Application of Feature-based Product Line Engineering in Information-Centric Domains

Jim Teaff
Senior Principal Engineer
Raytheon Intelligence, Information and Services
James.K.Teaff@Raytheon.com

Dr. Paul Clements
VP of Customer Success
BigLever Software
clements@biglever.com

Abstract. Feature-based systems and software product line engineering and management (FBPLE) is a way to engineer a portfolio of related products in an efficient manner, taking full advantage of the products' similarities while respecting and managing their differences[1]. While modern FBPLE can trace its methodological roots to the software industry, FBPLE is equally applicable to all categories of goods and services. This paper will describe how tailored information products can be "manufactured" using a product line "factory", and the benefits derived from this approach using a hypothetical multi-modal transportation system as an example.

Introduction

The information age has extended the concept of a product beyond the traditional hard goods and services. Digital content delivered on demand and tailored for individual consumers is now a key part of an organization's product portfolio, either provided solely or – more typically - bundled with other goods and services. And, the current 4th Industrial Revolution is driving the creation of cyber-physical systems, which enable "augmented intelligence" – seamless human and machine interactions leveraging a fusion of "real world" and "virtual world" elements – for which information products are key elements.

Today, data alone is mostly a commodity, available to consumers without charge. Data, when aggregated and enriched into reusable elements of information products, augments an organization's portfolio of goods and services such that the organization can deliver holistic solutions – combinations of "hard" goods, "soft" goods, and services. As an example, transportation systems rely heavily on information, from the earliest travellers using paper charts and maps, to today's multi-modal transportation systems. These systems rely on integrated information streams, including navigation, weather, traffic congestion, construction, and more which consumers are increasingly expecting to be tailored specifically for them, delivered when they need it, and in the form they need it. Given the commodity nature of raw data, corresponding information products must be produced at a cost that maintains the profitability of the overall product portfolio. This paper proposes that this goal is most easily achieved when tailored information products are automatically "manufactured" using a product line engineering (PLE) "factory".

Raytheon's Intelligence, Information and Services (IIS) organization has a long history with product line engineering, from a product line asset base for ground-based spacecraft command and control systems [6], to a family of transportation solutions that includes aviation weather; aircraft precision navigation and landing; air traffic management; vehicle intelligent transportation systems; and more. Over the years IIS has continued to incorporate internal lessons learned, as well as industry best practices and tools, transitioning from 1st generation PLE to modern or feature-based systems and software product line engineering and management (FBPLE).

We use as an example application of FBPLE a hypothetical multi-modal transportation system as depicted in Figure 1. The system enables point-to-point transportation using vehicles (including autonomous vehicles), aircraft, ships, and rail. Information products in the example portfolio include navigation data, weather data, food and lodging data, airport and rail terminal data, marina data, and more.

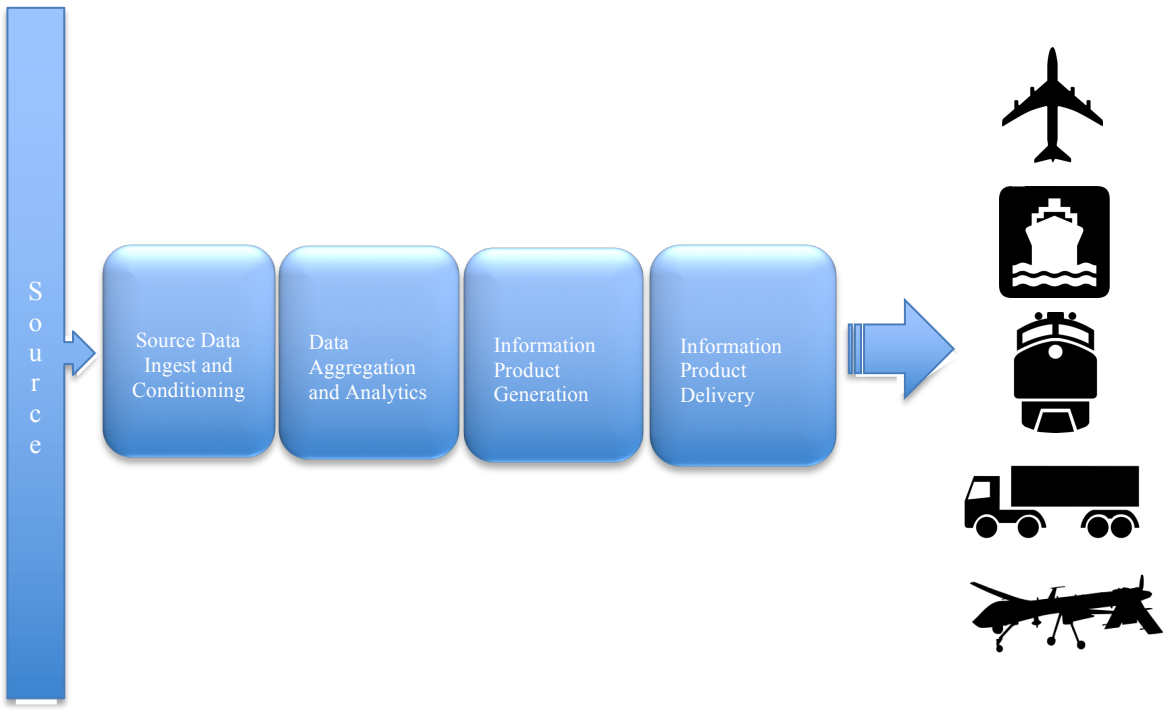

Figure 1 Multi-modal Transportation System

What Is Product Line Engineering?

