CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Software Engineering Principles for Complex Systems

TDA594

SPLs Agile development & Industry automation

Sam Jobara
jobara@chalmers.se
Software Engineering Division
Chalmers | GU

Learning Objectives

- Introduce Agile Product Line Engineering APLE
- Present APLE industry practices
- Enabling APLE by automation tools

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Agile Product Line Engineering APLE

A Quick Review

Agile Product Line Engineering APLE

The big upfront design associated with (Product Line Architecture) PLA conflicts with the current need of "being tolerant with change=Agile" *

To make the development of product-lines more flexible and adaptable to changes, many **large companies** are leaning to adopt APLE principles.

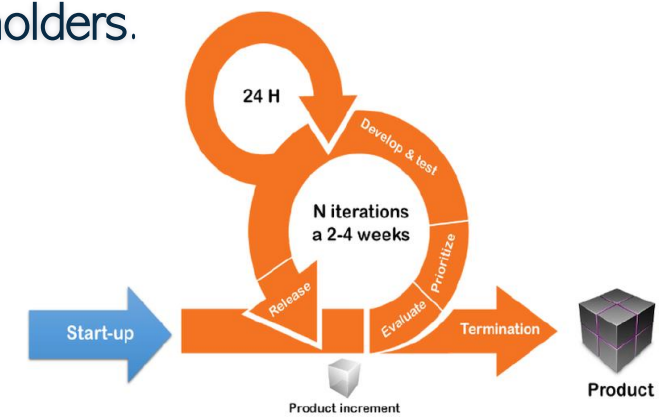
However, to put APLE into practice it is still necessary to introduce a **suitable framework** to assist and guide the agile practice and the evolve fully to APLE.

* Agile product-line architecting in practice: A case study in smart grids
Jessica Díaz ⇑, Jennifer Pérez, Juan Garbajosa, Technical University of Madrid-Universidad Politécnica de Madrid (UPM), E.U.

Agile Product Line Engineering APLE

Agile Software Development (ASD)* is an approach that is intended to enable rapid and flexible development, plans, are kept at a minimum, and work is organized in **short iterations developing the software product in increments, which is continuously tested and delivered in collaboration with all stakeholders**.

Agile methods are scaled up to approaches for large organizations, such as the **Scaled Agile Framework** (SAFe) or Scrum of Scrum (SoS). This is challenging whenever the software development is distributed all over the world.



Basic ASD process

** "Agile Product Line Engineering: The AgiFPL Method" Hassan Haidar1, Manuel Kolp1 and Yves Wautelet2, 1LouRIM-CEMIS, Université Catholique de Louvain, Belgium, 2KULeuven, Faculty of Economics and Business, Belgium*

Agile Product Line Engineering APLE

The common concepts in Scrum are:

- *Scrum roles:* "Product Owner", "Scrum Master", "Development Team", and "Stakeholders".
- *Scrum artefacts:* "Product backlog", "Sprint backlog", "User Story" (US), "Task", "Burn down charts", "Impediment backlog" and "Definition of done".
- *Scrum meetings:* "Sprint planning 1", "Sprint planning 2", "Daily Scrum", "Sprint review", "Sprint retrospective", and "Backlog refinement".

Scrumban has been used to help teams and organizations accelerate their transitions to Scrum from other development methodologies.

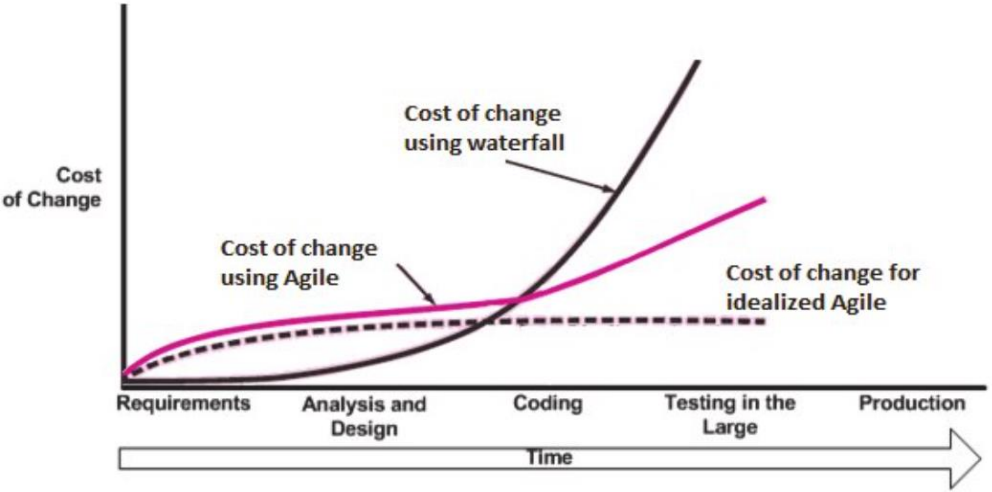
Scrumban more responsive to change than Scrum,
Scrumban retains all the roles and meetings of Scrum.

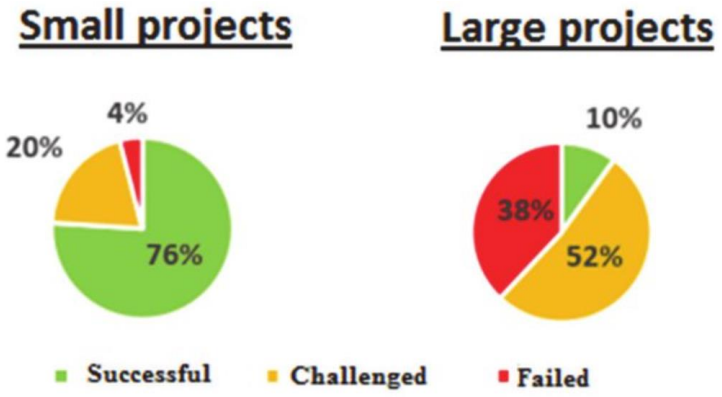
Agile Product Line Engineering APLE

One of the most important challenges consists of designing and evolving the PLEs while complying with the agile principles.
In fact, this challenge is also open in software architectures for single products, in which the **reconciliation of software architecture and agile communities** is a controversial issue that has been extensively discussed.
The conflict arise from the resistance of old schools of sequential waterfall design mentality, and lack of interest from old guards in traditional large industries.

Agile Product Line Engineering APLE

The graph* below shows that a waterfall model is much more sensitive against the requirements quality. The cost curve of waterfall model projects is growing exponentially – the later the defect is found, the more expensive it will be to fix it.





* Impact of Requirements Elicitation Processes on Success of Information System Development Projects Līga Bormane1 , Jūlija Gržibovska2 , Solvita Bērziša3

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Agile Product Line Engineering APLE

The AgiFPL FRAMEWORK

Agile Product Line Engineering APLE

The AgiFPL FRAMEWORK*

AgiFPL is an agile methodology designed to improve the agility within the PLE and to meet effectively any newly emerged business expectations. Its main goal is to **move teams from the classical approach to a more evolved APLE framework**.

Using a goal-oriented requirement engineering approach (GORE), will provide the mechanism to easily evolve PL architectures in an agile context and will establish a suitable reuse strategy.

