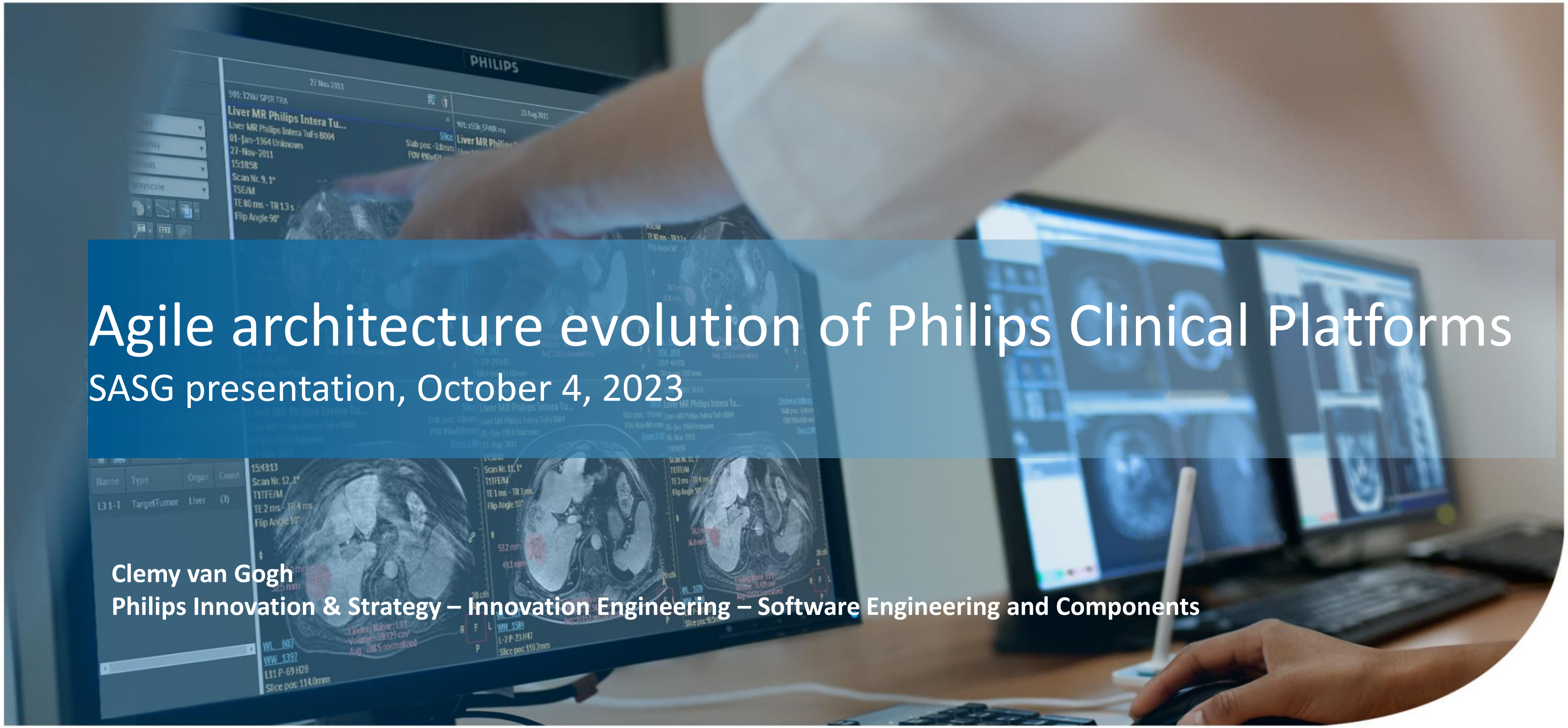


Agile architecture evolution of Philips Clinical Platforms

SASG presentation, October 4, 2023

Clemy van Gogh
Philips Innovation & Strategy – Innovation Engineering – Software Engineering and Components

innovation + you

PHILIPS

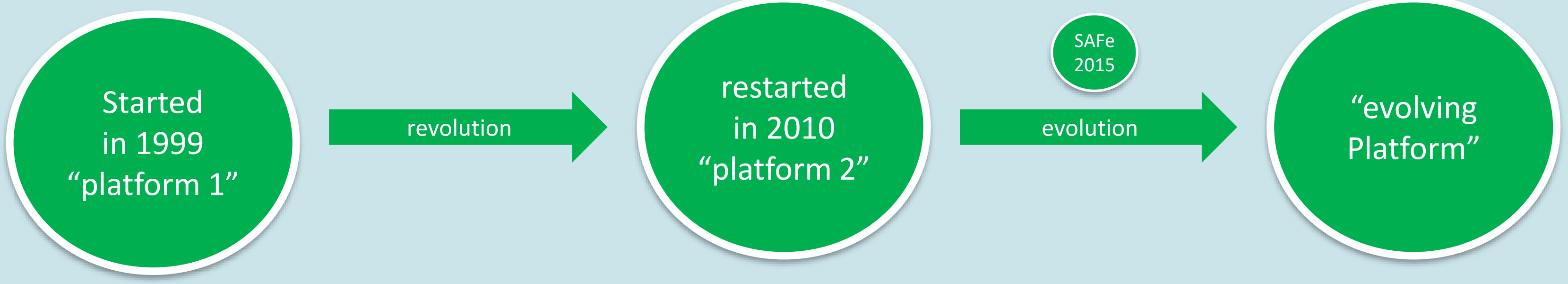
Contents

- **Introduction**
- Agile architecture evolution in Philips
- Challenges
- Summary

