Systems engineering and PLM

Basics

What is Systems Engineering?

The term Systems Engineering is today becoming more and more used for the development, testing and validation of complex technical systems in multidisciplinary cooperative activities. The systems themselves can be almost anything – a cell phone, a dishwasher, a machining center, a car or even a satellite. The most important of the disciplines involved are those of mechanical engineering and electronics and electro-technology, as well as informatics.

The term may well be gradually replacing that of mechatronics, which arose in the 60s to define described systems comprising mainly mechanical, electronic and electro-mechanical parts. At that time, only the beginnings of informatics existed and it played almost no role in the electronic systems of those years. Today, informatics – particularly in the form of embedded systems – takes pride of place in many respects. However the term mechatronics, which does not actually relate to informatics, has retained usage even for the ultramodern products whose development is driven by informatics. Normally, when explaining the term mechatronics today, software is included as a further central component.

An oft-quoted example: VDI Directive 2206, which came into being under the chairmanship of Professor Juergen Gausemeier of the Paderborn Heinz Nixdorf Institute, is a practical guide to the systematic development of mechatronic products. Wikipedia describes the subject as “Mechatronics is the combination of mechanical engineering, electronic engineering, computer engineering, software engineering, control engineering, and systems design engineering in order to design, and manufacture useful products.”
One of the common representations of mechatronics. Source: Technikum Vienna.

The term Systems Engineering is even older than mechatronics. It came into prominence after WWII, particularly for the American and European space programs. It was an attempt to describe a methodology that was suitable for the development and implementation of almost unmanageably large projects. This had mainly to do with the large numbers of participating companies – the normal case in today's globalized world – and the large time spans across which projects had to be planned. It also specified how to measure success against the originally defined requirements and was a response to what seemed to be the increasing impossibility of getting time and costs under control in such large projects.

When the term is today used for a wide variety of systems, the size of the system, and the time it takes to create, no longer play a role. Very small systems such as cellphones, whose model ranges appear at ever-shortening intervals, have become good examples of systems that are in contrast to the usual understanding of what a system means. What is crucial in today's world is much more the multidisciplinary complexity of the system, its subsystems and components, and most recently its “intelligent” interaction with other systems, with the environment, and with the user or operator.

Further details on the term Systems Engineering can be found on the homepage of the International Council on Systems Engineering (INCOSE) or the German Society for Systems Engineering (GfSE), which uses the following definition:

*Systems Engineering comprises the main engineering activities necessary for the development of complex products. In order to successfully integrate a wide range of functions into a manageable and cost-effective system, widely differing requirements must be accommodated across the entire system lifecycle. To do so, the multidisciplinary and interdisciplinary nature of system design must be taken into account, for example by integrated interdisciplinary product development teams. Project management and Systems Engineering go hand-in-hand, since one of the main accomplishments of*
Systems Engineering has been to give the project manager well-founded planning fundamentals.

The “V-Modell”

In the last 30 years, the science of informatics has thoroughly accepted Systems Engineering, and in it has found a method to describe the concept, development, detailing, validating and completion of a system. This is done, for example, with the help of the so-called “V-Modell”. It is based on the waterfall model developed by the American software engineer Barry Broehm in 1979 for describing the software development procedure. It splits overall development into phases which must be sequentially undertaken, with each one ending with acceptance. This model was later extended with a branch that again leads upwards, mirroring each specification phase with a test phase. The diagram looks like a V, and this gave the model its name.

The “V-Modell” is a registered trademark of the Federal Republic of Germany. Developed in 1986 by a federally-owned company (IABG) as part of a Federal Defense Ministry project, it was later extended to V-Modell 97 and finally to V-Modell XT (2005) to include usage in civil and private system development. Today, the V-Modell XT is provided by the Commissioner of the Federal Government for Information Technology.