Product Line Engineering (PLE) is a way to engineer a portfolio of related products in an efficient manner, taking full advantage of the products' similarities while respecting and managing their differences [1]. By "engineer," we mean all of the activities involved in planning, producing, delivering, and deploying, sustaining, and retiring products. Modern PLE has incorporated lessons learned over the past decades, and resulted in an advanced set of explicitly defined product line engineering and management solutions, which have been referred to as Second Generation PLE [1][2][5]. Recently, a distinct set of techniques and tools have emerged to form what the community has termed "feature-based systems and software product line engineering and management", or FBPLE. FBPLE is based on the concept of *features* that describe products and relies on automated product variation creators (*PLE configurators*) to instantiate shared assets (*digital engineering artifacts*) across all aspects of the system development lifecycle in order to support specific products [1].

FBPLE stands in contrast to classical product-centric development, in which each individual product is developed and evolved independently from other products, or (at best) starts out as a cloned copy of a similar product that is then changed to suit the new product's specific needs. Product centric development takes very little advantage of the commonalities among products in a portfolio after the initial clone operation. In particular, it derives very little benefit from commonality in a product's sustainment or maintenance phase, where data show that most products consume up to 90% of their project resources [1].

Product features are a key concept of FBPLE, and provide a common language – a *lingua franca* - among all stakeholders in the product line, from requirements engineers to testers, from marketers to executives, from designers to customers. Product features primarily express customer-visible or end-user-visible behaviours among the products in a product family. Product features can also express implementation capabilities not directly visible to a customer or end user except through non-functional differences such as performance, usability, colours, materials, price, etc. In product line engineering (PLE), the methodology literature uses the unqualified term of a "feature" to describe differences or *variations* among products, and we will adhere to that definition. The concept of feature allows a consistent abstraction to be employed when making choices from a whole product configuration all the way down to the deployment of software components within a low-level subsystem in the architecture.

A product variation specification – a bill-of-features analogous to a bill-of-materials for hard goods - captures the set of feature variation choices you make from the variable features that are available in the feature catalogue for a specific tailoring of a product, analogous to choosing an automobile's exterior color, seat material, navigation and entertainment bundles, etc.

FBPLE as a Factory

An analogy with factory-based manufacturing serves to illuminate the important concepts (see Figure 2). Manufacturers have long used engineering techniques to create a product line of similar products using a common factory that assembles and configures parts to produce the varying products in the product line. For example, automotive manufacturers can create thousands of unique variations of one car model using a single pool of parts carefully designed to be configurable and factories specifically designed to configure and assemble those parts.

In FBPLE, the configurator is the factory's automation component; the "parts" are the assets in the factory's supply chain. A statement of the properties desired in the end product tells the configurator how to configure the assets.

