Beyond Information Silos

Challenges in Integrating Industrial Model-based Data

Ali Shahrokni, Jan Söderberg

ali.shahrokni@systemite.se, jan.soderberg@systemite.se

Abstract.
Fragmented data management is commonplace for handling development and production data in large organizations today. While automated configuration and build mechanisms have made the implementation data more integrated, on higher levels of abstraction data is still fragmented into silos such as files or isolated databases. In the recent years, many companies, including a large number of automotive corporations, have realized the potential of integrating high-level data when it comes to saving cost, lowering time to market, increasing the efficiency of the processes, and increasing the quality of products. To achieve the integration, many initiatives on tool provider level and on standardization level have started. OSLC is one example of underlying architecture to enable integration of data that is stored in a distributed fashion. Systemite has more than 15 years of experience in central integrated data management, largely in the automotive industry. In this paper, we discuss our experience on some of the challenges that we have faced during the past years when it comes to integrating model-based and fine-grained data. Finally, we provide two approaches to move forward towards open ecosystems for tool interoperability.

Keywords: Model-based information management, tool interoperability, integrated information management, OSLC, meta modeling

1 Introduction

In most industries the predominant way of managing the complexity of product and process data during development is still the use of a file based approach, where data is fragmented into multitudes of computer files. As the amount of data increases and the nature of the products and processes become more complex, a new and separate type of complexity emerges from this file based approach that is not related to the complexity of the product and the organization. This complexity is a result of the scattered data. Finding the correct and up to date data in a large organization with thousands of documents can be a cumbersome task that can consume 14-30% of an engineer’s time [1, 2, 3]. Finding the right version of the right data with a high confidence in the sea of data produced by parallel projects and historic data is often an impossible task. At the same time, using incorrect and obsolete data will produce inconsistencies that decrease the quality of the product or raise the verification costs.

As the products, processes, and development and manufacturing data becomes more complex, the cost of fragmenting data into files or other silos increases. This cost will rise as a result of difficulties to keep the data up to date in several files or several databases. At the same time organizations need to perform increasingly advanced analysis over the silo border, not the least as a result of external requirements such as safety standards or legal requirements. Another need from a process perspective is to have a realistic and correct view of the progress of large projects that is based on more reliable data than estimations. The need and the trend is clear that organizations are moving more towards generation of reports, analysis, visualization, and views of data instead of manually creating them.

An example that stresses the need for efficient data management is testing of large systems. The challenge here is that the results of several processes meet and touch in the testing activities; the test cases, the specifications that are used as references for the test cases, the artifacts comprising the System Under Test, the test environment and equipment to mention the most important. The realization of individual artifacts is typically uncorrelated, on the scale of time relevant for testing activities: new artifacts arrive every day, requirements and test cases constantly change, and regression tests have to be performed daily. The rate of change means that formal waterfall, baseline based configuration management is not effective nor efficient, since there will be many changes included within each iteration, between each formal baseline. This rate of change means that the test data representing the developed system will change frequently. This also means that defining configurations based on labels and performing check-out, check-in, and merge operations required for file based configuration management is not adequate.

These are just a few of the challenges that organizations are facing and all of these challenges point to the need for traversing the data not only inside one information silo, but over the borders of the silos (tools, files, databases). Therefore, there are many industrial initiatives to facilitate the interoperability between tools. The CRYSTAL project[1] is one of these initiatives. Crystal aims at standardizing interoperability between tools and integration of product and process information relying on the architecture of Open Services for Lifecycle Collaboration (OSLC)[2]. OSLC is an open community that exists to define specifications and mechanisms to integrate disjoint tools and workflows. The purpose of the integration is to save time and money through integrating data in different tools with the aim to keep the data more consistent and transparent over the borders of different tools. The underlying mechanism in OSLC is inspired by the web architecture in the sense that it is distributed and link based.

Systemite is a company with more than 15 years of experience in integrated data management. The company has an extensive experience in managing development data in the embedded and especially the automotive industry. Systemite provides a product called SystemWeaver is a model-based data management tool where all data that is stored or linked exists in a context and is integrated with its surrounding. The

[1] http://www.crystal-artemis.eu/

[2] http://open-services.net/

cornerstone and basic philosophy of SystemWeaver is to keep data integrated, traceable, and consistent while it provides a real-time and high performance platform for engineers and developers to enter and view data in its context in contrast to fragmenting data in different databases based on the type of data (for instance requirements and tests). At the same time the collaborative, high performance and scalability aspect are essential parts of the SystemWeaver platform. In our opinion, neglecting any of these aspects renders an industrial solution a liability rather than a resource as the quantity and complexity of data and organizations increase.

