technical governance to Agile programs and teams for certain overarching technical constructs. These provide for commonality of approach, elimination of redundancy, and higher robustness of the full system.
  - Organizations must respond to new business challenges with larger-scale architectural initiatives that require some intentionality and planning.

Architectural Guidance

In systems of Enterprise scale, poor strategic technical planning, communication, and visibility can cause poor overall business systems performance, making significant redesign of multiple systems necessary. Poor economics and poor business systems performance are likely results.

- To prevent this, enterprise-class systems are well served by having some amount of **Intentional Architectural Runway*** built into their larger systems, and the larger system of systems, to support current and near-term business needs.
- In addition, some **architectural governance**—for example, to drive common usability and system behavioral constructs across the enterprise's Solutions—is beneficial.
- To address parts of this problem, **SAFe highlights the role of System and Solution Architects in providing much of this guidance** at the Program and Value Stream Levels.

* Another kind of runway

Emergent Design is not enough

As Agile practices mature and are adopted by larger teams and teams of teams, there comes a point at which ***emergent design*** is an insufficient response to the complexity of large-scale system development, and these problems start to occur:

- Excessive redesign and delays; flow bottlenecks
- Different architecture constructs in support of the same capabilities, increasing maintenance costs
- Reduced collaboration and synchronization among teams
- Low velocity
- Systems that are difficult to integrate and validate
- Deterioration of system qualities (Nonfunctional Requirements, or NFRs)
- Low reuse of components; implementation redundancies

Both are Needed.

- ***Emergent design*** enables fast, local control so that teams react appropriately to changing requirements without excessive attempts to future-proof the system.
- ***Intentional architecture*** provides the guidance needed to ensure that the system as a whole has conceptual integrity and efficacy.

A deep reciprocity between emergent design and intentional architecture can occur only as a result of *collaboration* between Agile Teams, System and Solution Architecture, Enterprise Architects, and Product and Solution Management.

Principle #1: Design Emerges, Architecture Is a Collaboration

- Clearly, it's best to have both: fast, local control of emergent design and a global view of intentional architecture. The combination provides the guidance needed to ensure that the system as a whole has conceptual integrity and efficacy.

Architecture Runway

Together, ***Emergent Design*** and ***Intentional Architecture*** create the ***Architectural Runway*** needed to enable teams to deliver business value faster and more reliably.

- An architectural runway exists when the enterprise's **platforms** have sufficient technological infrastructure to support the implementation of the highest-priority Features and Capabilities in the backlog without excessive, delay-inducing redesign.
- Enable programs to create and maintain large-scale solutions.
- To mitigate the risk, programs must take care to ensure that the necessary architectural underpinnings for the most innovative new features are *already* in the system when planning for the PI.
- *Build* some runway, *use* it, and *extend* it

Building the Architecture Runway

Innovative or Greensfield Platforms:

- The **System or Solution Architect/Engineer** plays a role in the initial definition and build-out of the runway.
- The new infrastructure is initially put in place with just one or two Agile Teams—sometimes with the architect/engineer serving as Product Owner.
- These teams iterate like every other Agile Team on the program.

To support a stable velocity, the architectural runway needs to be continuously maintained and extended.

- Enablers extend the Runway;
- Architects define the new **Enabler Epics**;
- These are split into **Enabler Features** and/or capabilities, which are ultimately implemented by individual ARTs;
- Each **Enabler Feature** must be completed within a PI, ensuring the *system always runs*, at least at the PI boundaries;

The Fragile and Temporal Nature of System Architecture

The initial success of a new architecture can be temporary and a number of natural forces will tend to cause the architecture to be consumed over time:

1. Agile Teams are fast. They have an unparalleled focus and ability to deliver new features, thus consuming whatever runway exists.
2. Product Owners and Product/Solution Management are impatient. They've invested some time on internal system capabilities, and they will quickly move backlog priorities to the features that users are willing to pay for.
3. Architecture itself is fragile and needs to be continuously evolved. Technologies change in short time frames. Stuff obsolesces.
4. Customer needs change quickly, too

Types of Enablers

Enablers exist on all four levels of SAFe:

- **Enabler Epics** at the Portfolio Level, cut across Value steams and PIs;
- **Enabler Capabilities** at the Value Stream and Program Level, fit into a PI;
- **Enabler Features** at the Program Level;
- **Enabler Stories** at the Team Level, and fit into an iteration.

- Enabler epics and enabler capabilities can cut across multiple value streams or ARTs respectively. If high risk may implement over a number of PIs. If cost of delay is high them implement concurrently across ARTs.

3 main types of Enablers:

1. **Exploration** – Exploration enablers are used to build understanding of what is needed by the Customer, to understand prospective Solutions, and to evaluate alternatives
2. **Architecture** – Architectural enablers are used to build the Architectural Runway in order to enable smoother and faster development
3. **Infrastructure** – Infrastructure enablers are used to build and enhance the development and testing (and occasionally deployment) environments, thereby facilitating faster development and higher-quality testing

Delivering Enablers

Enablers are often created by architects or by systems engineering at the various levels, whether by Enterprise Architects at the Portfolio Level or by Solution and System Architects/Engineering at the value stream and program levels.