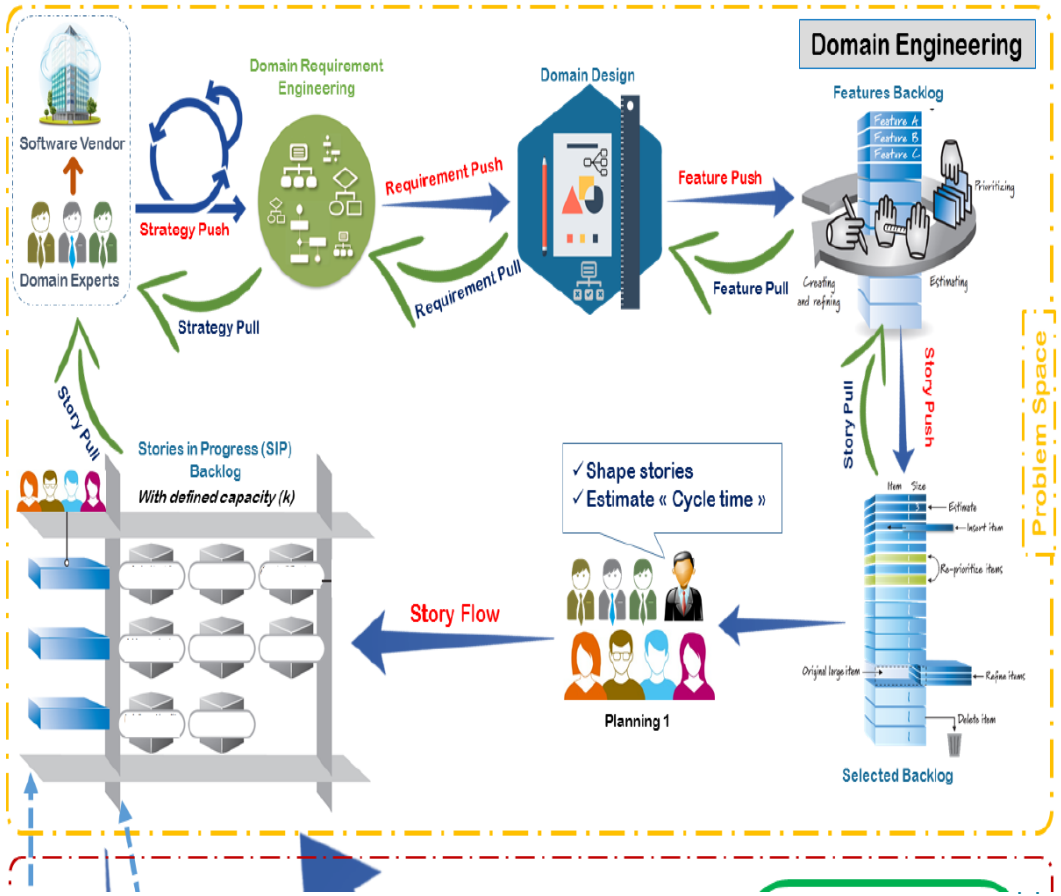
AgiFPL implement is an iterative process that uses Scrumban. GORE has been used for requirements engineering, business process reengineering, organizational impact analysis, and software process modeling.

** "Agile Product Line Engineering: The AgiFPL Method" Hassan Haidar1, Manuel Kolp1 and Yves Wautelet2, 1LouRIM-CEMIS, Université Catholique de Louvain, Belgium, 2KULeuven, Faculty of Economics and Business, Belgium*

The AgiFPL FRAMEWORK* Domain Engineering DE tier

The domain solution constitutes of *"Domain requirement engineering" (DRE).*

1. Start at *"Software Vendor"*
2. *They develop business strategy*
3. *Which drive the DRE*
4. *The "Domain Design" (DD) is derived*
5. *FM & "Features Backlog" (FB) defined*
6. *Document F. into "User Stories" (US)*
7. Define *"Selected Backlog" (SB)*
8. *Planning-1 (What) is select* next
9. "Development Team" structures user stories *"Stories In Progress SIP Backlog"*
10. *Planning-2 starts (How)*



The AgiFPL FRAMEWORK* Domain Engineering DE tier

10- "Planning 2" meeting, the team establishes the "Production Flow"

11- Daily Scrum: Done, Planned, Problems

12- Team" hold a *"Review" meeting* to review the work accomplished.

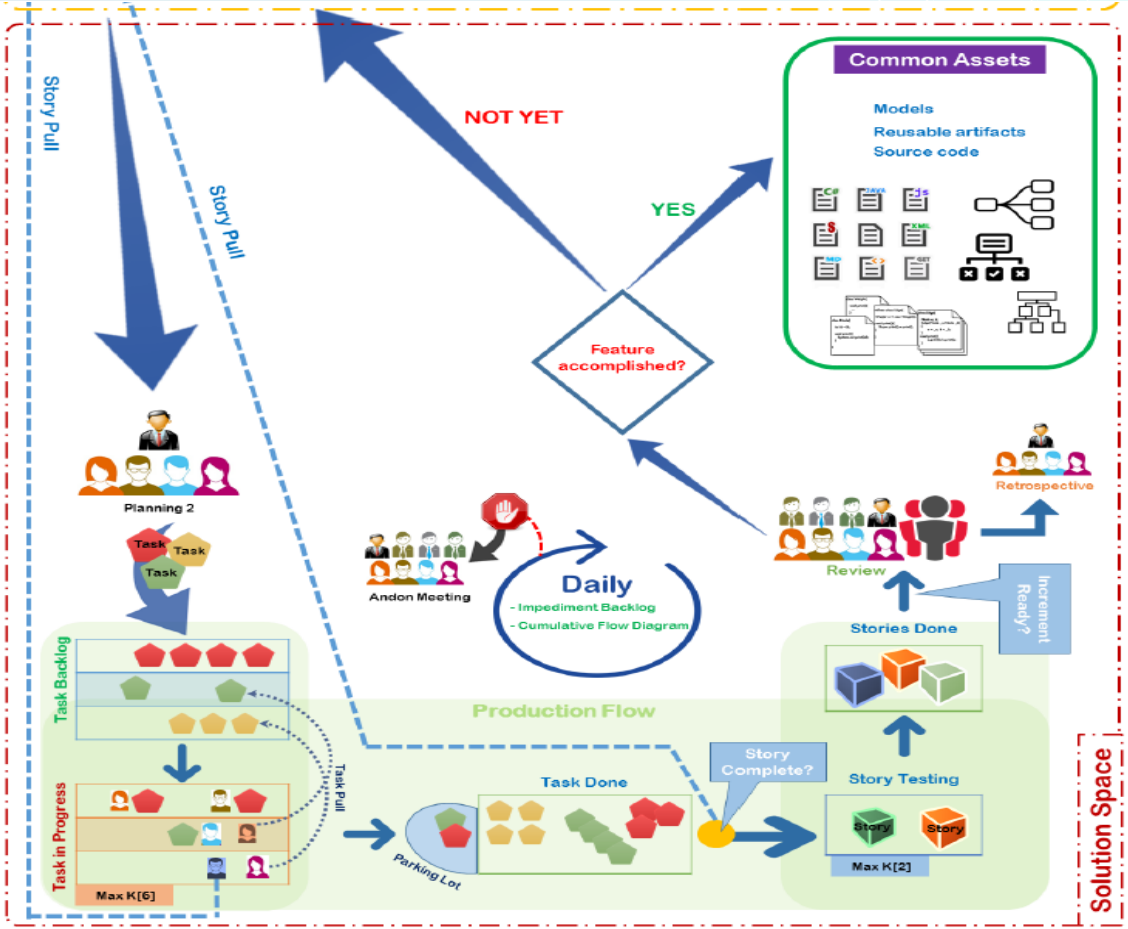
13- Pass: *"Common Assets Warehouse".*

14- Hold the *"Retrospective" meeting*

15- Incomplete Features pushed to SIP

16- SIP story can be pulled if it needs only further testing.