Clemy van Gogh

- MSc Computer Science University Nijmegen
- 30+ years of experience in ASML, Thalys, Philips Healthcare
- 20+ years of experience as product, system and platform architect in Philips Healthcare
- Lead architect Philips Clinical Platforms
- Contributed to Agile process rollout in Philips for agile Architecture and Platforms

Facts & figures



30+
Platform assets/ services

11+
Philips businesses served

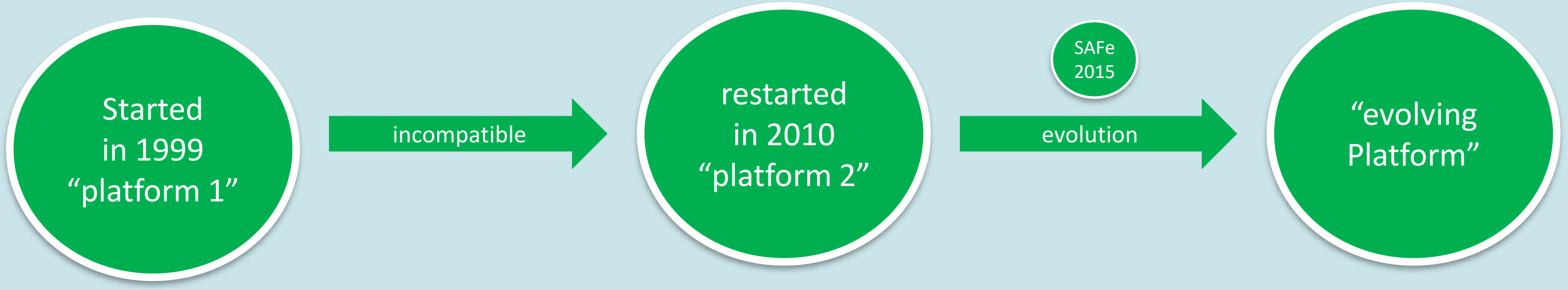
~10
releases per year on demand

2
R&D sites Eindhoven Bangalore

150
Trained in SAFe 4.0 CI/CD

2
weekly sprint releases

Facts & figures



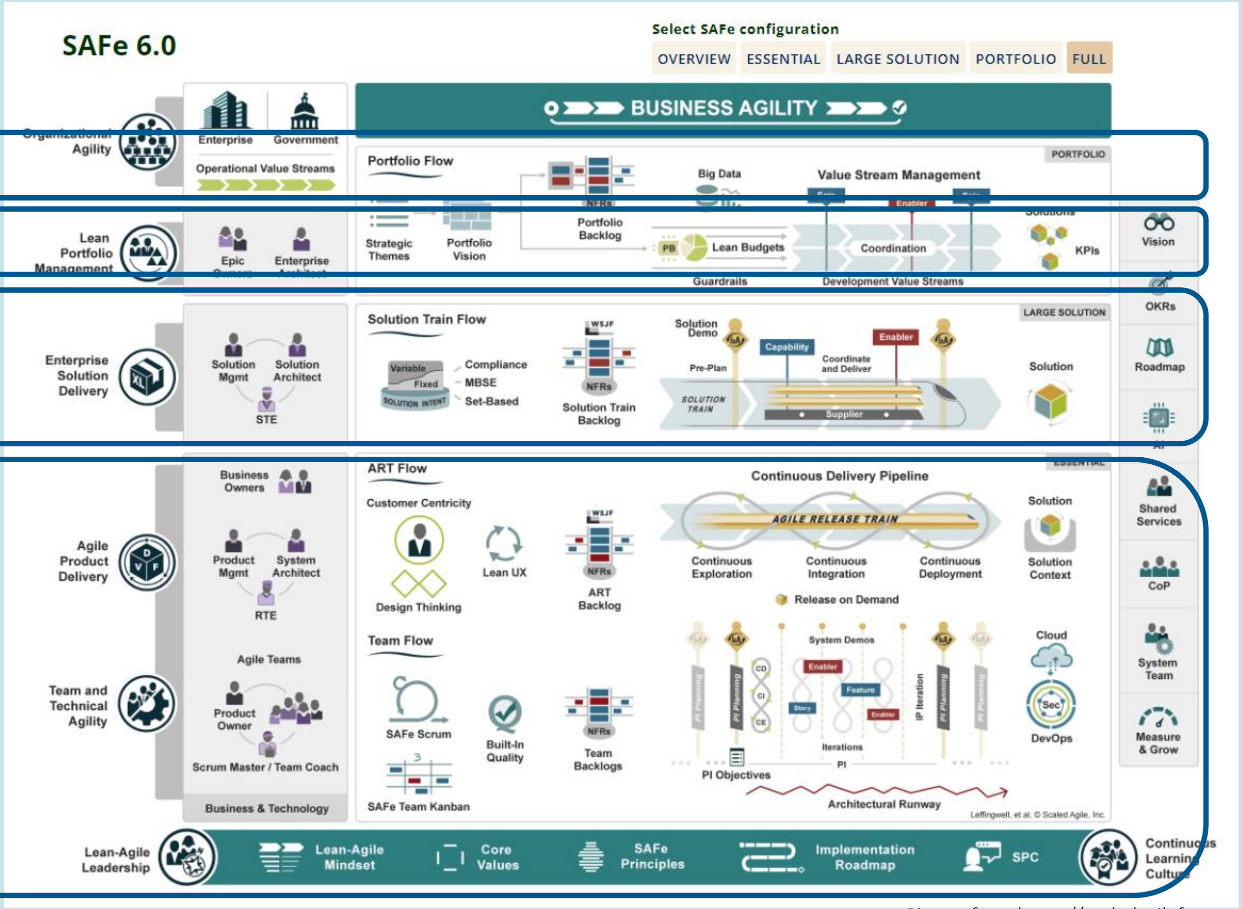
Philips Clinical Platform technologies:

