Introduction to Model-Based Engineering

What does a good model smell like?

Dr. Bruce Powel Douglass, Ph. D.

Principal

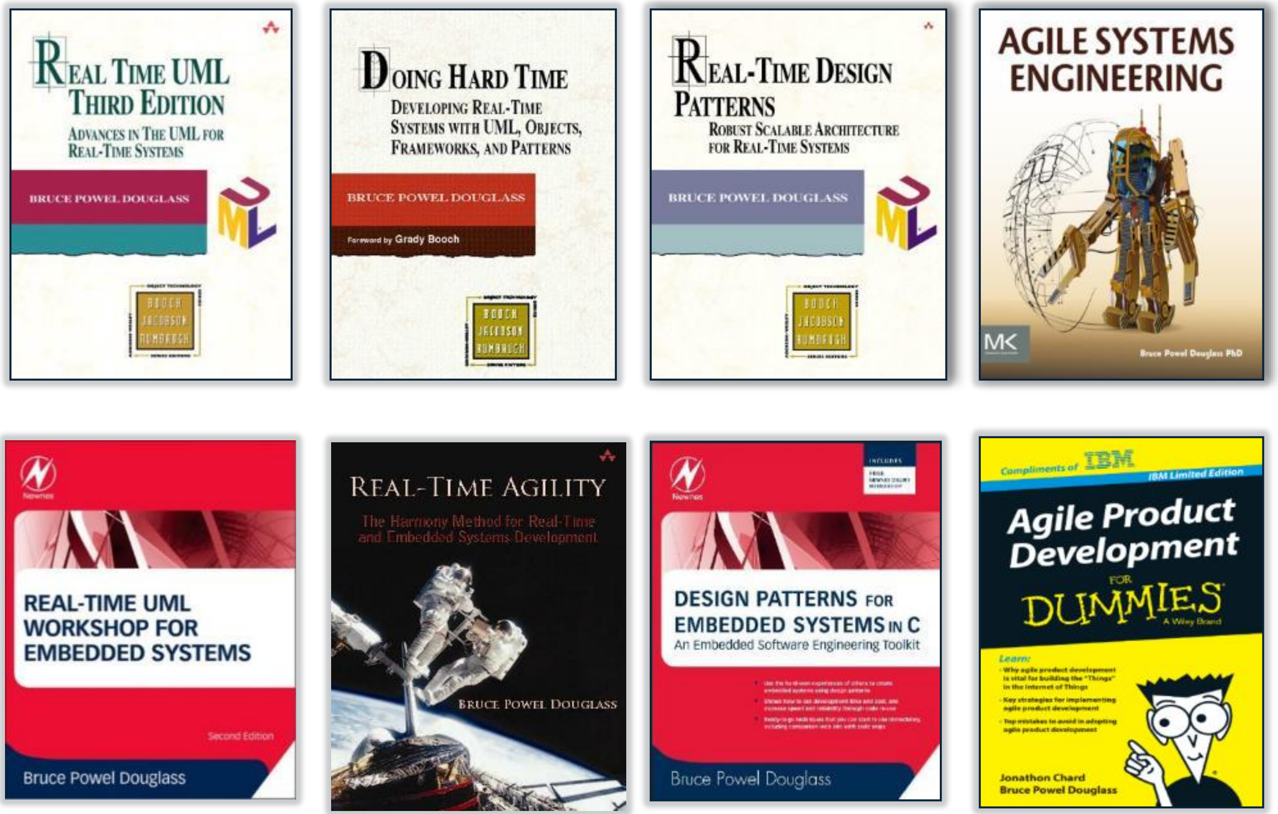
A Priori Systems

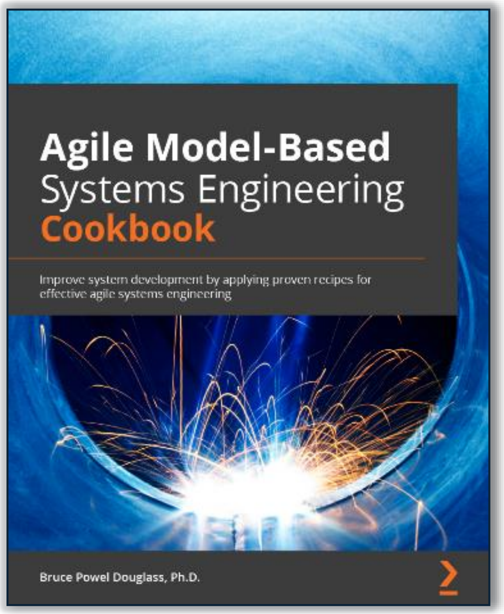
www.bruce-douglass.com

A Priori Systems

Real-Time Agile Systems and Software

About the Author





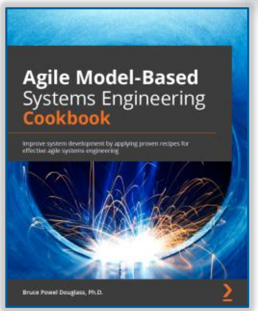
Bruce Douglass, Ph.D.