A feature can consist of multiple USs



The AgiFPL FRAMEWORK
Application Engineering AE tier

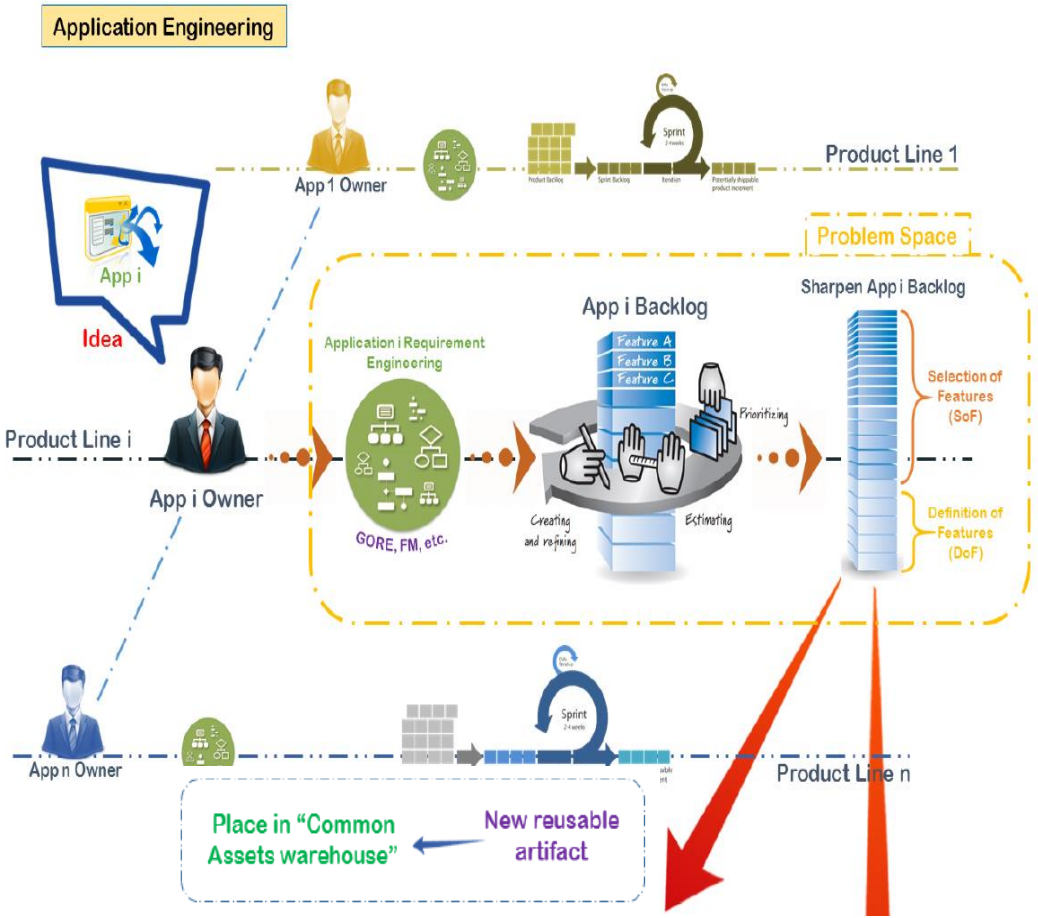
It includes several product lines where the outputs are client applications (products).

Starts at the *"App Owner"*. the *"App i Requirement Engineering" (ARE i) phase*

Two types of features coexist:

1. Features exist in "Common Assets Warehouse". called *"Selection of Features" (SoF)*.
2. Features that are to be developed called *"Definition of Features" (DoF)*.

The DoF move to AE Solution domain for development.

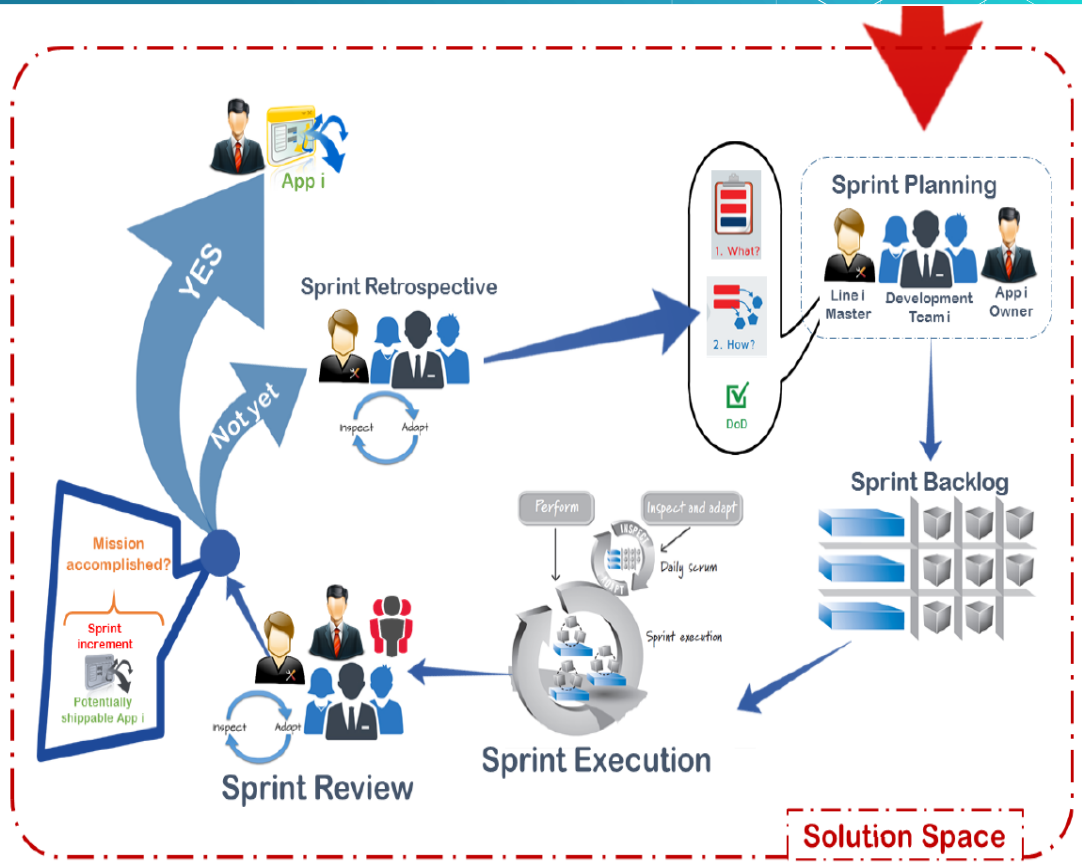


The AgiFPL FRAMEWORK Application Engineering AE tier

The "Sprint" starts with the *"Sprint Planning"*. The team start "Sprint Planning" most important subset of "App backlog" items selected to the next sprint.