- The architects who create the enablers steer them through the Kanban systems, providing both the guidance needed to analyze them and the information needed to estimate and implement them.
- Some enablers will emerge locally from the needs of the Agile Teams, Agile Release Trains (ARTs), or value streams to improve the existing solution.

3 options to deliver Enablers:

1. enabler is big, but there is an incremental approach to implementation. The system always runs
2. enabler is big, but it can't be implemented entirely incrementally. The system will need to take an occasional break
3. enabler is really big, and it can't be implemented incrementally. The system runs when needed; do no harm

Strangler Pattern is one way to deliver enabler incrementally over legacy system.

Principal #2 – The Bigger the System, the Longer the Runway

In order to achieve some degree of runway, the enterprise must continually invest in extending existing platforms, as well as building and deploying the new platforms needed for evolving business requirements.

- The enterprise commits to implementing architectural initiatives incrementally in the main code base. Doing so means that **Architectural Initiatives** must be split into **Enabler Features**, which build the runway needed to host the new business features.

- Ensuring the system always runs means that new **architectural initiatives** are implemented piecemeal and may not even be exposed to the users in the PI in which they are implemented.
- In this way, the architectural runway can be implemented and tested behind the scenes, allowing shipment throughout, and then exposed to users when a sufficient feature capability exists in the next PI or so.
- The concept of runway illustrates how intentional architecture and emergent design effectively complement each other at scale: intentional, high-level ideas in support of future functionality are adapted and instantiated by Agile Teams; they are empowered to figure out the optimal emergent design.

    - **Note**: Safe also uses the terms : “Enterprise Architectural Runway” and “Intentional Architectural Runway”.

ART/Team Organisation

ARTs should focus on a single products or solution objective.

2 main types:

- **Capability**, e.g. customer enrolment
- **Subsystem**, e.g. mobile app

Then organise teams by:

- Feature
- Component
- Platform
- Architecture Layer
- Programming Language
- Middleware
- UI
- DBMS
- etc

} Main Approach

SAFe Estimation

Its all down to Story Points

Agile Estimating and Planning:

- The formerly too-detailed business cases, too-early requirements specificity, and too-detailed work breakdown structures are replaced with Agile estimating and planning, **using the currency of Story points**, applied consistently through the team, program, value stream, and portfolio.

Estimating Larger Work Items:

- At the Portfolio and Value Stream Levels, it is often necessary to estimate larger work items (Epics and Capabilities) to determine the potential economic viability of these initiatives.
- In addition, development of ARTs and Value Stream Roadmaps requires both a knowledge of estimating (how big is the item) and ART velocity (how much capacity does the ART have to do it).
- In order to do this, Kanban teams break larger initiatives into stories for estimating, just as the Scrum teams do.

Spotify

Spotify Matrix Model

Squad – small (<8 people) cross functional self organising team - autonomous. 3 main types:

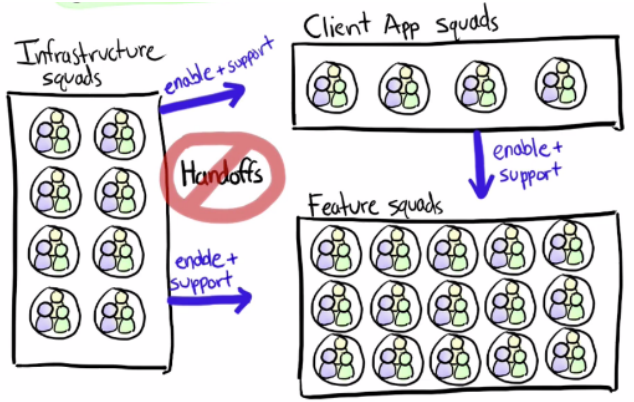
- **Feature** – the trains that deliver, e.g. search.
- **Client App** – support feature squads, for a platform
- **Infrastructure** – support above squads, e.g. CD, operations, testing.

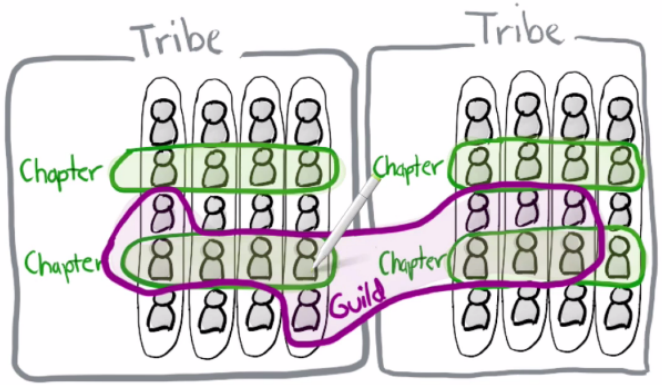
Synchronisation: Each Client App has a release train that may include a number of features.

Across Squads are:

- **Tribes** - a collection of squads that work in related area (sized based on Dunbar) managed by scrum of scrums.
- **Chapters** - people with similar skills, local to a tribe, and working within the same general *competency area*. Meet regularly to discuss their area of expertise and their specific challenges, e.g. testers, QA, web development or backend.
- **Guilds** - *Communities of Interest*, a group of people that want to share knowledge, tools, code, and practices. More wider reaching than Chapters, across whole org, e.g. Web Technology Guild.

