EVOCEAN
guiding you beyond

Challenging Complexity with better support for Reuse

Configuration Management across Engineering Lifecycle Management

Reaping the benefits for higher speed and better collaboration

Products and systems are becoming ever more complex and connected. This is a challenge for bringing new products to market effectively and for product development to be efficient. Further, the market and innovation push drive the necessity of building and supporting increasingly more variants of systems to address specific market needs, each variant also being released in different versions to address new needs in time. As a result, we have already two dimensions at a top level view: the variants in space and the version in time. However, it is more complex.

A system is made up of sub-systems and a subsystem is a collection of parts at a given time or a sub-system again is built of sub-systems and parts at a given time. And you can continue adding further levels of systems of systems. These levels further increase the complexity by adding additional dimensions, as all of the systems, subsystems and parts etc. evolve over time and are ever more connected and dependent on each other. On the other hand, effective reuse of parts or subsystems is crucial to drive new products or systems to market in-time and commercially viable. This is especially true for markets with high compliance and certifications needs.

As a result, various parts (also known as components) will be reused in various products or sub-systems, some as is, others again in different variants and some will be unique to an individual product, but each of them will have their own life in time, that is in “versions”, as they evolve with additional needs being met.

For the ease of reading we will only use systems and not products in this article.

As mentioned not only the end system needs to be managed, but also all the parts building the end-system and all the artifacts needed to build the parts etc. In addition, each asset’s domain needs to be developed and maintained during its lifespan, for example requirements, design, implementation, tests etc. This is where Configuration Management comes into the game, which needs to include all three dimensions the “type” ie. system, sub-system, part, artifact, the variants of these (as mentioned above these are the different editions in space) and versions (the different editions in time) for it to be effective. To make it even more complex, you will have versions and variants on different abstraction levels as discussed above, ie. from subsystem or system view and also from the view of parts or components making up the system respectively the sub-system.

Architectural design of the system will be key for how well the system will survive in time. In this article we will not dive go into the architectural aspects, however we would like to emphasize the importance of architecture which is mainly driven by non-functional requirements such as performance, scale, team, etc. which will also finally have an important impact on how reuse will be supported.

I want to get back to the configuration management concepts we need to think of for a system to be built from various parts. For this, I want to first reflect on the challenges we are seeing in the market, which can be summarized as follows:

- Complexity of Systems is increasing needing better traceability over different domains in engineering (Digital thread), especially as they are more and more connected to each other and as such dependent of each other.
- Distributed teams require better support for collaboration and parallel development across engineering disciplines is required to support shared artifacts, contributing to overlapping releases or product variants.
- Change is the norm so better support for change management to address change impact, change analysis and tradeoff studies.
- Better effective reuse is required at all levels.

To meet these challenges improved ways of collaborating across disciplines will be key for better outcomes. Especially reducing the need of cloning artifacts, such as but not limited to requirements, tests etc. which result in maintenance challenge and hinders effective reuse.

One of the key technologies helping to meet these challenges is given in the standard for Open Services for Lifecycle Collaboration (OSLC), which is an open standard, now managed by the OASIS Open Standards Network, an international nonprofit consortium that brings stakeholders together to solve communication challenges. OSLC is a key enabler to achieve the digital thread across domains, applications and organizations. For more information on OSLC please go to www.open-services.net.

www.open-services.net

Let us define some important terms we are going to use more specifically in the following sections:

"Local" Configuration: With this we mean configuration within one domain or one tool. Local Configuration are with respect to artifacts. For example, traditionally configuration management as used in software in order to manage versions of software. A configuration as such is set of versioned artifacts, either uneditable baselines or editable streams.

"Global" Configuration: A global configuration assembles other configurations (local ones) within and across applications. It thus represents a particular state of a system, sub-system or component across the various artifacts (i.e. across various disciplines).

Component: A component represents a physical or logical part of a system or sub-system. It is a unit of organization of artifacts and are the basis for reusable parts. This can also be called a global configuration.

Stream: Streams are modifiable configurations of artifacts from one or more components. Team members deliver their changes to a stream when they want to make their changes visible to other team members. These can be related to global and local configurations. Streams are the bases for baselines. New streams are always created from baselines.

Baseline: Baselines are snapshots of artifacts (cannot be modified) from one or more components. Baselines are useful for enabling a team to work with a known configuration, or as an initial state for some new stream of work. These again can be related to global or local configurations.

