PMR 5020

Modelagem do Projeto de Sistemas

Aula 12: From DSLs to Language Systems

Prof. José Reinaldo Silva
reinaldo@poli.usp.br

Objetivos do Curso

A proposta básica do curso era justamente para diferenciar o projeto de sistemas comparado ao processo de projeto mais intuitivo e familiar que é direcionado a produto. Isto abre espaço para inserção de serviços ou produto-serviços. Outro aspecto importante é a real importância do formalismo, especialmente para projetos de automação.

O projeto orientado a "produtos"

O projeto orientado a "produtos" é eminentemente funcional, ainda quando o resultado não é de fato um produto mas um software. O que importa de fato não é a natureza do artefato final mas a constituição de um processo de projeto "product-oriented" que originou de fato termos como a "fábrica de software". Portanto trata-se de dar "vida" a um objeto (ou a um artefato conceitualmente tratado como tal) cuja existência própria gera uma mudança no ambiente por sua capacidade de própria de interferência.

O projeto de sistemas é feito top-down, e “wide oriented”, isto é, evita-se aprofundar qualquer que seja o aspecto antes de verificar a sua relação com os demais aspectos do mesmo “nível" ou de níveis próximos.

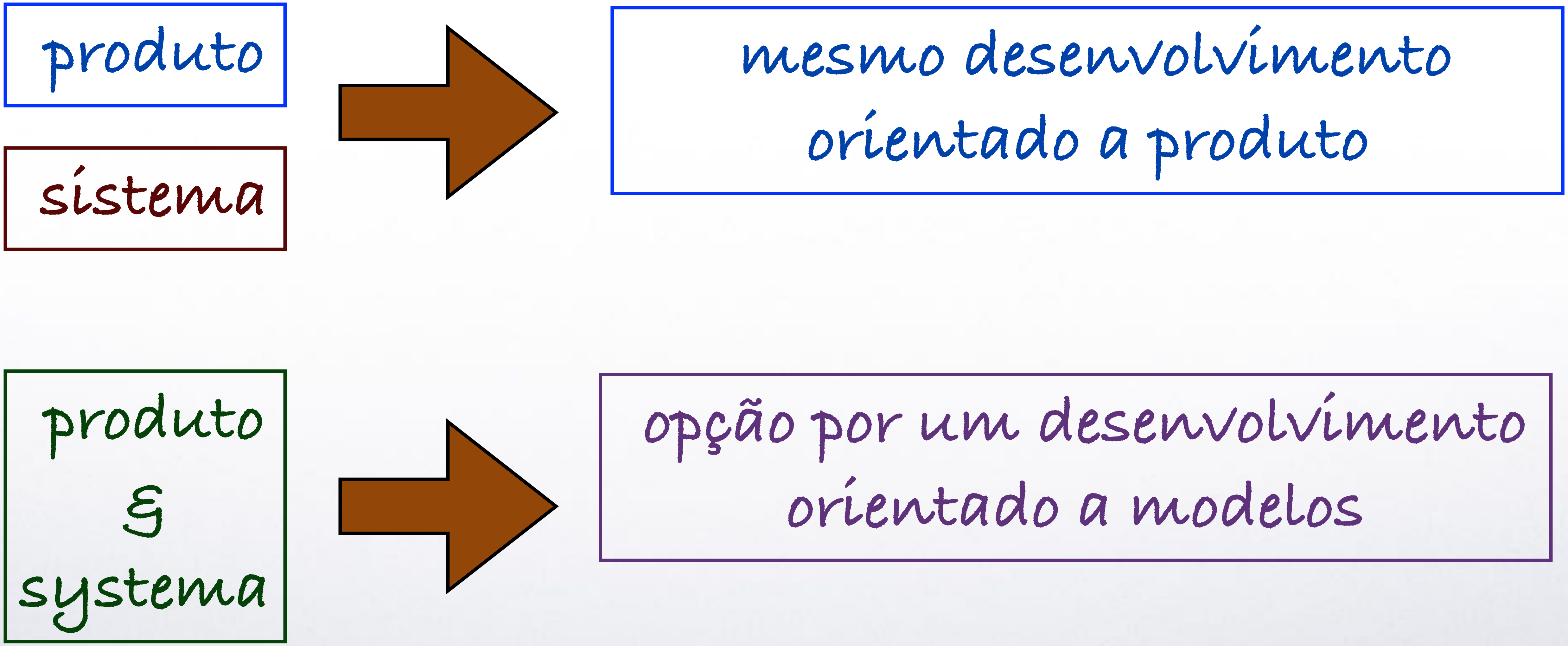
O Conceito de Sistema e de "Modelo"

A mudança de paradigma de “produto" para sistema também nos leva a pensar em *modelos* ao invés de protótipos e assim nos desligar de uma funcionalidade intrínseca para buscar uma harmonia entre elementos distribuídos (objetos) capaz de produzir de forma flexível e adaptável comportamentos distintos, sejam estes resultado da contribuição direta dos seus componentes ou resultado da ação conjunta. Portanto também não se pode tratar este novo elemento como uma junção de pequenos produtos mas sim como sistemas de sistemas.

A dicotomia hardware/software e produto/sistema

Historicamente hardware e software tiveram o seu desenvolvimento separado e ao mesmo tempo "integrado", tipicamente uma abordagem produto-dominante. A automação como é vista hoje não poderia se desenvolver restrita a este paradigma.