- Programming languages: C#/.NET, C++, TypeScript/JavaScript/Angular/React
- Windows, Linux (partially) and AWS Cloud/Kubernetes (partially)
- Deliverables: libraries, Windows executables, Docker containers
- GitHub & innersource

Some numbers:

- ~2.8M product LOC
- ~2.2M test LOC / 70K test cases
- 20 builds per day / 10 acceptance test runs per day
- Full test suite execution once per day

Clinical Platforms context in Philips



Philips Strategic Architecture framework

Philips & Business portfolio management

Business units have their own dev trains and solution integration

Clinical Platforms operates as separate organization in Philips, working according SAFe;
Internal supplier in Philips

Picture from https://scaledagileframework.com/

Platform Product management and collaborations

Clinical Platform portfolio management
- Review of EPICS
  - Value for platform
  - Value for business/Philips
  - Architecture fit
  - Opportunity enablement
- Life Cycle Management
- Priority & roadmapping

Philips enterprise guidance
- Strategic direction
- Product portfolio & platforms
- Architecture guidelines

Business engagement (per business)
- Participate in product/solution architecture
- Customer surveys (NPS)
- Yearly and quarterly roadmap & budget alignment

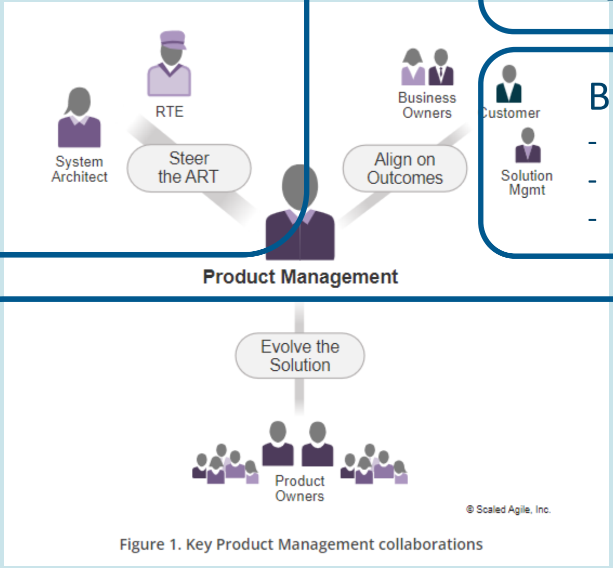


Figure 1. Key Product Management collaborations

Functional & Architecture EPICs

Realization (ART ~125fte)

Picture from https://scaledagileframework.com/

Contents

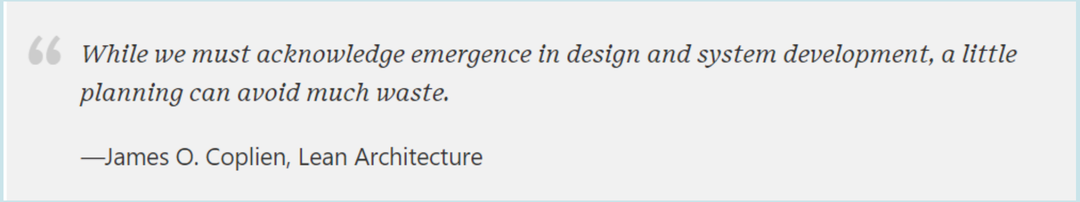
- Introduction
- **Agile architecture evolution in Philips**
    - Agile architecture definition
    - Critical success factors for evolution
- Challenges
- Summary

What is Agile Architecture?