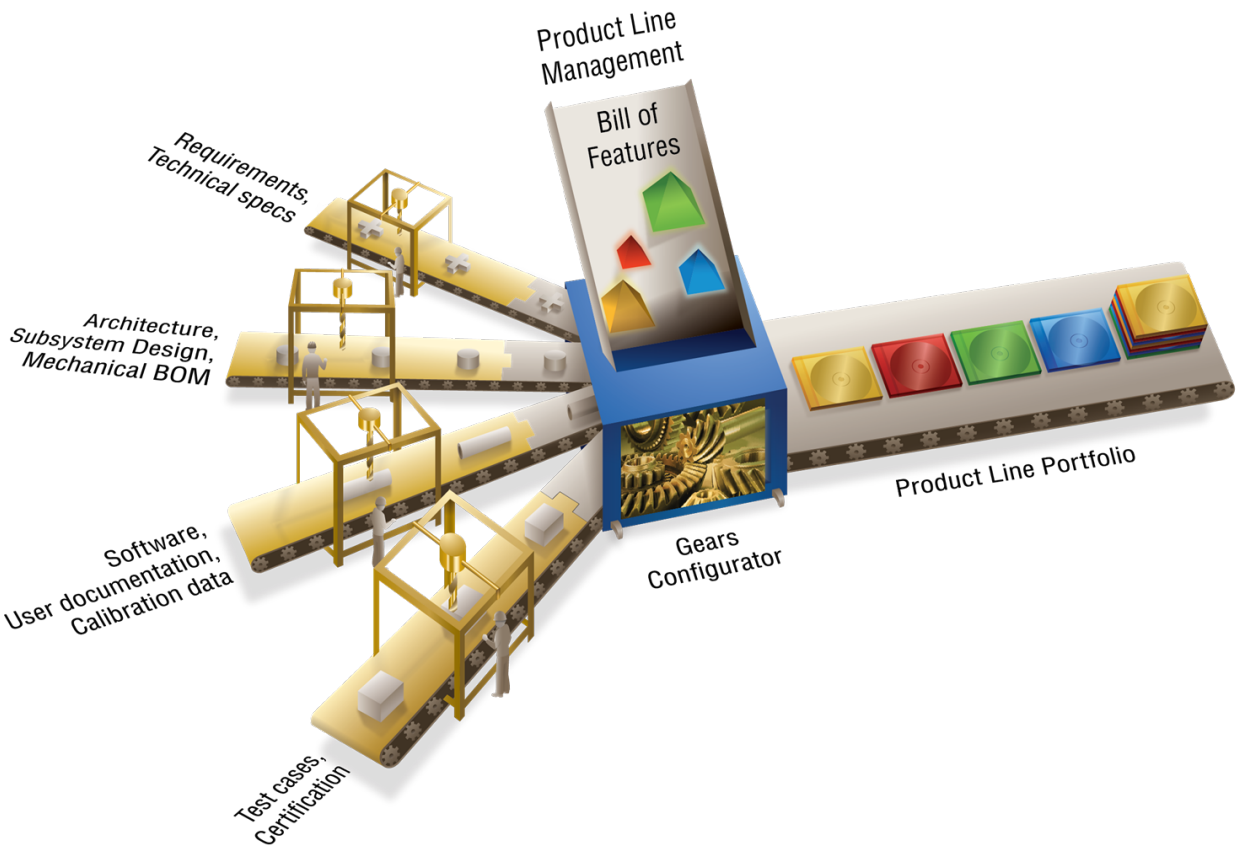

Figure 2 - FBPLE Factory © BigLever Software

The factory's supply chain is at the left, in the form of shared assets that are configurable because they include variation points that are expressed in terms of the features available in each of the products. A product specification at the top tells the configurator how to configure the assets coming in from the left. The resulting products, assembled from the configured assets, emerge on the right. This enables the rapid production of any variant of any of the assets for any of the products in the portfolio. Once this production line capability is established, products are instantiated – derived from the shared assets – rather than manually created.

The products in the portfolio are described by the properties they have in common with each other and the variations that set them apart. The products can comprise any combination of software, systems in which software runs, or non-software systems that have software-representable artifacts (such as requirements, engineering models, or development plans) associated with the engineering process that produces them.

Key Information Product Consumer and Organizational Needs

As mentioned, today's multi-modal transportation systems use data fused from multiple sources including navigation, weather, traffic congestion, road construction, airport terminal maintenance, and more. Moreover, consumers expect the information to be tailored specifically for them, and delivered when and how they need it. Additionally, sovereign states require some or all of their data remain physically resident on their soil, or managed solely on their computing infrastructure, due to its sensitive nature. Although our hypothetical product portfolio roadmap targets cyber-physical systems as its penultimate, we will keep our illustrative example simple by using only two types of data – navigation data, and weather data.

The key information product consumer needs we apply for our illustrative example include:

- Avoid information overload – provide only the information needed for that specific consumer's needs. For example, commercial airline pilots do not need detailed information – approach procedures and the like – for airports that do not have runways long enough to land their aircraft. Excluding that information means the pilot no longer needs to filter through it on the flight deck (critical in emergency situations).
- Accommodate edge devices such as handheld devices; embedded avionics; et al. These devices periodically work disconnected from the network, and frequently have limited resources (e.g. storage).
- Provide the ability to manage data and deliver information products when the data physically remains resident within a sovereign state.
    - From this stakeholder need we derive requirements to deploy an instance of the enterprise system to a customer. In order to protect intellectual property and comply with the various import and export regulations we derive requirements to deploy tailored versions of the software.
- Deliver information products on demand in specific time windows e.g. while an aircraft is at an airport gate, or when a maritime ship has wireless network connectivity, etc.
    - From this stakeholder need we derive requirements to deploy instances of our Information Product Generation and Delivery sub-systems optimized for specific customers e.g. tailored software deployed to specific AWS regions and Availability Zones.

The key organizational needs we apply for this paper's example include:

- Keep production and delivery costs low such that the overall product portfolio of goods and services remains profitable.
- Enable increasingly feature-rich information products through easy incorporation of new categories and formats of data.
- Augment the organization's ability to effectively and efficiently pilot and market new portfolio goods and services leveraging information products.