A mudança de paradigma



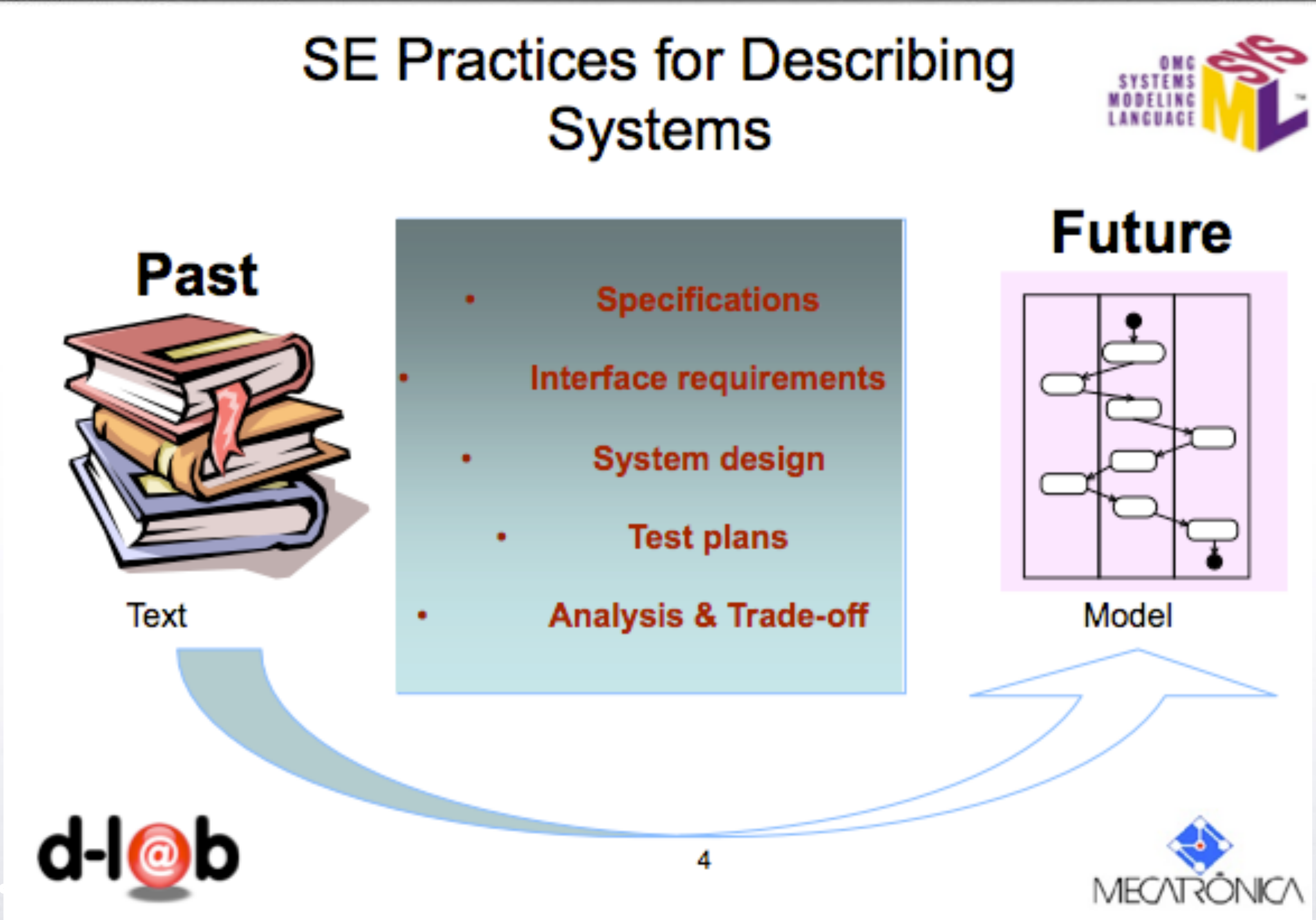
produto

sistema

mesmo desenvolvimento orientado a produto

produto & systema

opção por um desenvolvimento orientado a modelos



SE Practices for Describing Systems

Past

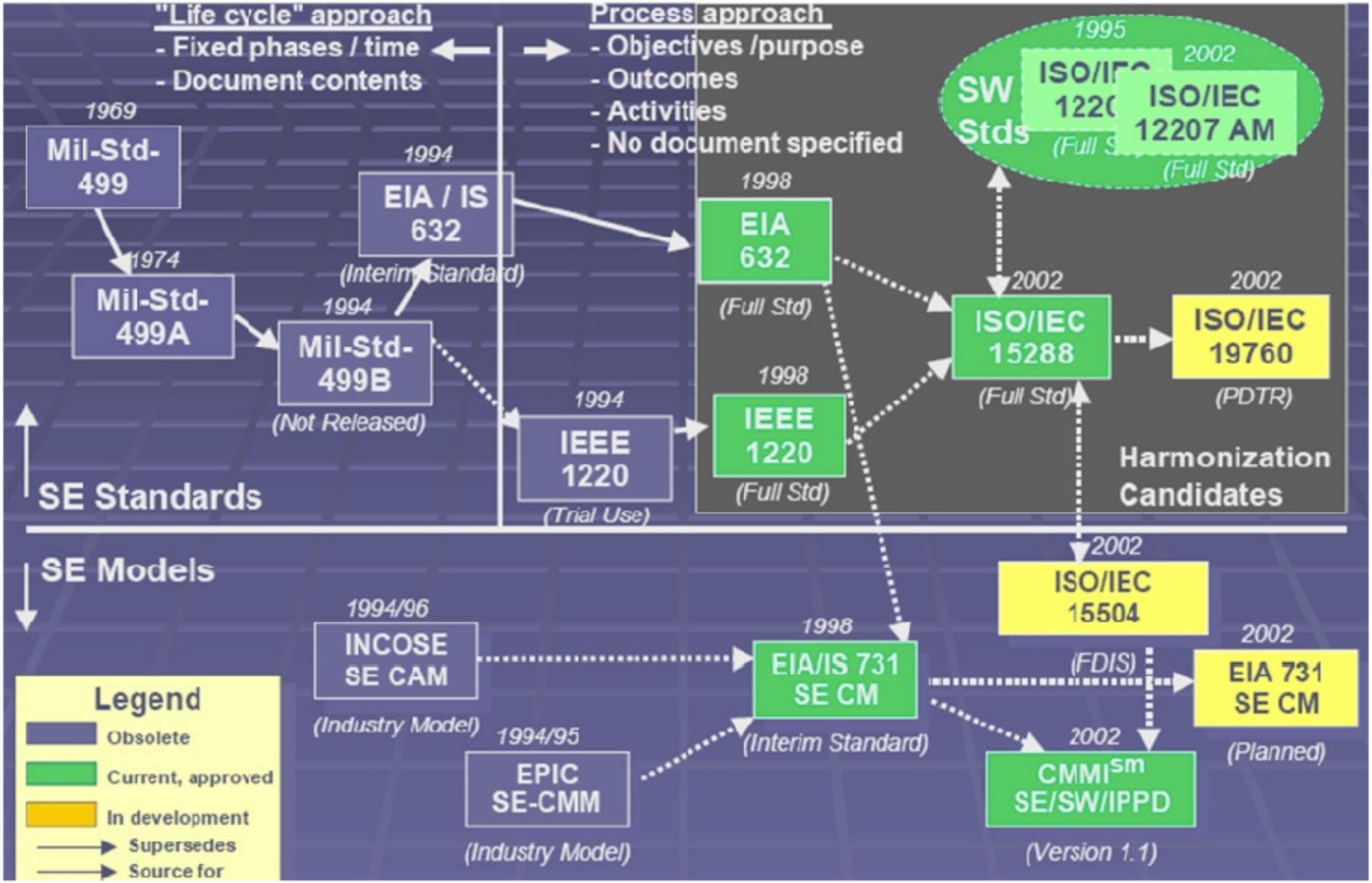
Text

- **Specifications**
- **Interface requirements**
- **System design**
- **Test plans**
- **Analysis & Trade-off**

Future

Model

4



"Life cycle" approach
- Fixed phases / time
- Document contents

Process approach
- Objectives /purpose
- Outcomes
- Activities
- No document specified

Fundamentação matemática para o MBSE

A. Wayne Waymore   (T3SD)

1927-2011

Wymore, A. Wayne, A Mathematical Theory of Systems Engineering: The Elements , John Wiley & Sons: New York, NY, 1967.

Wymore, A. Wayne, Model-Based Systems Engineering , CRC Press, Inc.: Boca Raton, FL, 1993.

Wymore, A. Wayne, “Contributions to the Mathematical Foundations of Systems Science and Systems Engineering,” Systems Movement: Autobiographical Retrospectives, The University of Arizona, Tucson, AZ, 2004.

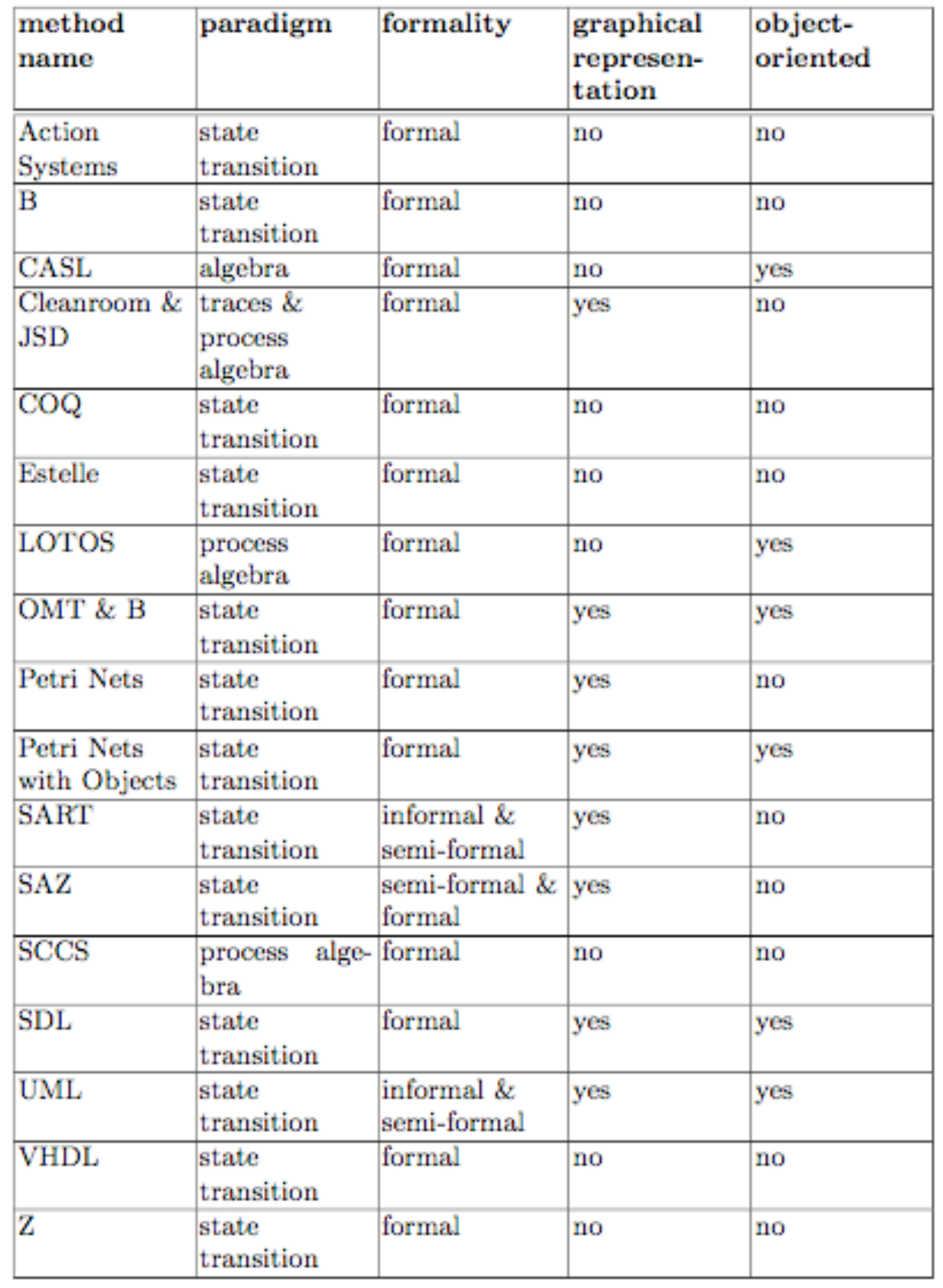
The basis of system theory and systems engineering, according to Wymore, is modeling, and the concept of "system" is human interpretation via senses (i.e., a mental model). A system model is a description that separates the perceived universe into two parts, the part "inside" the system and the part "outside" the system. From the "outside" the system receives inputs. To the "outside" the system delivers outputs. The "inside" of the system is described, initially, as states. The state of the system at any time is a function of its state at a previous time and the intervening inputs (including "noise"). System designs are system models. When modelers describe some part of reality as a "real" system, it means that their system mode "adequately" represents the reality. For the purpose of accurate communication, mathematical definitions of various classes of system models are postulated.

Mas o problema abordado intermitentemente neste curso é como vamos implementar um formalismo que vá além das fronteiras acadêmicas e que possa mostrar sua efetividade também no mercado. Aí é que se coloca o problema de “esconder" o formalismo e/ou acelerar a interpretação de sentenças formais dentro de um dado domínio de aplicação.

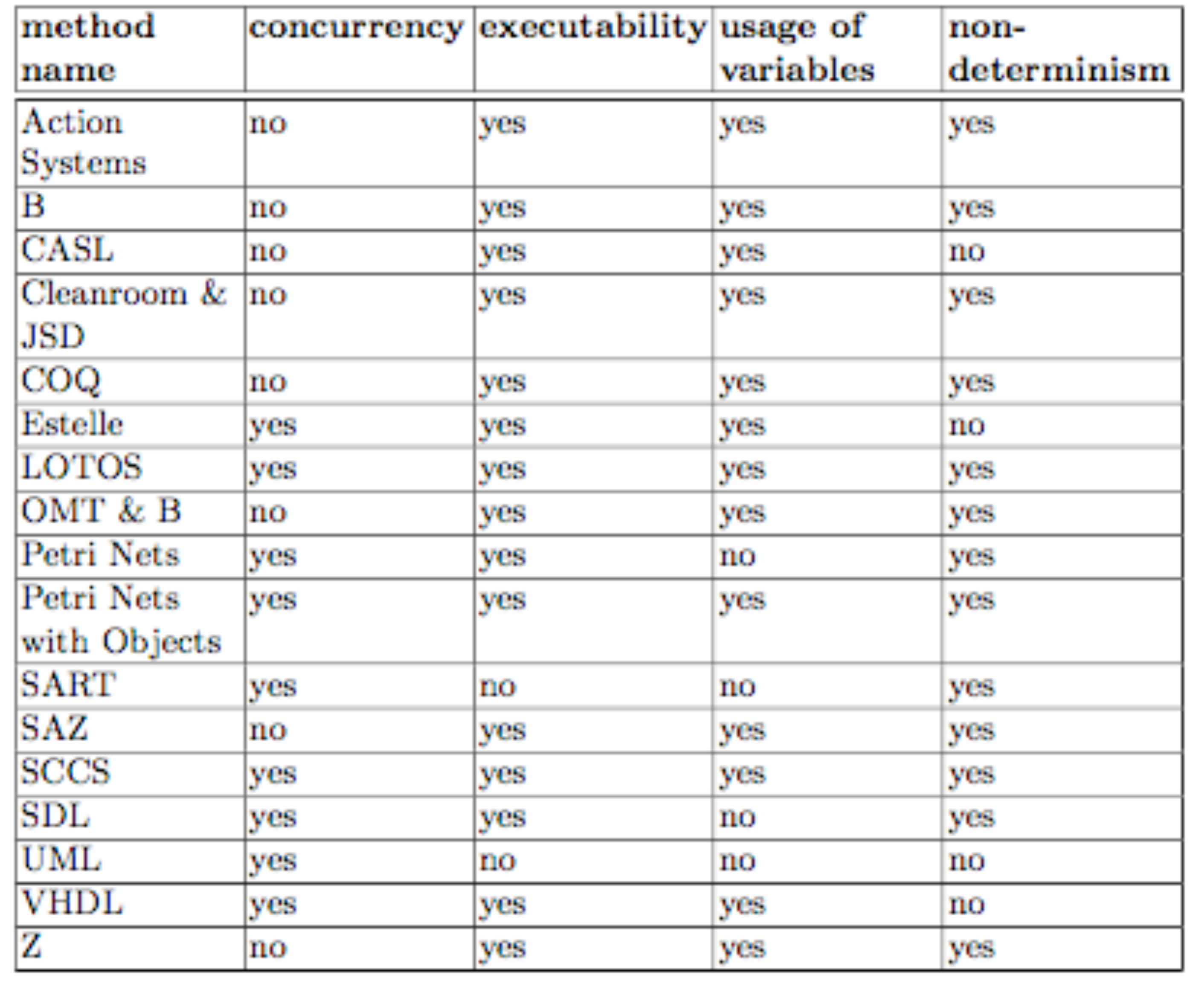
Conceptual Model of the Globalization for Domain-Specific Languages

Tony Clark(1), Mark van den Brand(2), Benoit Combemale(3), and Bernhard Rumpe(4)
1 Middlesex University, London, UK
2 TU Eindhoven, Eindhoven, Netherlands
3 University of Rennes and Inria, Rennes, France
benoit.combemale@irisa.fr
4 Software Engineering, RWTH Aachen, Aachen, Germany

Abstract. Domain Specific Languages (DSL) have received some prominence recently. Designing a DSL and all their tools is still cumbersome and lots of work. Engineering of DSLs is still at infancy, not even the terms have been coined and agreed on. In particular globalization and all its consequences need to be precisely defined and discussed. This chapter provides a definition of the relevant terms and relates them, such that a conceptual model emerges. The authors think that this clarification of terms and the meaning will foster the field of efficient DSL definition and evolution in the future.