- **Senior Principal Agile Systems Engineer**
  - Systems Engineering Tech Center
  - The MITRE Corporation
  - Can be reached at *bdouglass@mitre.org*
- **Contributor to UML standard**
- **Contributor to SysML standard**
- **Developer of UML Dependability Profile**
- **Former Cochair RTAD Task Force for the OMG**

A Priori Systems
Real-Time Agile Systems and Software

INCOSE Lunch and Learn Series

- Introduction to Modeling
- Introduction to Agile and Model-Based Engineering
- Engineering Agile Requirements: Epics, Use Cases, and User Stories
- Improving Requirements with Use Cases
- Model-Based Interface Control Documents
- From Systems to Downstream Engineering: The Hand Off
- Model-Based Testing
- MBSE and Safety Analysis



A copy of **Agile Model-Based Systems Engineering Cookbook** will be given away at the end of the session. ***You must be present to win.*** *If you do not acknowledge your presence when called, another attendee will be selected.*

A Priori Systems
Real-Time Agile Systems and Software

Starting Definitions

- **Model**
  - is a representation of a system of interest from a particular viewpoint, capturing attributes for a specific purpose. A model is always an abstraction in that it focuses on properties of interest at the expense of properties not of interest and at a specified level of precision (detail).
- **MBE (Model-Based Engineering)**
  - "An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle." (Final Report, Model-Based Engineering Subcommittee, NDIA, Feb. 2011)
- **MBSE (Model-Based Systems Engineering)**
  - "The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." (INCOSE MBSE Report, September 2007)
- **MDD (Model-Driven Development)**
  - The use of models for the specification and design of software-based systems.

A Priori Systems
Real-Time Agile Systems and Software

Starting Definitions

- **DE (Digital Engineering)**
  - Digital engineering is the ability to perform discipline-specific engineering by collaborating with other disciplines and by leveraging authoritative system data in digital form from those disciplines within my tools of choice and in the right format.
- **DE Platform**
  - A standard platform for projects which includes pre-installed tools, tool integrations, processes, and links to training and other knowledge / skill resources with the intent of allows quick start up of internal and sponsor-related projects.
- **Digital Thread**
  - A connected set of models of a system in different lifecycle stages, including specification, design, operation, and maintenance.
- **Single Source of Truth**
  - Each important datum is located in a singular, authoritative place and is connected, via navigable links, to all other relevant data in that or other repositories. Note: this doesn’t mean that all data are in the same repository but the authoritative source for each datum is singular.

A Priori Systems
Real-Time Agile Systems and Software

Modeling For Beginners

Drawing vs Modeling

What's a model?

Models & Views

A Priori Systems
Real-Time Agile Systems and Software

Foundational Concept of Modeling

A drawing is a picture with only imagined semantics and no underlying repository of information

Once you're done drawing, then go do the "real work"

Drawing ≠ Modeling

A model focuses on system aspects of interest and ignores others

Stores underlying semantics in model repository

Uses a precise language

Generates any needed documentation from the model repository

Supports verification through review, execution and/or formal methods

Note: it IS possible to use a modeling tool solely for drawing and not modeling, but it's not a good idea!

So What IS a Model exactly?

Modeling is the development of a set of system data of relevant systems and their properties

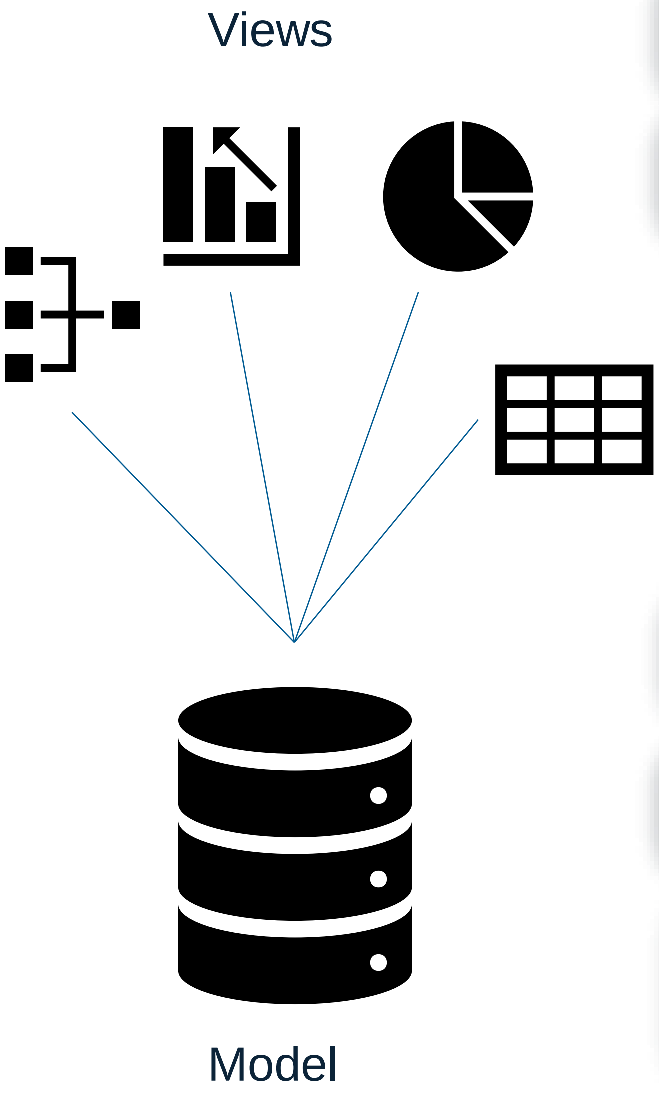
Models have views (e.g. diagrams)

Diagrams show subsets of eng. data

Diagrams have singular purpose

Diagrams answer questions

Diagrams support specific reasoning



Models have scope

Models have purpose

Models have precision

Models have accuracy

Models have fidelity

Models are falsifiable

Models are verifiable

Models *are interconnected data!*

A Priori Systems
Real-Time Agile Systems and Software

So What IS a Model exactly?