Multi-Modal Transportation System Feature Catalogue

Our illustrative Feature Catalogue includes the following data subsets and variations:

- Vehicle navigation data. Varies by vehicle category, including military, private, and autonomous.
- Aviation navigation data. Varies by aircraft category, including military, civil (commercial, business, general aviation), as well as manned, and unmanned.
- Ship navigation data. Varies by ship category, including seagoing and freshwater; military and civil (commercial, private).
- Weather data. Varies by category and intended use, including current and forecast data; and weather specific to land transportation, aviation, or marine operations.

Additionally, our Feature Catalogue includes variations for product delivery formats and mechanisms. For example, aviation navigation data can be delivered in legacy ARINC 424 Navigation System Data Base Standard format; or the more modern Aeronautical Information Exchange Model (AIXM) format.

The Product Line Model

The BigLever Gears model of our product line captures the Feature Catalogue and the superset of reusable assets used by our FBPLE Factory to automatically generate tailored software and information products. The assets in our product line include reusable "data generators" that emit information product subsets; and "data formatters" which transform data subsets from a canonical internal format to the requisite output format. The tailored data generators and data formatters are in turn physically deployable to a multitude of different physical and virtual environments e.g. as cloud native services hosted within a specific AWS region and Availability Zone; or as database stored procedures that can efficiently respond to on-demand pulls of data.

Our Feature Catalogue is modularized into a set of model "mixins" encapsulating the data generators, data formatters, and named customers (for driving tailoring) as depicted in the figure below.

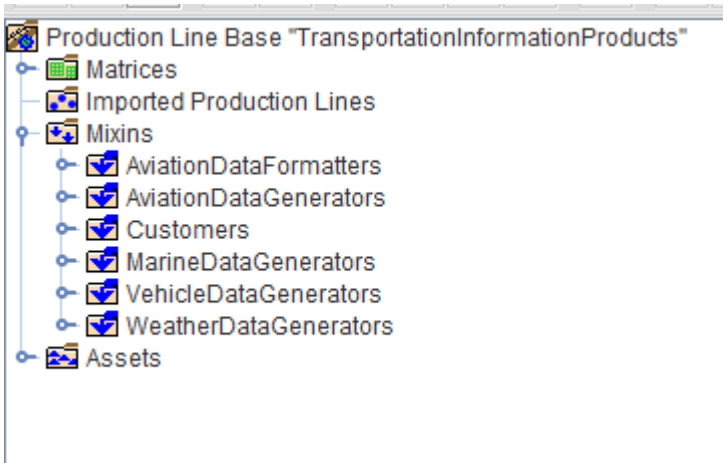

Figure 3 - Multi-modal Transportation System Feature Catalogue

For aviators, navigation data generators and formatters are included in the production line for navigation data and for weather data. The Aviation Navigation Data Generators mixin encapsulates the variations by aircraft type as depicted in the figure below. Each data generator feature is simply modelled as "selectable" or "not selectable" for subsequent use creating the Bill-of-Features Portfolio, which in turn will be used by the product line's Configurator to auto-generate the information products.

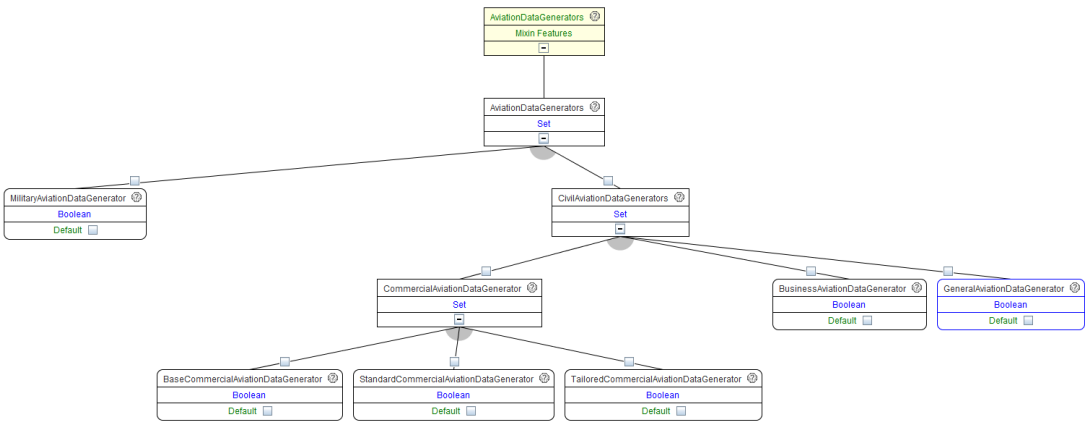

Figure 4 - Aviation Navigation Data Generators

The Aviation Data Formatters Mixin encapsulates ARINC 424 and AIXM navigation data formatters as depicted in the figure below. Other data formatters can be easily added, extending the product line's capabilities, such as data formats mandated by a consuming contract.