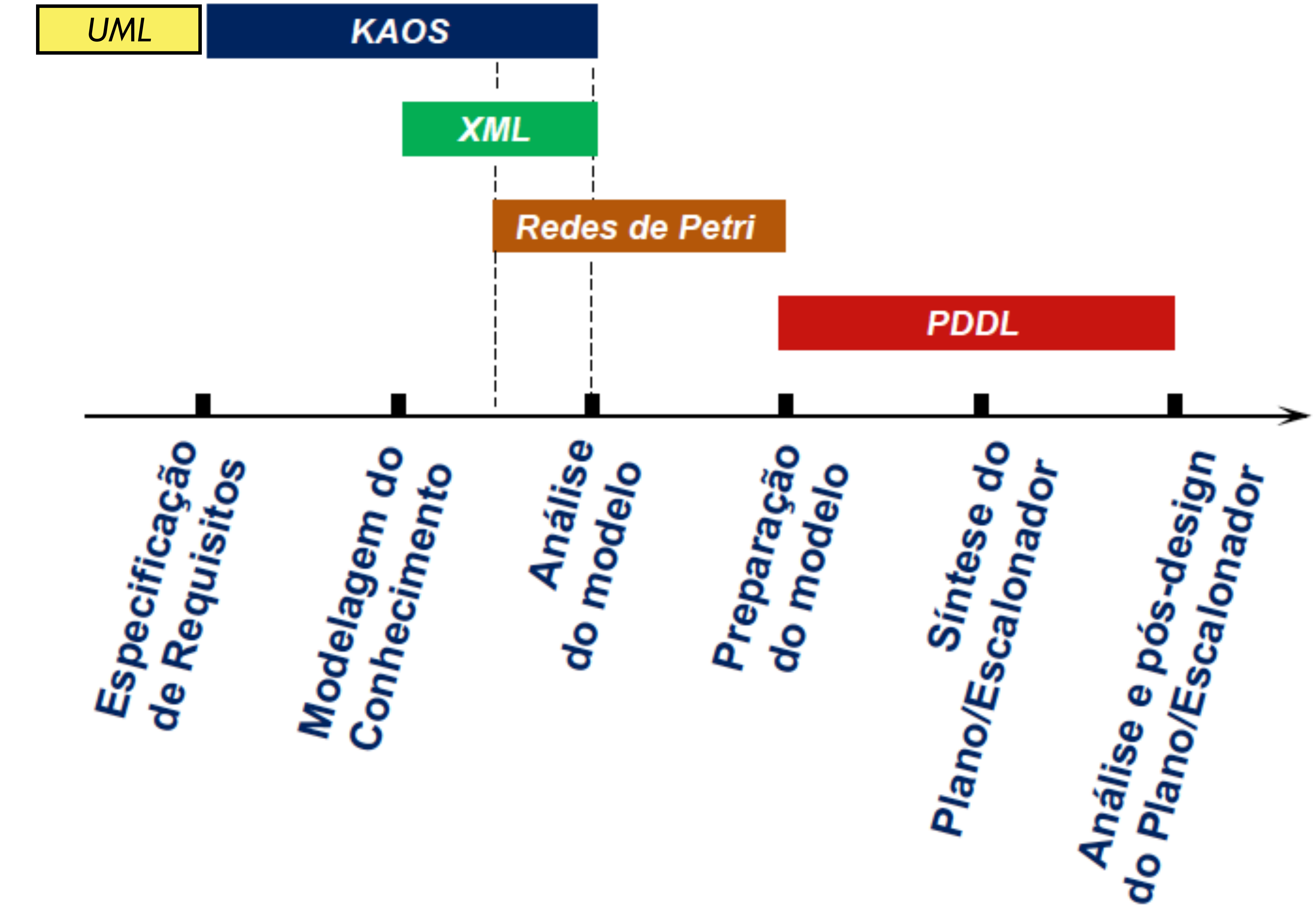
method name	paradigm	formality	graphical represen-tation	object-oriented
Action Systems	state transition	formal	no	no
B	state transition	formal	no	no
CASL	algebra	formal	no	yes
Cleanroom & JSD	traces & process algebra	formal	yes	no
COQ	state transition	formal	no	no
Estelle	state transition	formal	no	no
LOTOS	process algebra	formal	no	yes
OMT & B	state transition	formal	yes	yes
Petri Nets	state transition	formal	yes	no
Petri Nets with Objects	state transition	formal	yes	yes
SART	state transition	informal & semi-formal	yes	no
SAZ	state transition	semi-formal & formal	yes	no
SCCS	process algebra	formal	no	no
SDL	state transition	formal	yes	yes
UML	state transition	informal & semi-formal	yes	yes
VHDL	state transition	formal	no	no
Z	state transition	formal	no	no

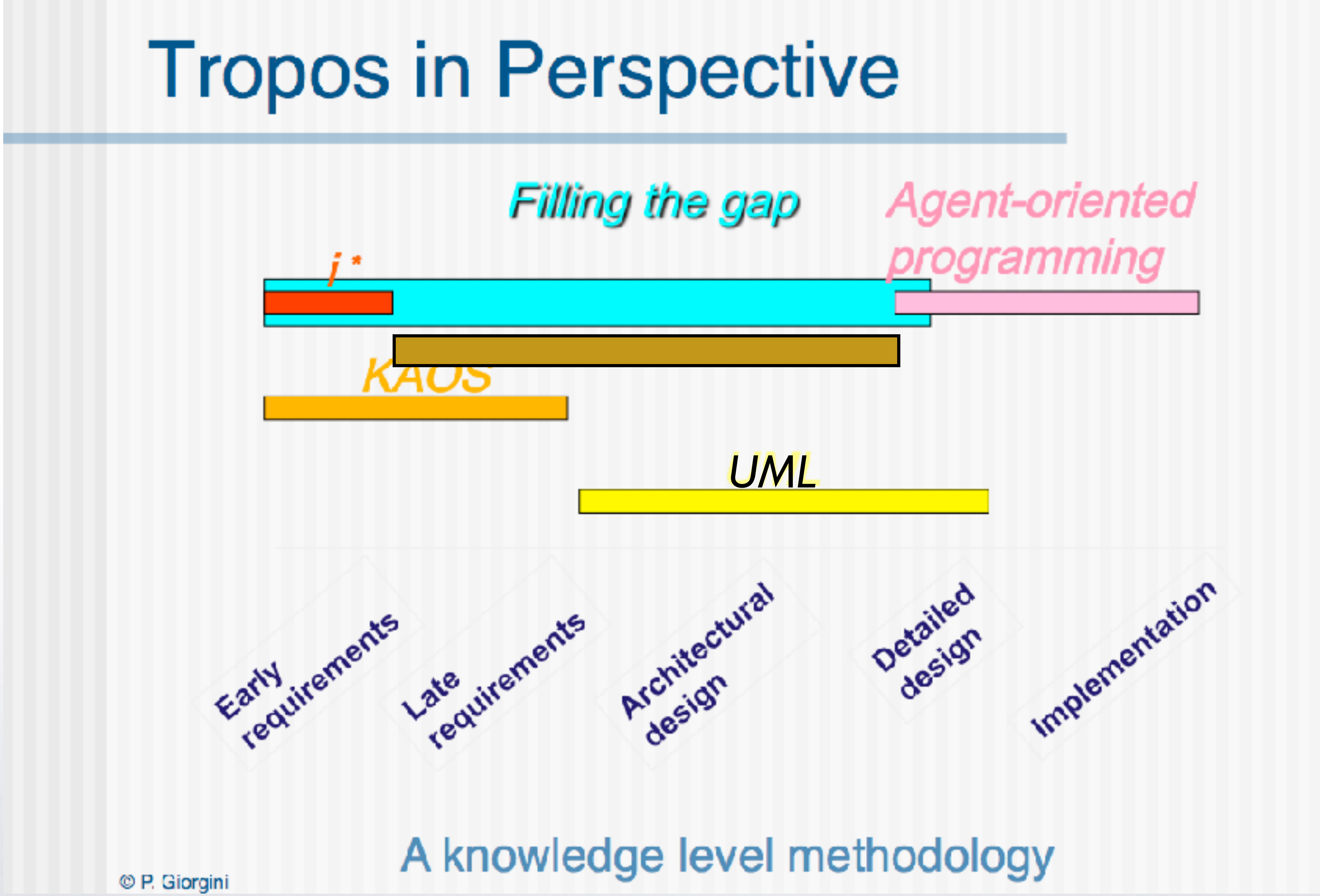


method name	concurrency	executability	usage of variables	non-determinism
Action Systems	no	yes	yes	yes
B	no	yes	yes	yes
CASL	no	yes	yes	no
Cleanroom & JSD	no	yes	yes	yes
COQ	no	yes	yes	yes
Estelle	yes	yes	yes	no
LOTOS	yes	yes	yes	yes
OMT & B	no	yes	yes	yes
Petri Nets	yes	yes	no	yes
Petri Nets with Objects	yes	yes	yes	yes
SART	yes	no	no	yes
SAZ	no	yes	yes	yes
SCCS	yes	yes	yes	yes
SDL	yes	yes	no	yes
UML	yes	no	no	no
VHDL	yes	yes	yes	no
Z	no	yes	yes	yes

?

Exemplos “práticos” de DSLs?


