

# Applications based on the Emerging Systems Paradigm: Software Development Productivity Toolsuite

## ***Abstract***

Staying competitive in software development today means managing a lot of complexity.

If you have a problem conquering complexity then we provide systematic and efficient management of your core system and its feature components. We provide the technology, know-how, tools and services that enable extreme reuse in industrial and financial sectors.

Benefits are:

- reduced system complexity by enabling isolated modification of system parts,
- automation-enabled reuse,
- shorter time to market,
- agile feature management,
- end-user tailored products,
- optimal resource management, smaller footprint / hardware costs
- less human errors, efficient and motivated development teams

## ***Introduction by Analogy***

Imagine if you could assemble your own car by choosing from a set of available parts. How would you go about it?

1. Let's assume you already have a car chassis at your disposal (Core System)
2. Choose the wheels you would like to mount onto the chassis (Feature1)  
just specify where the wheels are to be fastened and which screws to use
3. Choose the steering wheel you would like to use in your new car (Feature2)  
just specify the size, shape and type of steering wheel you need
4. Choose the engine you want (Feature3)  
just specify the power, speed, fuel consumption you require in your new car
5. and so on, you may specify feature after feature...

After configuring your new car using the Product Specification, you can execute the System Builder to compose the car in less than 60 seconds! You can always go back and change your choices or modify the features in isolation and rebuild the system anytime.

You continue with the Platform Specification to specify the kinds of roads you will drive your new car on. When done, just generate the implementation for the complete system. Now you are ready to drive your newly assembled car on the roads of your own choice!

For more information and a complete analogy of concepts, please see Figure 1.

## ***Detailed description***

Systems always get bigger and fatter over time and any exceptions only confirm this rule. Growth of system functionality is hard to manage unless done systematically using an appropriate architecture and process. This is ever more important when an increase in the number of features or products leads to an exponential rise in system complexity.

This risk may be fatal to sustainable company growth. Our approach mitigates this risk.

We see your system made up of the Core System and Features as parts. Core System, occurs always and only once in each of the products you are assembling within your product line. Features and feature variability are specified in Product Specifications. The System Builder composes the target system according to the Product Specification thus making each product minimally complex. This results in a product with minimal resource requirements e.g. footprint and hardware not to mention performance enhancements. No conditional code is necessary to select features or to configure subsystems.

Core & Features are contained in functional and glue models and elaborated in full detail. All models are formally and completely described using static and dynamic viewpoints and are designed to manage the required variability. Models are executable, at least with respect to testability, making automatic testing of composable, target systems easy.

The central part of our system is the System Builder, that interprets a Product Specification and processes all input functional and glue models to produce the target system.

We define an iterative, incremental process consisting of 7 steps:

1. Apply a single feature profile to an input model:  
Input model is overlaid with a specification of feature functionality
2. Materialize feature functionality into a target model:  
Interpret feature overlay and create a target model with emerging character, with additional feature functionality
3. Generate the implementation model i.e. complete source code:  
Create a platform specific model according to platform specification
4. Build, deploy, test and debug the model:  
Breakpoints may be used in e.g. state machine transitions
5. Retract feature profile:  
Results in a clean target model enabling its reuse as input model, thus making the process iterative and incremental
6. Choose a new feature to add to the incremental target
7. Go to step 1. and use the clean incremental target as input model

After step 2 it is possible to generate an implementation model for a platform of choice. We consider a platform to be a stack of hardware, software drivers, operating system, VM, libraries/utilities and programming languages. The platform is specified using a platform