With the field of informatics now being the driving force behind innovation and product development, Systems Engineering – and the V-Modell itself – have now entered the realm of product development. In fact it can be met almost everywhere. It is an easy to understand, very strongly simplified model (that, by the way, exists in thousands of variants), for representation of the special characteristics that must be taken into account in the development and manufacture of an interdisciplinary system.
This model makes very clear that it is no longer just about a clean definition of the phases of system development, but that these development phases are worked through in parallel by diverse specialist disciplines. And this is where the great difficulty lies: Every single discipline has across the decades developed its own methodology on how to get from the concept to the ready-made product. All of them have developed and implemented IT tools to support these methodologies, however these methodologies are not open to, and cannot be transferred to, other disciplines. The supporting tools are in no way compatible and can only interoperate in the most seldom cases; mostly they cannot communicate with one another at all. To this must be added the fact that the management of development, as well as the resulting models and data, is also generally a matter for the specialist discipline concerned. This does not permit synchronization of the development and test phases, although these are running in parallel.

What does Systems Engineering have to do with PLM?

We have now arrived at the question of the relationship between Systems Engineering and Product Lifecycle Management. PLM has the task of supporting, managing, and if possible controlling the entire product creation chain, i.e. the development and manufacturing of the product and its use and maintenance through to recycling or scrapping. Because products in the past were primarily mechanical in nature, PLM has in its turn evolved out of this discipline. However PLM also exists for electronic or electro-technical products and we must not forget Application Lifecycle Management (ALM) for software products. As of December 2011, however, there is still no methodology or IT tools that support the interdisciplinary development of complex systems.
The relationship of Systems Engineering and PLM to one another is therefore of great significance:

Because PLM claims to manage the entire product creation process, it must answer the question of how the disciplines involved in complex product systems are synchronized and matched to one another during the product creation process, and in particular why a way must be found to simulate, test and validate systems in regard to their multidisciplinary functions. There must also be a possibility to manage the relationships of all digital models and data created during the development, production planning and manufacturing processes.

Specifically, the following tasks – in accordance with the most important phases of system development – must be dealt with:

- Requirements acquisition and analysis
- Description of the functions and modeling of the system architecture
- Breaking down the functions for distribution among the individual specialist disciplines
- Synchronization of component development
- Simulation of all parts of the system
- Validation of subsystems and components against requirements

Industrial companies cannot find solutions for the handling of such tasks without outside help, since the development methods available were not developed for complex multidisciplinary systems. For this reason there is intensive cooperation between industrial research departments and scientists and researchers in universities and other institutions, with many projects and initiatives having been called into being, each of which deals with (partial) answers to the great challenge of Systems Engineering. Nevertheless, the problems in this field are not the same everywhere and neither is the degree of experience or maturity of the processes. These depend on the sector, the type of product, the proportion and specific type of informatics involved, the type of usage and the way that intelligent linking to the Internet or other networks is carried out.

It is clear that to address this major challenge, the industry needs strong IT support that offers more functionality than the systems currently available in the PLM field. This is true in all sectors, and the general challenge for PLM suppliers – or perhaps for as-yet-to-emerge IT branches – will be to enter the market in the coming years with tools and solutions that can help solve the problem.

In the Systems Engineering and PLM whitepaper compilation, a range of suppliers present their position and their strategies with regard to Systems Engineering. Those agreeing to participate presently include AUCOTEC, Contact Software, Dassault Systèmes, PTC, SAP and Siemens PLM Software.

Comments

Comment by Joel | 24.05.2014

I just thought I should highlight that although Items & Files do indeed share the Lifecycle ability they do differ somewhat. File Lifecycle definitions are fully customisable with a seemingly infinite number of variations & combinations. However the Item lifecycle definition is fixed, you can change the name of each of the 4 available States and set a revision bump transition rule to on or off (Primary, Secondary or
Tertiary Revision bumps are available), but that is about it. There is also no link between Files & Items in either direction when it comes to Releasing' or changing the state of a file (or Item) they are independent of each other. Which ultimately means you have to do the same thing twice. There are some very good reasons for this which I won't go into (i personally don't agree with them, since I'm certain it could be reworked by Autodesk), however, there is a lot to gain from using Items and maintaining their Lifecycle state in addition to the File lifecycle state.