UML
KAOS
XML
Redes de Petri
PDDL
Especificação de Requisitos
Modelagem do Conhecimento
Análise do modelo
Preparação do modelo
Síntese do Plano/Escalonador
Análise e pós-design do Plano/Escalonador

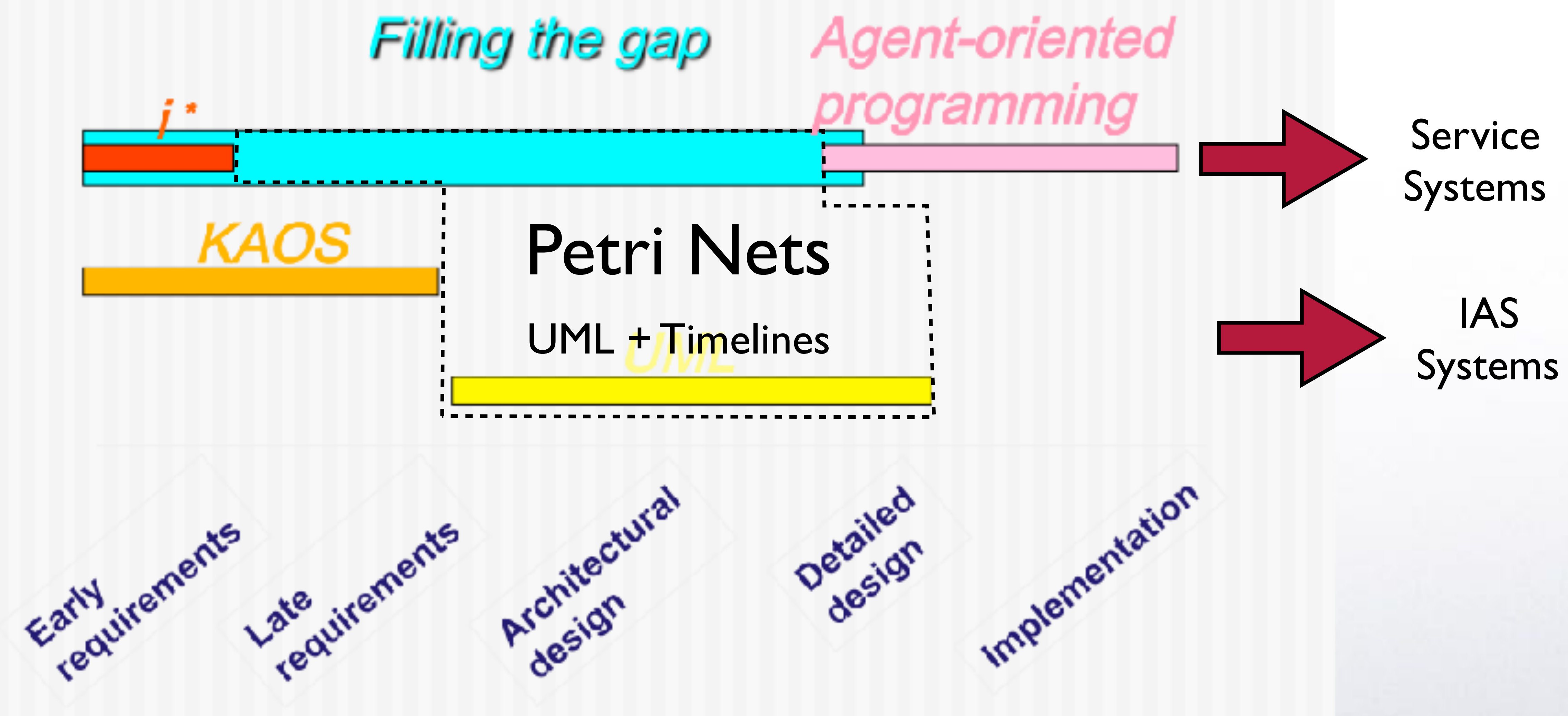


Tropos in Perspective

A knowledge level methodology

© P. Giorgini

Tropos in Perspective



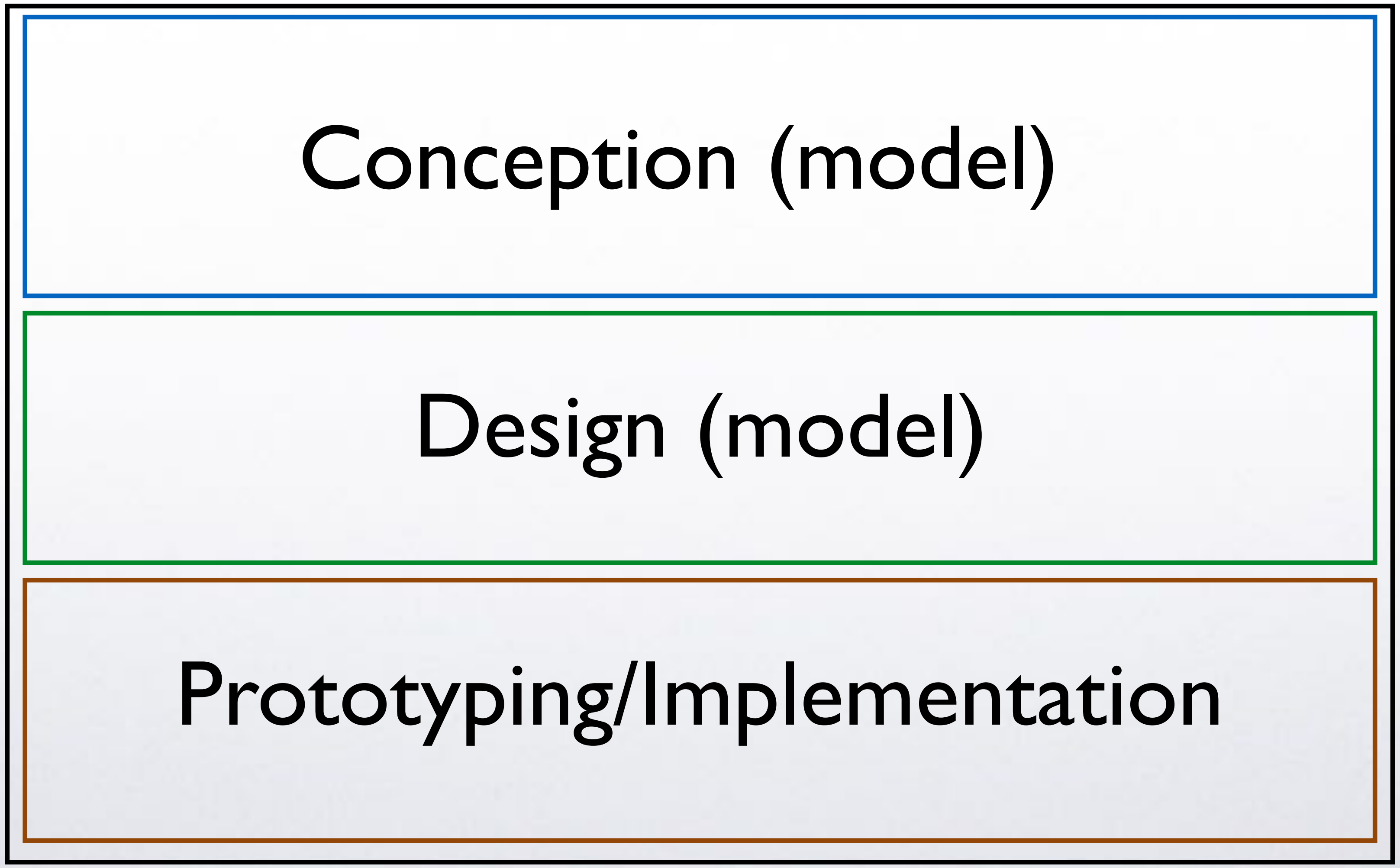
A knowledge level methodology

© P. Giorgini

Key points for using DSLs

(Systems) Design Reusability

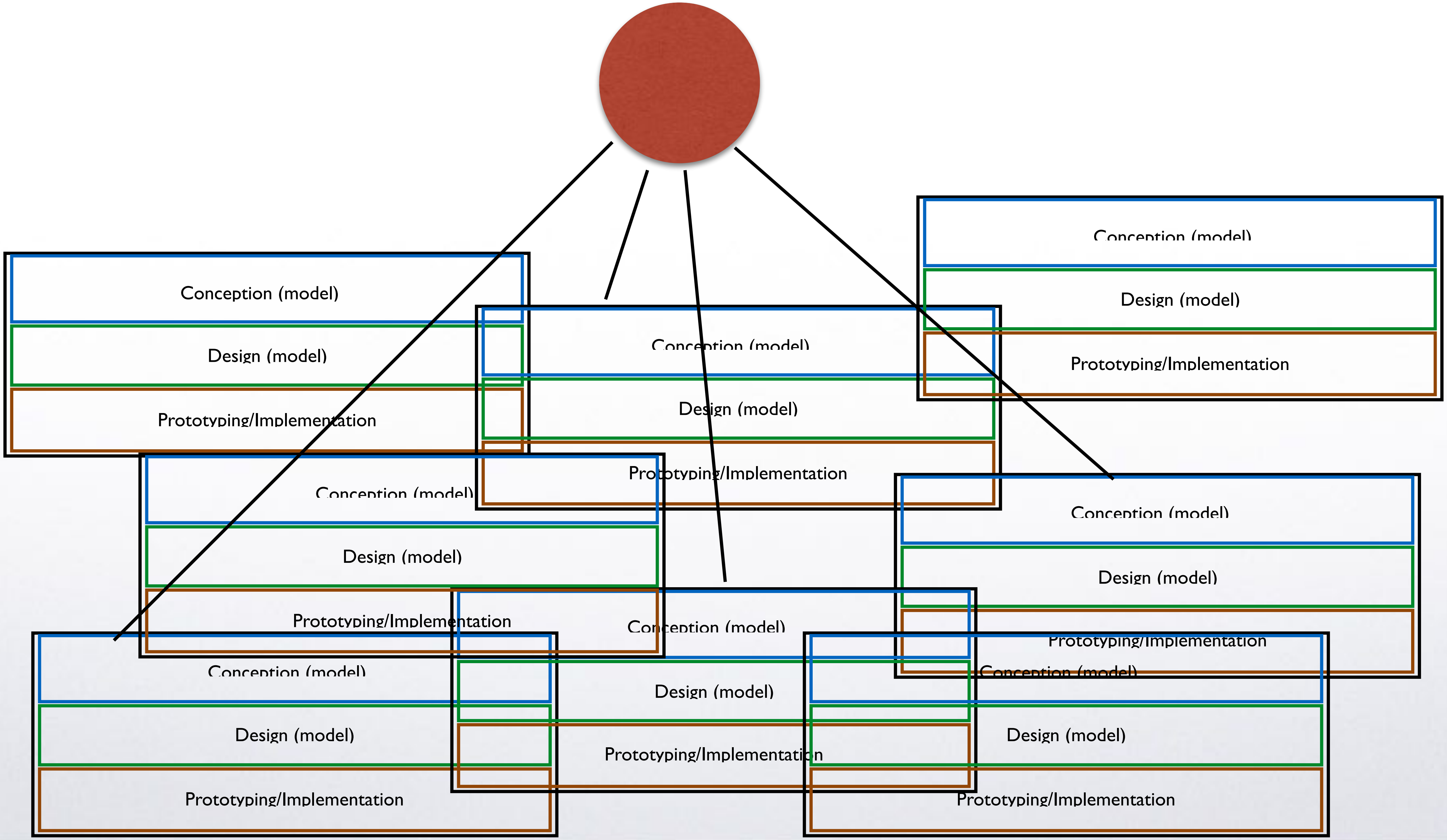
Layered Approach



Conception (model)