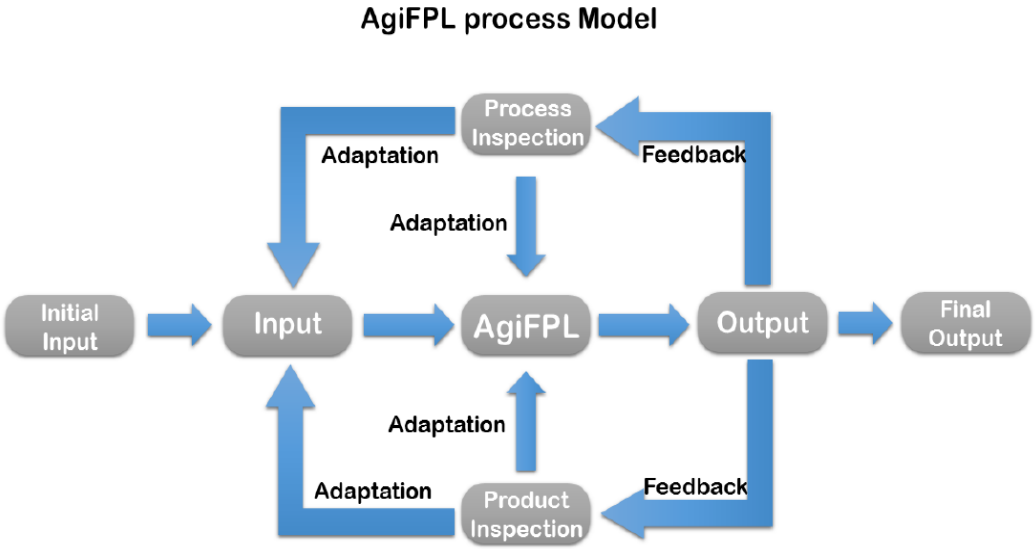
At end of the "Sprint Planning", the "Sprint Backlog" is defined and the *"Definition of Done" (DoD)* list is established.

In fact, DoD is a checklist of activities required to declare the implementation of a story to be completed.



The AgiFPL FRAMEWORK
AgiFPL processes

Each process of AgiFPL is based on an iterative and incremental development as shown



CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Agile Product-Line Architecting (APLA)

Towards an Agile compliant architecture

Agile Product-Line Architecting (APLA)

APLA is the integration of **a set of models for describing, documenting, and tracing Product Line Architectures PLAs***, as well as an algorithm for guiding the change decision-making process of PLAs.

The APLE development process requires that PL-architectures

- are incrementally and iteratively designed, and
- welcome unplanned changes. The PL-architecture has to evolve in each of the iterations to incrementally include all the features of the product-line.

* Agile product-line architecting in practice: A case study in smart grids
Jessica Díaz ⇑, Jennifer Pérez, Juan Garbajosa, Technical University of Madrid-Universidad Politécnica de Madrid (UPM), E.U.

Agile Product-Line Architecting (APLA)

The objectives of APLA:

1. Provide software architecture with **flexibility and adaptability** when defining software architectures to **facilitate change during the incremental and iterative design** of PLAs as well as their evolution (unanticipated change).

2. **Facilitate architectural concerns, such as dependencies, rationale, constraints, or risks, that may be impacted by change.**

3. Provide guidance in the decision-making process during constructing and evolving PLAs to **facilitate change impact analysis in terms of architectural components and connections that may be impacted by change**.

Agile Product-Line Architecting (APLA)

To achieve these objective, we need first to define some terms:

Plastic Partial Component PPC address agile architecting by defining architectural components in an iterative and incremental way.
The variability of a PPC is specified using **variability points**, which hook fragments of code to the PPC known as **variants, and weavings which specify where and when to extend the PPCs using the variants.**
Weavings are defined outside from PPCs and variants, so that these **PPCs and variants are independent of the weaving or linking context**.
As a result, PPCs reduce dependences and coupling between components and their variants, while enable easy and cheap (un-)weaving of variants.

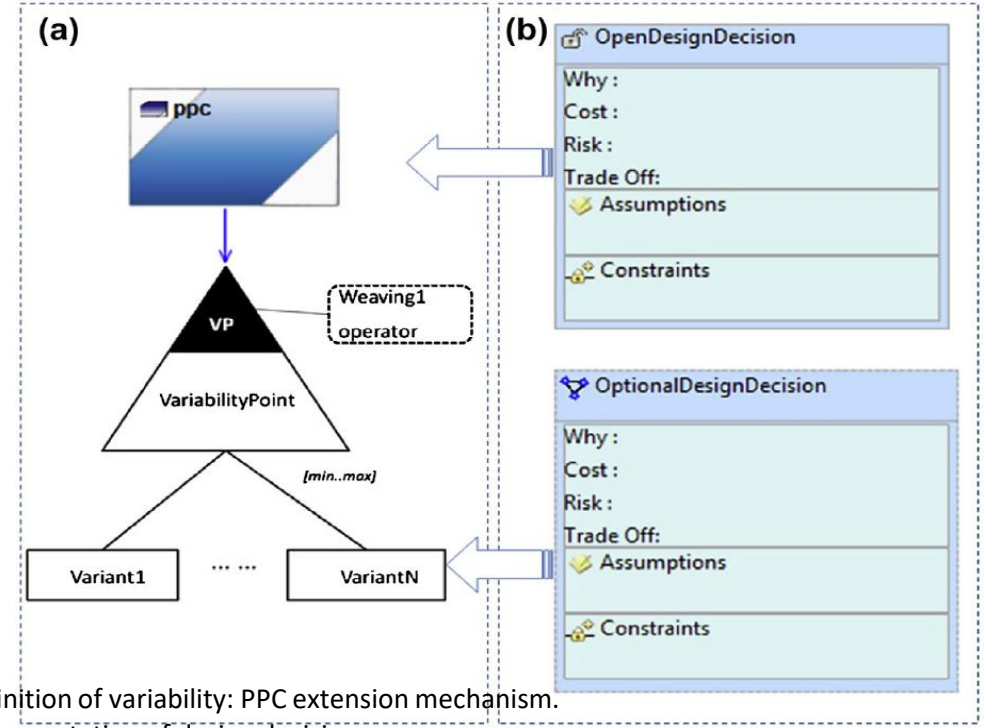
Agile Product-Line Architecting (APLA)

The graphical representation of a PPC that defines a variability point and n variants.

Variability points behave as extension points and variants behave as extensions.

Namely, **the PPC variability mechanism behaves as an extension mechanism to flexibly compose pieces of software**, as if we were building a puzzle.

Since they are unaware of the linking context they can be easily changed, without coupling concerns.



(a) Definition of variability: PPC extension mechanism.
(b) Documentation of design decisions

Agile Product-Line Architecting (APLA)

APLA objective 1: Describing flexible & adaptive architectures

PPC variability mechanism supports incremental development of architectural components through the incomplete specification of components and their extension by hooking new variants.

PPCs get closer and closer to **meeting customers' needs** by specifying the variants only when they are strictly required by a working product.

As a result, **working architectures can be incrementally and iteratively designed and evolved in each iteration by weaving/unweaving extensions,** and/or by modifying the architecture configuration through optional components and connectors.

This continuous architecting may help reduce big upfront architecture design and keep the system in-sync with changing conditions.