Models have scope

Modeling i
a set of s
systen

pose

cision

Models ha

ccuracy

Diagrams

lelity

Diagram

able

Diagram

rifiable

To be clear, you do NOT model in Visio or PowerPoint; you can only draw pictures

Diagrams support specific reasoning

Model

Models *are interconnected data!*

A Priori Systems
Real-Time Agile Systems and Software

Uses of Diagrams and Tabular Views

- Data Entry
  - Drawing diagrams or entering data into tables/matrices is a way of entering information into the model
  - When you create an element on the diagram, the model either
    - Refers to an existing element, and updates it based on your actions, or
    - Creates a new element in the model repository
- Model visualization
  - Creating a diagram or tables allows you to create a view of a subset of the model information
- Simulation / Execution Debugging & Execution Control
  - Some modeling tools provide special diagrams and tools to control execution, insert events, change values, set breakpoints, etc.

A Priori Systems
Real-Time Agile Systems and Software

Executable Models

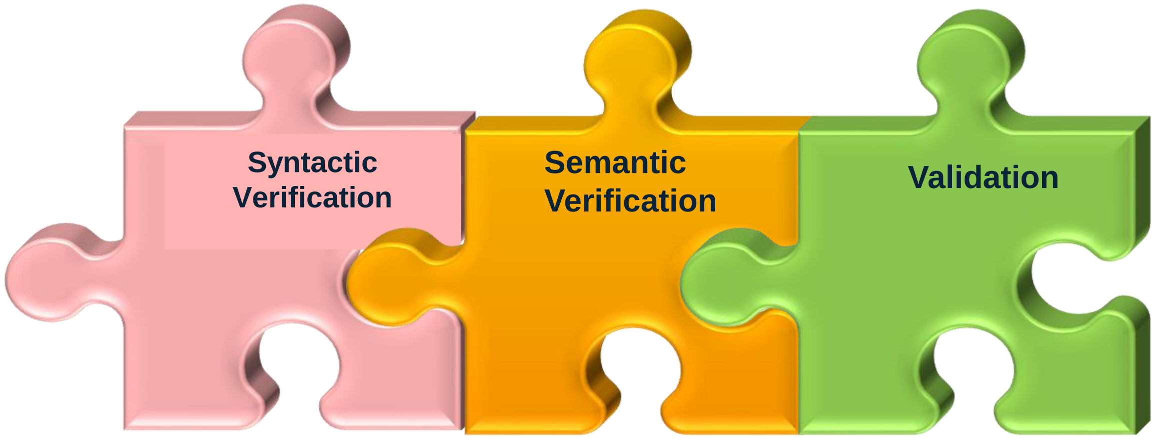
- WHY
  - To make sure the model isn't stating "utter nonsense."
  - Models make declarative and imperative statements of truth
  - It is *absolutely crucial* that we have a means by which we can verify that the statements of truth made by the model can be verified or demonstrated to be true
    - Such models are said to be "falsifiable"; this means that there is a way to demonstrate that a false model is indeed false
  - The larger the model, the more important this is
  - The more significant the impact of the model or system, the more important this is
- Rhapsody, Magic Draw, and Sparx Enterprise Architecture can build and execute models (with differing levels of fidelity)

"Any language rich enough to say something interesting is also rich enough to say utter nonsense that at first glance sounds reasonable. "
- **Douglass' Paradox**

Declarative statements identify what you want to happen; *imperative statements* identify how to make something happen.

A Priori Systems
Real-Time Agile Systems and Software

What do we mean by “verification & validation” of work products (e.g. models)?



Semantic: Is the content correct?

- *Compliance in meaning*
  Performed by engineering personnel

Three basic techniques

- **Semantic review** (subject matter expert & peer) – most common, weakest means
- **Testing** – requires executability of work products, impossible to fully verify
- **Formal methods** – strongest but hard to do and subject to invariant violation

Syntactic: Is it well-formed?

– “Compliance in form”
  Performed by quality assurance personnel

- **Audits** – work tasks are performed as per plan and guidelines
- **Syntactic review** – work products conform to standard for organization, structure and format

Valid: Does it solve the right problem?

- Validation = “meets the stakeholder need”
  Performed by customer + engineering

