

Domain-Frontier Architecture

Aleksandar Orlic, Founder and Chief Consultant of Craftware Ltda.

Abstract:

Enterprise architecture is a multidimensional puzzle. Domain complexity, business rules, information technology diversity, existing system structure, overall scale and abstraction, number of stakeholders, everlasting changes are some of those dimensions.

In order to resolve this puzzle and optimize enterprise operations, business and software architects must work closely together. In this work, an enterprise architecture concept, called Domain-Frontier Architecture, is proposed as a possible approach to a modern enterprise architecture modeling and construction.

Key words:

Business enterprise, process simulation, business architecture, integration, software architecture, BPMN, UML, Executable UML, MDA

Author Biography:

Aleksandar Orlic holds a M.Sc. in Computer Science from the University of Belgrade, Serbia and has over 15 years of experience in enterprise architecture consultancy and software development. His professional interests are related to executable model based tools and methodologies, as a support to enterprise architecture. He lectured at OMG's conference on Model Driven Architecture, in Orlando, 2004.

Aleksandar is founder and chief consultant of Craftware Ltda, consultancy firm based in Santiago, Chile (www.craftware.net). He currently lives in Berlin, Germany, and works as Project Manager in Digitote GmbH.

Domain-Frontier Architecture

Business and Software Architecture

In the heart of every modern enterprise is a software system. Business performance highly depends on how it is supported by software.

During the last decades and over several generations of technology and enterprise methods, enterprise architecture has evolved into a large multidimensional puzzle. Domain complexity, business rules, information technology diversity, existing system structure, overall scale and abstraction, number of stakeholders, everlasting changes are some of these dimensions.

*A major problem for most of the modern enterprise architecture methods is the lack of an efficient way to **separate** all those dimensions, to analyze and understand them, and finally be able to **integrate** them back into a consistent whole.*

This work introduces a conceptual framework, supported by a method and a tool, used by our company in our enterprise consultancy projects. It is called **Domain-Frontier Architecture**, and is designed to support "enterprise construction process", providing:

- a road-map to build a two-fold integrated model
 - business as a whole
 - business-software symbiosis
- a set of software tools for building those models in fully executable form

In other words, Domain-Frontier Architecture provides a way to *integrate by separation*.

Domain-Frontier Structure Map

Simple scheme, shown on Fig. 1, is the starting point of Domain-Frontier Architecture. It identifies the minimal set of elements needed to describe a business enterprise including a software system which drives it (or should drive it).