Agile Product-Line Architecting (APLA)

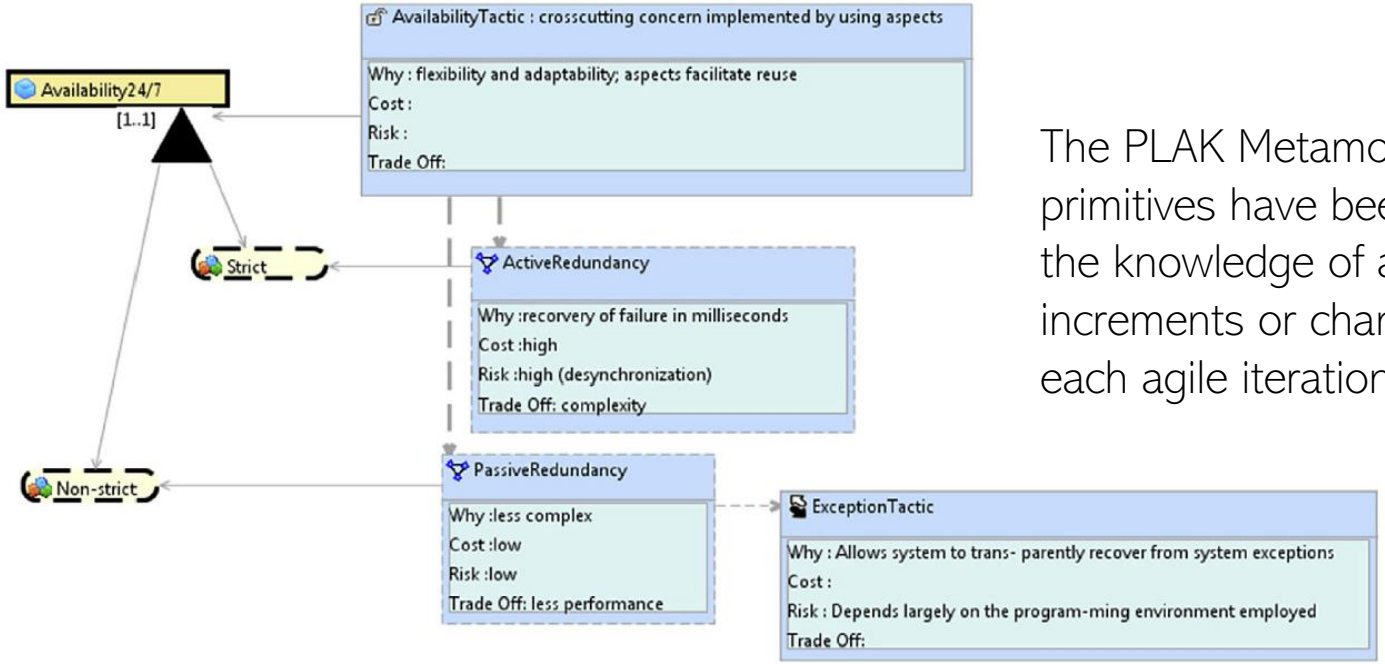
APLA objective 2: Documenting flexible & adaptive architectures

Storing the knowledge created during the architectural design would support the rationalization of architectural decisions taken during solution design.

The main types of architectural knowledge are the design decisions driving the architecture solution, their dependencies, and rationale.

The concept of Product-Line Architectural Knowledge (PLAK) provides both the documentation of design decisions with variations and the capability to trace the history of these variations.

Agile Product-Line Architecting (APLA)



An example of a PLAK model. The figure shows the design decisions for documenting the rationale of different availability tactics.

The PLAK Metamodel and its modeling primitives have been reused to capture the knowledge of adding feature increments or changing features in each agile iteration.

Agile Product-Line Architecting (APLA)

The APLA process

The APLA process has been deployed in Scrum , where APLA advocates the role of the architect in the agile team.

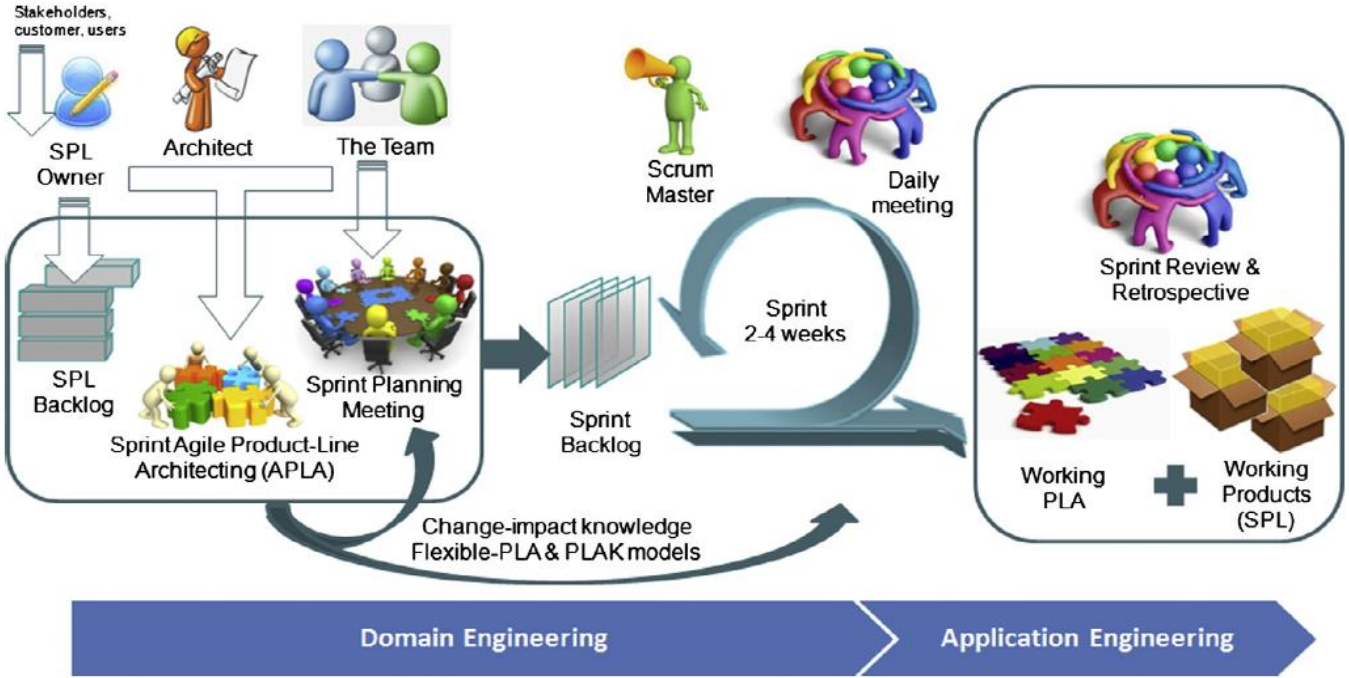
In this way, architects are part of the agile team are **tracking architectural concerns, such as constraints, risks, or viability**, and balancing them with the business priorities during the decision-making process.

The customization of Scrum with APLA divides its sprints into two main phases: Domain Engineering and Application Engineering* (see next figure).

* Agile product-line architecting in practice: A case study in smart grids
Jessica Díaz ⇑, Jennifer Pérez, Juan Garbajosa, Technical University of Madrid-Universidad Politécnica de Madrid (UPM), E.U.

Agile Product-Line Architecting (APLA)

The APLA process



APLA: a customization of Scrum for agile product-line architecting.

The Sprint APLA activity is supported by the Flexible-PLA and PLAK modeling mechanisms as well as the change impact analysis technique which guide architects in the agile product-line architecting process

Agile Product-Line Architecting (APLA)

The APLA process

In summary, the APLA process consists of three main steps:

STEP 1. Architects analyze the Working PLA of the previous sprint by using the change impact analysis algorithm. They can re-prioritize the features based on their impact.

STEP 2. The features assigned to the sprint are incorporated into the Working PLA. Architects use Flexible-PLA modeling primitives to iteratively and incrementally construct and evolve the working PLA through the sprints.

STEP 3. Architects use PLAK modeling primitives to document and trace the design decisions, dependencies, constraints, trade-offs, risks, etc. through the sprints.

