CASE STUDY

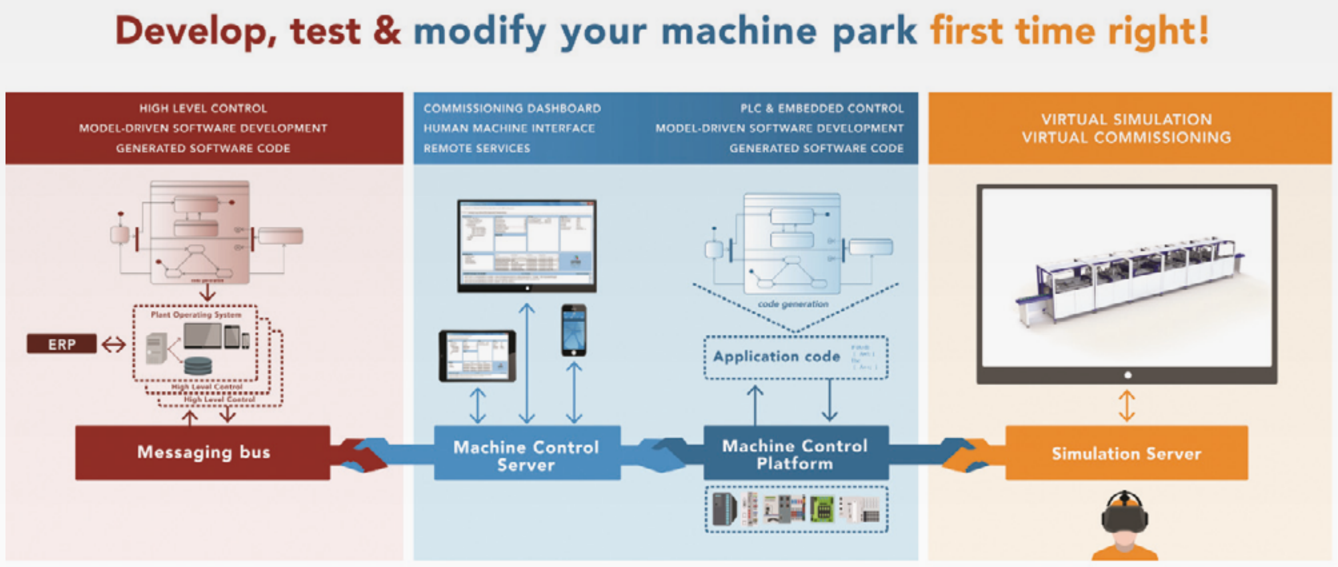
Softwareport: the three musketeers of a Model Driven Integrated Solution

Softwareport provides a software development platform that combines high-level control, embedded software and virtual prototyping into one integrative solution. The Softwareport is a joint venture of three software companies - Cordis, Festa Solutions and Unit040 - that uses a Model Driven Engineering approach, saving time, costs and programming headaches.

Combining high-level and low-level control logic

Softwareport's solutions are especially perfectly suitable for situations where high-level and low-level control operate together, for example in the distribution of luggage in an airport. The high-level control of coordinating incoming and outgoing flights and the low-level control of operating the luggage belts have to cooperate seamlessly. It could also be used in the operation of a series of bridges, where the arrival of the ships (high-level control) and the mechanics of opening and closing the bridges (low-level control) have to work in tandem. Another example is a smart factory (Industry 4.0), where robots, 3D printers

ICT smarter solutions



Softwareport offers a complete solution to combine high- and low-level control software plus digital simulation into a single package using a Model Driven Engineering approach.

and automated guided vehicles (AGV's) work together to manufacture and assemble products, increasing overall efficiency.

Developing software for these systems is painstaking and error-prone due to the complex behaviors that have to be managed and integrated. Through a Model Driven Engineering approach, Softwareport uses an elegant and simplified solution that saves time, energy and human coding mistakes.

Why Softwareport?

Softwareport can accelerate the transition to Industry 4.0, where manufacturing processes are smart and fully automated. Softwareport is able to develop both hardware and software for smart factories in less time and with reduced costs. Because of its automatic software coding based on functional design, hardware and software development run parallel instead of sequential.

The option of virtual simulation using a digital twin enhances efficiency by eliminating the time consuming and costly process of physical prototype testing and bug fixing. The use of running a digital twin parallel to the physical factory also makes maintenance more predictable. In later stages technical modifications and factory updates can also be virtually tested for risk-free implementation in the factory.

How do the companies work together?