Design (model)

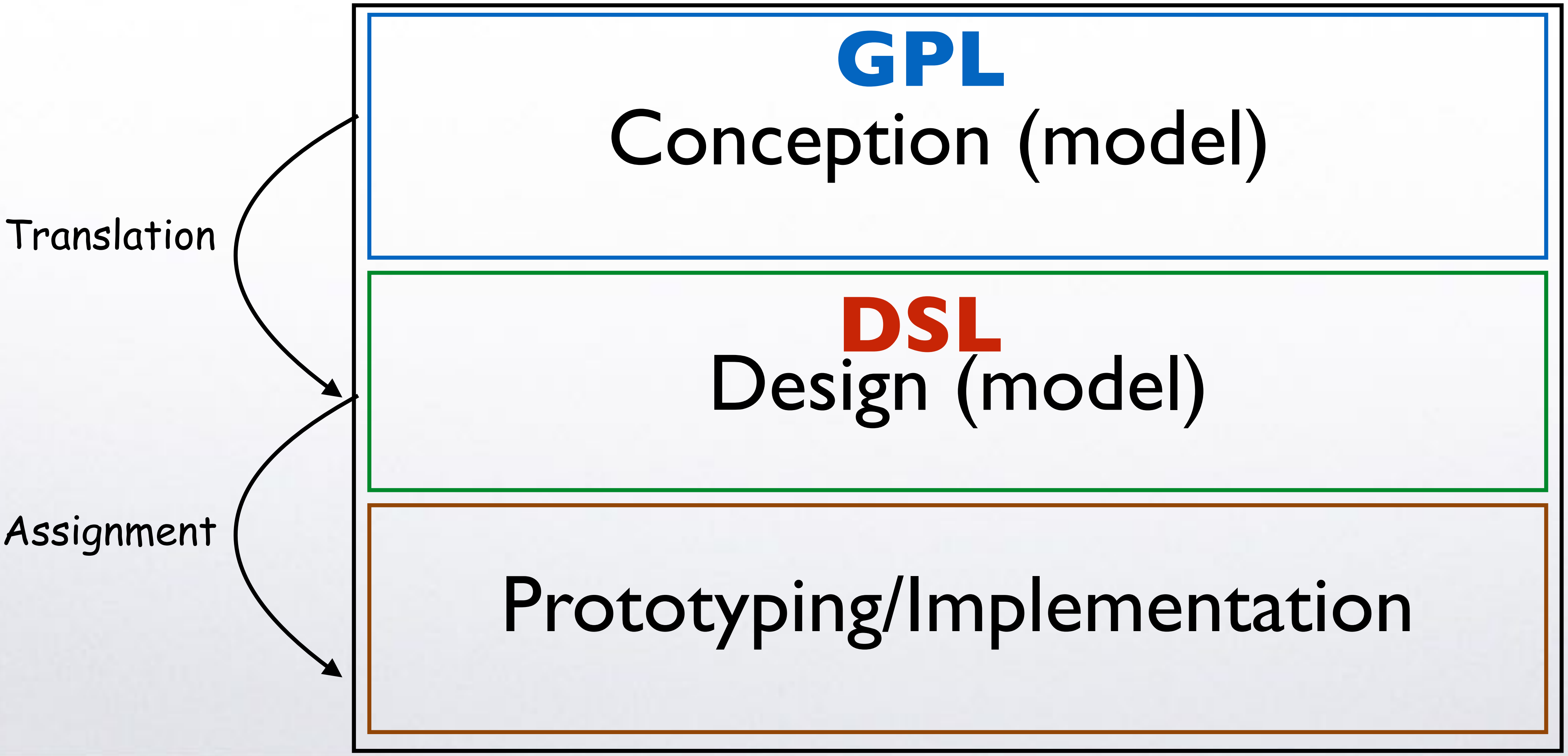
Prototyping/Implementation


Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation
Conception (model)
Design (model)
Prototyping/Implementation

Using different languages

Diversified Design Background

Layered Approach



GPL

Conception (model)

DSL

Design (model)

Prototyping/Implementation

Practical ideas do Sys Design

What that really means?

To make GPL(s) standard for the initial phases

DSLs are naturally diversified, but could have a standard meta-description

To “solve" (or propose a new theory) for the assignment problem

Basic Definitions

Definition 1 (Model). A model has three characteristics: There is an original that it models. The model is an abstraction with respect to the original. The model has a purpose with respect to the original. (Definition to Stachowiak, 1973)

Definition 2 (Language). A language is a means for communication between humans, machines, and humans and machines. A language describes the set of possible sentences that may be communicated between the stakeholders.

Definition 3 (Domain Specific Language (DSL)). A DSL is a language that is specifically dedicated to a domain of interest.

Definition 4 (Language Definition). A language is defined by the following concepts:

– Concrete syntax: e.g. in textual, tabular, or graphical form describing the set of sentences of the language.
– Abstract syntax: describing essential concepts and structure of the sentences without semantically irrelevant concrete sugar.
– Static semantics (or context conditions): Is a boolean predicate based on the concrete respectively abstract syntax. Sentences that fulfill the static semantics are called well-formed. They obey the context (scope, type system, etc.)
– Meaning (or dynamic semantics) of the sentences e.g. as operational, denotational, or axiomatic semantics.

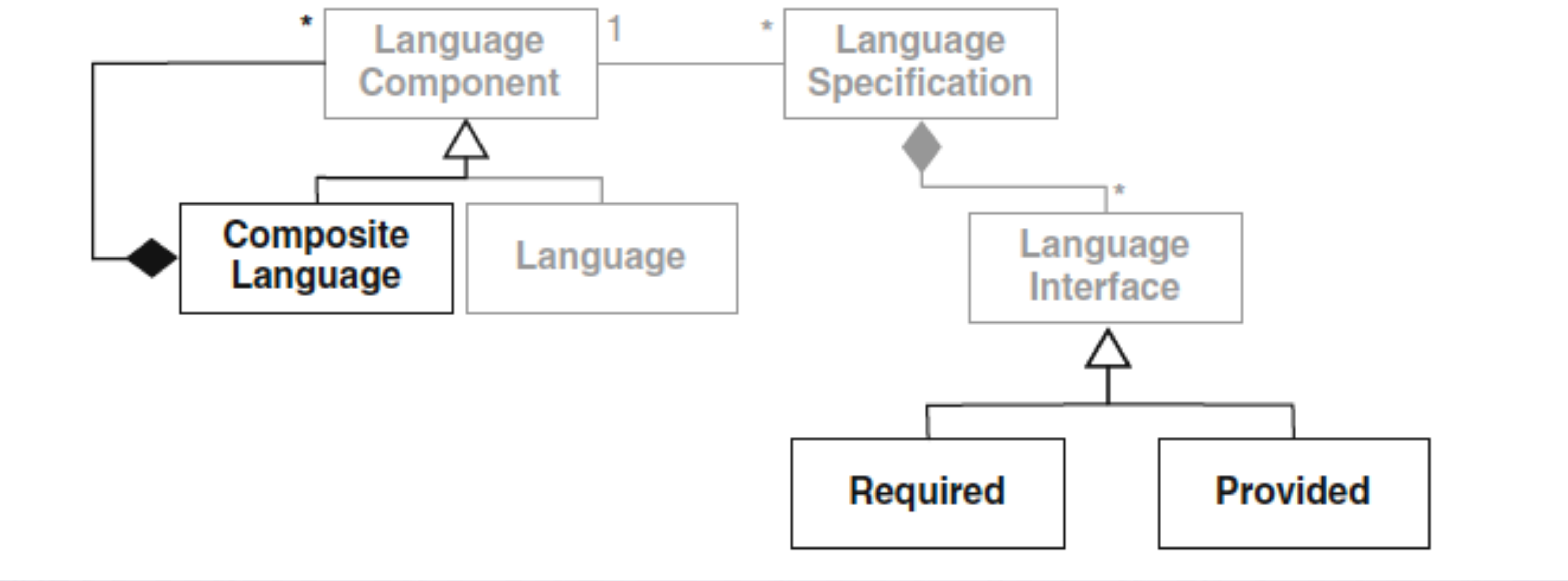
Definition 5 (Language Component).

A language component (aka. language module or language unit) is a reusable encapsulation of a, possibly incomplete, language.

A language component includes a language definition and might include explicit provided and required language interfaces.

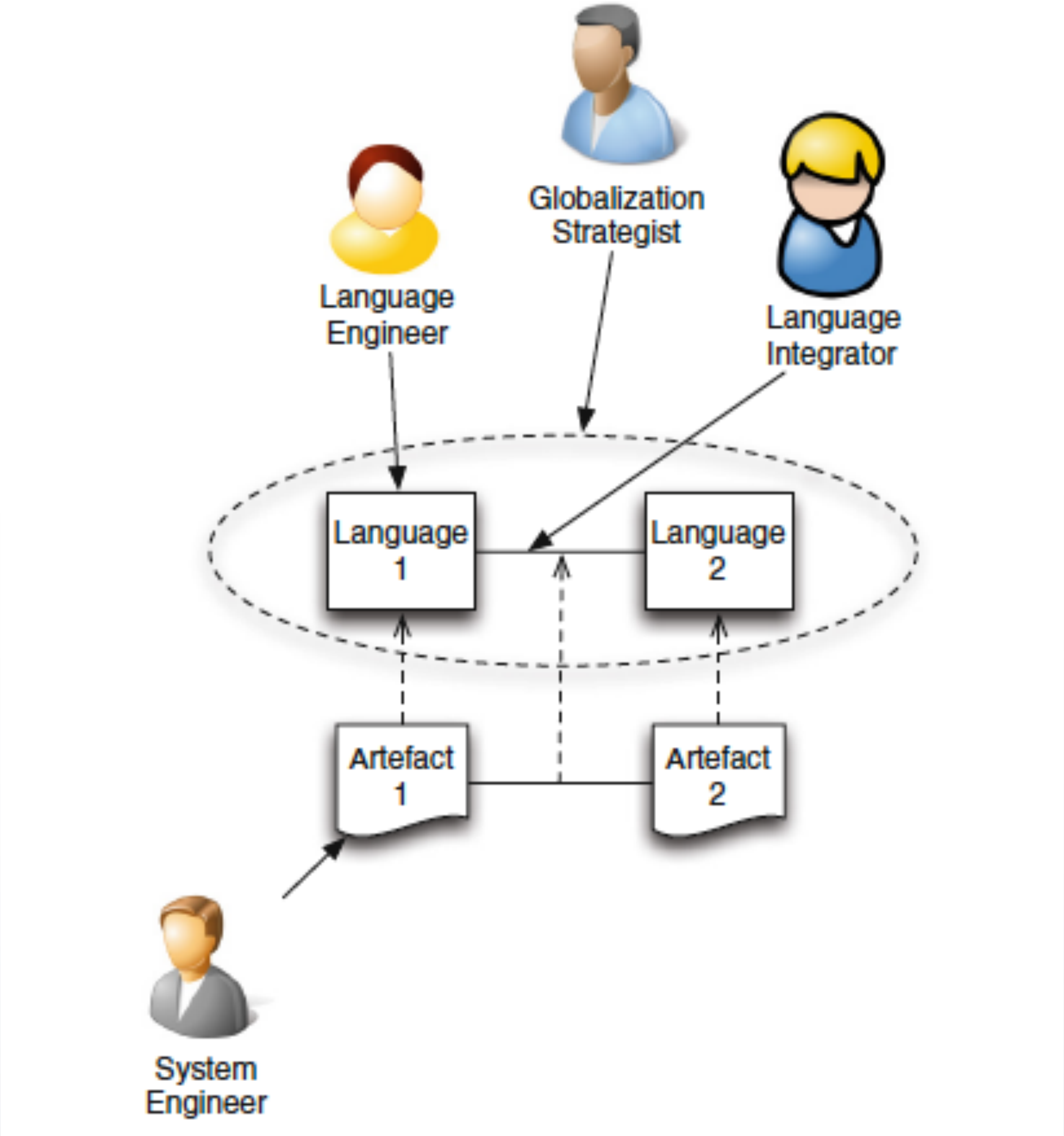
Definition 6 (Language Interface).