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

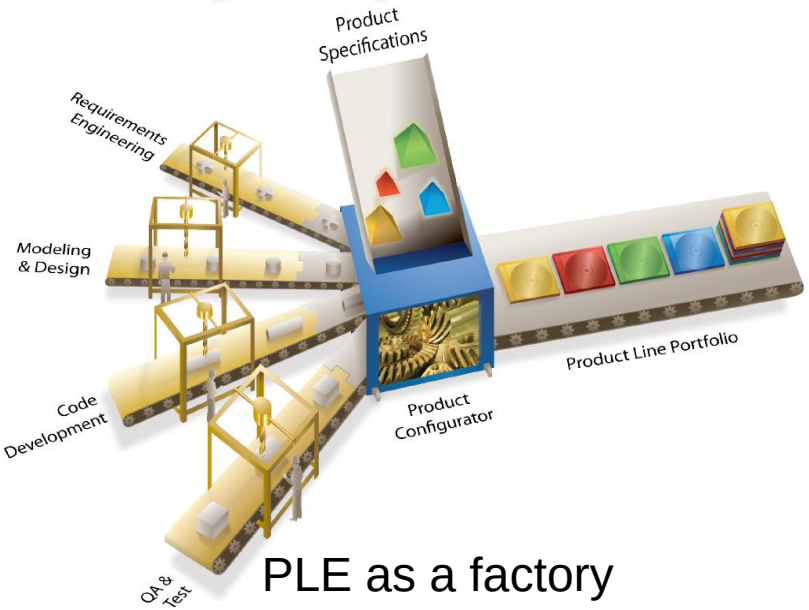
Agile Development and Product Line Engineering at Lockheed Martin*

*"The Best of Both Worlds: Agile Development Meets Product Line Engineering at Lockheed Martin" Susan P. Gregg, Rick Scharadin
Lockheed Martin, 26th Annual INCOSE International Symposium (IS2016)
Edinburgh, July 18-21, 2016

Agile Development and Product Line Engineering at LM

We discuss here the experience of Lockheed Martin as it introduced large-scale agile development practices on one of its largest product line engineering.

The Factory uses shared assets, requirements, design specifications, design models, source code, build files, test plans and test cases, installation guides, project budgets, schedules, and work plans, product calibration and configuration files, data models, parts lists, and more. Assets in PLE are engineered to be shared across the product line.



PLE as a factory

Agile Development and Product Line Engineering at LM

Lockheed Martin calls a member of its product line a configuration. A configuration is an instance of the system to be deployed for a specific platform or site.

The product line approach, then, exists to produce configurations for customers. This product line comprises roughly 100 such configurations.

Each configuration that is deployed to its final operational user community comprises nine major software components, working together to achieve the functionality of the full system. These software components are, in the context of their projects are called products.

Agile Development and Product Line Engineering at LM

For requirements, the product line employs a common specification repository (a DOORS* database) that contains all requirements for all configurations, with varying requirements captured in feature-based variation points.

This model allows for multiple configurations to share requirements while having the flexibility for each configuration to have unique requirements as well.

During the test and verification phase, the product line utilizes a consolidated testing approach to maximize efficiency of common requirements and capabilities. This results in **tailored regression testing based on changed functional area. Common test efforts are leveraged.**

* IBM® Engineering Requirements Management DOORS® (DOORS) is requirements management tool to capture, trace, analyze, and manage changes.

Agile Development and Product Line Engineering at LM

In late 2014, corporate management decided that Agile was the way to strengthen Lockheed Martin's competitive position on its large legacy product line program.

At LM the agile challenge is serious:

- The products –are large by any standard, comprising some 10 million lines of code and costing tens to hundreds of millions of dollars.
- The teams, spread across the products, are sizable. The largest component, for example, involves over 200 engineers. Some 800 engineers are involved in the product line.
- The build cycles, under the product line's are four months long.

* IBM® Engineering Requirements Management DOORS® (DOORS) is requirements management tool to capture, trace, analyze, and manage changes.

Agile Development and Product Line Engineering at LM

How to bridge the gap between large-scale PLE and small-scale short-iteration Agile?
Here there are three answers, each detailed in a sub-section below.

1- Small Teams

Small teams resulted from decomposing the products' **large teams into Scrum teams of size seven to ten**. The teams, considered together, do the same work as they did before, but the work has also been **divided into smaller chunks that match the team size and short iteration (sprint) schedule.**
The teams are **self-organized by product, mostly around areas of domain expertise** or functionality, or around specific elements in the architecture.

* IBM® Engineering Requirements Management DOORS® (DOORS) is requirements management tool to capture, trace, analyze, and manage changes.

Agile Development and Product Line Engineering at LM

The Scrum Team

- Typically 5 - 9 people, no more than 10 (4.6 has been found to be optimal team size)
- Cross-functional – SE, SW, Test, DB, System Admin, UI, etc.
  - Every skill team needs to implement, test, and deliver its user stories
- Members should be full-time
- Team is self-organizing, self-managing
- Team commits to its work and is responsible for it
- Works in an open, collaborative, co-located space
- Demonstrates work results to Product Owner and/or Customers
- Membership should only change between releases (if then)

Scrum teams

Agile Development and Product Line Engineering at LM

However, instead of nine large product teams, the decomposition resulted in 100 or so small ones. To ensuring everyone is working towards common large-scale goals, without devoting all their time to management overhead tasks, becomes a concern of the first order

To address this problem, Lockheed Martin has trained its leaders on the Scaled Agile Framework (SAFe), a “knowledge base for implementing agile practices at enterprise scale.”

It defines the “individual roles, teams, activities, and artifacts necessary to scale agile from the team, to teams of teams, to the enterprise level.”

Agile Development and Product Line Engineering at LM

2- Small iterations

While inter-team coordination is resolved by SAFe, work inside each team follows classic Scrum, meaning that planned work is organized into two-week sprints. A certain number of sprints add up to a release.

How many sprints? Lockheed Martin has adopted a four-month release cycle thus, eight sprints constitute a release.

The work accomplished in each sprint is, again from Scrum, defined in terms of *user stories*. Users stories are used to accomplish *features*. A feature is a sort of large-scale user story that completes within a release.

Agile Development and Product Line Engineering at LM

PL release tempo is 4 months = eight 2-week sprints (may shorten later)

Epic – Spans Multiple Releases, Often Major Capabilities or Types of Work
Large User Story (can have sub-epics)
Feature – Completed within Release, Often tied to Program IMS(s), Often Spans Teams
User Story – Each Team owns their set, Completed within Sprint
Tasks – Generally at the Team Level, 2 to 8 worst case 16 hours

Decomposition

EPIC								EPIC							
Feature				Feature				Feature				Feature			
User Story		User Story		User Story		User Story		User Story		User Story		User Story		User Story	
Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task	Task

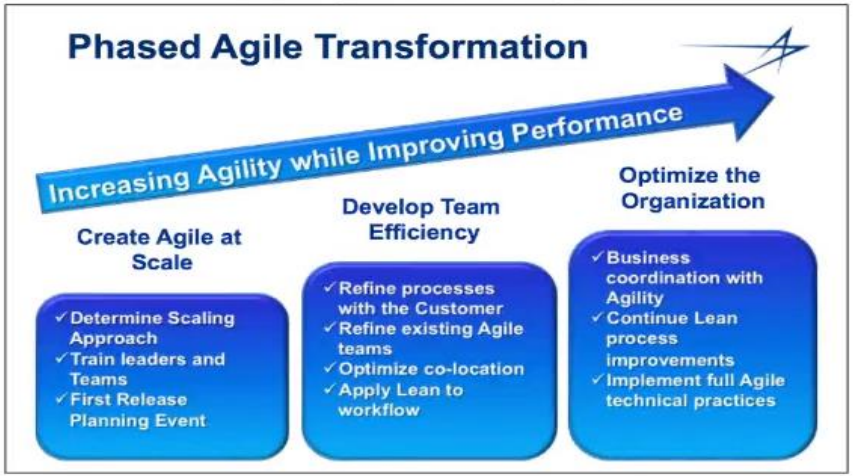
Following Scrum methodology, each sprint includes daily scrum meetings and culminates in a sprint demo/review and a sprint retrospective.