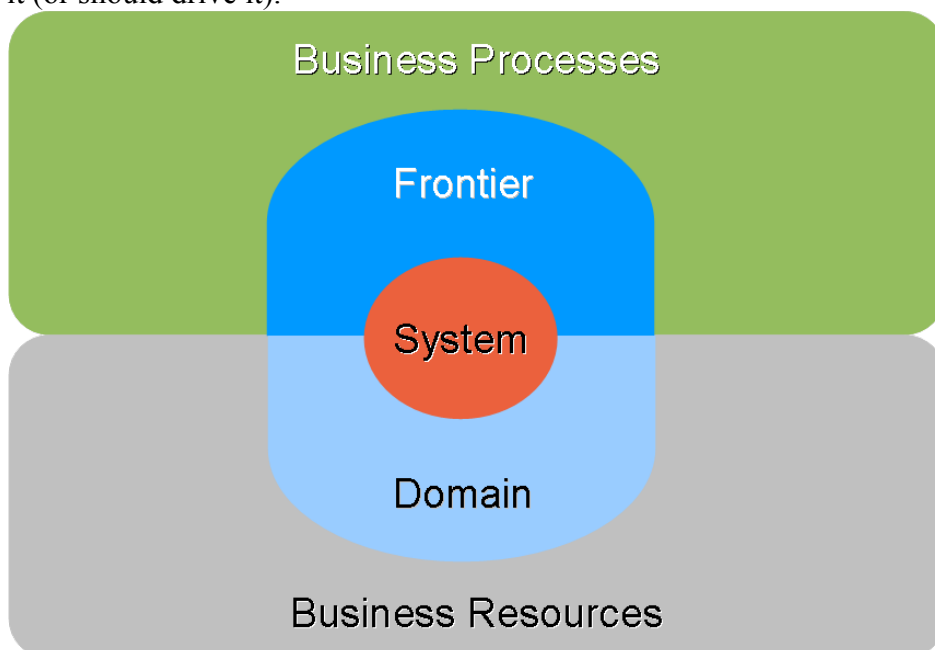


Fig. 1 – Domain-Frontier Structure Map

Business enterprise is seen through two views. **Processes** consume, produce, transform **resources** or are executed by them. Examples of business resources are *Invoice, Purchase Order, Credit Card, Contract, Money*, but also *Supervisor, Development Area, CEO*.

Domain-Frontier is a technology independent abstraction of a software system, a functional prototype of the software implementation. **Domain** is an abstraction of the complex structure of business resources and their relationships. **Frontier** is an abstraction of the user-interaction part of the software, that is, a link between the business process and the underlying technology.

System is the actual software solution, which drives the enterprise. Its details are out of focus of Domain-Frontier Architecture. However, as it will be seen later in this work, Domain-Frontier model is a working prototype of the software system and its automatic transformation is a matter of time. There are plenty of methodologies focused on this technique, such as MDA, Executable UML, DDD, etc.

Domain-Frontier Architecture defines an abstract conceptual framework to help Enterprise Architects focus on their work, freeing their creativity and minimizing restrictions and dependencies, especially on implementation technologies. They can freely choose their favorite notation and modeling technique for each view. In Tab. 1 are shown some of the possibilities. The traceabilities between the elements of the different views are left apart for simplicity.

Abstraction Layer	View	Representation method
Business Enterprise	Business processes	BPMN, UML Eriksson-Penker, UML activity diagrams
	Business resources	UML class diagrams, ER-models, organization charts
Domain-Frontier	Frontier	UML use cases, Interaction maps
	Domain	UML class diagrams, UML state diagrams
Software Enterprise		

Tab. 1 – Domain-Frontier Architecture implementation

Domain-Frontier Process

Domain-Frontier process is top-down, executable, model based, testing-rich and customer-oriented methodology.

Top-down because it starts from the highly abstract enterprise view (business mission and vision, strategy and organization chart), gradually adding more and more details to describe processes, resources, and later concepts, business-system interactions, getting down to the software implementation.

Executable models, easy to **test** are produced in each project phase. More details about this feature will be exposed in the next section.

Thanks to executable models, **non-technical customers** can easily check the ongoing progress, test models and give their feed-back from the very beginning of the project. This leads to extremely low defect-rate and less re-work at the end of the project.

On Fig 2 is shown the Domain-Frontier Process. Its phases follow the structure of the Domain-Frontier Map on Fig. 1. Each phase has a clear output-milestone, an executable and testable model which is carefully and deeply verified before diving into the next phase. Just like the structure map,

the process is abstract and flexible, so Enterprise Teams are free to use their own methodologies within each phase. Waterfall, incremental, iterative, spiral, agile or completely custom-made are welcome.