"If each squad was fully autonomous and had no communication with other squads, then what is the point of having a company?"

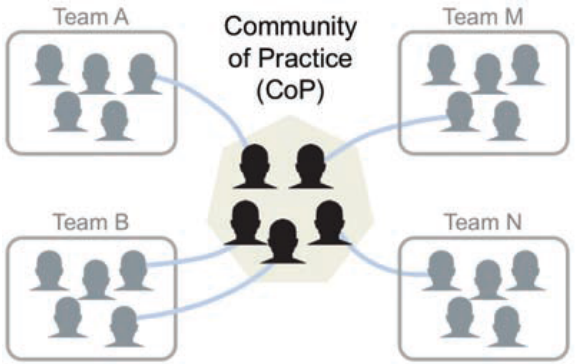




SAFe - Communities of Practice

A **Community of Practice (CoP)** is an informal group of team members and other experts, acting within the context of a program or enterprise, that has a mission of sharing practical knowledge in one or more relevant domains.

- Like a Spotify 'Guild'



Spotify and Architecture – an aside

Spotify is mainly SOA based:

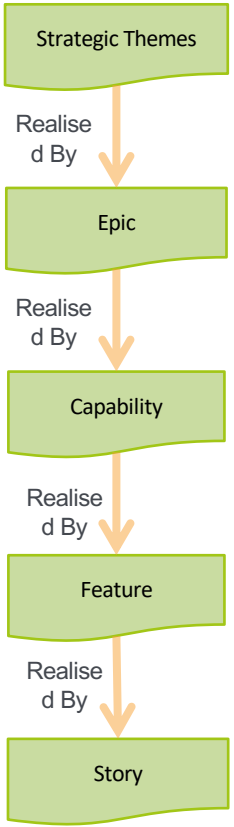
- Feature teams often need to update multiple systems to build a new features;
- Risk is, architecture gets messed up unless someone focuses on integrity of the systems as a whole;
- So each system has a 'System Owner', usually a dev + ops pair, they focus on quality, documentation, technical debt, stability, scalability, and release process.

Also have a chief architect role, a person who coordinates work on high-level architectural issues that cut across multiple systems.

- reviews development of new systems to make sure they avoid common mistakes, and that they are aligned with our architectural vision.
- feedback is always just suggestions and input - the decision for the final design of the system still lies with the squad building it.

Appendix: Requirements

Requirements Types – Functional Breakdown



Strategic themes are specific, itemized business objectives that connect the SAFe portfolio to the enterprise business strategy. They provide business context for decision-making within the portfolio, effecting the economic framework and investments in value streams and ARTs. They serve as inputs to the budget, portfolio, solution, and program backlog decisions.

Epics are significant initiatives that help guide value streams toward the larger aim of the portfolio. They are investment intensive and far ranging in impact. They require a formulation and analysis of cost, impact, and opportunity in a lightweight business case, as well as financial approval before implementation.

Capabilities are similar to features; however, they account for higher-level behaviors of the solution, which often spans multiple ARTs. Capabilities are maintained in the value stream backlog and are sized to fit in a program increment, so that each PI delivers solution value

A feature is a service provided by the system that fulfills stakeholder needs. Each is developed by a single Agile Release Train. They are maintained in the program backlog and are sized to fit in a program increment so that each PI delivers conceptual integrity. Each feature includes a statement of benefits and defined acceptance criteria.

Stories are the primary artifact used to define system behavior in Agile development. Stories are not requirements; they are short, simple descriptions of a small piece of desired functionality, usually told from the user's perspective and written in the user's language. Each story is intended to support implementation of a small, vertical slice of system functionality, supporting highly incremental development.

Enablers

- **Enablers encapsulate the exploration and the architectural and infrastructure development activities necessary to support some future solution capability.** They occur at all levels of the framework and are described as enabler epics, enabler capabilities, enabler features, and enabler stories, depending on their level.

- **Enabler Capability**
- Enabler capabilities occur at the value stream level, where they capture work of that type. As these enablers are a type of capability, they share the same attributes, including a statement of bene ts and acceptance criteria, and they must be structured so as to fit within a single PI.
- **Enabler Epic**
- Enabler epics are a type of epic, and as such are written using the value statement format defined for epics. They tend to cut across value streams and PIs. They require a lightweight business case to support their implementation. They are identified and tracked through the portfolio Kanban system.
- **Enabler Feature**
- Enabler features occur at the program level, where they capture work of that type. As these enablers are a type of feature, they share the same attributes, including a statement of benefits and acceptance criteria, and are structured so as to fit within a single PI.
- **Enabler Story**
- Enabler stories, as a type of story, must fit in iterations. However, while they may not require user voice format, they have acceptance criteria to clarify their requirements and support demonstration and testing.