A language interface is a relevant
abstraction for a specific purpose of a provided or required part of a language component.
An interface can be defined manually in a separate artifact or inferred automatically from the language component definition.


*
Language Component
1
*
Language Specification
Composite Language
Language
*
Language Interface
Required
Provided

Definition 7 (Globalization). Globalization deals with the purposeful construction, adaptation, coordination and integration of explicitly defined languages, to be amenable to mechanical and cognitive processing, with the goal of improving quality and reducing the cost of system development.

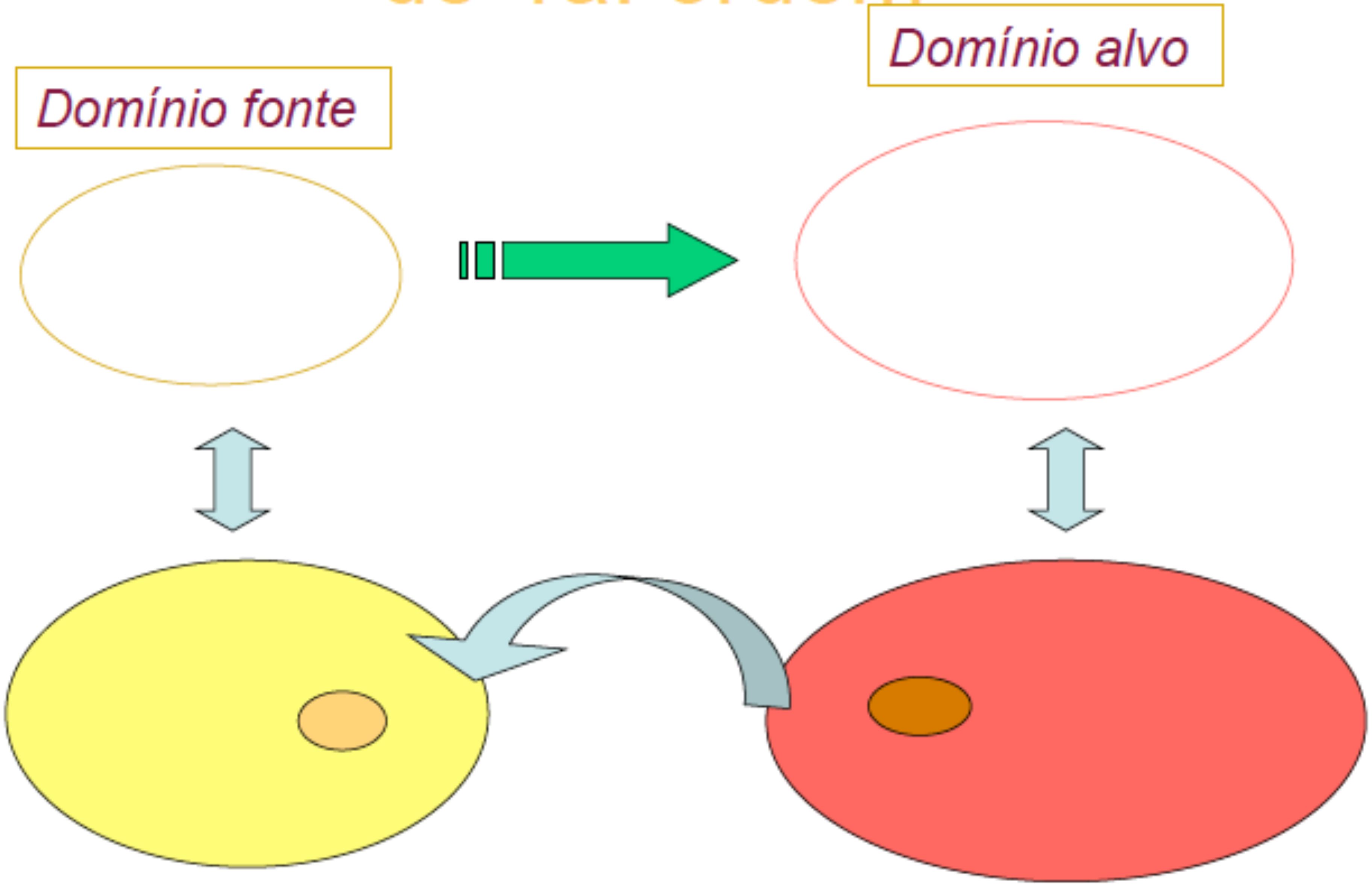
Definition 8 (Globalization Stakeholder). Any person who is affected by the definition or use of a language or its components is a globalization stakeholder.


Globalization Strategist
Language Engineer
Language Integrator
Language 1
Language 2
Artefact 1
Artefact 2
System Engineer

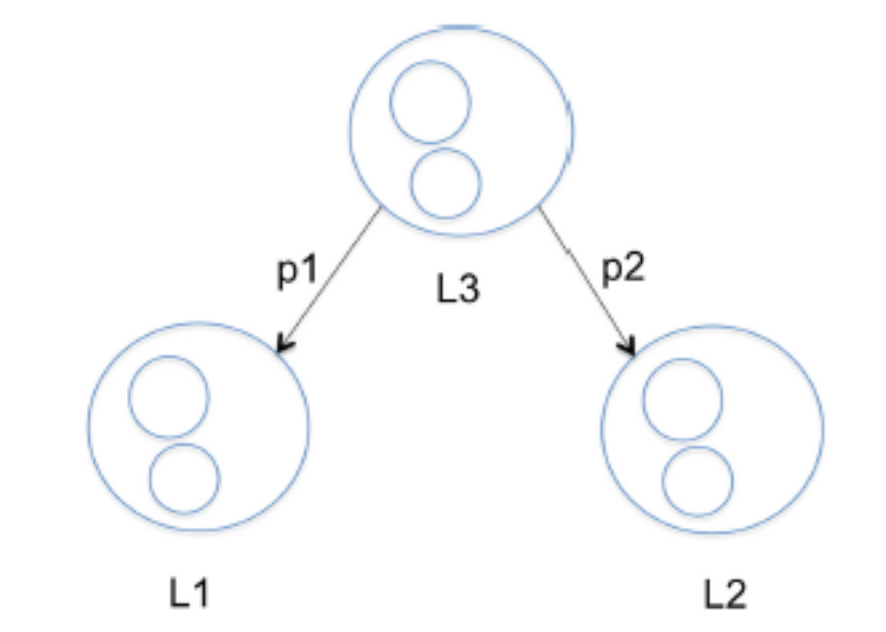
Definition 9 (Language Relation). A language relation relates the sentences of multiple languages.

Definition 10 (LanguageMapping). A language mapping is a language relation that has an algorithmic, effectively executable realization that maps sentences of the source languages to sentences of the target languages.

Abordagem em lógica de predicados de 1a. ordem



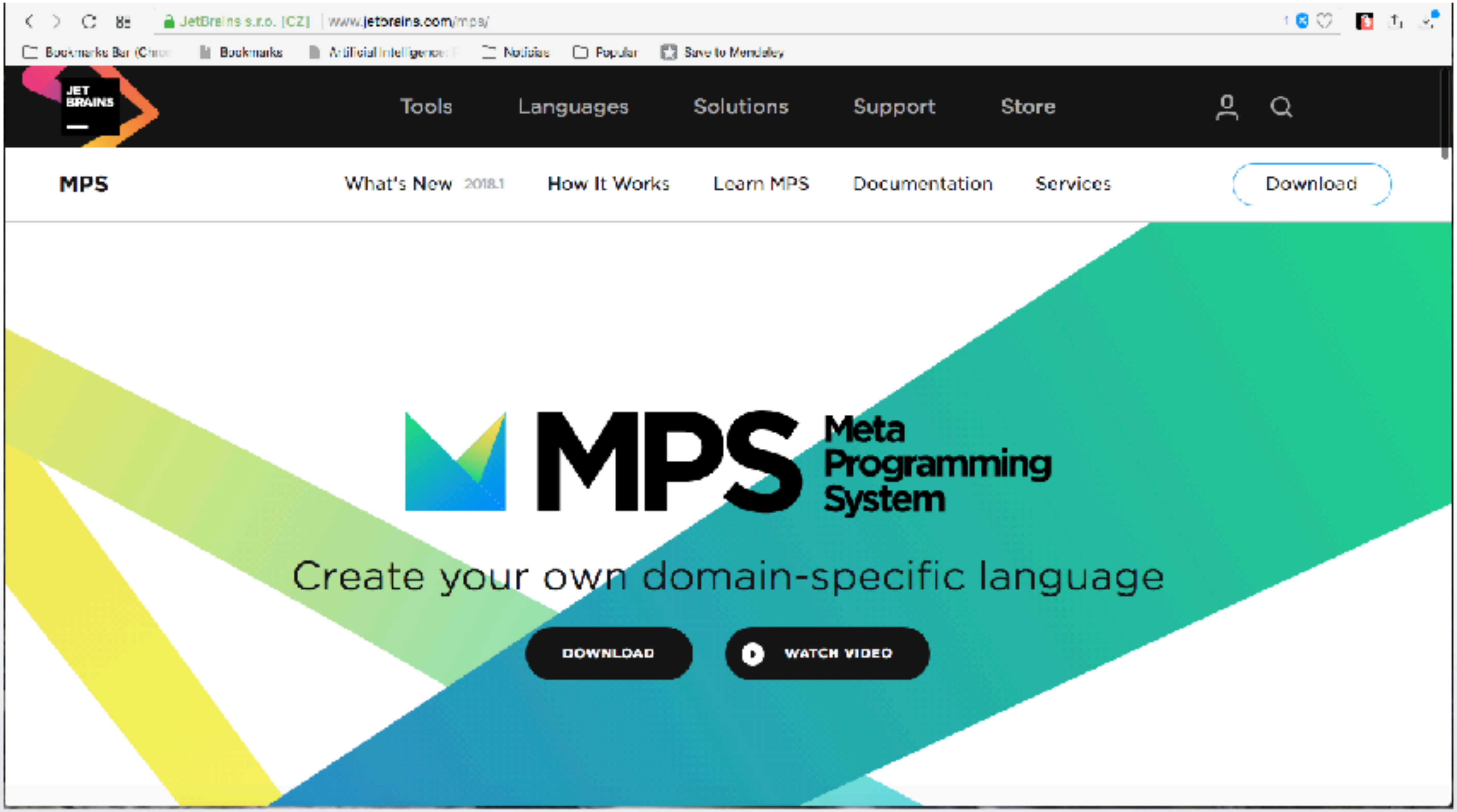
Definition 11 (Language Composition). This is an abstract concept that achieves globalization in terms of multiple languages working together to achieve a common goal.