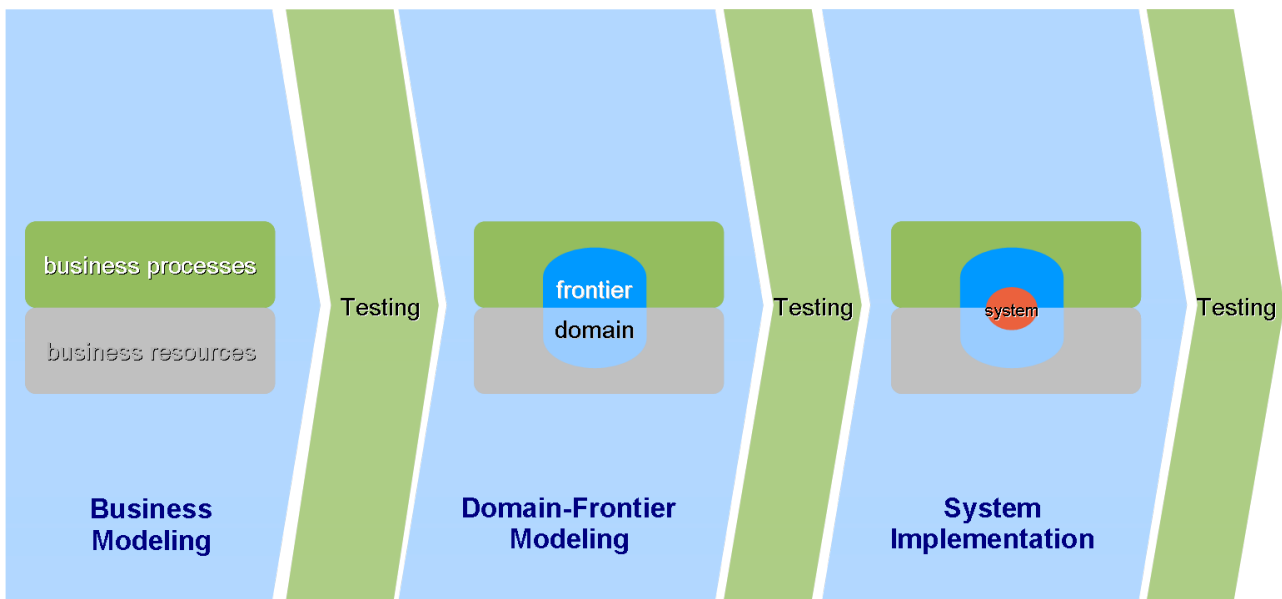


Fig. 2 – Domain-Frontier Process

Domain-Frontier Tools

Lack of formal modeling tools are one of the biggest challenges in modern Enterprise Architecture. Rapidly growing problem's scale has made impossible to efficiently develop, check and maintain textual documents and even informal models. These artifacts are too ambiguous, unclear and are becoming useless. On the other side, it is not acceptable to wait for fully implemented system to be able to test an enterprise operation.

Domain-Frontier Architecture follows model-driven approach. Executable models are easy to check and maintain and therefore only valid outputs.

We have developed a software tool to create executable prototype of the **Domain-Frontier model**. It receives standard UML diagrams as input and produces executable output. This tool fully supports the main phase in the process (as shown on Fig. 3), by building a platform independent bridge between the enterprise business and enterprise software solution.