NFRs

NFRs

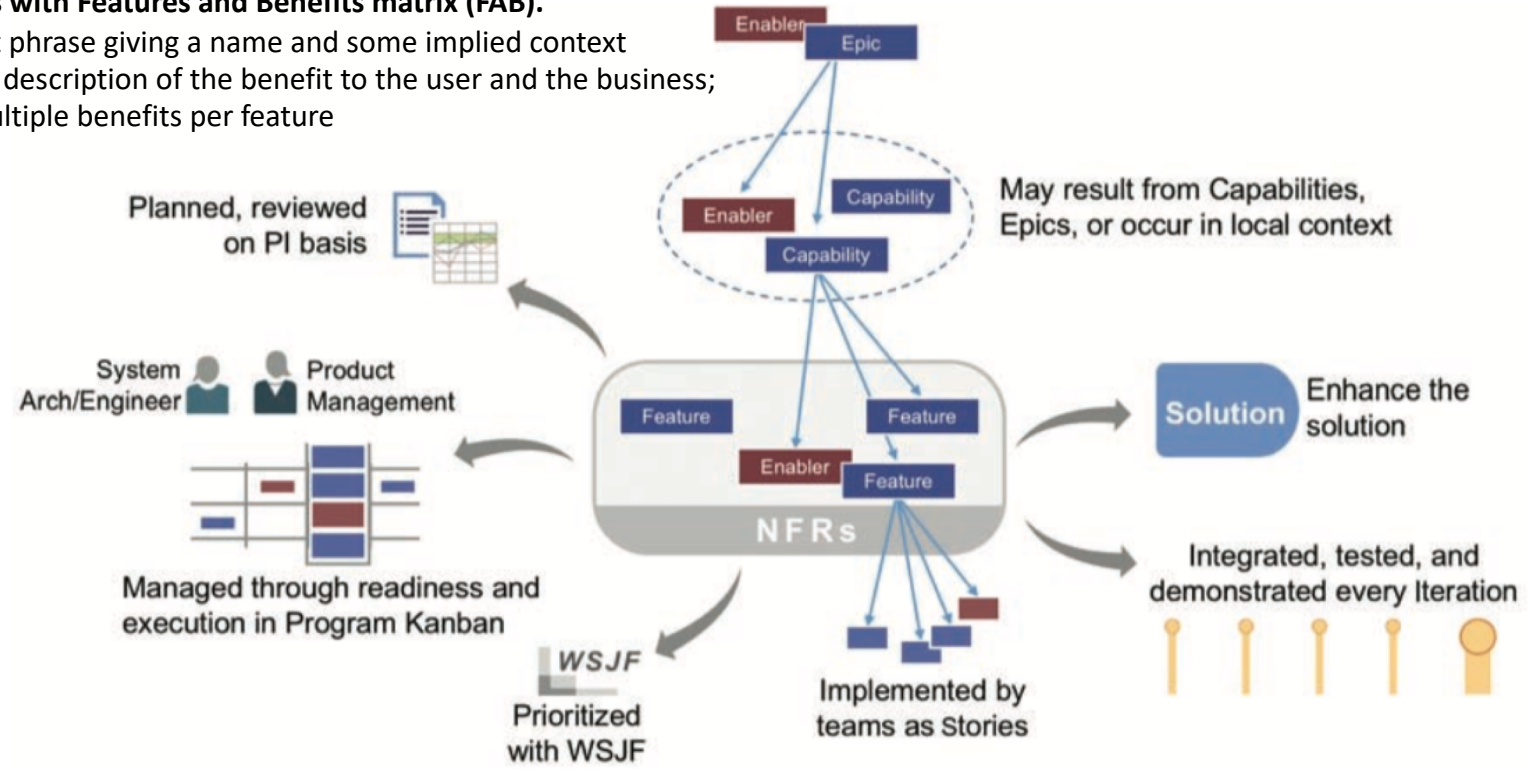
Nonfunctional requirements describe system attributes such as security, reliability, performance, maintainability, scalability, and usability. They can also be constraints or restrictions on the design of the system.

- NFRs are just as critical as the functional Epics, Capabilities, Features, and user stories, as they ensure the usability and efficacy of the entire system; failing to meet any one can result in systems that do not meet internal business, user, or market needs or that do not meet mandatory requirements imposed by regulatory or standards agencies.
- NFRs are persistent qualities and constraints and, unlike functional require- ments, are typically revisited as part of the Definition of Done for each Iteration, Program Increment, or Release.
- NFRs exist at all four SAFe levels: Team, Program, Value Stream, and Portfolio.
- NFRs are significant attributes of the solution that the Agile Release Train and Value Streams create, the most obvious representation of NFRs are at the Program and Value Stream Levels. System and Solution Architect and Engineering are often responsible for defining and refining these NFRs.
- NFRs play a key role in understanding the economics of fixed versus variable solution intent.