Definition: “Agile Architecture is a set of values, practices, and collaborations that support a system's active, evolutionary design and architecture.”

Agile Architecture:

- Evolves while supporting the needs of current users
- Avoids overhead and delays associated with phase-gate and BDUF methods
- Ensures the system always runs
- Balances emergent and intentional design
- Takes a systems view across the entire value stream



While we must acknowledge emergence in design and system development, a little planning can avoid much waste.

—James O. Coplien, Lean Architecture

Source: Advanced Topic - Agile Architecture in SAFe - Scaled Agile Framework

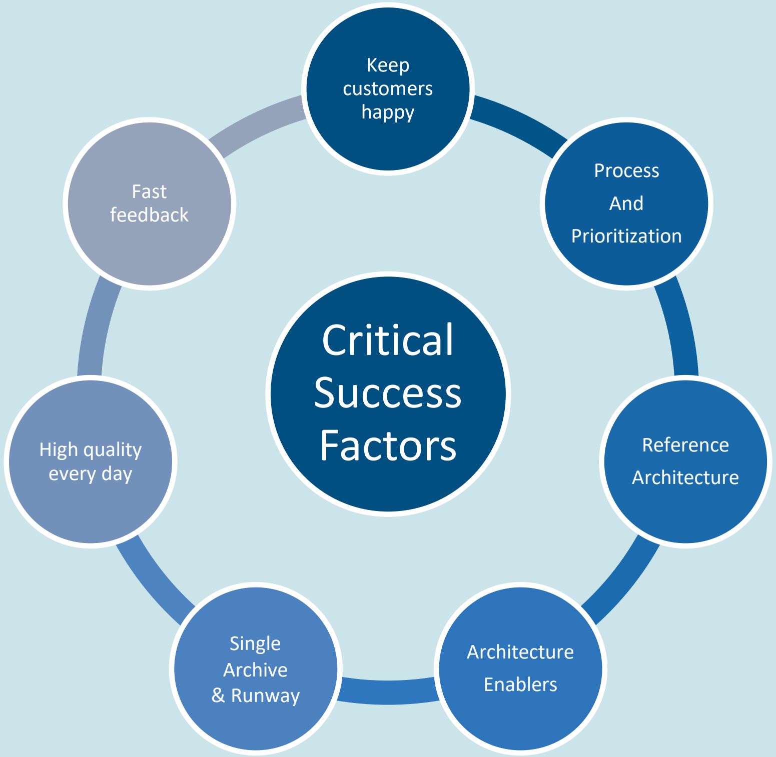
Architecture evolution

Major changes in the Philips platform

From	To
“monolithic black box with APIs”	Independent subsystems
Windows	Windows & Linux
Device/Premise	Device/Premise & Cloud
Windows UX/WPF	Client-Server & Web UX
Executables	Containers
.NET APIs & WCF	.NET and REST APIs

Agile Architecture Evolution

Critical Success Factors



Keep customers happy
Process And Prioritization
Reference Architecture
Architecture Enablers
Single Archive & Runway
High quality every day
Fast feedback
PHILIPS
Critical Success Factors

Keep customers happy during evolution

Philips business unit (BU) developers are the key customers

General expectations

- Easy to integrate & use
- High quality
- Nonfunctional: performance, resource usage, operational cost
- Close collaboration

Evolution related expectations

- Continuous integration
- Upgradeability: stable APIs & functionality
- Migration path in case of breaking changes
- Prepared for BU technology transitions
- [Support of older versions for installed base]

Process and Prioritization

SAFe Principle #1 – Take an economic view

Economic view is also applied for architecture changes & runway.

Part of standard "portfolio process"
Tradeoff with functional features.
Based on:

- Time criticality
- Business value / Enabler
- Effort / cost

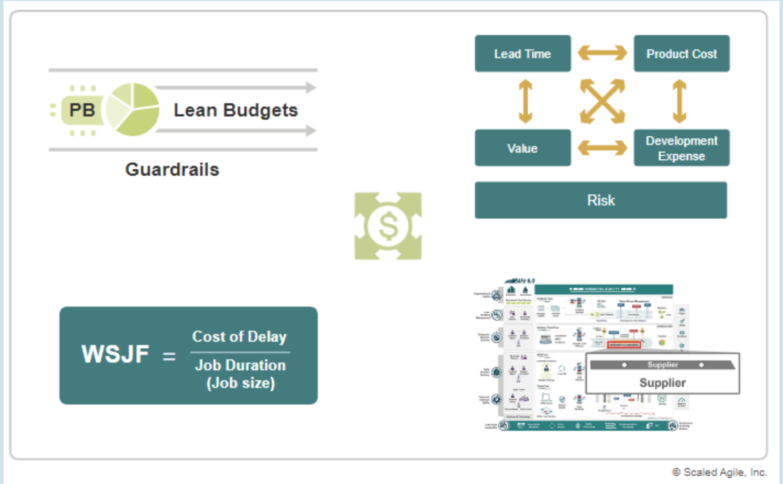


Figure 4. The four primary aspects of the economic framework

Incremental delivery with incremental value " deliver early and often"