Definition 12 (Language Coordination). Language Coordination is a form of composition where individual sentences of the coordinated languages are collaborating to achieve a common goal.

Definition 13 (Language Integration). Language Integration is the production of a new language from a set of individual languages.

www.jetbrains.com/mps/



Create your own domain-specific language

Basic Tools for MBSE

MDE is a relatively new engineering approach with some expectations and challenges to be addressed in the next years.

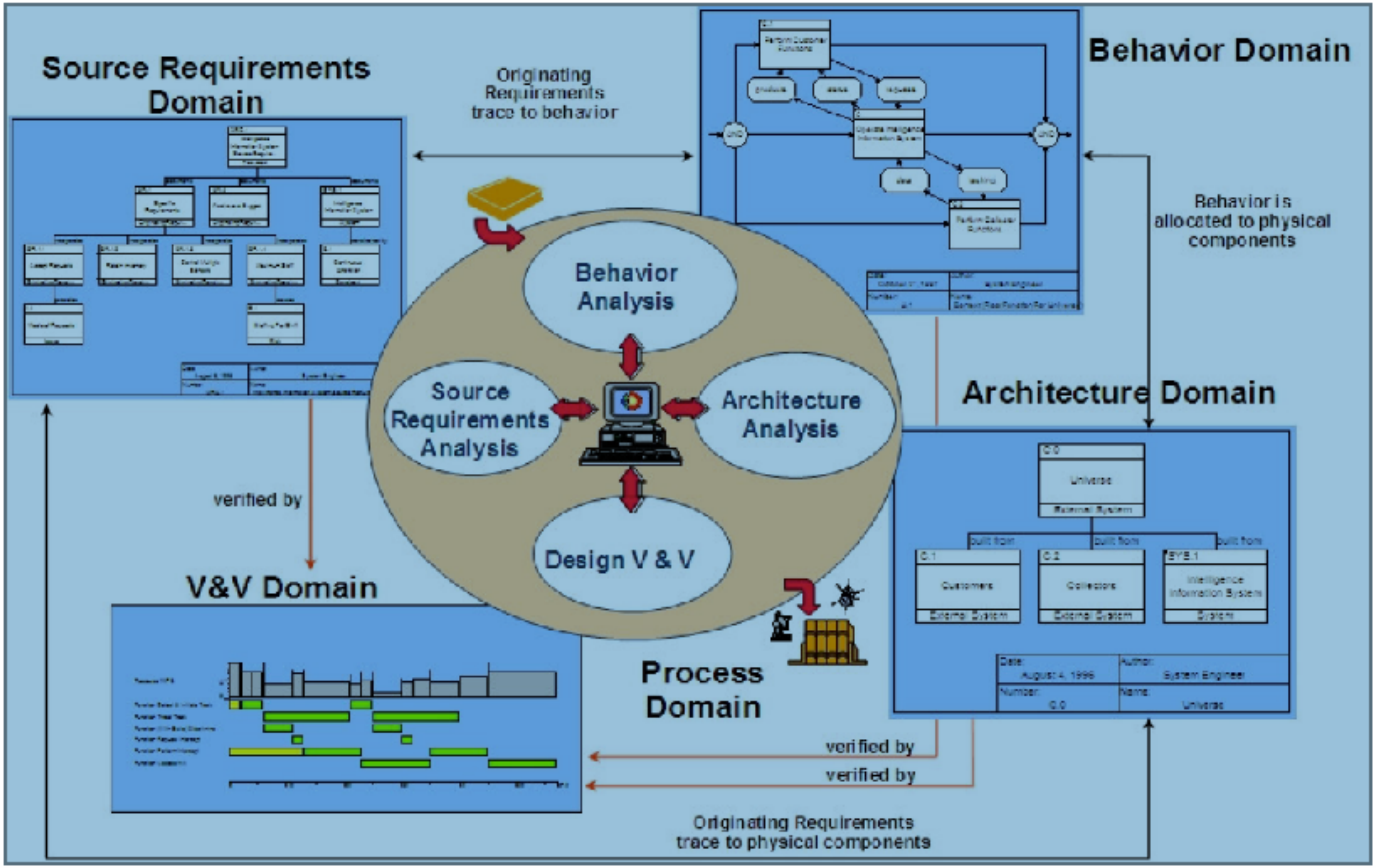
A variety of tools that embody the main ideas of MDE have been developed and improved over this last decade. Some of them correspond to tools developed in an academic environment, as is the case of experiments carried out under GME, ProjectIT, VMTS, MetaSketch, or AtomPM. Other tools are commercial, such as the case of Microsoft Visual Studio Visualization and Modeling SDK, Sparx Enterprise Architect, Metacase Meta Edit+, or ObeDesigner. Beyond these, it is worth to highlight some tools and technologies currently developed around the Eclipse Modeling Project and the JetBrains MPS.

Vitech

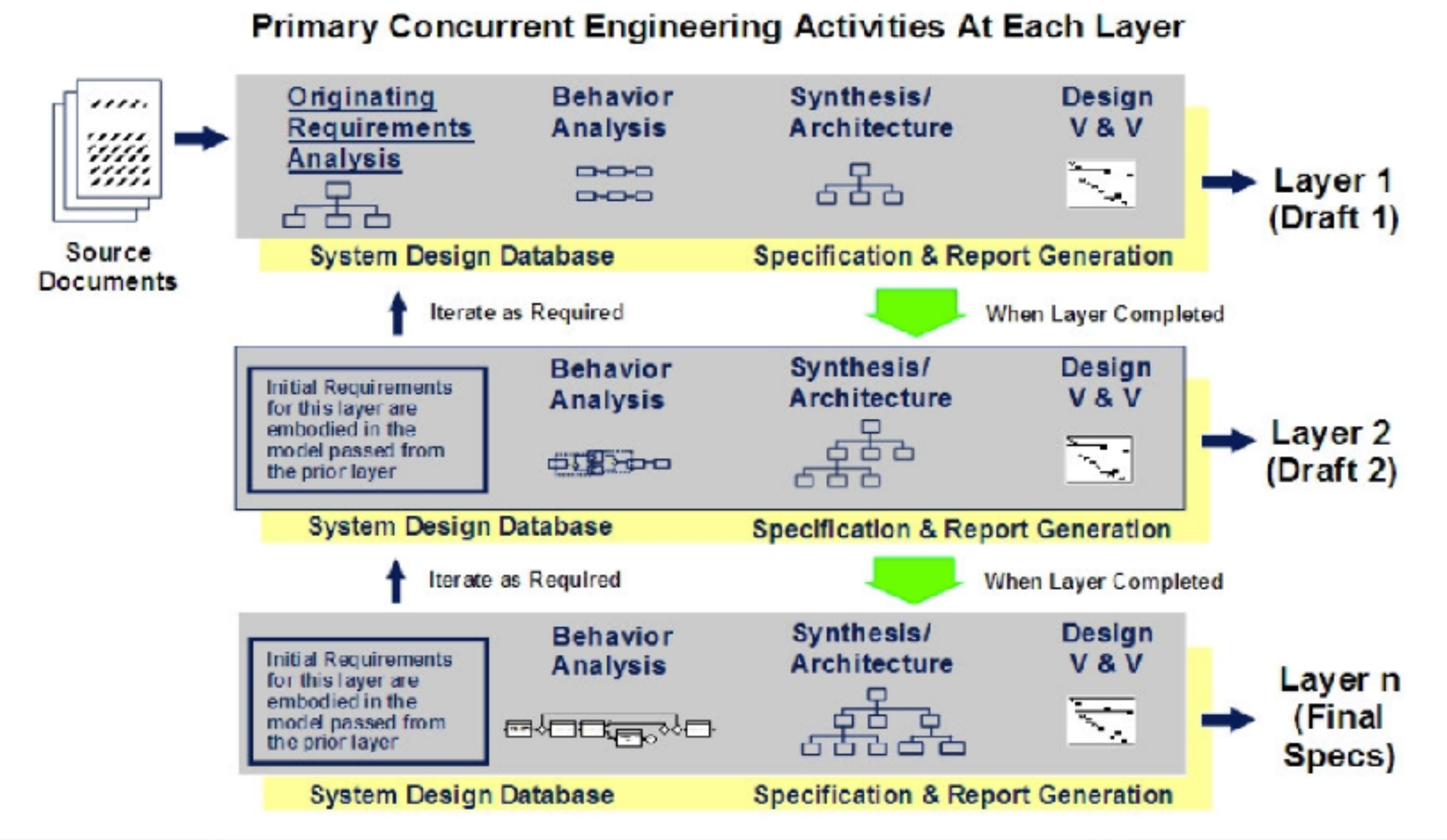
Insight through integration

NORTHROP GRUMMAN
FedEx
SAAB
Booz | Allen | Hamilton
DEPARTMENT OF TRANSPORTATION UNITED STATES OF AMERICA
SIEMENS
QinetiQ
BAE SYSTEMS
USAA
Sikorsky
A United Technologies Company
cyberpoint
AEROSPACE
DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA
GOODRICH
NASA
AIRBUS
Raytheon
SAIC
From Science to Solutions

VITECH Model (CORE)