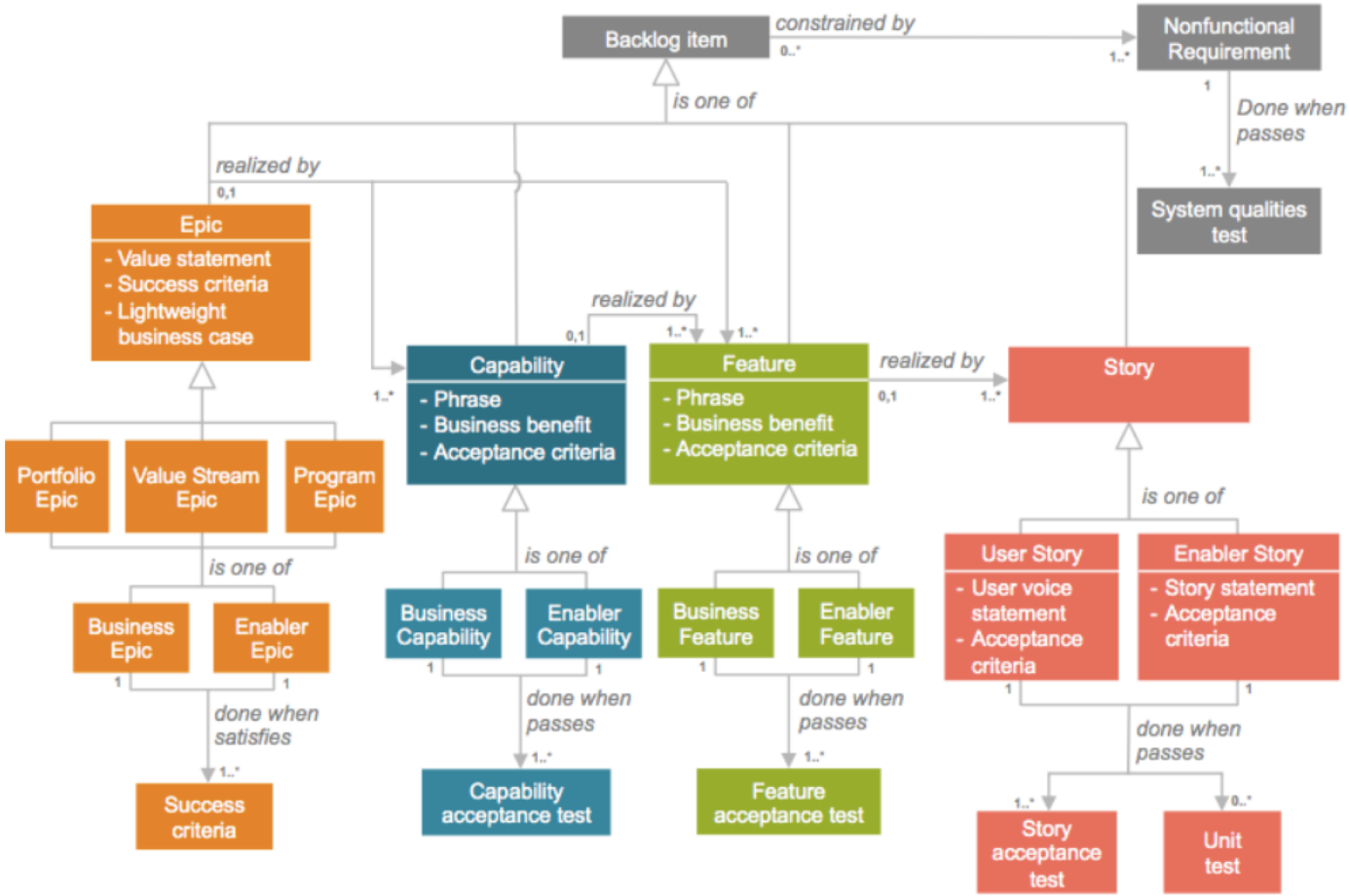
Features

Describing Features with Features and Benefits matrix (FAB).

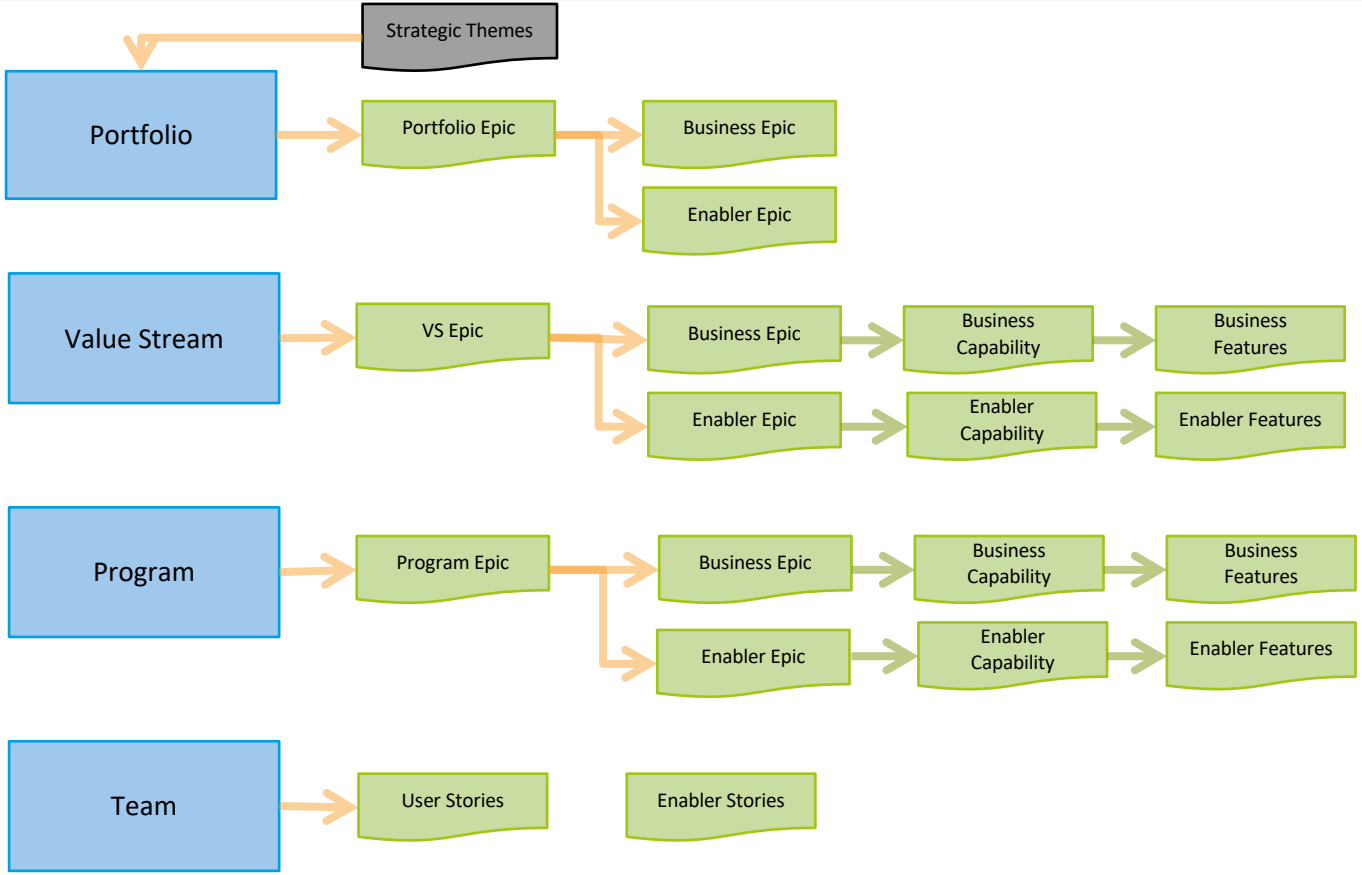
- Feature – A short phrase giving a name and some implied context
- Benefit – A short description of the benefit to the user and the business;
- There may be multiple benefits per feature