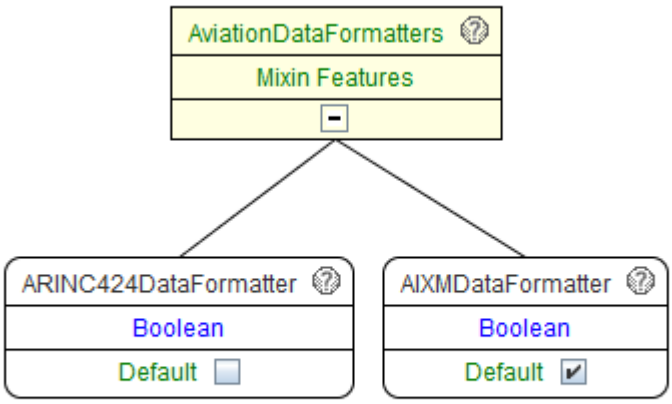

Figure 5 - Aviation Navigation Data Formatters

The Weather Mixin encapsulates the different types of weather data available for inclusion in our information product, from the current weather to weather forecast data, including specializations for aviators and mariners as depicted in the figure below.

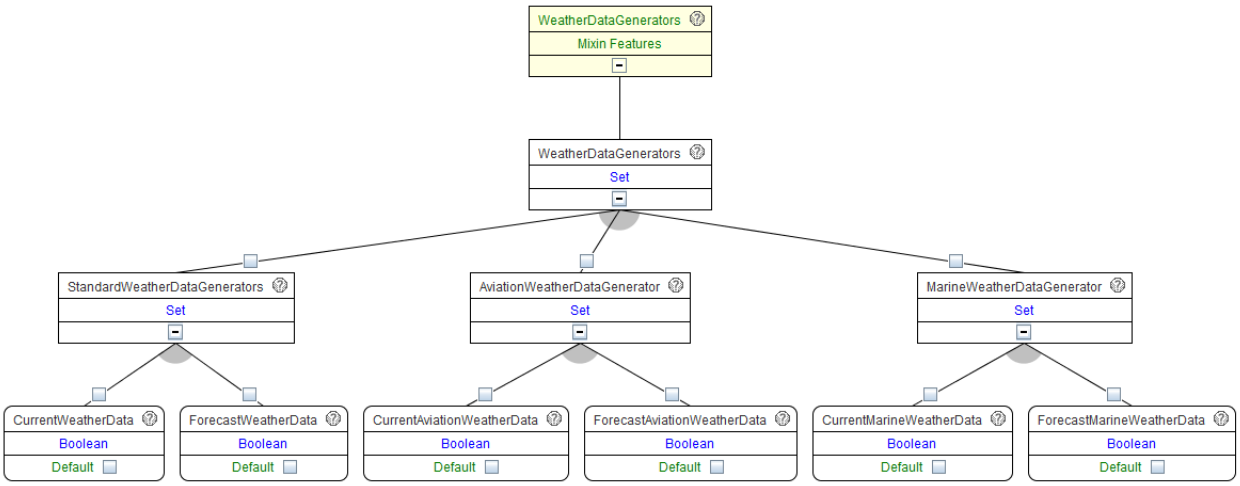

Figure 6 - Weather Data Generators

The Assets folder in our product line encapsulates the reusable system elements that are associated with our features. In our simple illustrative example the assets are sparse Java code modules, but in an "industrial strength" model the assets would be realized and deployed as cloud native micro-services and/or database stored procedures, et al.

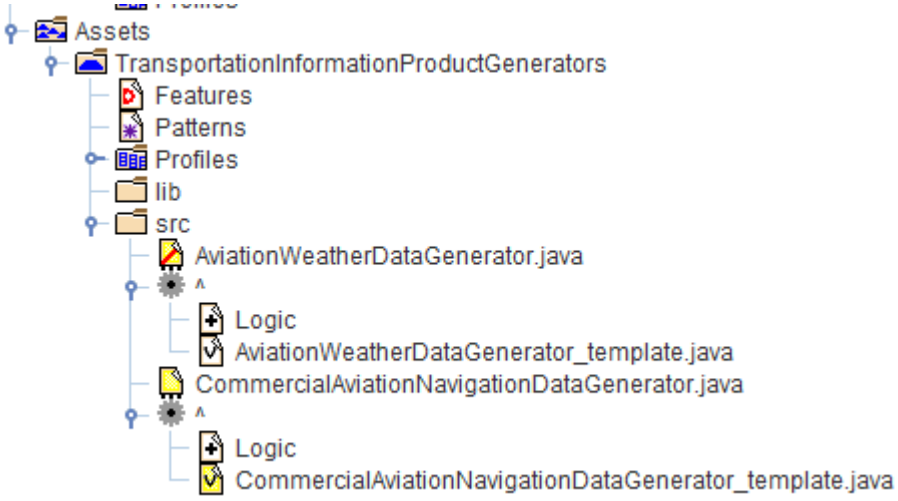

Figure 7 - Reusable Product Line Assets

BigLever Gears includes a rich "variation language" for specifying the variation point logic associated with each of the digital assets. The variation point's logic uses the variation language to specify the feature's variations to the product line configurator. We demonstrate two: data generator inclusion / exclusion which results in entire data subsets being included or excluded from the emitted information product; and Java code block deletion which results in specific customizations being included or excluded from a data subset.