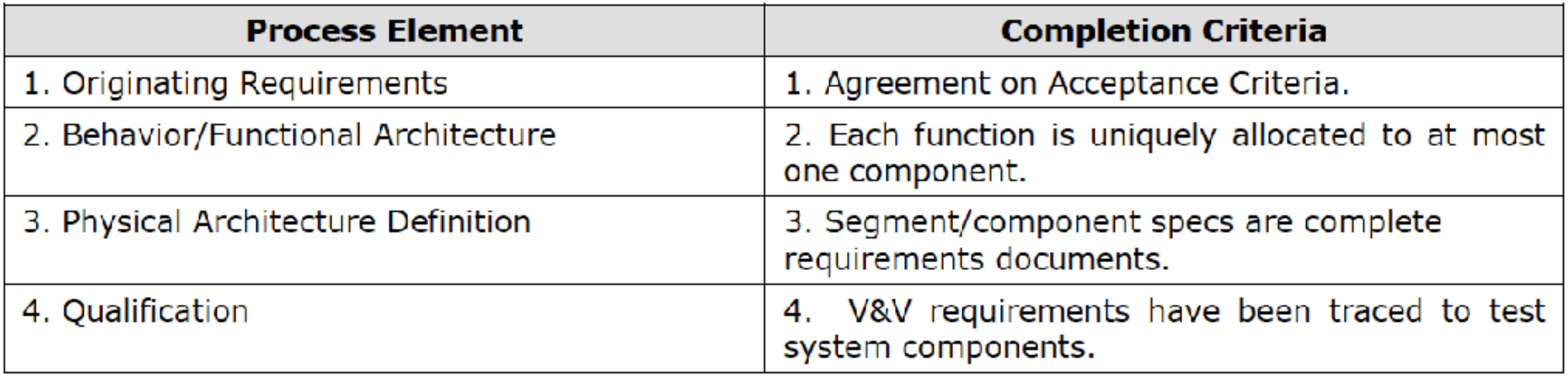

Source Requirements Domain
Originating Requirements trace to behavior
Behavior Domain
Behavior is allocated to physical components
Behavior Analysis
Source Requirements Analysis
Architecture Analysis
Design V & V
Architecture Domain
verified by
V&V Domain
Process Domain
verified by
verified by
Originating Requirements trace to physical components
Universe
Customers
Collectors
Intelligence Information System

VITECH Onion Model

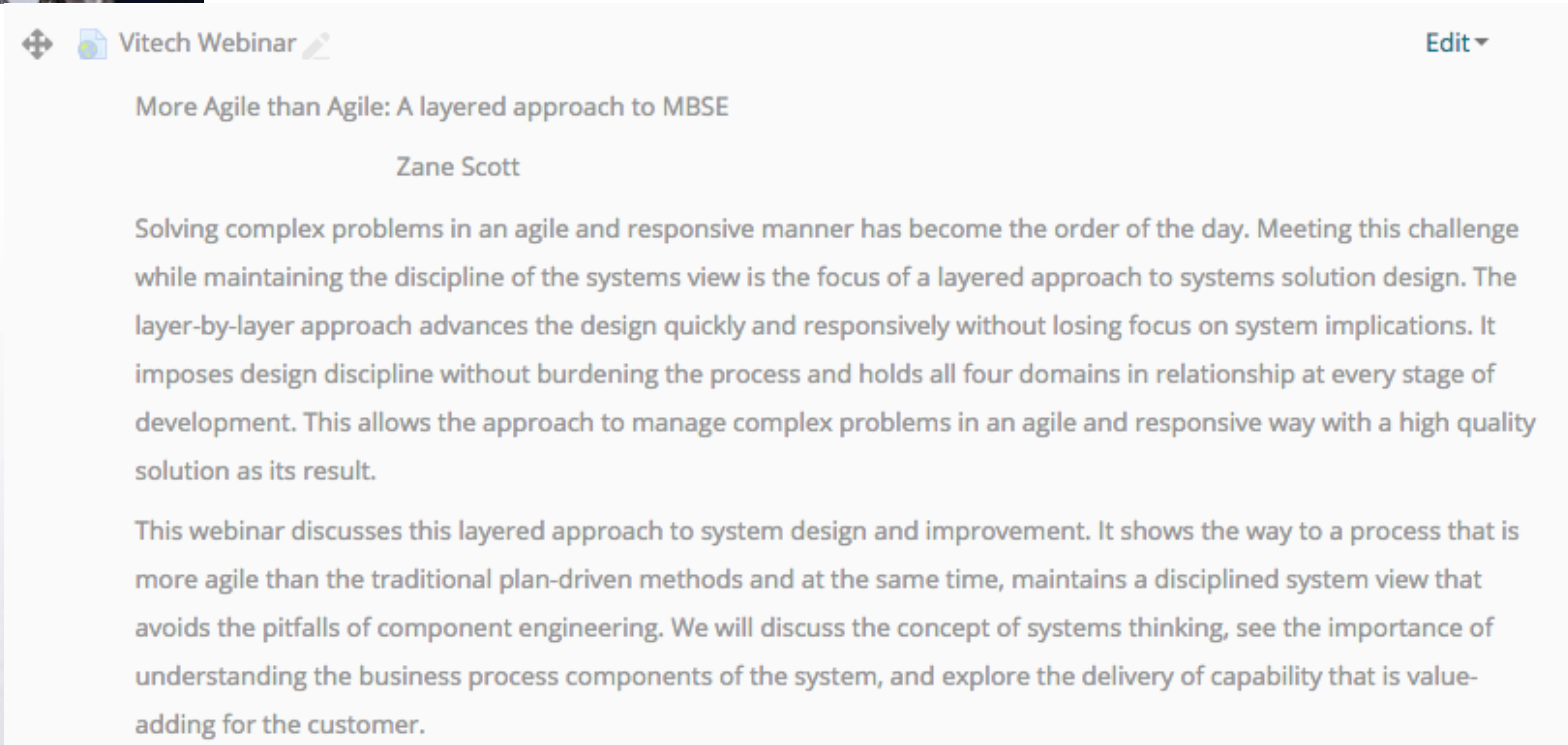


Primary Concurrent Engineering Activities At Each Layer

VITECH completion criteria



Process Element	Completion Criteria
1. Originating Requirements	1. Agreement on Acceptance Criteria.
2. Behavior/Functional Architecture	2. Each function is uniquely allocated to at most one component.
3. Physical Architecture Definition	3. Segment/component specs are complete requirements documents.
4. Qualification	4. V&V requirements have been traced to test system components.

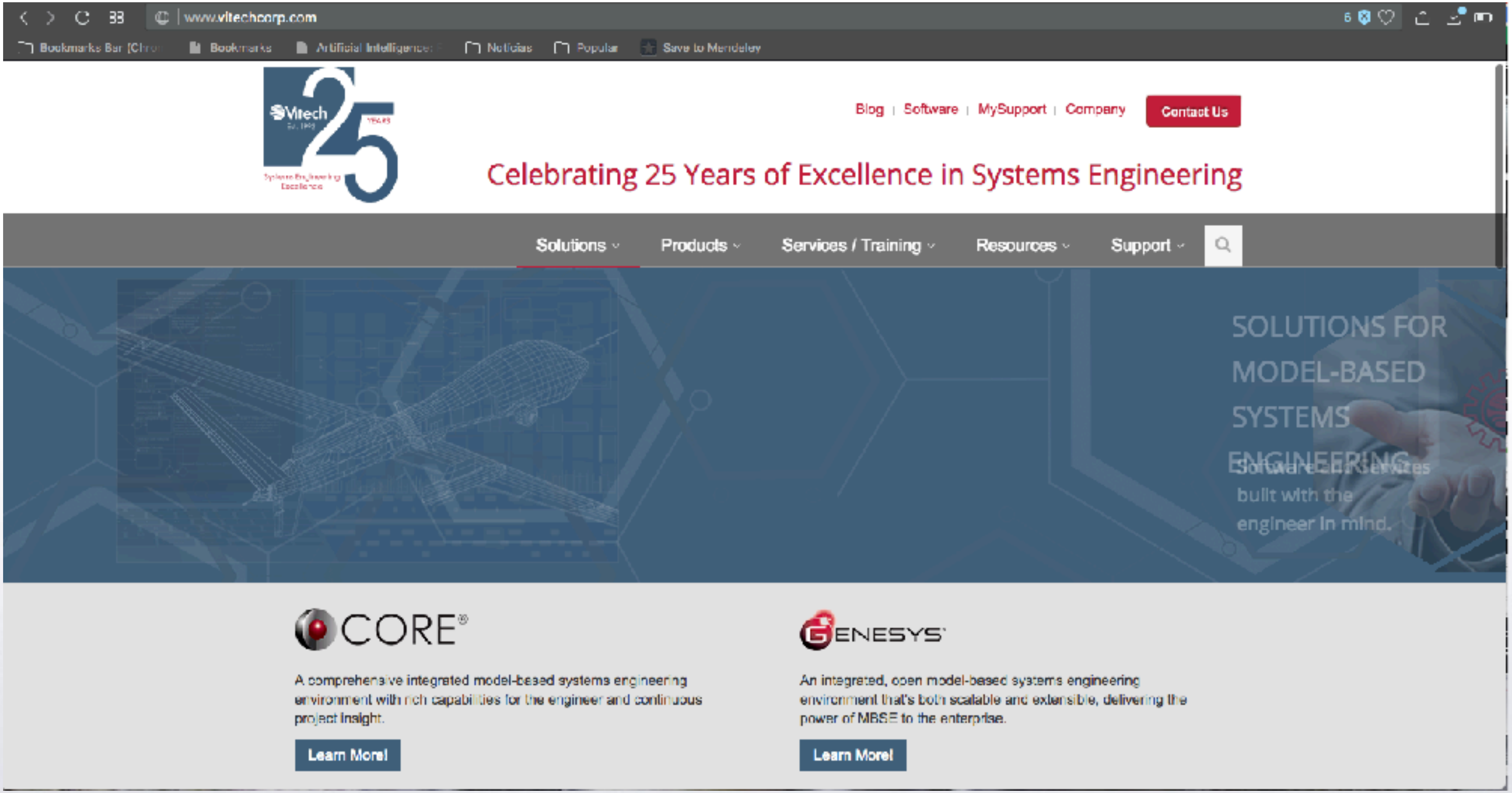


More Agile than Agile: A layered approach to MBSE

Zane Scott

Solving complex problems in an agile and responsive manner has become the order of the day. Meeting this challenge while maintaining the discipline of the systems view is the focus of a layered approach to systems solution design. The layer-by-layer approach advances the design quickly and responsively without losing focus on system implications. It imposes design discipline without burdening the process and holds all four domains in relationship at every stage of development. This allows the approach to manage complex problems in an agile and responsive way with a high quality solution as its result.

This webinar discusses this layered approach to system design and improvement. It shows the way to a process that is more agile than the traditional plan-driven methods and at the same time, maintains a disciplined system view that avoids the pitfalls of component engineering. We will discuss the concept of systems thinking, see the importance of understanding the business process components of the system, and explore the delivery of capability that is value-adding for the customer.



Celebrating 25 Years of Excellence in Systems Engineering

SOLUTIONS FOR MODEL-BASED SYSTEMS ENGINEERING

Software and Services built with the engineer in mind.

CORE®

A comprehensive integrated model-based systems engineering environment with rich capabilities for the engineer and continuous project insight.

GENESYS

An integrated, open model-based systems engineering environment that's both scalable and extensible, delivering the power of MBSE to the enterprise.

Calendário da PG-EPUSP

Inicio do Primeiro 1o. ciclo 19/02 para nós 22/02

Término (oficial) do 1o. ciclo 25/05 para nós 24/05

Prazo para entrega de notas ~~25/07~~ 15/07

Prazo final para os artigos 02/06

Entrega de notas 22/06

Obrigado

Reinaldo