As the industry moves towards more integrated strategies for data management, we want to discuss some of the challenges in the area of data and share our experience on some of the obstacles that the industry needs to overcome to unleash more of the potentials of integrated data management.

2 Structured Data Management

The idea to manage system and product models in structured databases is not new. In the early 80's there were the CASE tools that relied on a centralized data dictionary or database server to manage and give access to the model data, which was typically developed according to some structured analysis and design methodology. (Example: Teamwork3). These tools were typically expensive and required the use of networked so called workstations, the high spec computers of the time. The high cost limited the use of the tools to specific, high margin industries such as military or aerospace. The arrival of PCs that offered low cost computing power coincided with the development of object oriented (OO) methods and tools such as Rational ROSE4. Frameworks of the 90s that relied on centralized data dictionaries, like the AD/Cycle of IBM or the open standard PCTE5 (Portable Common Tools Environment) were never used widely and were all ousted by the new object oriented tools running on the inexpensive PCs. A common feature of most of this type of tools, apart from being OO, was that they relied on storing and managing the model in computer files. This made the tools accessible when used in small projects, but the use of computer files also introduces the complications described in this paper. Throughout this period software development has remained a file based affair. Even modern ALM (Application Lifecycle Management) solutions manage the program source code as computer files, although information like change requests and configurations are managed with a database approach.

A common industrial method to deal with complexity is to minimize dependencies and interactions between subsystems, so that these can be largely developed independently. This practice of development on the sub system level leads to problems like sub-optimization, since analysis on the system level becomes difficult when all data is hidden on the sub system level. Any changes that need to break the subsystem barrier also tend to be very difficult to handle, calling for negotiation between sepa-

[3] Keysight Technologies
[4] IBM
[5] ISO/IEC 13719-1

rate development teams. Integration is not done until the subsystems have been developed, leading to late discovery of integration problems.

A special characteristic of configuration management of software is that the focus of the system generation (build) process is to compile and link single executables. It is indeed possible to extend the generation process to higher levels to collections of executables. However, since these higher level configurations do not have any specific semantics, from the perspective of the single executable and its software, they are rarely, if ever, used. Even when the development is according to concepts that support system level description and composability aspects, like AUTOSAR[6], the detailed development is today done on the subsystem (Electronic Control Unit) level. A reason for this is the lack of tool support for collaboration, configuration management and integration of data on the system level. Data is instead managed in multitudes of computer files, according to state-of-the-art software development practice. Another reason for the late integration is that the organizations responsible for the development of systems (and subsystems) are themselves organized in a way similar to the structure of the fragmented data. If you organize your system according to the technology required for the different parts, you probably have a development organization made up in the same way. This can even mean that there is actually no one in the organization responsible for the system level aspects. If the development of the subsystems is done by separate corporations, which is common in the automotive world, this situation gets even more accentuated.

3 Fragmented Data Management

According to our experience, file based data management is by far the most common way of handling data in the industry. This also applies to industries such as automotive that are more mature in data management. One reason for the spread of the file based approach is that many development tools still store their data in files, and the scale of individual development activities is small enough to be carried out in the traditional approach. However, the growing scale of systems like automotive electric/electronic systems combined with tight schedules in large development projects, and the need for integration and collaboration between different development activities has made the traditional approaches less feasible.

In traditional file based development individual artifacts are defined in separate files. File based versioning can keep track of changes and versions of such files, while also offering basic configuration support, usually using some label mechanism, where different files in a configuration are tagged with a label representing the configuration. A limitation with this approach is that the actual system configuration has to be built outside the versioning system, using some build script/mechanism, like the traditional ‘make’ command for software. This approach works well for development of software where an individual developer can develop a module of the software contained in a file, for a period of hours or days. For a complex system, or system-of-

[6] http://www.autosar.org/

system context, this level of granularity and level of collaboration becomes a bottleneck.

Many companies use tools for managing specific types of data such as requirements, tests, designs, or change requests. These tools are another source of fragmentation as they operate under the paradigm of information silos. Information silos have no or limited connection to their surrounding and are not aware of their organizational context. We call this approach silo-based information management tools. The data in these tools is stored on a finer granularity level than a file, which enables configuration and version management on a more detailed level than a file. Organizations use these tools increasingly to manage their data.