Reserved percentage for keeping the software at sufficient quality (e.g. test coverage, technical debt)

Central Philips budget to prepare in advance for major changes

Source: SAFe Lean-Agile Principles - Scaled Agile Framework

Keep customers happy
Process And Prioritization
Reference Architecture
Architecture Enablers
Single Archive & Runway
High quality every day
Fast feedback
Critical Success Factors

Role of Reference Architecture

- Defining evolution guardrails
  - Strategic interfaces and technology choices
- Reference check for portfolio decisions
  - Framework for scope & architecture decisions
- Driving harmonization across portfolio
- Defining the "to be state" and transition path
  - Providing guidance to platform team
  - Providing guidance to BU solution architects
- Aiming at high decoupling
  - Preferably cross process
  - Technology independent APIs

Philips platform approach:

- Platform reference architecture document
- Yearly update
- Guideline during EPIC creation and (API) review

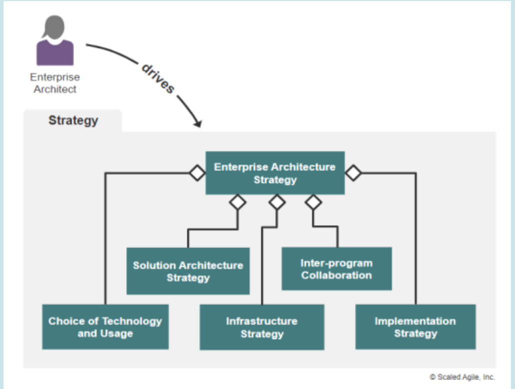
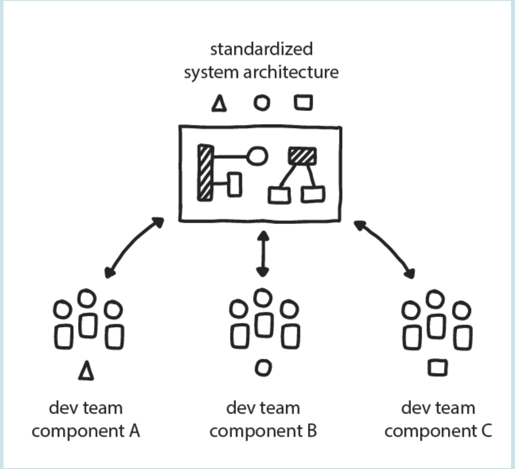

Enterprise Architect
drives
Strategy
Enterprise Architecture Strategy
Solution Architecture Strategy
Inter-program Collaboration
Choice of Technology and Usage
Infrastructure Strategy
Implementation Strategy
© Scaled Agile, Inc.