Requirements Model



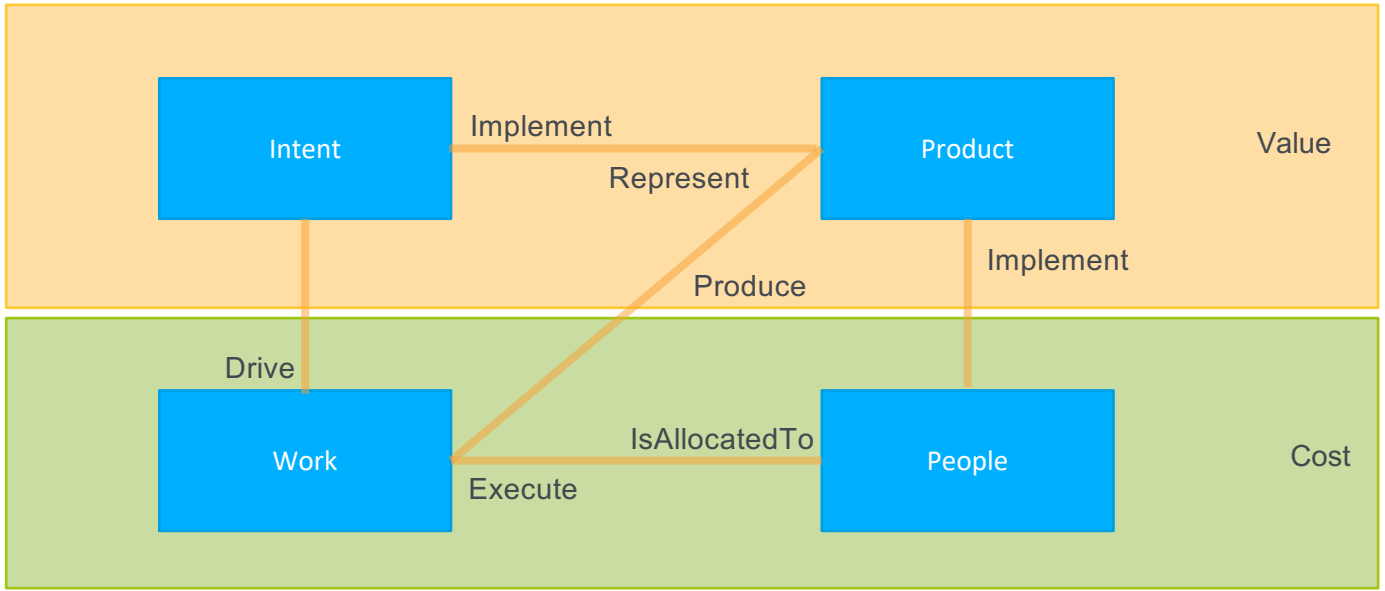
Requirements Levels



Split Into

Appendix: KRUSHEN

4 Concepts in Software Development – a Model



4 Concepts

1. **Intent** – what the project is trying to achieve, the intentions of the stakeholders, the vision, requirements, specifications, constraints, tests. Change requests modify the intent.
2. **Product** – the outcome of the project, what has been achieved. The software and other artefacts e.g., installers, manuals, training.
3. **Work** – The discrepancy between Intent and Product drives a project, the process, activities, tasks and steps.
4. **People** – systems development is a human intensive activity

Each has 3 Attributes:

- Time
- Quality
- Risk & Uncertainty

Appendix: Agile At Scale

Other Scaled Agile Approaches

There are many methods to Scale ‘Agile’ methods:

- Less,
- DAD,
- SoS,
- Spotify,
- Leading Agile,
- Slim,
- SCARE,
- FAST

What is the common ground, the essentials ?