In our illustrative example we include or exclude entire subsets of data for our information product in the bill-of-features captured in our mixin's "profiles". As depicted in the figure below, the "Commercial Airline Alpha" profile specifies inclusion of BaseCommercialAviationData and TailoredCommercialAviationData while excluding all other categories of aviation data (e.g. military, business, and general aviation). The emitted Java code would then be deployed to the AWS region and Availability Zone appropriate for the thousands of Commercial Airline Alpha flight planners and pilots to pull on demand their customized information products.

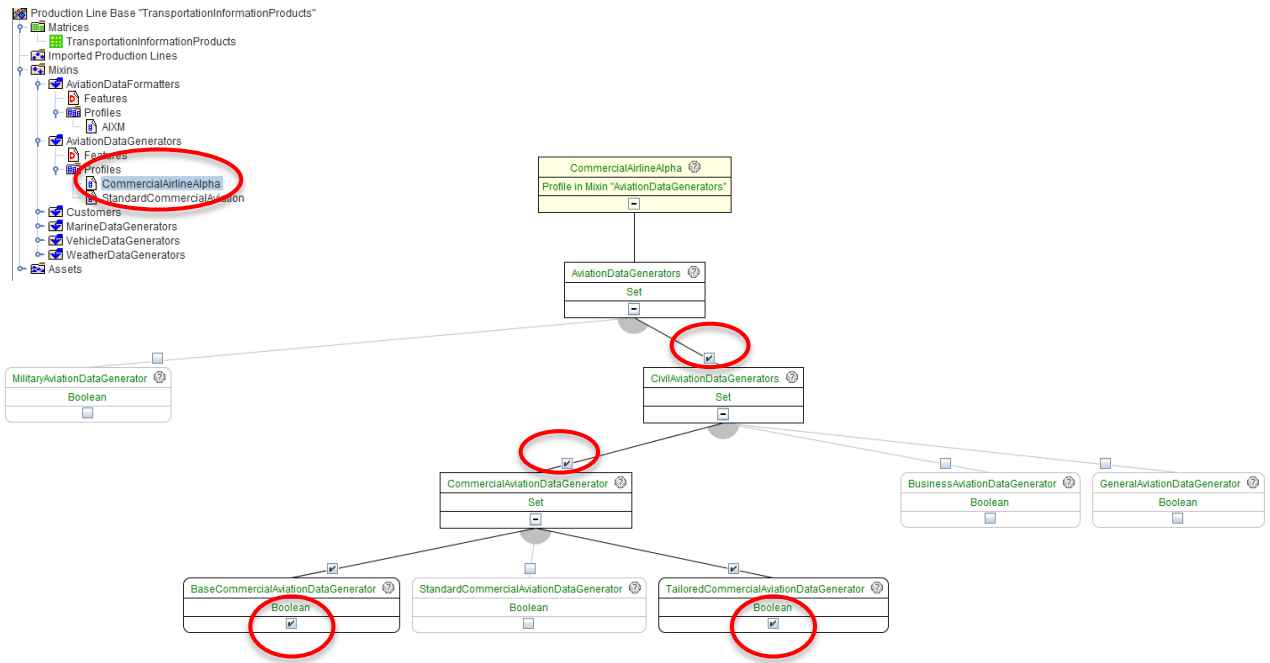

Figure 8 - Including / Excluding Data Subsets

Variation mechanisms are available which manage variations within individual digital assets. These include string substitution; text block inclusion / exclusion; and more. For our example we will include or exclude a block of Java code based on whether a "standard" or "tailored" data subset is required. The figure below depicts defining the variation logic for the selection of Java code blocks based on the Commercial Airline Alpha Bill-of-Features.