4 Integrated Data Management

As systems and systems of systems become larger and more complex, organizations realize the significance and difficulty of keeping data consistent. Data is produced and developed in teams that are spread geographically, and work across different disciplines and domains. These teams still need to have a shared understanding of the product and the organization. At the same time the data changes increasingly rapidly. In this reality, duplicating data creates inconsistencies unless advanced mechanisms are put in place to keep all copies of the data in sync. Also, data does not exist in isolation. Much of the value of the data comes from its context and how it is connected to other data. Hence, linking data sources instead of copying them is a natural solution, and there are clear trends, for instance the OSLC initiative, suggesting that industrial tool vendors are moving in this direction. IBM’s Jazz platform and PTC Integrity are examples of this trend.

A few of the advantages of contextual and longitudinal traceability of data is to:

- trace the data back to the rationale and why it was created. This is important to assess the validity of data in a new context,
- see how a component, product, or process has changed over time. Organizing historic data is not only important in order to improve the future work, but it is in many cases valid since that data describes products that are still operating and being used in the market,
- see how a component is connected in a specific context,
- analyze the different uses of a component in different products,
- analyze the impact of the change to a component or requirements or any data in the whole organization

On a higher abstraction level one common solution to hold the data together and linked is to use a Product Lifecycle Management (PLM) approach where data is managed in a coherent framework. In this approach the actual system configuration, for example as defined by requirements, test cases and test results, is explicit in the PLM system, with no need for a separate build process like the one used in software configuration management. This means that data produced by test activities can update the configuration in real time, by direct access to the representation of the developed system in the PLM system. This kind of real-time access and collaboration is not limited to human intellectual interaction, but can also be used for automated processes like regression tests. PLM systems typically do not manage development data such as software development data and test automation for complex embedded systems today.

As the complexity of systems increases, the industry has no choice but to improve their handling of low level data and also improve the handling of high level Systems Engineering data in their PLM systems. In this reality, challenges arise to connect these two worlds into a holistic data management ecosystem that works on an industrial level; solutions that not only can describe an organization, its information, and how the information is connected in one project or one product, but also over time.

4.1 Tool Interoperability & Data exchange

Intra-organizational communication across different groups, domains, and disciplines is a major challenge for many organizations. While fragmented data management strategies have tried to address this challenge, they often give rise to new types of complexity in form of consistency problems and major rework instead of reuse, by dividing the data into silos; no matter if the silo is a file or a database.

Inter-organizational communication such as supplier management and customer management is another important aspect of most development and manufacturing processes. The medium for this communication is often files. In this reality, data management tools pack the relevant data into collections of files such as requirements or test specifications. These files are sent to the supplier and are either processed in their existing form or unpacked into receiving tools on the supplier side. Depending on the tools involved and the nature of the data, these supplier interaction files have different levels of formalism. Packing and unpacking data creates challenges to keep the data correct and consistent during round-trips.

An alternative to packing data to files and unpacking on the receiver’s side is that customers and suppliers use the same tool or use tools that use the same formalism for their data (for example EAST-ADL[7] or AUTOSAR) or even tools with formalisms that are transformable to each other. In this case the data can theoretically be exchanged with finer granularity than file. However, even this approach results in inconsistency in the data, as the root problem is duplication of data on a storage level and not the format or type of data. Duplication of data cuts the relationship between the data and the context that gives rise to it and the context in which it was created. The disconnection limits the ability to keep the data consistent and coherent and it makes the data unanalyzable for any foreseeable future until big data analysis methods become advanced enough.

One of the characteristics of complex systems, like embedded systems, is the multitude of aspects that need concern during development. The classical (minimal) Product Data management (PDM) approach is to manage the main product structure of the system, where detailed data is kept as proprietary, black box representations for each block in the structure. This solution is established and functioning for CAD data

[7] http://www.east-adl.info/

of mechanical systems. While, for complex multi-disciplinary systems the product or module structure is one of the least important structure to be managed. Other more or equally important structures or viewpoints are, to mention a few:

- Connectivity – interfaces between components defining responsibilities between them
- Interaction – how the component interact through the interfaces in order to achieve desired functional properties.
- Allocation – how functionality or responsibility is allocated to the physical structure
- Dependability – how the system components and requirements fulfill the safety goals of the system
- Requirements – how the properties of the system components and derived requirements fulfill product requirements
- Test and verification – the implementation status of the system as proven by test

To enable interoperability of tools involved in the development process all the above structures must be open. Open means that they must be visible and accessible. Moreover their semantics must be defined. One efficient approach to achieve this is to share a common meta model, accepted by the product domain. The level of required interoperability, from the needed viewpoints (listed above), defines the level of granularity of this openness, and a coherent domain specific definition of this openness for embedded (automotive) systems has been defined in the EAST-ADL meta model. The common formalism is also the cornerstone of the OSLC architecture.