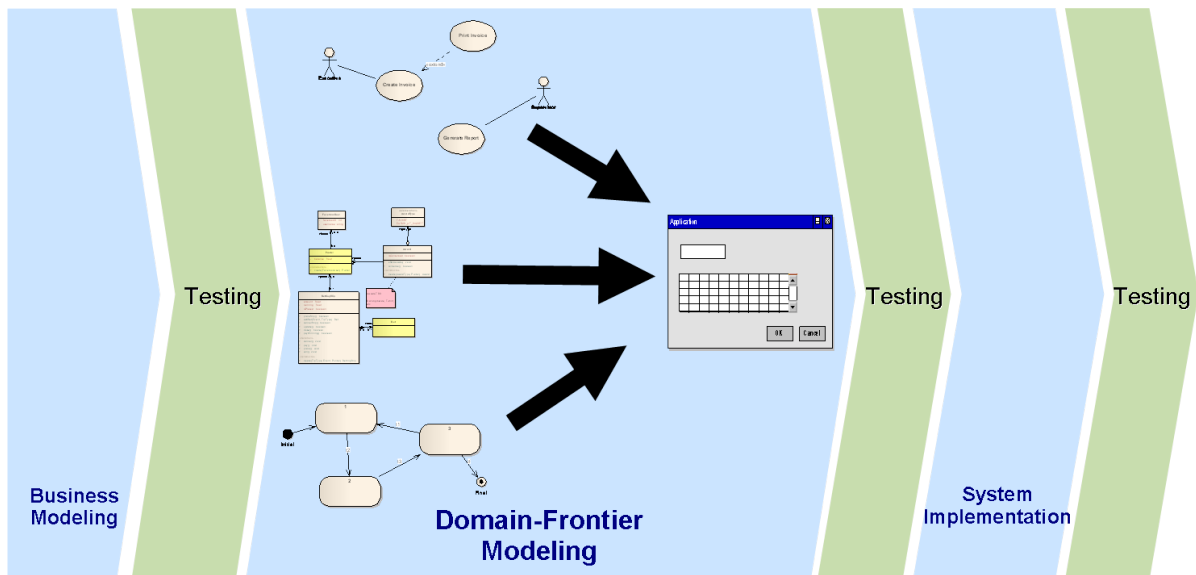


Fig. 3 – Domain-Frontier Tools

In the near future, we plan to extend our tool to cover the rest of the process.

In the **Business Modeling** stage, the processes are first simulated with the real operational data and then optimized. At this time, there are several tools on the market, able to execute BPMN diagrams.

In the **System Implementation** stage, our aim is to achieve near 100% of the code generated from the Domain-Frontier Model. Meanwhile, there is plenty of existing tools for code generation.

Domain-Frontier in Practice

In Craftware, we have employed this method in several projects by now, typically executing the first 2 phases. Our delivery is an executable and tested **integrated system prototype**, built on previously simulated and tested business process model.

This delivery is afterwards taken by a software development provider, and typically broken in several smaller units, to be gradually transformed into integrated information system.

The whole process is resumed in the Tab. 2, showing the main outputs and progress across the project phases.

Project phase	Model progress			
	Business process model	Conceptual Model	Use case model	Code Model
Business Modeling				
Domain-Frontier Modeling				
System Implementation				

Tab. 2 – Domain-Frontier Architecture implementation

Business modeling phase. In this phase the intensive research about the enterprise is performed, the process-structure is discovered; the resources are precisely scanned and classified. We apply top-down, iterative approach, and work intensively with enterprise participants in order to describe

their organization.

In the early iterations our aim is to describe the enterprise structure, objectives and strategy (high level conceptual model). In the mid-iterations we trace those to processes and resources (BPMN for processes and UML for resources). A real-world example of high-level BPMN process representation is given on the Fig. 4.