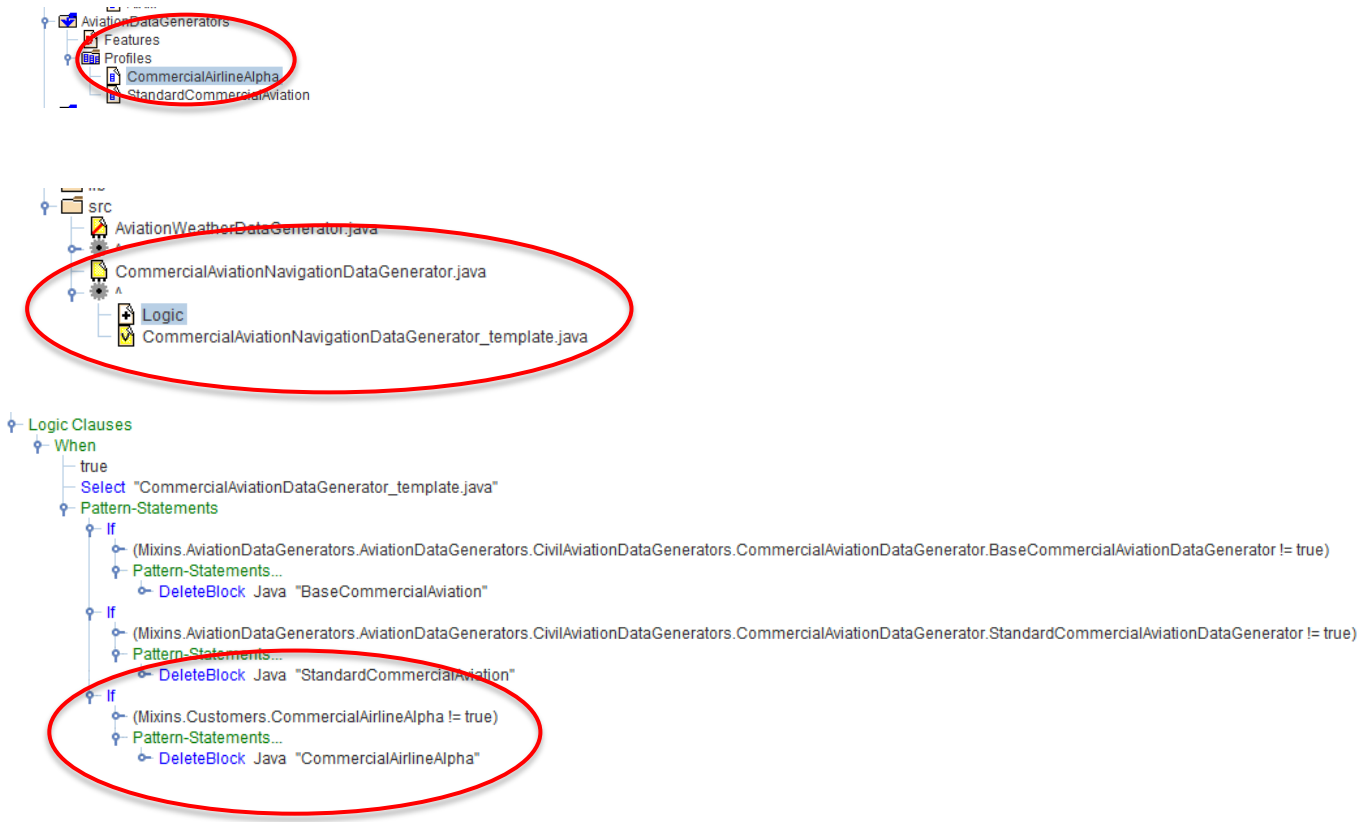

Figure 9 – Variation Logic for Selecting / Deselecting Data Generator Code Blocks

The figure below depicts the Java code instrumented with the code block include / exclude tags: "BeginGearsBlock Java CommercialAirlineAlpha" and "EndGearsBlock Java CommercialAirlineAlpha".

```
try {

    FileWriter fw = new FileWriter(FILE_URL);

    ///* BeginGearsBlock Java BaseCommercialAviation
    generateBaseCommercialNavigationData(fw);
    ///* EndGearsBlock Java BaseCommercialAviation

    ///* BeginGearsBlock Java StandardCommercialAviation
    generateStandardCommercialNavigationData(fw);
    ///* EndGearsBlock Java StandardCommercialAviation

    ///* BeginGearsBlock Java TailoredCommercialAviation

    ///* BeginGearsBlock Java CommercialAirlineAlpha
    generateTailoredCommercialAirlineAlphaNavigationData(fw);
    ///* EndGearsBlock Java CommercialAirlineAlpha

    ///* EndGearsBlock Java TailoredCommercialAviation

    fw.flush();
    fw.close();
} catch (Exception e) { System.out.println("Caught exception " + e); return false;}
```

Figure 10 - Code Block Include / Exclude

Now that the Feature Catalogue, Bill-of-Features, and variation points have been created, the Products in the Product Line are specified in a product matrix, and the production line is actuated (run) as depicted in the figure below. Two cycles are performed: the first cycle actuates within the developer's integrated environment and unit tests executed; the second cycle actuates to a staging area where the tailored product is available for downstream build, test, and deployment.

Figure 11 - Actuating the Product Line

Information Product Benefits

Once the FBPLE Factory has been matured the organization will reap numerous benefits. First and foremost is accommodating the previously mentioned commoditization of data through a substantial

decrease in the operational expense for developing and delivering information products bundled with other portfolio goods and services.