4.2 Challenges in connecting tools and silos

As already discussed, linking distributed data in a way that is useful in an industrial context is a major challenge. Service-oriented architectures such as OSLC provide a possible first step of integration. In this step, loose links between two silos are created. The source tool (Consumer) can retrieve specific data from an OSLC service connected to a target tool (Provider) by providing the address to the desired data. This link can then be stored on the consumer side to establish a standing relationship with the provider.

Shallow links and deeper contexts. These links can only be used for navigation. For instance, while it is useful to see what requirement a test comes from, it is not possible to see the connection between a requirement and design or implementation items while standing in a test tool. In other words, a limitation with these links between silos is that an item can only be aware of the links it directly owns and therefore the depth of the visible links from the perspective of each item and tool is one (connecting only two silos at a time). These links per se can therefore not be used to generate reports or analysis on deeper contexts or structures.

Consistency and up-to-date data. In a distributed linking architecture, links are directed. Information about the target item and about the link are stored in the consumer tool. In this situation if the target item is deleted the link will be invalid.

Furthermore, there are no standard versioning concepts across tools. Many tools lack any versioning and configuration concepts at all. The rest of the tools handle these concepts in a heterogeneous way. For example, if there is a new version available of the target item, the consumer should be notified of the change.

Following the same train of thought it is clear that while the concept of weak links to connect fragmented data is useful from a user's perspective and enables the user to navigate quicker between two tools, there are many challenges to keep the links consistent between two silos. If this aspect is not considered creating and maintaining OSLC-like links can be a costly and inefficient concept in an industrial setting. Links to growing and live data need frequent updating either by use of bidirectional links with a notification mechanism or by a centralized data hub that keeps track of all links between silos.

Semantics and specifications. As mentioned earlier, predefined semantics and meta models is one solution to agree upon exchange and linking formats. However, in practice we see the difficulty to agree upon these semantics. Different organizations have different products and different processes, which leads to the need for different semantics even inside the same domain. AUTOSAR is a good example of this phenomenon. Although AUTOSAR is a widely spread standard, different organizations have different versions and interpretations of AUTOSAR. Organizations want the flexibility to decide their own processes and formalisms. Meanwhile, too much flexibility makes the tool integration difficult if not impossible.

SystemWeaver has a programmable meta model, which enables us to extend and integrate specific meta models in a single platform, rather than integrating different tools. Historically, we have built the interoperability in SystemWeaver, for tool and process integration, by defining a meta model to be used within SystemWeaver.

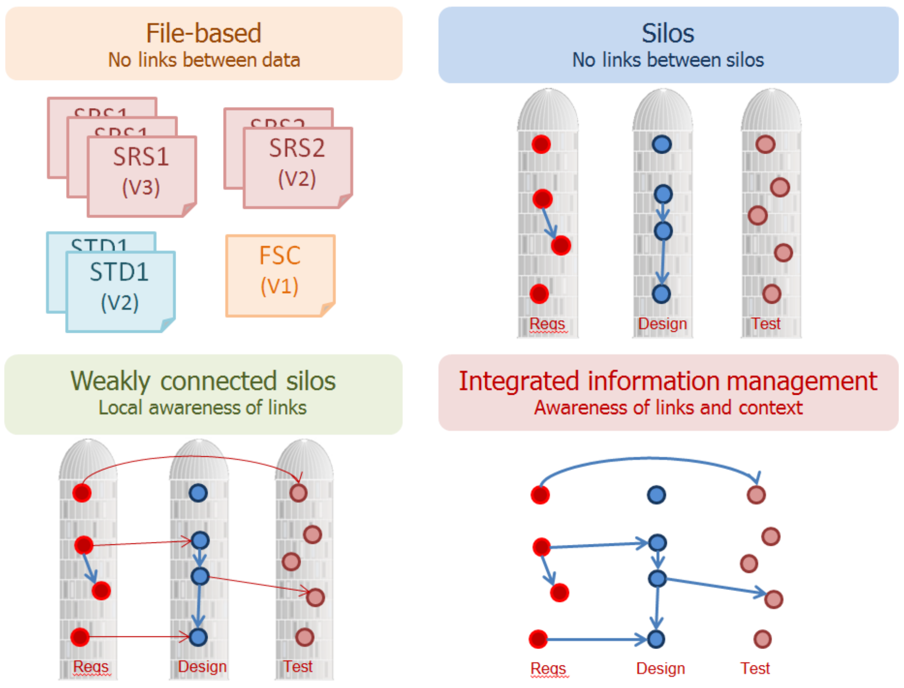
By the use of industrially accepted meta models e.g. EAST-ADL and means of technical integration as offered by OSCL this level of integration can be achieved also between different tools. Note that provisions for technical integration are not enough since there must be a shared definition of the semantics of the shared data.