Agile Development and Product Line Engineering at LM

Sprint events

Daily scrum meetings	Sprint demo/review	Sprint retrospective
• Display Visible Progress Indicators for team to review • Scrum Guidelines • Daily stand-up • 15 minutes • Not for problem solving • Three questions: What did you do yesterday? What will you do today? Is there anything in your way? • Schedule follow-on meetings to discuss details. • Whole world is invited • Only team members, Scrum Master and Product Owner can talk • Scrum Master writes down and tracks issues and requests for additional conversations	• Team "shows off" working product resulting from current Sprint (what is "done") • Typically takes the form	• Short meeting held at the end of each Sprint • Celebrate the completion of the Sprint • Take a look at what is and

Three-year transition plan



General Motors PLE's automated support *

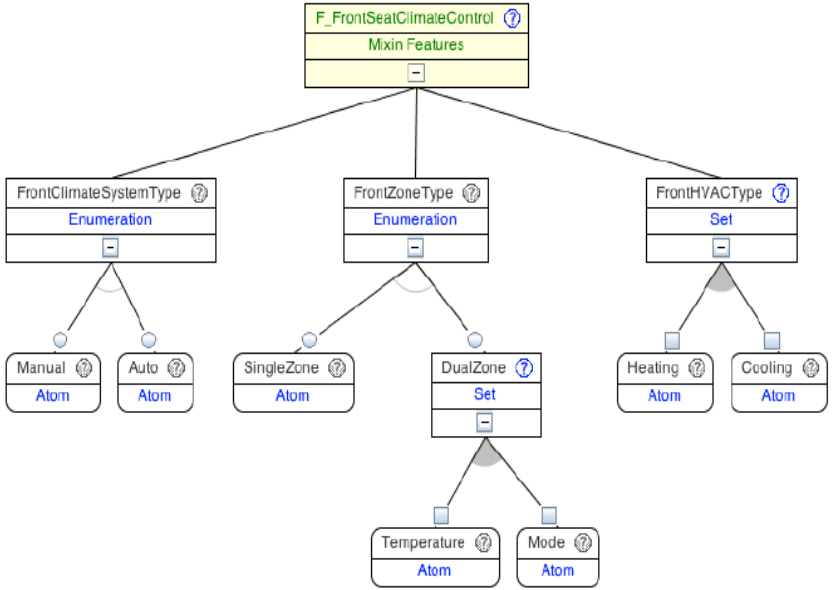
*Mega-Scale Product Line Engineering at General Motors
2012 Software Product Line Conference
Best Paper Award in Industrial Track

PLEs Industry case

General Motors PLE's automated support

GM deals with mega-scale product lines and **keeping track of the variation in each system** and the feature interactions among systems leads to too many possible systems variations.

GM needed a tool to **manage variation points** in their engineering artifacts and **help configure vehicle-specific engineering products**. For this, **they chose Gears from BigLever Software**.



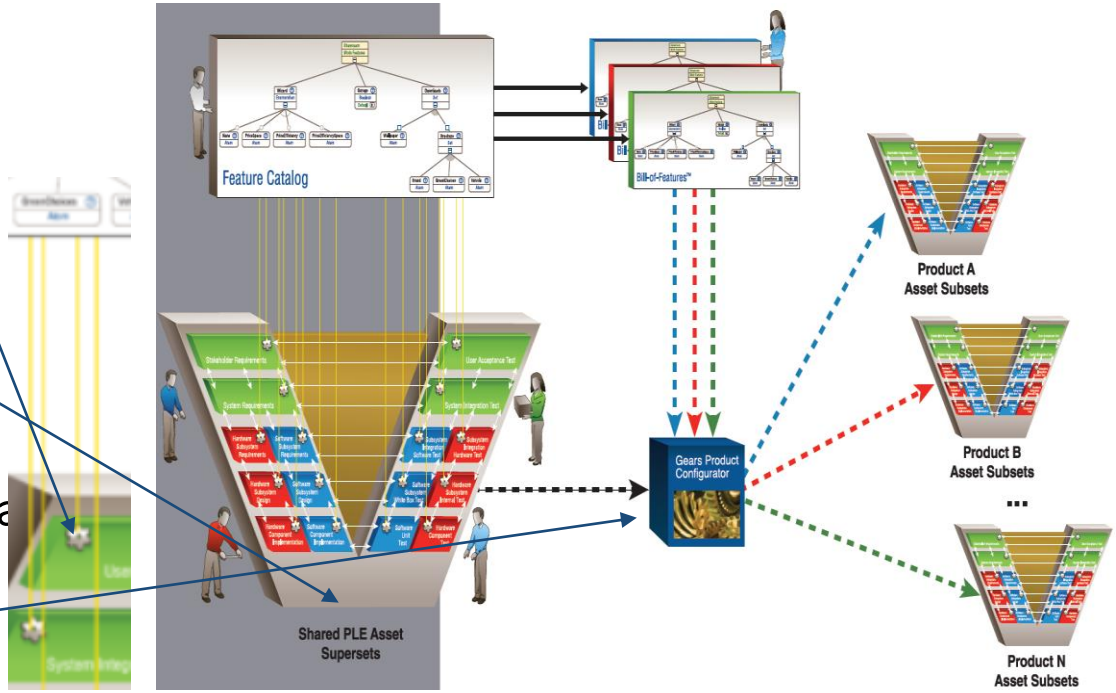
A feature model for the front seat climate control Feature

PLEs Industry case

General Motors PLE's automated support

A "**superset**" supply chain of reusable digital assets is used, with **variation points** (illustrated by the small gear symbols) defined in terms of features in the product line's Feature Catalog. The features chosen for each product are specified in the **Bill-of-Features** for that product.

The Gears product configurator creates a product instance by exercising variation points according to the features selected.



PLEfactory, based on Gears

PLEs Industry case

General Motors PLE's automated support

The bill-of materials for a vehicle's electronics is generated from its bill-of features.

To capture features, here is the set of feature modeling constructs (provided by Gears) that GM is using for its product line work:

Feature declarations are parameters that express the diversity in the product line for a system or subsystem.

Feature profiles are used to select and assign values to the feature declaration parameters for the purpose of instantiating a product.

Assets are the abstraction for systems and software artifacts in a production line.