Figure 2. Five elements of enterprise architecture strategy

Philips level Architecture guardrails

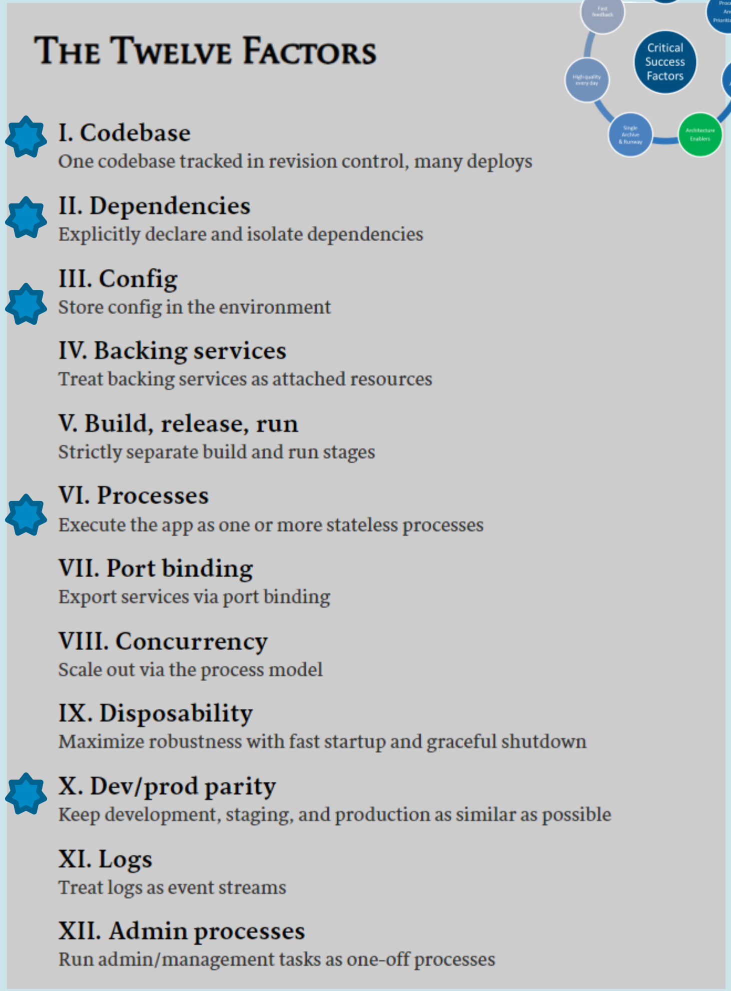


Platform level choices (aligned with BUs)

Source: SAFe Lean-Agile Principles - Scaled Agile Framework
Source: https://www.cnpatterns.org/

Architecture Enablers
The Twelve factor App

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.



The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

Source: The Twelve-Factor App (12factor.net)

Architecture Enablers
SAFe Principle #3 – Assume variability; preserve options

Upfront: agree upon integration interfaces as early as possible

Adapt based on integration feedback

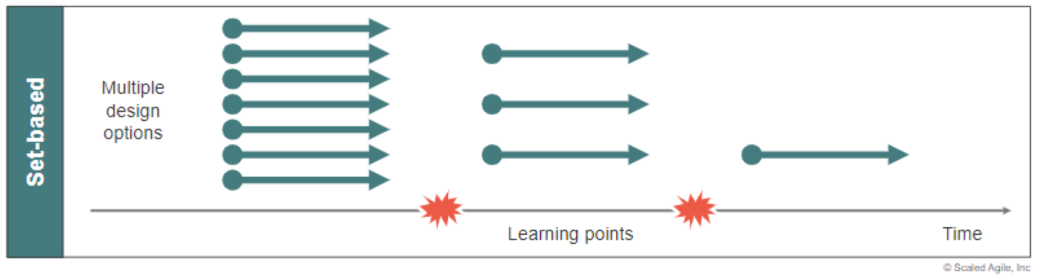


Figure 2. Set-Based Design provides multiple design options

Evaluate options, together with BU (e.g. performance evaluation)

Retain flexibility in platform development “decentralized decision making” (principle 9)

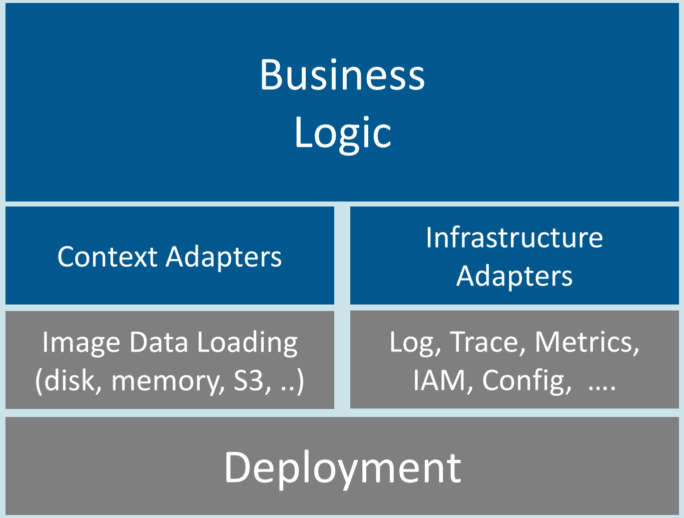
Platform choices:

- API first: hide implementation details behind the API
- Decouple deployment from business logic
- Support Philips deployment platforms (device, premise, cloud)

Source: SAFe Lean-Agile Principles - Scaled Agile Framework

Architecture Enablers

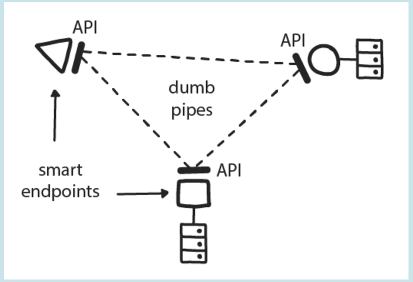
Decouple deployment from business logic


Business Logic
Context Adapters
Infrastructure Adapters
Image Data Loading (disk, memory, S3, ..)
Log, Trace, Metrics, IAM, Config, ….
Deployment

Philips platform choices:

- Flexible: pluggable / injection
- Deliberate choice on flexibility level (some designed upfront, some later)
- Limited to Philips deployment platforms (device, premise, cloud)