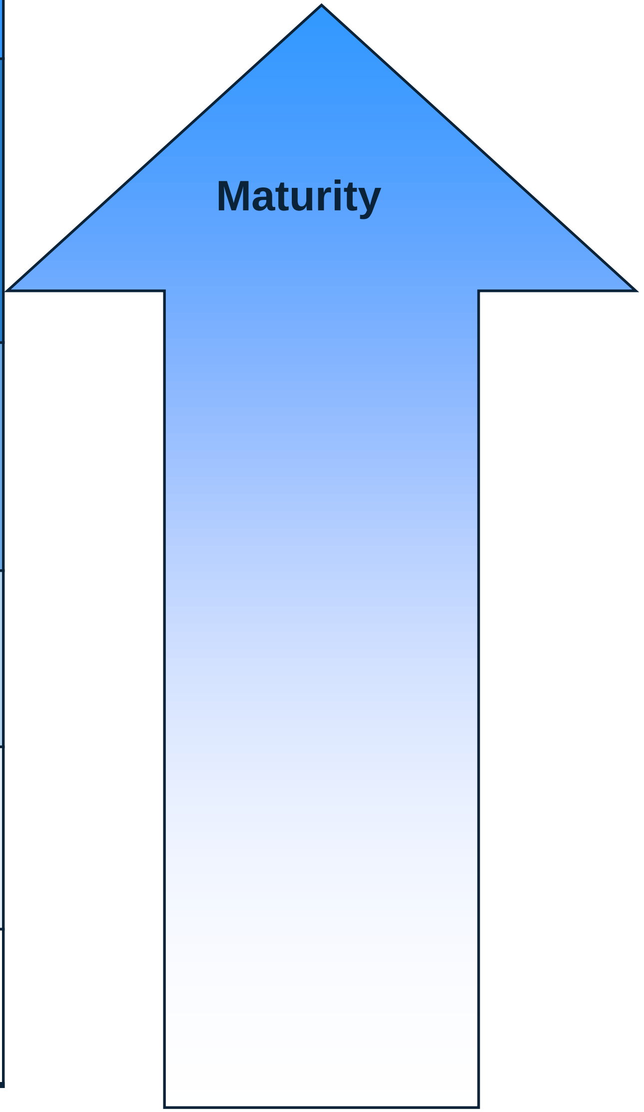
Some common techniques

- **Review** – (subject matter expert & customer) – most common, weakest
- **Simulation** – show simulated input → outputs
- **Sandbox** – exploratory usage in constrained environment
- **Flight test** – demonstration of system capabilities
- **Deployment –** early usage of system of partial capability

A Priori Systems
Real-Time Agile Systems and Software

INCOSE Organizational Model-Based Capabilities Matrix

Level	Benefit	Focus	Technologies
4 Enterprise wide capabilities	High	Employing modeling as an organizational standard approach; managed reusable DE Assets	DE Platform infrastructure, tooling, training, and processes widely applied across organization
3 Program/project wide capabilities	Moderately High	Wide-scale use of modeling throughout projects	Model integrated with other functional disciplines, digital threads defined and digital twin
2 Modeling standards are applied	Moderate	Standardizing use of modeling	Integration of modeling into processes, standardized reviews and quality assurance
1 Limited use of modeling	Low	Answer specific questions during development	Modeling efforts address specific objectives and questions
0 No MBSE capability	None	Litlte or no use of models in systems engineering efforts; use of document-based, siloed data.	