Essence defines 3 Key Concerns

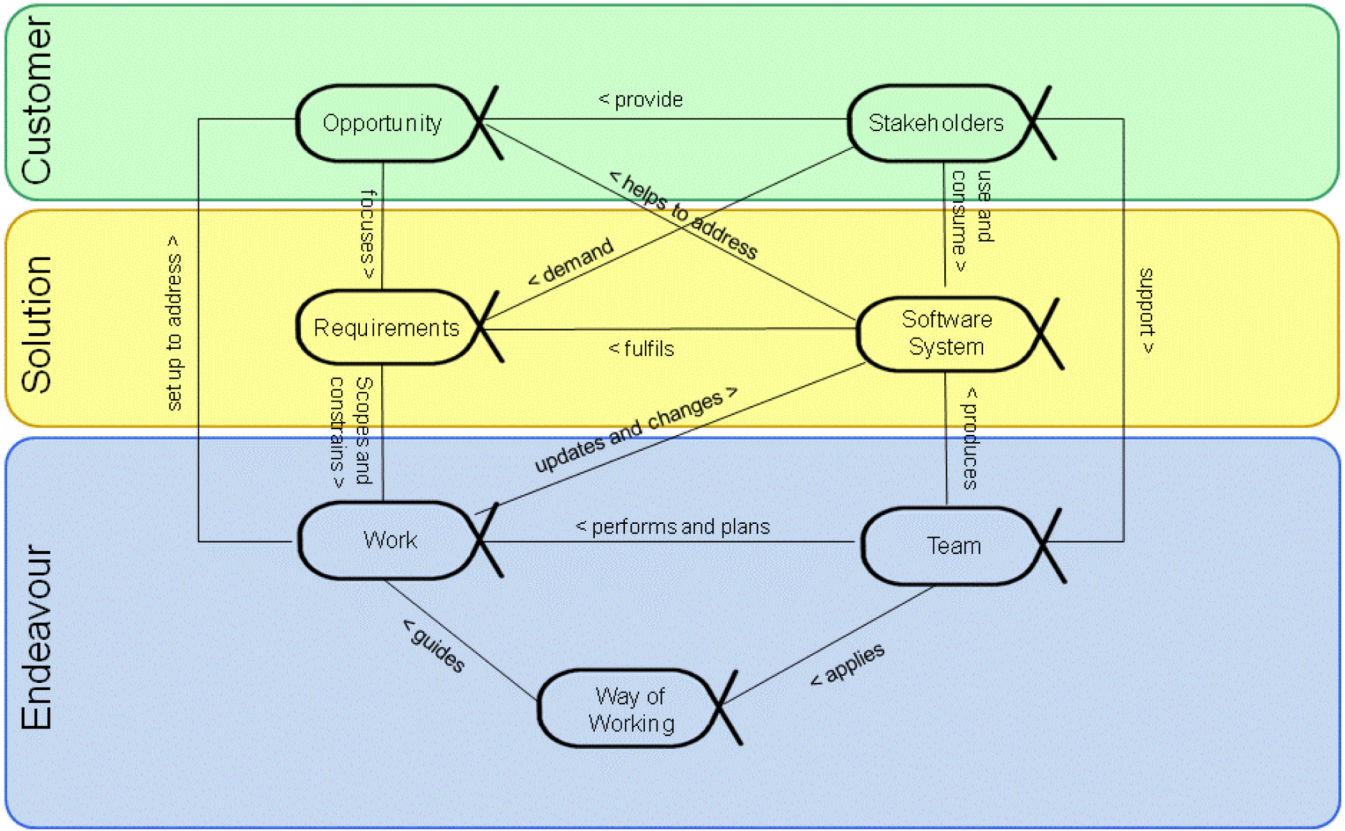
Undertaking this Endeavour to create a Solution with the Customer