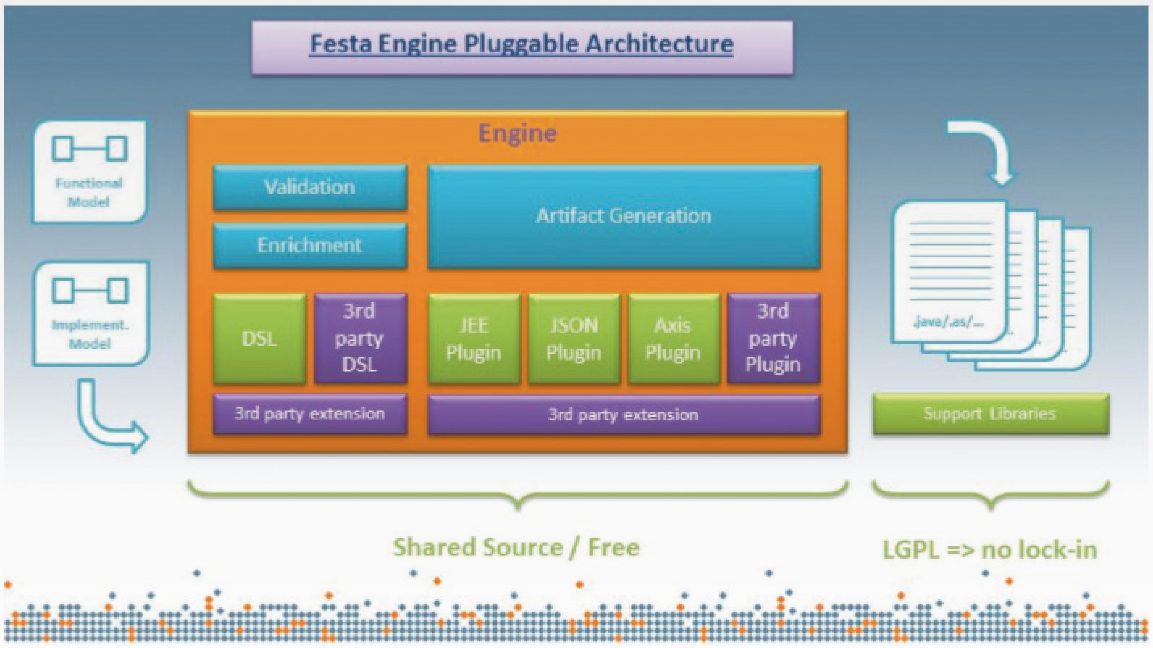
Each of the three companies comprising Softwareport takes care of one aspect of the integrative solution. Festa Solutions specializes in developing high-level control software, Cordis takes care of low-level control logic and Unit040 provides a visualization and simulation tool ('digital twin') to guarantee functional implementation of the software.

ICT Group works with Softwareport and trains software engineers in the Model Driven Engineering approach. For more information about Cordis and Unit040 and their practical application please see our Case Study about Cordis Suite. Below we will explain the advantage of working with Festa Solutions for developing high-level control software in a real-world scenario.

Case Study: developing software for an Air Traffic Control system using the Festa Engine

Festa Solutions specializes in developing High-Level Control (HLC) software systems from design to development, hosting and maintenance. Based on a Model Driven Engineering approach Festa offers higher quality, flexibility and functionality at lower costs and less throughput time. Festa's HLC systems are especially geared for use in smart factories, where internet-of-things technology and Big Data require vertical integration of IT systems.

ICT Group's software engineer Annie Jovitha Arulanandam enrolled in an internship with Festa Solutions to become more familiar with its software tool, the Festa Engine.

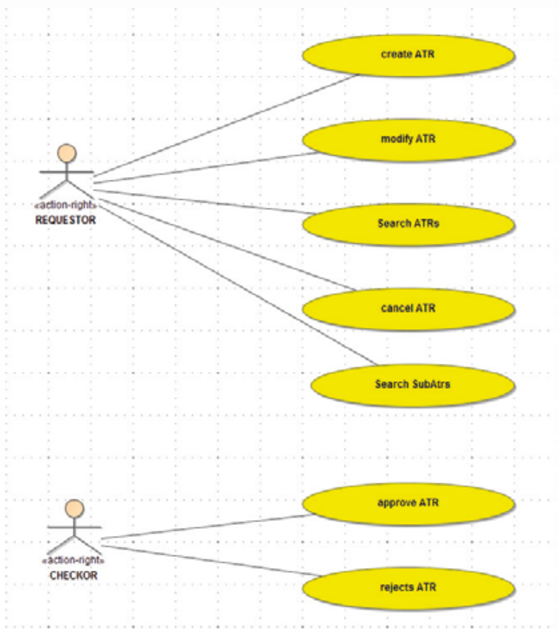


The Festa Engine uses a Model Driven Engineering approach, generating flawless code for software artifacts from predesigned models.