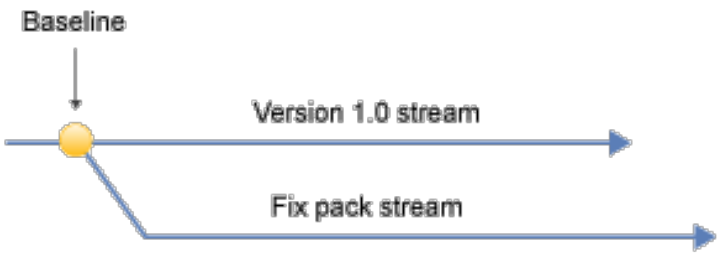
Working on artifacts with Streams and Baselines

Artifacts in a stream can be modified, and the stream always presents the artifact version that is currently selected by that stream (other streams or baselines can select other artifact versions). By maintaining the artifacts for a project in a stream, teams can share their work and ensure they are working with the artifact version that they need.

When the work on a set of artifacts in a stream is complete or has reached some milestone, you can create a baseline of that stream to capture the specific artifact versions selected by the stream at that time. The artifact versions in the baseline cannot be modified, nor can the set itself. In the future, the team can use the baseline to create a new stream for its work.

Configurations enable parallel development

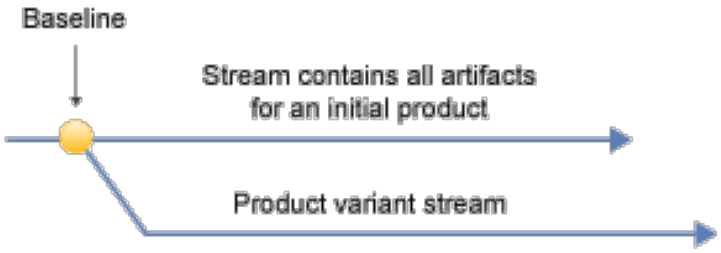
For software applications, the current stream might be used for the next version of the application or product while a new branched stream is used for the first maintenance fix pack. Changes in the maintenance stream can be delivered to the current stream, so the next release includes all changes from the maintenance fix pack. For many years systems and software development teams have used configuration management in this way to develop parallel releases.



To branch a stream, first create a baseline of it; then, create a new stream from that baseline.

In a product line context, the initial stream might hold all artifacts associated with one product while the new stream holds the artifacts and versions associated with a new product variant under development. Several variants can exist at the same time. This is how teams manage versions and variants.

In the new stream, artifacts are reused by reference, rather than being copied. Only new or modified artifacts have versions unique to the new stream.

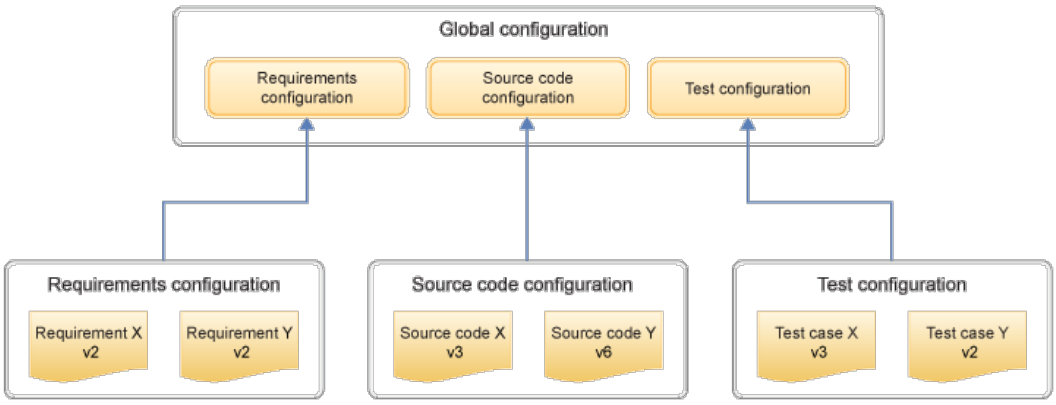


Assemble configurations to represent a specific version of a component or product

You can thus assemble configurations that contain other configurations from one or more OSLC enabled tools. Assembling configurations offers teams two main benefits:

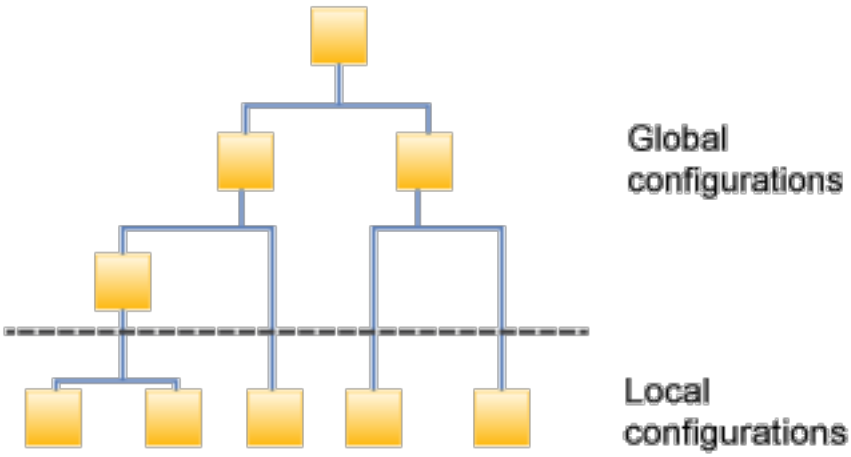
- Provides your team with a holistic view of the artifacts (requirements, test artifacts, source code) for a specific version of a component or product. The view enables your team to develop multiple versions of an application or product in parallel. As you work within and across tools, you see only the artifacts relevant to the version of the component you are working on.

- Creates a hierarchical component structure that reflects the system under development and enables subsystems to be updated independently. By grouping multiple configurations from different tools, you can represent the artifacts associated with a complex and aggregated subsystem, such as an engine, or complex component-oriented applications.



Introducing Global Configurations

With reference to the OASIS Open Services for Lifecycle Collaboration (OSLC) specification the term “global configurations” is used to distinguish them from the local configurations that contain only artifacts. Any tool can contribute configurations to a global configuration if it supports the OSLC configuration management specification.



These new features help teams:

- Work in parallel development streams
  - Create and reuse otherwise unrelated configurations of artifact versions, within or across tools
  - Start design and analysis for the next release while developers are still using the artifact version they need for the current release
- Incorporate changes to artifacts in an isolated environment; then, after they are approved, finalize and control where they are delivered. You can compare changes to see what has changed since an earlier milestone, or to see changes across streams. When you are ready, you can deliver your changes to the appropriate stream.
- Improves linking and navigation across tools. The technical vision for cross-tool configuration management includes the ability to link and navigate across tools within a configuration context, but only to the artifacts and links that are in the configuration you are working in.
- Create cross-product baselines that capture the versions of artifacts at a particular milestone or in a specific product version or variant. When you select a baseline configuration you recreate your development environment. You can determine, for example, which type of artifact was the source of a defect reported by a customer: Is a requirement inaccurate? Did a test case miss an essential capability or misinterpret the requirement? Does the source code contain the defect?

Create a branch from an existing product or library of core assets to create a variant.

Conclusions

Developing products and systems is becoming ever more complex. The need for better reuse and analysis of impact coming from changes and new requirements is driving better support of traceability and reporting across the different engineering domains and tools. A key enabler making this possible is OSLC and the support of configuration management from different levels (global and local) as explained above. This will also help to manage change through easier dependency management.

As a result, time consuming effort of analyzing changes will be reduced as tools support ever more the insights and impacts of changes. Further better support for reuse will help innovation and drive new solutions to meet customer and market needs faster and more effective.

This support will help teams to work better in parallel on multiple development streams, to re-create a development environment from the past, or to build a hierarchical component structure of a system that you are developing, while managing versions and variants across product lines.

Tool will thus help you, as you navigate across the lifecycle applications, to see the appropriate artifacts and links for the global configuration you are working in.

As a result, global configurations assemble all the relevant artifacts (requirements, designs, tests, source code, other global configurations) for a specific component release or product version. This allows to connect teams and their work across applications and offers a fast way to link different versions of requirements, tests, designs and implementation. Teams can work in their familiar applications yet stay connected to the larger context of their system.

These new capabilities allow configuration management across your Engineering Lifecycle helping to reduce and manage the complexity of working in teams across different disciplines on different versions and variants of product or system developments.

Next Step

If you have questions or want more information about solutions addressing this, please contact Philip Zollinger at EVOCEAN. He will be happy to share more information with you.