5 Discussion and Conclusion

A common way to solve a big problem is through the reductionist approach of breaking the problem down into smaller problems and dealing with them in isolation. Engineers define responsibilities and interfaces of a system early in the project on a high level in systems engineering. Early definition of interfaces requires simple and manageable interactions between subsystems. This leads to a development process with less need for integration of development data. Also, often we see that a problem is never really recognized as a problem until there is first a solution available. We think there is increasing awareness of the problems associated with fragmentation as discussed in the paper. A main driver for this awareness is the need for more cost efficient systems with higher performance, systems that require a higher degree of integration, and optimization of system properties.

In this paper we discussed four main approaches to data management. These four approaches are presented in **Fig. 1**. The file based and information silos that lead to fragmented data are the most commonly used data management paradigms.

Fig. 1. Four approaches to data management



As discussed, there are initiatives and a sense of urgency in the tool provider community to go beyond fragmented data. This need and urgency is a strong pull from the customers since the cost and complexity of fragmenting data is becoming clearer as systems become more complex. One approach to remove the fragmentation is to store and link all the data in one tool and platform and another is to create an ecosystem where different tools can coexist and cooperate, similar to the development in the smart phone app industry. Although the latter alternative seems more feasible and desirable, an open and distributed approach to tool integration is a large endeavor that requires many years of research and development. Two research questions that need to be answered are:

- How to create data links across tools to be able to traverse them in order to generate specifications and reports, export structured data to other tools, perform differ-

ent types of analysis, and etc. The traversal requires awareness of data links external to a single tool.
- Harmonized formalism between tools in order for the links to have semantics and in order to be able to define structures for the generated specifications, reports and analysis.

Tool interoperability has all the problems and challenges of classical data round-tripping (state of the information between export and import). These challenges manifest clearly in a weakly connected silo solution unless elaborate mechanisms are deployed. We can see two main solutions to address these challenges in order to create an industrial tooling ecosystem:

1. Bidirectional links with notification mechanism to keep the links between two silos consistent. The reason for bidirectionality is for the target item to be aware of the link in order to notify the source tool about potential changes.
2. Centralized data hub where there is a central/federal source that is aware of inter-tool links in an organization. In this case all tools need to notify the central hub about the relevant data changes and the central hub is in charge of keeping a consistent and up-to-date view of the links.

As discussed in the paper, there are many challenges such as versioning, configuration management, and data consistency that need to be addressed for both of these solutions to work. For example, when it comes to versioning, a decision is whether to point to the new version of the item or the old one. There are different types of links from a versioning perspective. A link that always refers to the latest version of an item, called floating link in SystemWeaver, or a link that points to a specific version of an item, called a stable Configuration Management (CM) controlled link.
When these mechanism are in place and the organizations can be confident about the validity of the links the next step would be to use the created link infrastructure to do analysis and generate reports, visualizations, and synthesize data. In case of solution 1, there is a need for peer-to-peer mechanism for generating reports that have data traces deeper than one link. According to our experience at Systemite, the ability to synthesis data that traverse deep structures is an important enabler for having fully generated reports and analysis and being confident about the correctness, completeness, and consistency of the generated reports and analysis.

References

1. Robinson, Mark A. "An empirical analysis of engineers' information behaviors. "Journal of the American Society for Information Science and Technology 61.4 (2010): 640-658.
2. Culley, S. J. "Court, AW, and McMahon, CA, 1992," The information requirements of engineering designers." Engineering Designer 18.3: 21-23.
3. Puttré, M. (1991, October). Product data management. Mechanical Engineering, 113(10), 81–83.