Software for a model Air Traffic Control System

The Festa Engine is especially particularly suitable to build software for systems that are use case driven, where different actors can perform different actions on the same system. Web applications with multiple users are a good example. Annie used the Festa Engine to develop software for a system that processes Air Traffic Control requests for logistics planning. This was a training assignment to give Annie practical experience with the Festa Engine. The resulting software would not be applied in the real world.

The system Annie designed serviced two users. The first user (called 'Requestor' in the model) makes air transport requests, specifying flight origin and destination, the number of passengers and/or the total weight of the cargo. The Requestor should be able to use functionalities such as 'create', 'search', 'modify' and 'cancel this request', which have to be programmed in the software. This request is then sent to a second user (called 'Checkor' in the model) that has to approve or reject the air transport request made by the Requestor.

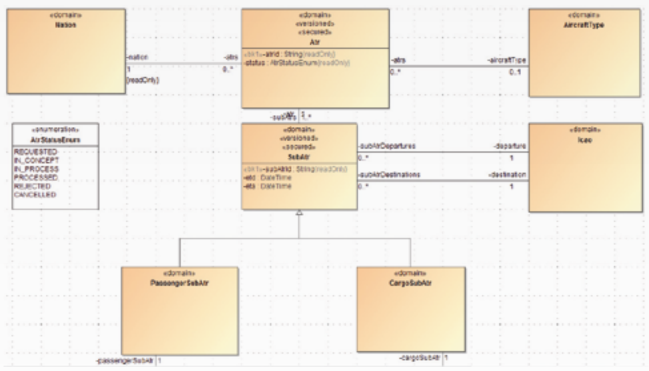


Use case model for the Air Traffic Control system. The model lists the two users Requestor and Checkor, each with their own specific actions.

Functional model: defining actions and class diagrams

In line with the Model Driven Engineering approach the Festa Engine uses graphical models that represent a functional overview of the software. Building software with the Festa Engine requires two modelling phases. The first phase is functional modelling, which is done by the domain expert, for instance a product analyst.

In this phase the domain expert generally creates two models. The first is a use case model, which defines the actions the model provides to the outside world. In our example these are 'create', 'search', 'modify', 'cancel', 'approve' and 'reject' air transport requests. The second model contains the class diagrams, which represent the main entities and their relations. In our example the entities are Nations, Air Transport Requests and Aircraft Type.



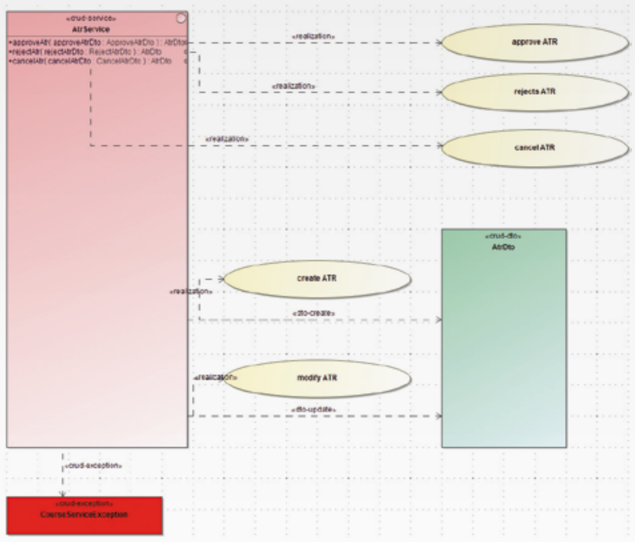
The class diagram above describes how a user from a Nation can create and own an Air Transport Request with a particular Aircraft Type.

Implementation model: transitioning data and generating reports

In the second modelling phase the implementation model is created, which is where the software engineer enters the picture. The software engineer now models the data service layer, security and authentication. The first modelling phase was all about the 'what', this phase concerns itself with the 'how', or in other words, how the data is transitioned in the system.