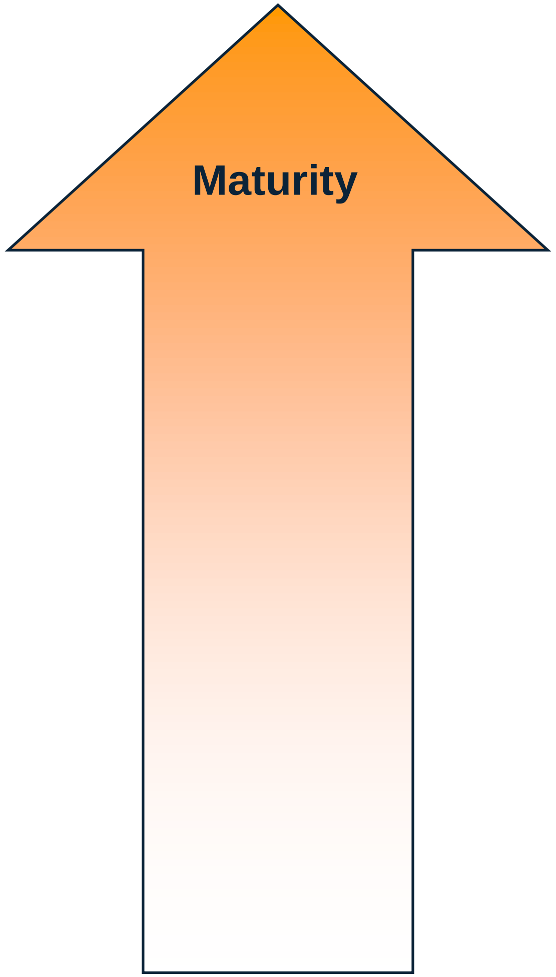


https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=MBCM

A Priori Systems
Real-Time Agile Systems and Software

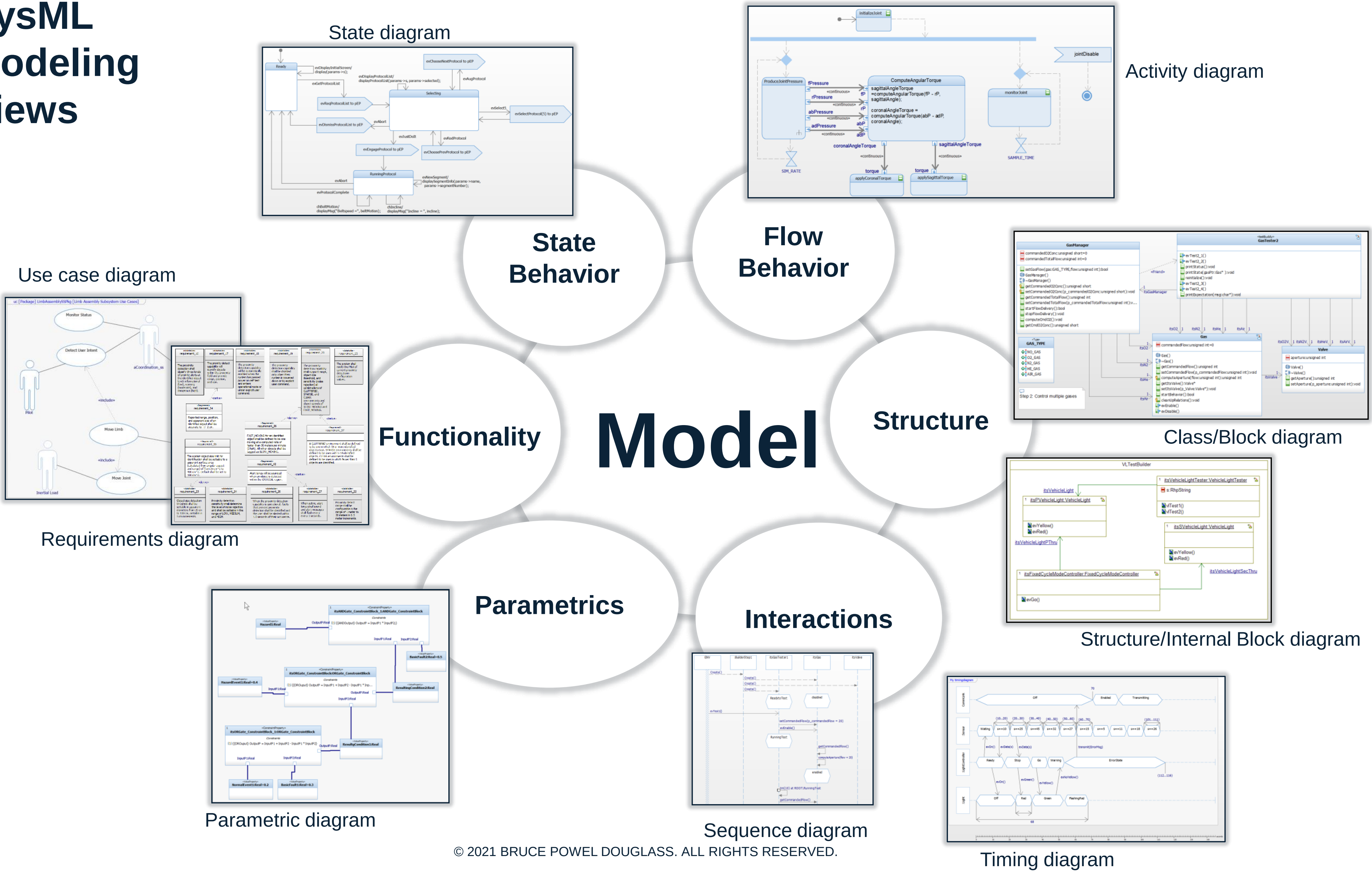
Project-Oriented Modeling Maturity

Level	Benefit	Focus	Technologies
4 Integrated cross platform modeling	High	Large-scale breaking down with federated models; connecting tools and data	DE Platform forms the core work environment
3 Executable	Moderately High	Use of verifiable, testable models;	Executable state and activity models, model-based test; use of quantitative metrics
2 Standardization	Moderate	Wide-spread use of modeling within the project; single source of truth	Use of modeling guidelines and standards, strong integration into engineering process
1 Visualization	Low	Visualizing engineering data	Reverse engineering, Picture drawing, “boutique engineering”
0 Textual / code-based/ siloed document-based development	None	Manual, time intensive heroic development with disconnected, siloed data	



A Priori Systems
Real-Time Agile Systems and Software

SysML Modeling Views



UML and SysML – The Preeminent Modeling Languages

SysML Pillar

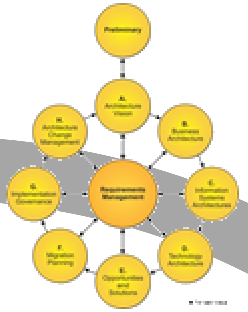
SysML Pillar	UML (software)	UML4SysML	SysML (systems)
Structural	Class Diagram Structure Diagram Deployment Diagram Class Object (Instance) Attribute	Package Profile Stereotype Interface Package Diagram Signal Operation Port	Block Part Value Property Units SI Units Model Library Block Definition Diagram Internal Block Diagram Proxy port Full port Interface Block
Functional (declarative)		Use case Use Case Diagram	Requirement Requirement Diagram Requirement Table Allocation Matrix
Behavioral (imperative)	Communication Diagram Timing Diagram Interaction Overview	State Diagram Activity Diagram Sequence Diagram State Event Action Control Flow	«continuous» «discrete» «control» «probability»
Parametric		Constraint	Constraint Block Parametric Diagram Parametric Constraint

At the UML 101/SysML 101 level, they are *the same*, except some elements are renamed