Customer

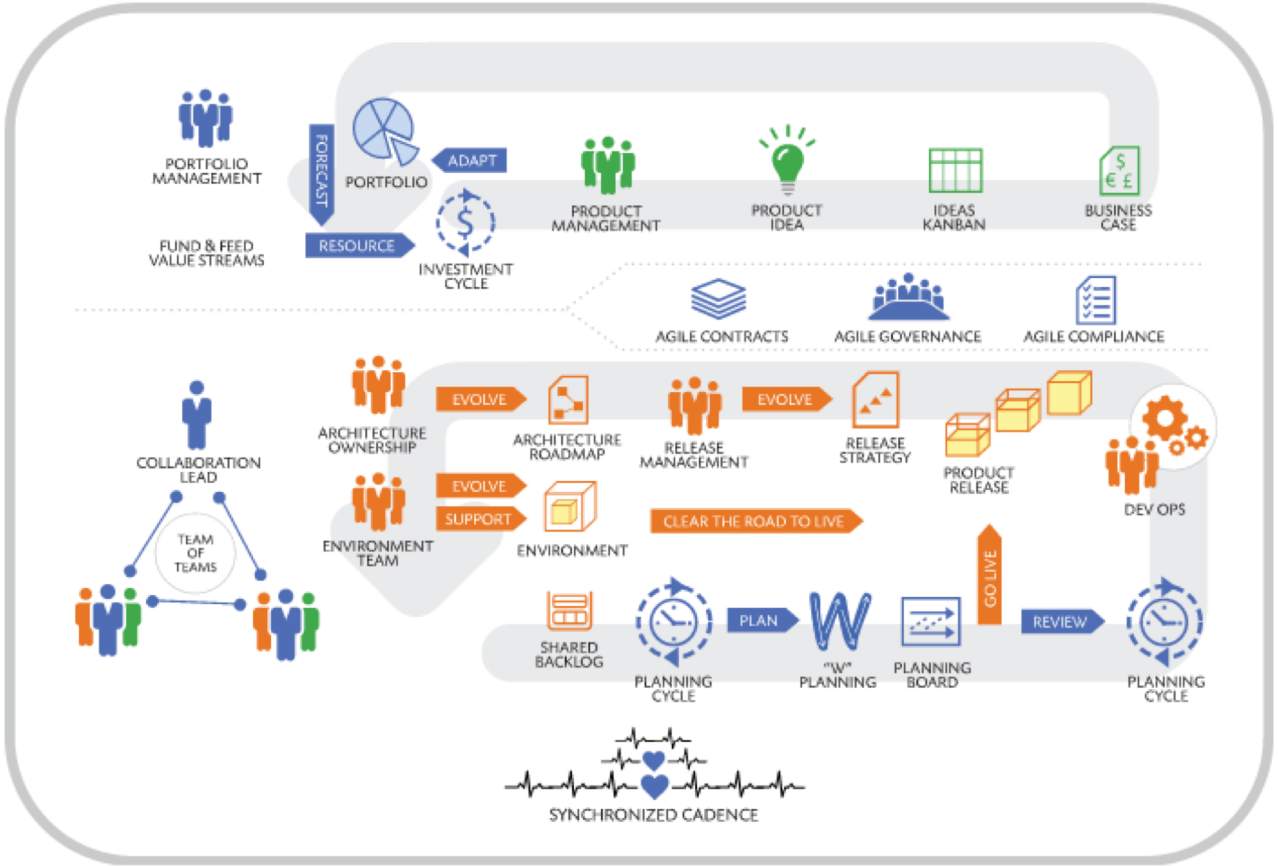
Solution

Endeavour

Add Core Elements – things to work with – 'Alpha's'

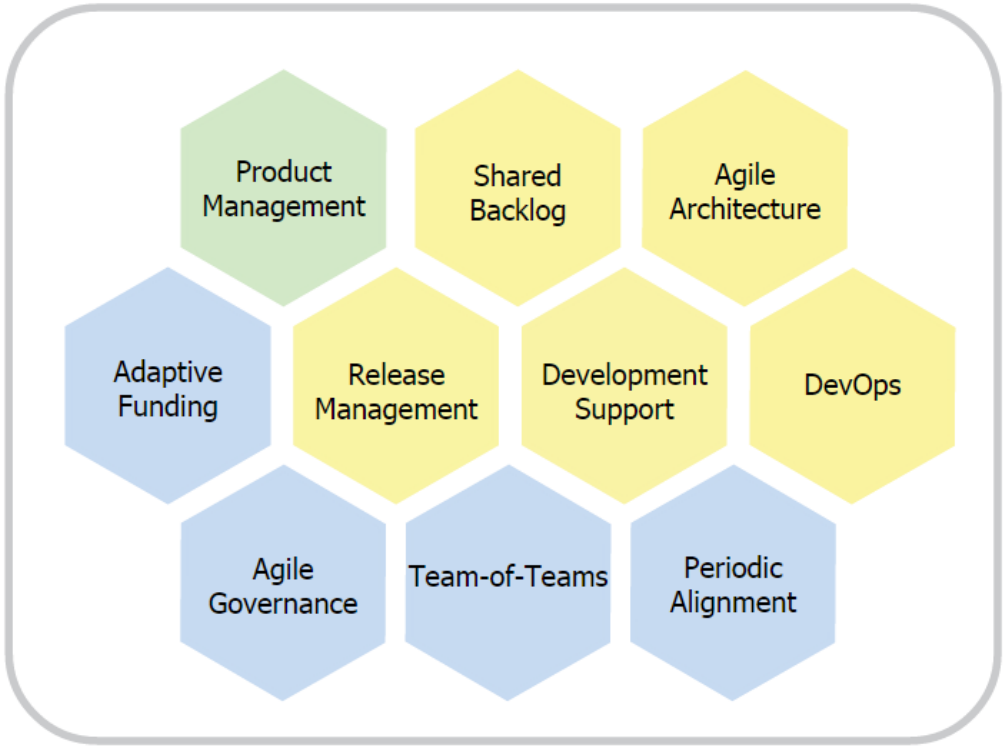


Agile At Scale Overview – finding the common ground.



Practices

The Agile at Scale Essentials ***practices*** provide a basic starter kit toolbox that covers all the common and critical aspects of scaled agile development.



The End