Architecture Enablers
CNCF PATTERN: Communicate Through APIs

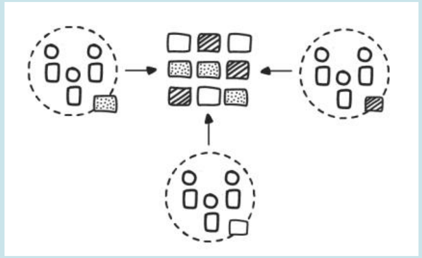


- Microservices should communicate with one another only through the network,using simple, consistent, and stable APIs.
- **Build stable APIs with backward compatibility.**
- **Place most of the service logic within the service itself, keeping the API simple and easily maintainable.**
- Smart endpoints, dumb pipes (most of the business logic is in the microservices themselves and not in the APIs).
- Ensure each microservice has no direct access to data of other microservices.
- Make sure there is version control and version management for APIs.

Philips platform choices:
- Categories of assets: library, "subsystem", application, microservice/container
- Technology independent interfaces (REST, industry standard)

Source: Communicate Through APIs (cnpatterns.org)

Architecture Enablers
CNCF PATTERN: Microservices architecture



Split applications into smaller, loosely coupled microservices that can be built, tested, deployed, and run independently from other components.

- Small and independent teams work on separate modules and deliver them with only limited coordination across the teams.
- Independent components allow different teams to progress at their own pace.

Philips platform choices:
- Split of “black box monolith” into independent subsystems (major evolution 2014-2018)
- “independent deployable assets” : initially executables for devices (Windows), Docker containers in cloud, moving to containers for devices (Linux)
- Independent asset lifecycles and upgrade

Source: Microservices Architecture (cnpatterns.org)

Single Archive and Runway

SAFe architecture evolution

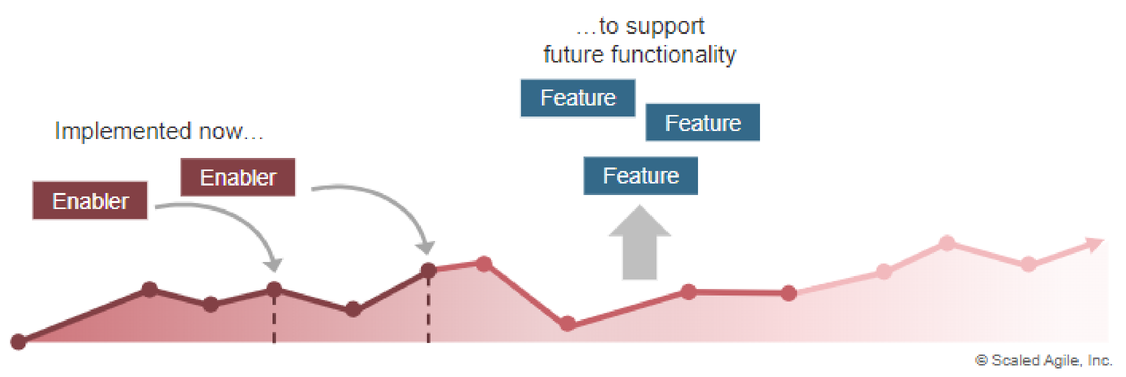


Figure 1. Architectural runway evolves in support of dynamic business needs

Philips platform principles:

- In archive changes, no code branches
- Backwards compatible
- Working software always, enabling continuous integration and release on demand

Source: Architectural Runway - Scaled Agile Framework

Single Archive and Runway

Architectural runway types

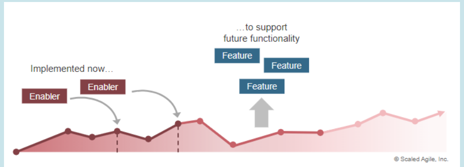
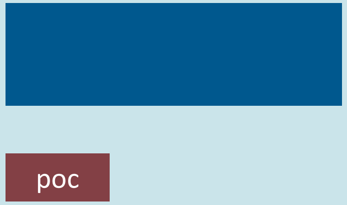
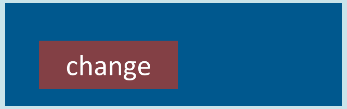
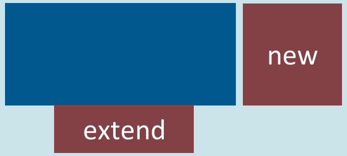


Figure 1. Architectural runway evolves in support of dynamic business needs

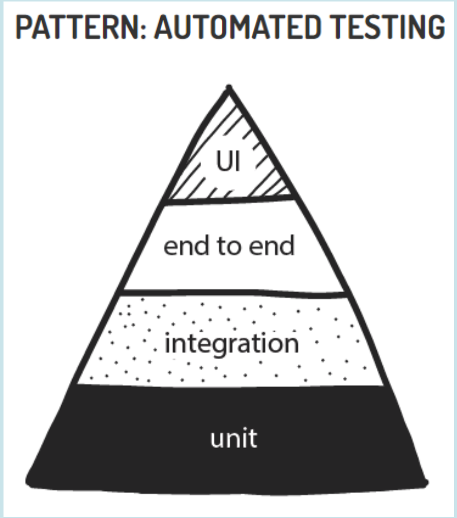
Source: Architectural Runway - Scaled Agile Framework