A Priori Systems
Real-Time Agile Systems and Software

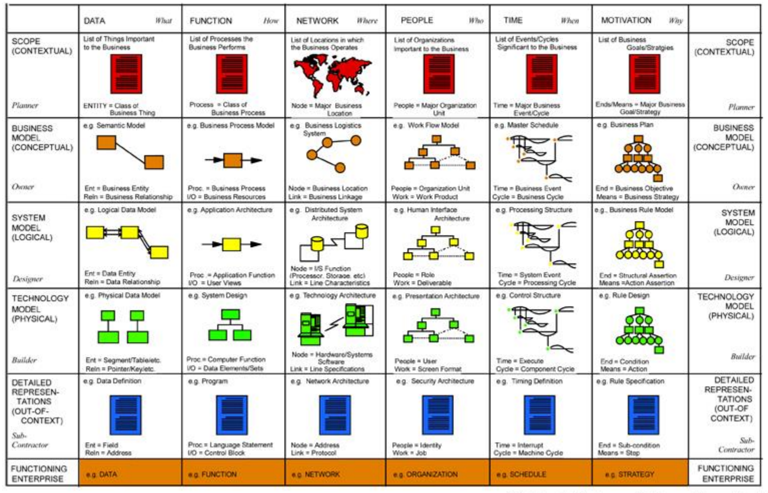
Architecture Frameworks

DnDAF



TOGAF

Zachman Framework



ENTERPRISE ARCHITECTURE - A FRAMEWORK ™

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*	
SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the Business ENTITY = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organization Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	List of Business Goals/Stratgies Ends/Means = Major Business Goal/Strategy	SCOPE (CONTEXTUAL) *Planner*
BUSINESS MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model Ent = Business Entity Reln = Business Relationship	e.g. Business Process Model Proc. = Business Process I/O = Business Resources	e.g. Business Logistics System Node = Business Location Link = Business Linkage	e.g. Work Flow Model People = Organization Unit Work = Work Product	e.g. Master Schedule Time = Business Event Cycle = Business Cycle	e.g. Business Plan End = Business Objective Means = Business Strategy	BUSINESS MODEL (CONCEPTUAL) *Owner*
SYSTEM MODEL (LOGICAL) *Designer*	e.g. Logical Data Model Ent = Data Entity Reln = Data Relationship	e.g. Application Architecture Proc .= Application Function I/O = User Views	e.g. Distributed System Architecture Node = I/S Function (Processor, Storage, etc) Link = Line Characteristics	e.g. Human Interface Architecture People = Role Work = Deliverable	e.g. Processing Structure Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model End = Structural Assertion Means =Action Assertion	SYSTEM MODEL (LOGICAL) *Designer*
TECHNOLOGY MODEL (PHYSICAL) *Builder*	e.g. Physical Data Model Ent = Segment/Table/etc. Reln = Pointer/Key/etc.	e.g. System Design Proc.= Computer Function I/O = Data Elements/Sets	e.g. Technology Architecture Node = Hardware/Systems Software Link = Line Specifications	e.g. Presentation Architecture People = User Work = Screen Format	e.g. Control Structure Time = Execute Cycle = Component Cycle	e.g. Rule Design End = Condition Means = Action	TECHNOLOGY MODEL (PHYSICAL) *Builder*
DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*	e.g. Data Definition Ent = Field Reln = Address	e.g. Program Proc.= Language Statement I/O = Control Block	e.g. Network Architecture Node = Address Link = Protocol	e.g. Security Architecture People = Identity Work = Job	e.g. Timing Definition Time = Interrupt Cycle = Machine Cycle	e.g. Rule Specification End = Sub-condition Means = Step	DETAILED REPRESEN-TATIONS (OUT-OF-CONTEXT) *Sub-Contractor*
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

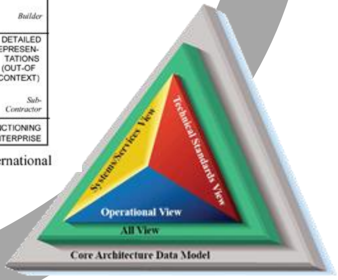
ⓒ John A. Zachman, Zachman International

NAF

UAF®

OMG UNIFIED ARCHITECTURE FRAMEWORK®

MODAF



UPDM

DoDAF

Definition: An architecture framework is an encapsulation of a minimum set of practices and requirements for artifacts that describe a system's architecture.

A Priori Systems
Real-Time Agile Systems and Software

What is UML?

- Unified Modeling Language
- Diagrams and views
- Model elements
- 4-Tier Metamodel Architecture

UNIFIED MODELING LANGUAGE™ UML®

What is UML?

- Unified Modeling Language see http://www.omg.org/spec/UML/2.5/PDF/
- Comprehensive full life-cycle 3rd Generation modeling language
  - Standardized in 1997 by the Object Management Group (OMG)
  - Incorporates state of the art Software and Systems development concepts
- Matches the growing complexity of real-time systems
  - Large scale systems, Networking, Web enabling, Data management
- Extensible and configurable
- UML supports but doesn't require object-oriented development
- UML is process agnostic
  - By design, the UML is meant to be used with any reasonable development process