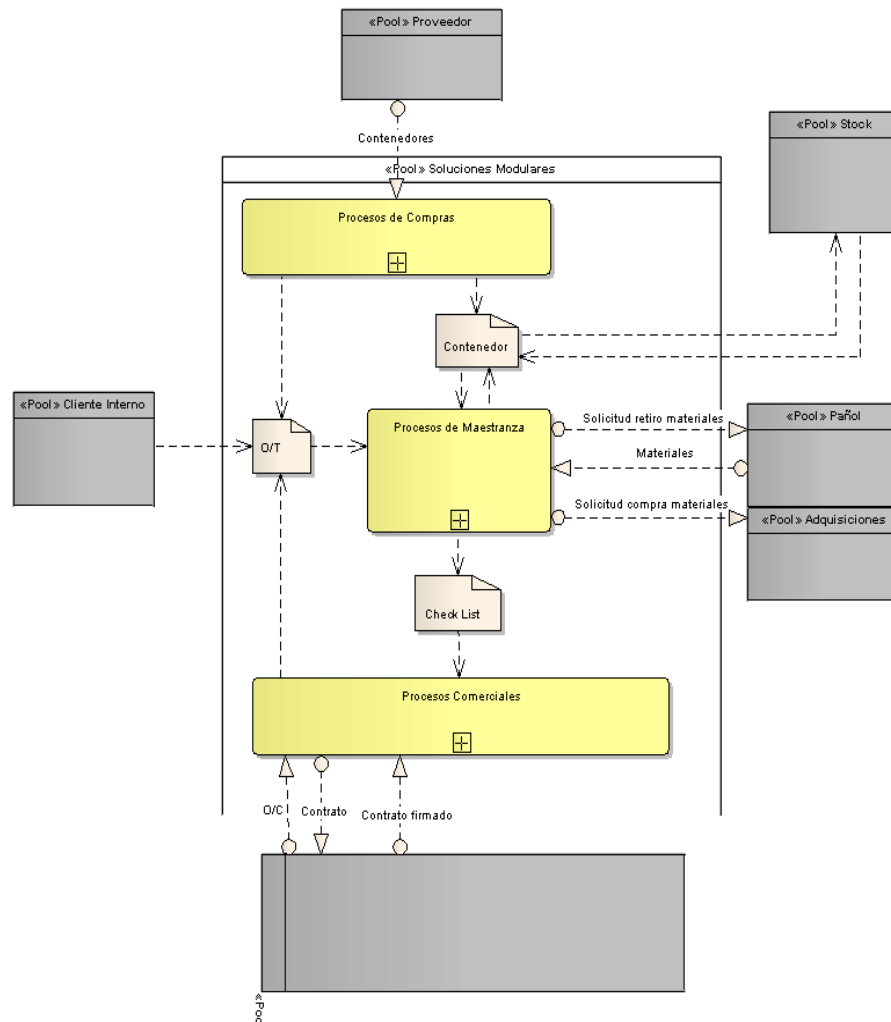


Fig. 4 – High-level enterprise process model

In the posterior iterations, the high-level processes are decomposed to the detail-level process specifications (BPMN) where the output is simulated and verified with real-world loads. In this phase the enterprise processes' optimization is done, to reconfigure it and prepare for the system implementation. Fig. 5 shows a real-world example of the detailed process description ready for simulation.