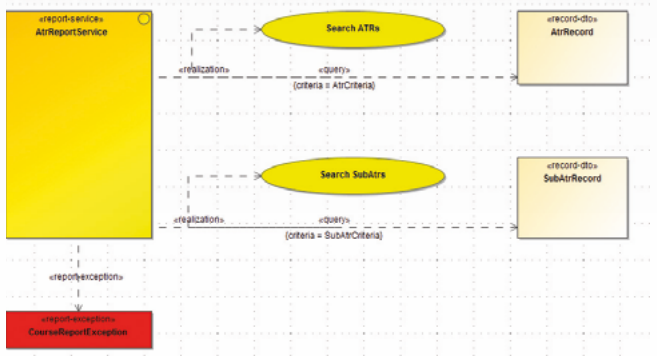
The implementation model creates the interfaces for the client in the form of a CRUD (Create Read Update Delete) matrix, is used to actualise the use cases or workflows. In this phase, entities and service models are added by the software engineer. Annie created several DTO's (Data Transfer Objects) that define how the data is transferred and stored in the database.

Next Annie generated reports to ensure that the search functions in the software - like 'Search air transport request' - would return the right values (passengers, cargo, destination, etc.). For this she created report diagram models. The report diagram is an API that acts as an interface for the Festa Engine, so the Engine knows how to



The implementation model above created by Annie describes the implementation of the CRUD services.

search and retrieve data. Finally Annie created a connector diagram to export the services to the outside world.



The report diagram model above describes the 'search' and 'retrieve' functions.

Coding business logic by hand

When all modelling was done Annie only had to press a button to generate the software code. The generated code represents 80% of the final product. The missing 20% consists of business rules and logic, which is not taken care of by the models. Annie had to program these rules manually (for example the rule that each nation can make multiple air transport requests). She also had to create unique business keys for all entities, to prevent the same request being created twice.

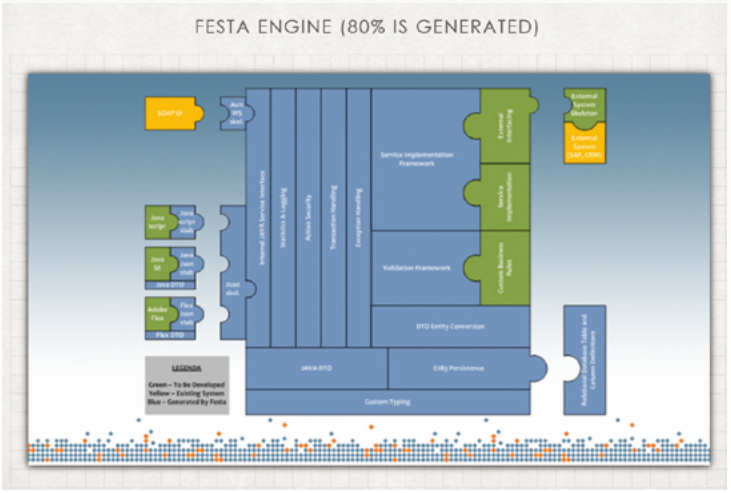
The code was then validated by the Festa Engine against the flow and rules created by the models. Finally the artefacts were created, ready to use in the application

ICT smarter solutions

server. The Festa Engine supports different programming languages. Currently these are Java, JavaScript, C# and Objective-C.

Flawless code generated straight from the models

The great advantage of Model Driven Engineering is that flawless code is generated straight from the models. Annie: "In traditional software development I would have to code everything by hand. But with the Festa Engine all the plumbing code and authentication is taken care of. If you want to make changes, you just change the model and with the push of a button you get new code. No copy-pasting is required."



The Festa Engine creates 80% of the software code, displayed above in blue. Annie still had to code the green pieces manually.

Eye-opener

The Festa Engine taught Annie many valuable new insights. One of them was how the use of models can facilitate software development. "The only models I had seen before were customer requirements on paper. In the Festa Engine I could actually give life to models and directly create code out of them. That was a real eye-opener. It also meant I had to model in domain specific language, which was new to me."

Using models facilitated the communication between the stakeholders, Annie found. "The models represent a language that both the domain expert and the software engineer understand. Since the model is actually an image, you graphically see what is required. Traditional requirements in text form are much harder to understand."

According to Annie, one of the biggest advantages of the Festa Engine is its faultless code generation. "I always had to be alert not to make manual coding mistakes. Since the model generates the code, it's one hundred percent error proof."

Curious about the possibilities of Model Driven Engineering and Festa Solutions? Feel free to contact our software engineer Annie Jovitha Arulanandam to discuss the opportunities for your organisation.

Contacts:

Annie Jovitha Arulanandam
Software Designer
E: annie.arulanandam@ict.nl

Ronald Wiericx
Operations Manager
E: ronald.wiericx@ict.nl

Professor Doctor Dorgelolaan 30
5613 AM Eindhoven