A Priori Systems
Real-Time Agile Systems and Software

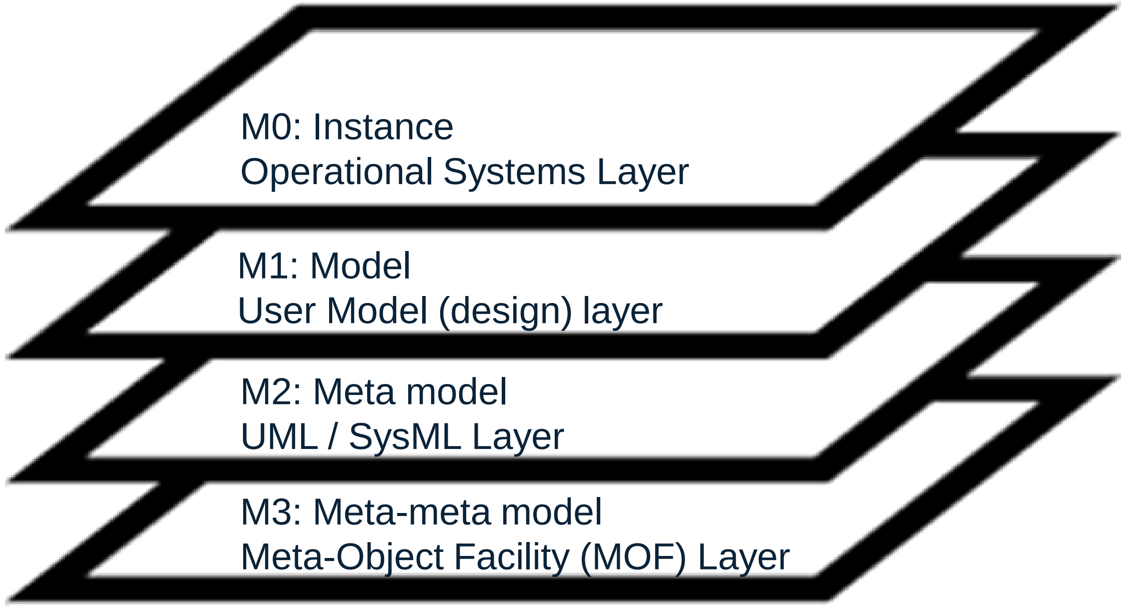
UML Features

- UML is a graphical language
  - **Diagrams** form the primary means by which models are created and understood
  - **Packages** are folders that contain model elements including both diagrams and the elements they portray.
    - This applies to the UML itself but also to the user models (designs) you create
  - The key is the underlying semantic repository of information about the system you're modeling
  - A **diagram** type is defined by the types of things that can be represented and their symbology
  - A **diagram usage** is the purpose for a diagram, which subsets the kinds of elements used
  - Example:
    - A class diagram is a type of UML diagram
    - Uses of class diagrams: class, structure, object, package, task, subsystem, architecture, interface

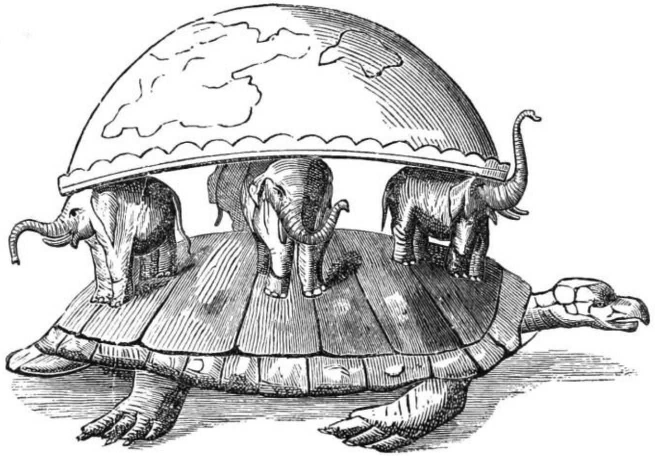
A Priori Systems
Real-Time Agile Systems and Software

UML Semantic basis

- UML is constructed using a 4-tier metamodel hierarchy
  - M3 – Meta-metamodel (MOF Core language)
  - M2 – Metamodel (UML Language)
  - M1 – Design model (model)
  - M0 – Instance model (deployed system)
- The UML definition itself is divided up into packages to support
  - Modularity
  - Layering
  - Partitioning
  - Extensibility
  - Reusability