Management of Complexity and Capture of Domain Knowledge

Our information age has resulted in an explosion of data, in terms of size (think BigData) and in complexity – both inbound and outbound - to consumers. FBPLE provides an organization with the ability to manage complexity at all scales, up to and including tens of thousands of features through the use of abstraction, modularization, and automated generation of products from a superset of configuration managed reusable assets [2]. Additionally, domain knowledge from subject matter experts is codified in the feature models and associated business rules - feature assertions describing constraints and dependencies among the feature declarations – that ensure products are fit-for-use.

New Product Pilots

Piloting a new variation is now reduced to modelling the new features to be marketed and adding the requisite digital assets and tests, then actuating (auto-generating) the new product in the product line. Contrast this with the typical pilot which requires standing up a new development environment, cloning the desired assets, refactoring and potentially performing a technical refresh on the clones, adding the new capabilities, creating new test harnesses, et al. With FBPLE a new product pilot can be in the hands of business development within an estimated hours or days rather than months.

Support for New Consuming Contracts

Similarly, creating a new variation for a new consuming contract is reduced to the requirements analysis, development, and deployment of new capabilities. Contrast this with a typical new project inception requiring formation of a new project team; acquisition and implementation of the required development, test, pre-production, lab, and other environments; generation of new project plans, schedules and metrics; and more. Sources including [3] and [4] show significant time to market / time to deployment compressions and development cost reductions of $40M USD annually. Further, the more categories of assets (e.g. requirements, architectural designs, hardware design specifications, bill-of-materials, and more) added to the production line, the greater the cost avoidance – referred to in [4] as "superlinear" cost avoidance.

Operations and Maintenance

Similarly, operations and maintenance costs and schedules are dramatically reduced using an FBPLE Factory. Rather than laboriously fixing the same defects across multiple, individual products, modifications are made to the reusable superset of assets and the product variants automatically regenerated [1]. Additionally, unified operations and maintenance (O&M) or field engineering business processes and training reduce costs across the portfolio.

Summary and Conclusions

Today's cyber physical systems fundamentally rely on information for their operation. Feature-based systems and software product line engineering and management (FBPLE) techniques and tools provide an organization with the ability to effectively and efficiently generate information products as a cost-effective component of a portfolio of "hard" goods, "soft" goods, and services that provide a holistic solution to consumers. With FBPLE the organization is able to deliver precisely the information the consumer needs, when they request it, and in the form or format desired in a cost effective and timely manner. Once the information product lines are matured, the factory paradigm results in a tenfold or more improvement in the ability to add new features and provide ongoing maintenance [4]. Use of FBPLE provides an organization with greater agility to meet market demands and provides a competitive advantage with capability-rich offerings in a cost efficient fashion.

References

[1] Krueger, C. and Clements, P. “Systems and Software Product Line Engineering,” Encyclopedia of Software Engineering, Philip A. LaPlante ed., Taylor and Francis, 2013, in publication, retrieved from http://www.biglever.com/extras/PLE_SE_Encyclopedia_2013.pdf

[2] Flores, R.; Krueger, C.; Clements, P. Mega-scale product line engineering at General Motors. SPLC 2012, 1, 259–268.

[3] Gregg, S., Scharadin, R., LeGore, E., and Clements, P. “Lessons from AEGIS: Organizational and Governance Aspects of a Major Product Line in a Multi-Program Environment,” Proc. SPLC 2014, Florence.

[4] Gregg, S, Scharadin, R., Clements, P. “The More You Do, the More You Save: The Superlinear Cost Avoidance Effect of Systems and Software Product Line Engineering”, Proceedings Software Product Line Conference 2015, Nashville, 2015.

[5] Lanman, J., Kemper, B., Rivera, J., Krueger, C., “Employing the Second Generation Software Product-line for Live Training Transformation,” Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011.

[6] Clements, P. & Northrop, L. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.

Biographies

Jim Teaff wrote his first line of code as a professional in 1982 while working for a small startup. Subsequently over the past three decades Jim has worked for numerous aerospace and commercial companies across the full system development lifecycle, from principal investigator and proposal writer to IPT lead, chief architect, requirements analyst, Scrum Master, programmer, tester, 2nd tier O&M support, and more. Jim holds a Bachelor of Science degree in Computer Science from Colorado State University, and a Master of Engineering in Engineering Management from the University of Colorado. Jim is an INCOSE Certified Systems Engineering Professional (CSEP), and an ISC^2 Certified Information Systems Security Professional (CISSP). Jim joined Raytheon in 2016 as the Intelligence, Information and Systems (IIS) Product Lines Champion, and is assisting the organization with the continued rollout of feature-based systems and software product line engineering and management.

Dr. Paul Clements is the Vice President of Customer Success at BigLever Software, Inc., where he works to spread the adoption of systems and software product line engineering. He was previously at Carnegie Mellon's Software Engineering Institute, where for 17 years he worked in software product line engineering and software architecture documentation and analysis. Clements is co-author of three practitioner-oriented books about software architecture as well as the field's leading text on software product line engineering.