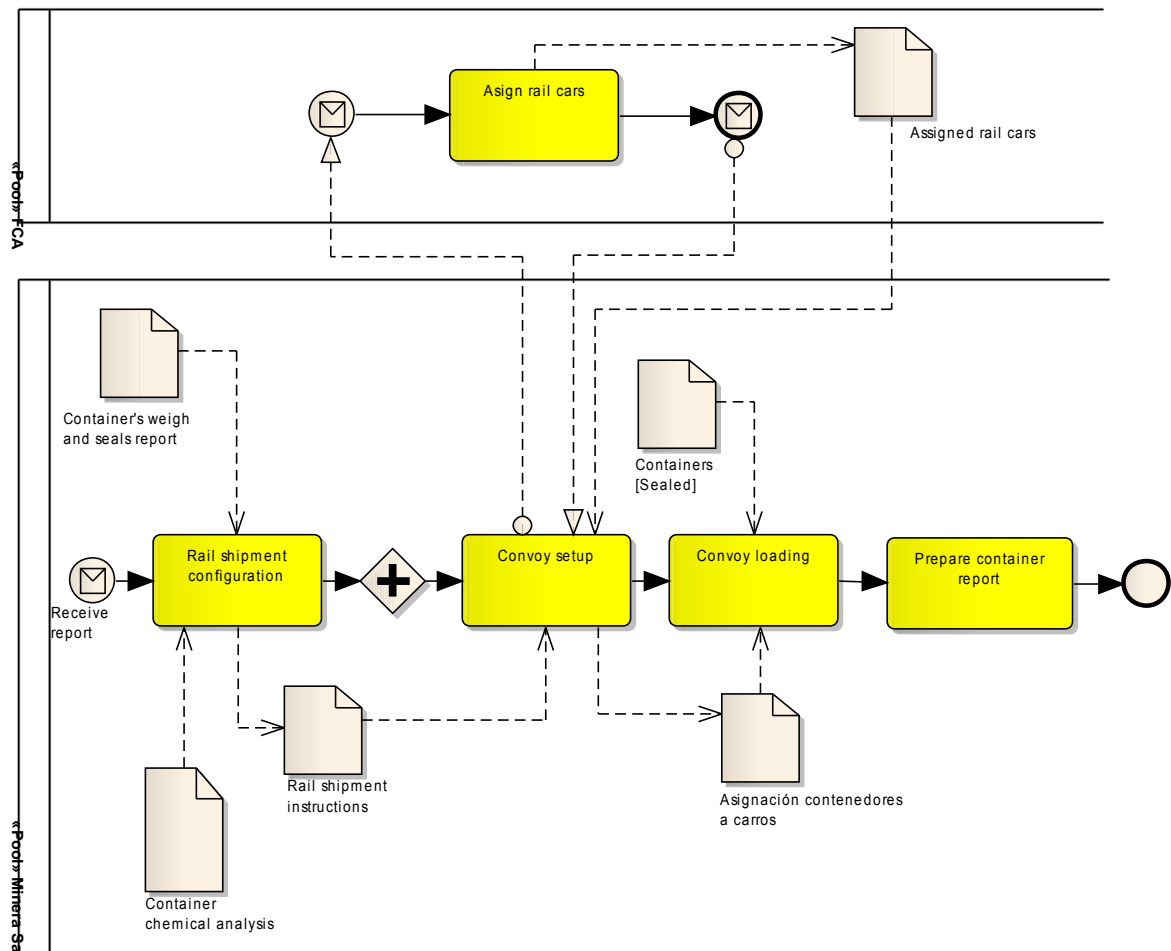


Fig. 5 – Detailed (simulation-ready) enterprise process model

Domain-Frontier phase. This is the crucial part of the method, in which the conceptual model is refined and detailed, as well as simulated in our tool Enterprise Analyst. This model is executed to assure its precision and correction. This is practically the information system prototype testing. Before even thinking about its technology characteristics and far before any implementation efforts, we are capable to test the system and avoid expensive defects, so typical for the software development projects.

This phase is planned and organized around the iterations, similar to the first phase.

The final **system implementation phase** is realized on the base of the Domain-Frontier prototype. The system architects can fully concentrate on the technology and other non functional requisites, freed of any functional complexity, which was tested earlier. There are a lot of tools for automatic code generation to be chosen from, or the manual programming can take place.

Domain-Frontier model may be implemented as a whole, or part by part. The integrated model is technology-free, so there is no risks of any kind of obsolesce.

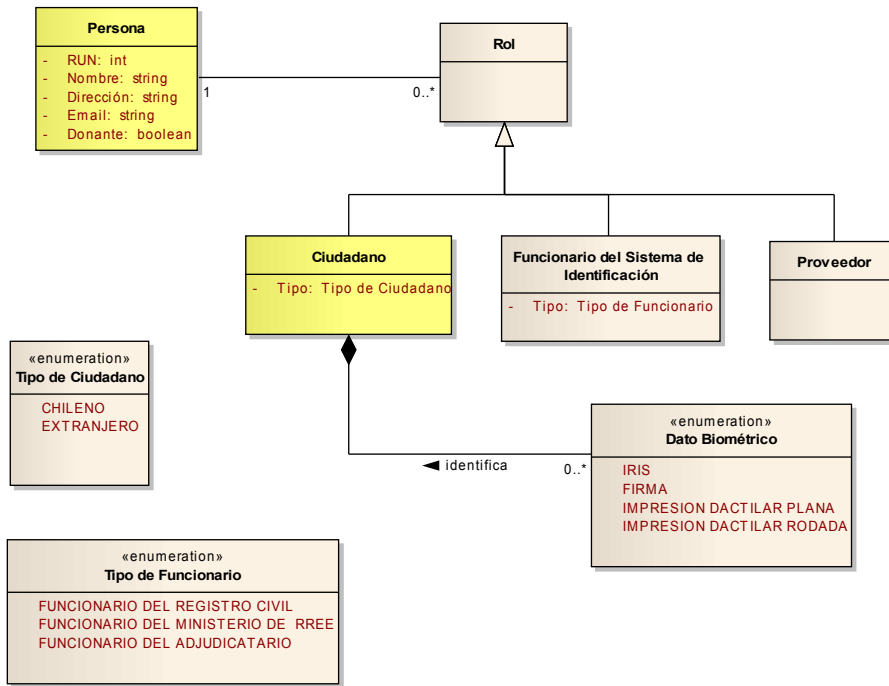


Fig. 6 – Detailed (executable) enterprise conceptual model

* * * * *

About the presentation of this work

If accepted, the 45 minutes presentation of this work will be complemented by a real example of an Enterprise Model constructed using Domain-Frontier Architecture method and tool, so the participant can perceive the benefits of the method.