Goal	**Technology or concept evaluation**	**Technology change/redesign**	**New delivery model or deployment**
How	• Small proof of concept • "Out of archive" (github repo)	• In archive • In place change • Backwards compatible	• In archive • Parallel track • Backwards compatible • New API side by side
Archive			

High quality every day

Automated testing, delivery and dashboards

- Continuous build and test
  - "shifting left"
  - Fast 'short' tests
  - Verifying backwards compatibility and compliance
  - Aiming for high code coverage via automated tests
  - Test pass rate: 99.5 %
- Continuous delivery
  - Moved from file shares to Artifactory & Docker registry
  - Daily / weekly versions for integration
  - Release on demand, to cover the Quality and Regulatory artifacts



Enabling architecture evolution

- Working software always, enabling fast feedback: functional and non-functional
- Goal: 100% requirements coverage and >80% code coverage via automated tests
- Verifying in-archive runway changes
- Verifying backwards compatibility

Source: Automated Testing (cnpatterns.org)

Fast feedback

Principle #4 – Build incrementally with fast, integrated learning cycles

Keep customers happy
Process And Prioritization
Reference Architecture
Architecture Enablers
Single Archive & Runway
High quality every day
Fast feedback
Critical Success Factors

Internally in Cinical Platforms

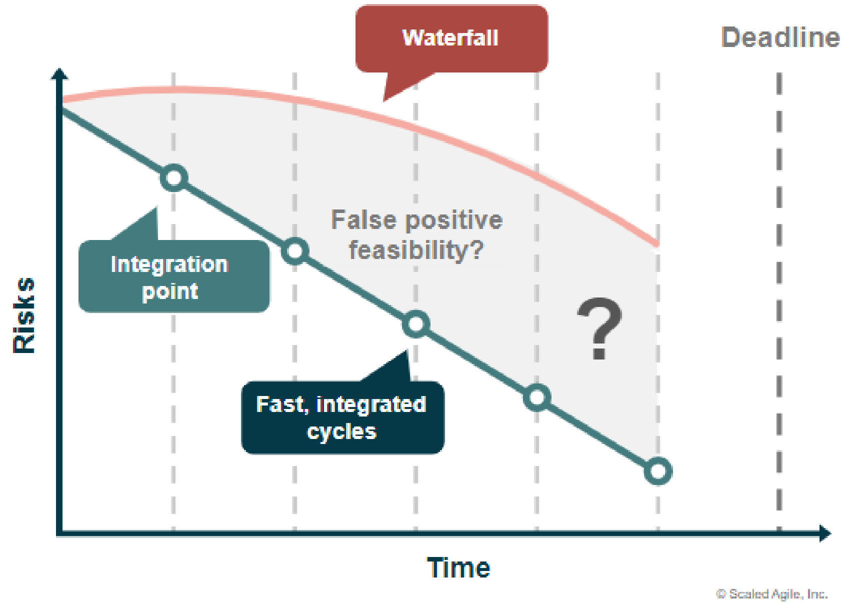


Figure 3. The lack of frequent integration points creates false positive feasibility.

Between platform and BU product development

Picture from https://scaledagileframework.com/

Contents

- Introduction
- Agile architecture evolution in Philips
- **Challenges**
- Summary

Challenges

- Business drive for functional features, hard to get priority for architecture runway
  - Tendency to delay runway till it becomes critical
- Higher development effort/cost to retain backwards compatibility
  - Runway development can be “perceived slow/costing more effort” which makes it more difficult to prioritize
- Archive/CICD must support technology transitions per asset
  - In few cases the use of new technology was delayed, as other assets could not move in the same pace
- Keeping up CICD infrastructure with technology changes
  - Extending build and test infrastructure for new deployments tends to have a long lead time
- Not able to enforce all architecture guidelines by the CICD pipeline
  - Surprises during development (leading to higher effort)
- Long integration / cycle time at the BU for medical devices
  - Delayed deprecation of old APIs/features
- Cloud native / serverless design
  - Are not portable to on premise deployments and lead to (partial) duplicate implementations

Contents

- Introduction
- Agile architecture evolution in Philips
- Challenges
- **Summary**

Summary

- Architecture evolution is a deliberate choice and mindset
- Evolving a large software stack has been a proven practice
- Evolution does not come for free
- Reference architecture is critical to guide the evolution (including deprecation)
- Automated verification and continuous integration is critical enabler
- Taking an economic view is key to prioritize the architecture runway at the right moment

Architecture evolution has helped Philips Clinical Platform to stay **up to date** and to address **evolving business needs** while keeping satisfied users

PHILIPS