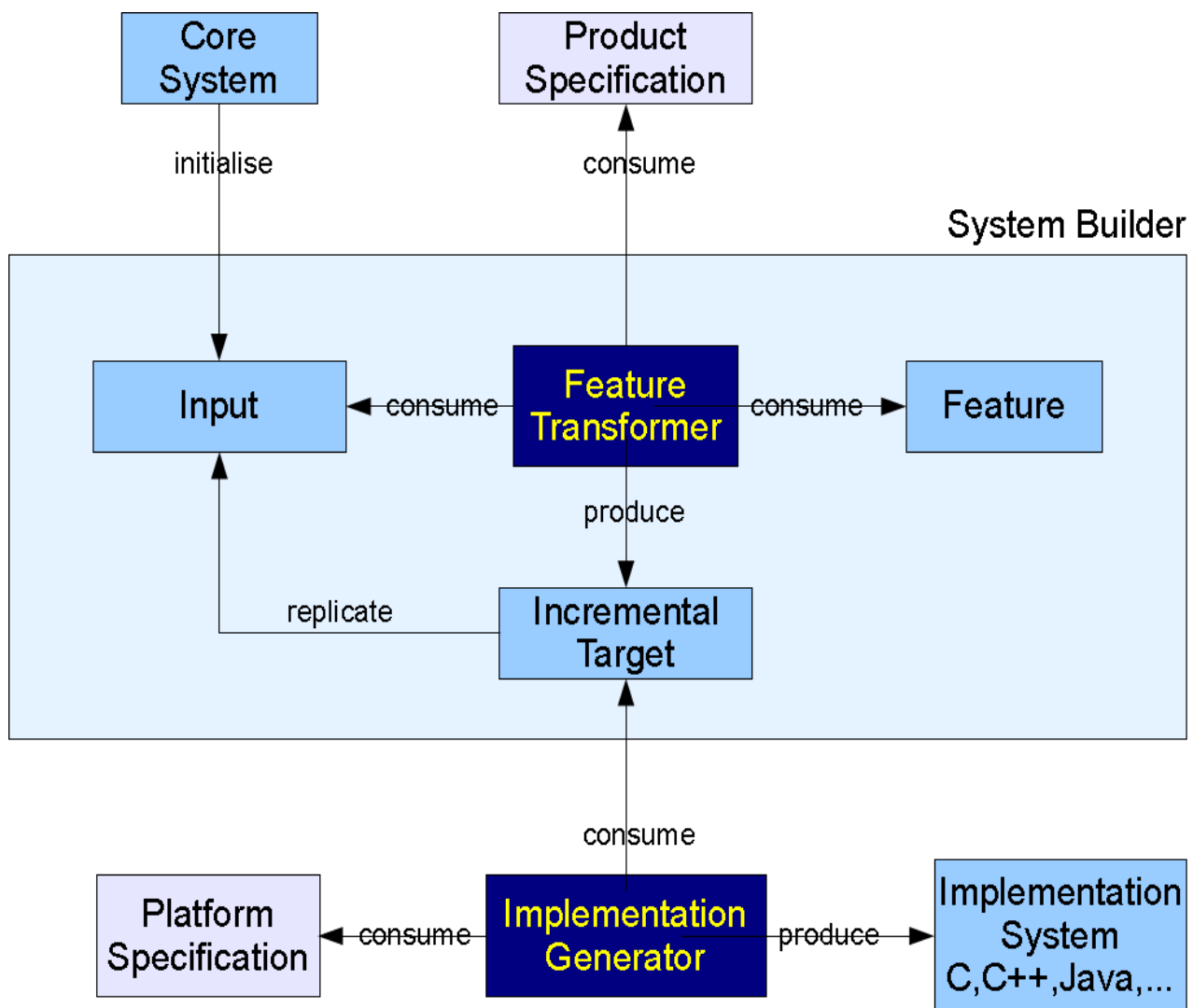
Specification and used by the generator to generate the complete source code. After generating this implementation model, in step 4 it can be built, deployed and tested using standard, mainstream, tools. Legacy source code may be included in the build process. Additionally, test cases and documentation may be generated for the composed features and test automation used during e.g. nightly builds.

The execution of the 7 steps above is fully automatic. No manual intervention is necessary.

Reduction in complexity is achieved by developing Core and Feature models in isolation from each other. System complexity is further reduced by defining interaction patterns using feature profile application specifications which are used by the transformations during the incremental target creation process.

The enabler for this kind of technology is a concept: "super-loose" component coupling.

### ***Simplified process overview***



*Figure 1: Main concepts and their relationships*

## ***Implementation Generator and Source Code***

The target models are fully executable meaning that it is not necessary to touch them by hand anymore. Thus, from a single set of target models, we forward engineer complete source code implementations, out of the box, for various programming languages e.g. C, C++ and Java. We allow for reuse of legacy source code by means of interface models.

Generated C and C++ source code is highly optimised at the expert programmer's level, making it fit for real-time, high performance, mission critical systems.

We also generate full source code artefacts for J2SE and J2EE platforms including Java Server Faces XHTML and Managed Bean source code, packaging and deploying on e.g. the Glassfish open source application server or application server of your choice.

Furthermore, we generate Web Service artefacts such as XSD schema, WSDL artefacts and SQL statements to populate various databases.

Upon request, we would be happy to extend the implementation generator for individual business domains, languages and platforms as needed.

## ***Keywords***

Emerging systems, super-loosely coupled components, isolated modification, managed components, model weaving, feature composition, multiple simultaneous model processing, executable models, analysis models, profile overlays.

## ***More Applications based on ESP coming up***

Finally, we see other productivity tools emerging from our concepts in addition to the mentioned application of system building from a set super-loosely coupled components:

1. Extreme reuse: enable feature-to-feature mapping as well as feature composition from a set of subfeatures using the same concepts as given above.
2. Evolution of reference architectures and variability management: in this scenario, developers implement only core business components. System builder embeds the components automatically into the predefined reference architecture. Change of reference architecture does not impact components but rather just the glue models.
3. Realisation of non-invasive modification mechanisms during software development
4. Front-end integration: a front-end application helps the architect develop / apply detailed profiles based on domain specific concepts. The system builder abstracts glue models from concrete models.

## ***Contact***

Contact us for a demonstration and get a hands-on trial version of the system.