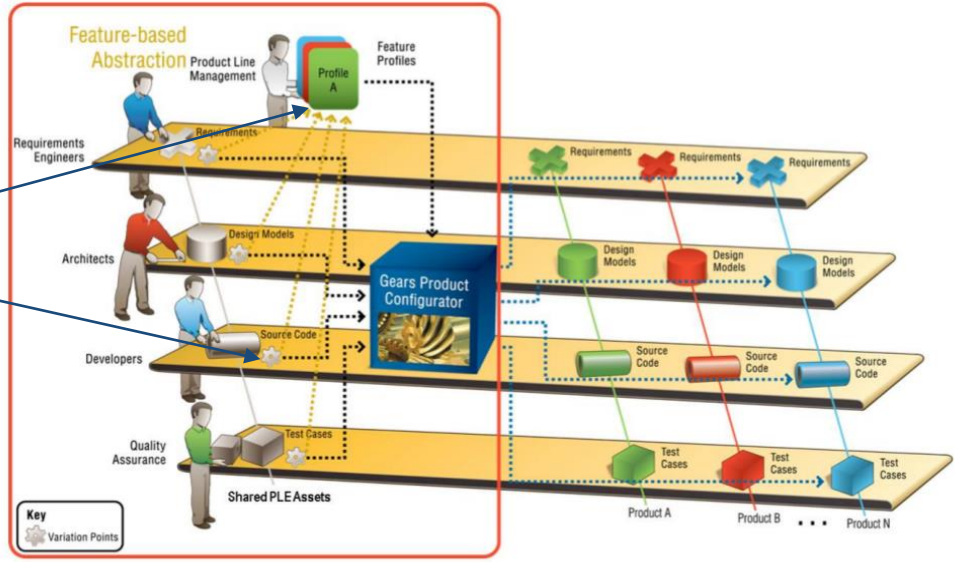
Variation points encapsulate the variations in the assets used to build products.

PLEs Industry case

General Motors PLE's automated support

Assets are built and maintained on the left; each is endowed with one or more variation points (indicated by the gear symbol).

Feature profiles determine how the assets are instantiated (by exercising their variation points) to produce product-ready artifacts.



Feature profiles drive the exercising of shared assets' variation points by the configurator to produce product-specific instances.

PLEs Industry case

General Motors PLE's automated support

Assets Traceability

The **list of artifacts at GM** include requirements, system architectures and designs, source code implementation, calibration parameters, test cases, and documentation.

Common representation of variation points is key to achieving traceability from requirements to deployment.

Traceability is of great concern for GM. **Every requirement needs to be traceable to one or more design elements that satisfy that requirements**, and each design element should be traceable back to one or more requirements that it satisfies.

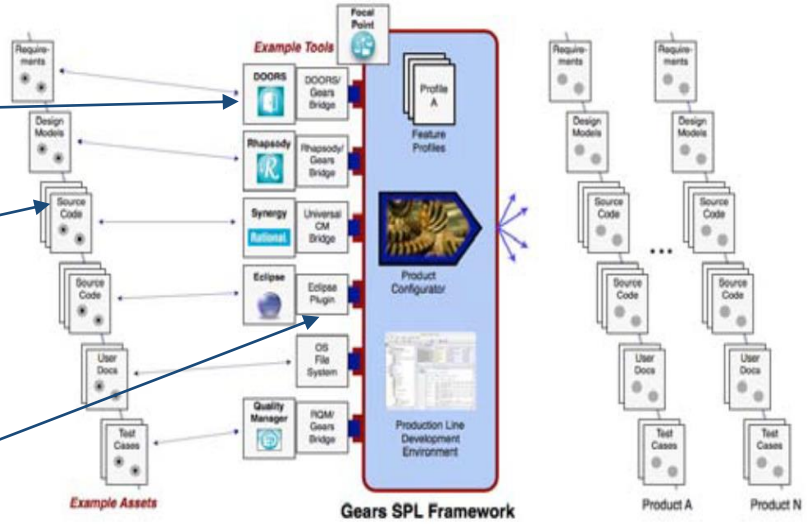
PLEs Industry case

General Motors PLE's automated support

GEARS interface

GEARS platform supports plug-ins, for example, DOORS requirements modules, Microsoft Word documents and Excel spreadsheets, and UML, and many more.

Gears supports various artifacts maintained under the proprietary of various tools. In that figure, a *bridge* is a piece of software that "knows" the other-tool representation and presents a "product-line-aware" user interface for that tool that allows product line engineers to insert variation points in the artifacts maintained by that tool.



Gears and its bridges to other lifecycle tools
(© BigLever Software, Inc.)

PLEs Industry case

General Motors PLE's automated support

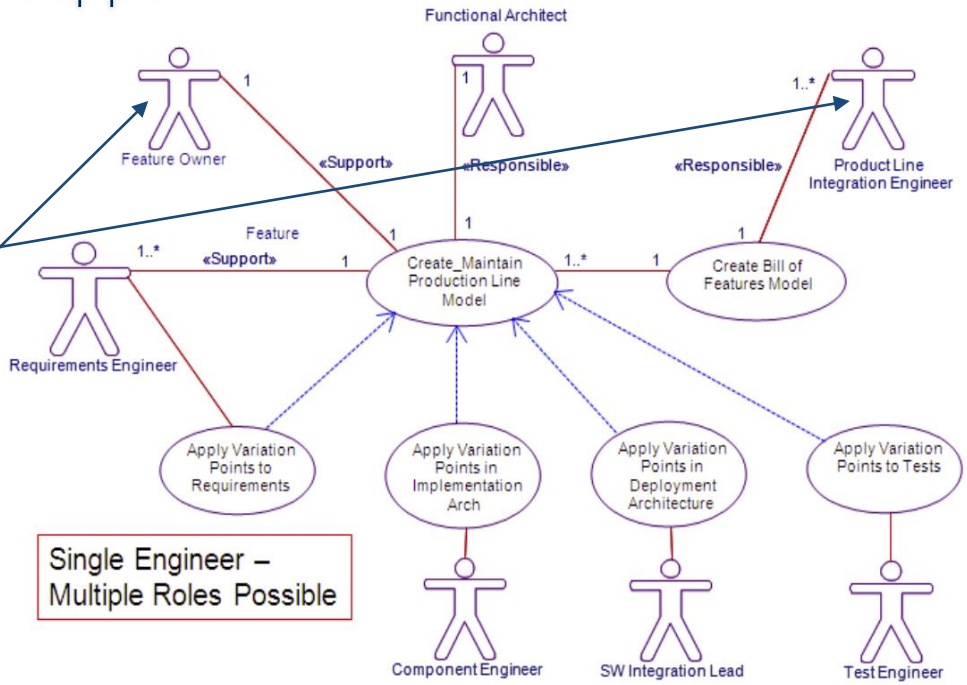
GEARS Roles at GM

The Figure sketches the major PLE roles and their broad responsibilities vis-à-vis maintaining the PLE models and artifacts.

PRODUCT LINE INTEGRATION ENGINEER

This engineer collaborates with Vehicle Product Teams in the selection of a '**bill-of-features**' for a vehicle being planned.

The product line integration engineer also collaborates with the **feature owners** in the identification of the top-level subsystem production line 'products' that will be offered up to vehicles.



PLEs Industry case

General Motors PLE's automated support

In conclusion

- A focus on features for variation and product selection; the "bill-of-features" replaces the "bill-of materials" as the key engineering artifact for product derivation.
- An emphasis on high-quality automation at the center of a production line, to quickly turn a bill-of-features into a set of instantiated lifecycle assets
- A CM and PLE approach geared to multi-baseline multiproduct management in a way to reduce the order of complexity from $O(n2)$ to $O(n)$.
- Taking multi-organizational management in stride, by providing feature model concepts such as mix-ins and imported (hierarchical) production lines, to reflect the structure of engineering activities and domain knowledge present in an ultra-large organization.

In Conclusion of SPLs Agile development & Industry automation

We have presented the following

Resolve PLA requirement conflict by deploying short CD per Agile requirement

Use AgiFPL framework to implement an iterative process that uses Agile Scrumban process

Used PPC variability mechanism to support APLA (incremental development of PLA)

How large organization can use scaled-up Agility for CD Agile transformation

How LM was able to transform to APLE with large number of small teams and short iteration

How GM is using PLE GEAR automation to manage their assets superset variation points