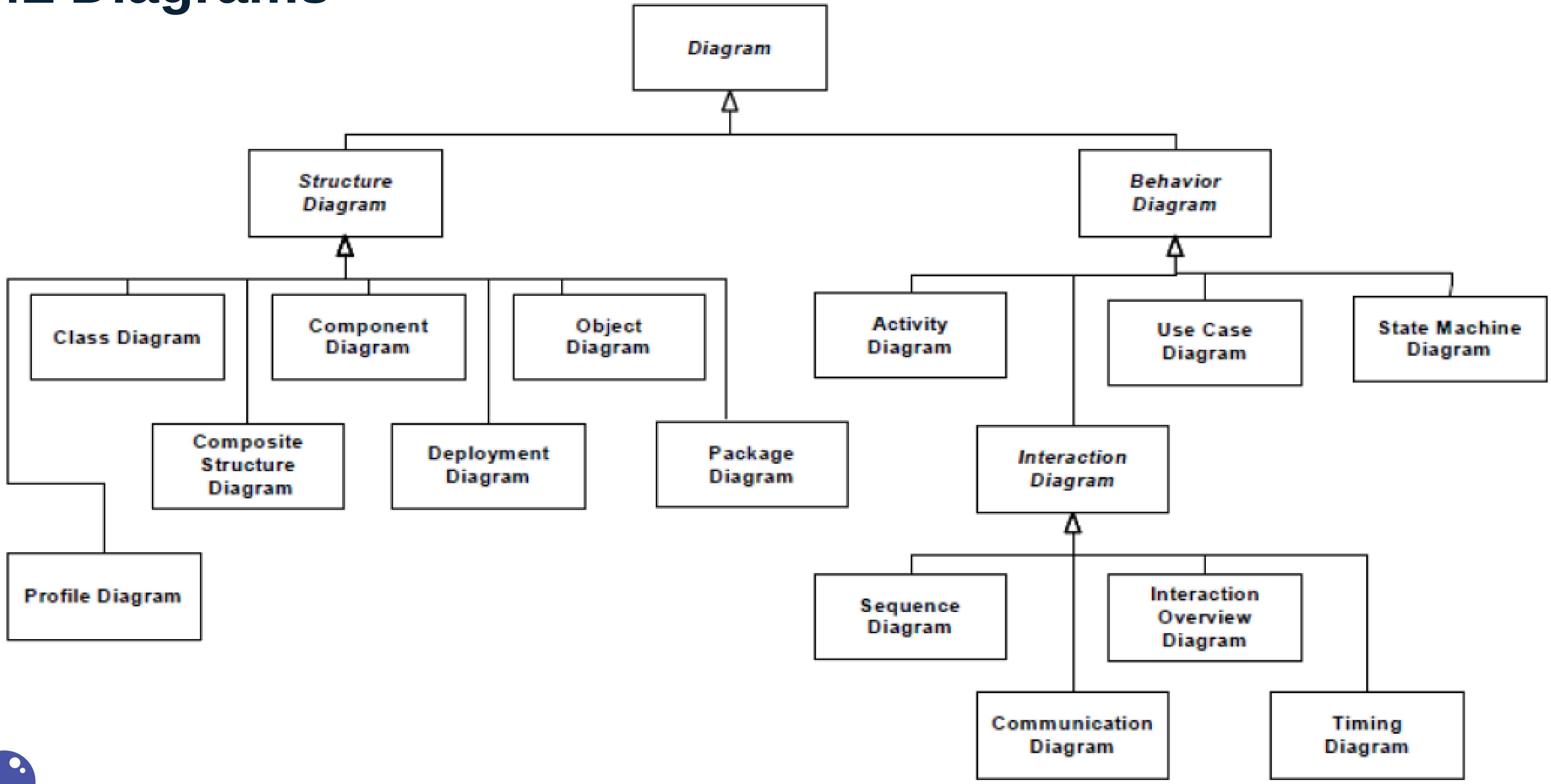


"It's Meta-Turtles all the way down"



A Priori Systems
Real-Time Agile Systems and Software

UML Diagrams



From UML 2.51 OMG Document Number formal/2017-12-05

A Priori Systems
Real-Time Agile Systems and Software

What is SysML?

SysML is derived from UML

SysML Timeline

UML vs SysML

A Priori Systems
Real-Time Agile Systems and Software

What is SysML?

- A graphical modeling language in response to the UML for Systems Engineering RFP developed by the Object Management Group (OMG), International Council on Systems Engineering (INCOSE), and AP233
  - a UML Profile that is both a subset and extension to UML 2
- Designed specifically for the Systems Engineering domain with extensions for requirements and analysis
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- SysML is the most common way to represent systems engineering information in a rigorous, structured way by storing the information in **models.** We discuss models in more detail shortly.
- The pervasive application of models for systems engineering is known as Model-Based Systems Engineering (MBSE)

Important! **At a basic level of use, UML and SysML are the same language,** with only minor naming differences between them.

- More advanced uses of SysML will highlight the differences between them.

A Priori Systems
Real-Time Agile Systems and Software

Like UML, SysML is a *language*
and is process-agnostic.

SysML History

UML 1.1 Adopted by the Object Management Group (OMG)

1995

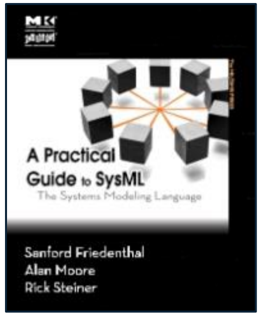
Bruce Douglass and Peter Hoffmann release **Harmony Systems Engineering** (Harmony SE) process

Initial release of SysML for adoption

2001

Work begun on SysML

2003

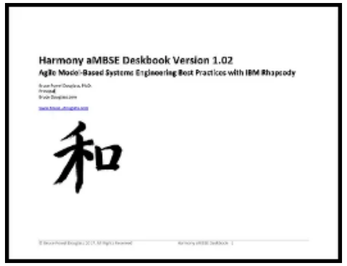


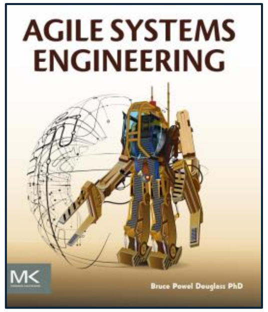
Friedenthal et. al. release **A Practical Guide to SysML**

2006

SysML 1.0 Adopted by the OMG

Bruce Douglass releases **Harmony Agile Model-Based Systems Engineering** (Harmony aMBSE) process



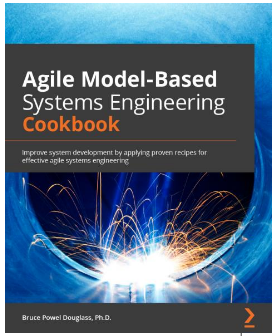


Bruce Douglass publishes **Agile Systems Engineering** book

Work begun on SysML 2.0

2019

SysML 1.6 released



Bruce Douglass publishes **Agile Mode-Based Systems Engineering Cookbook**

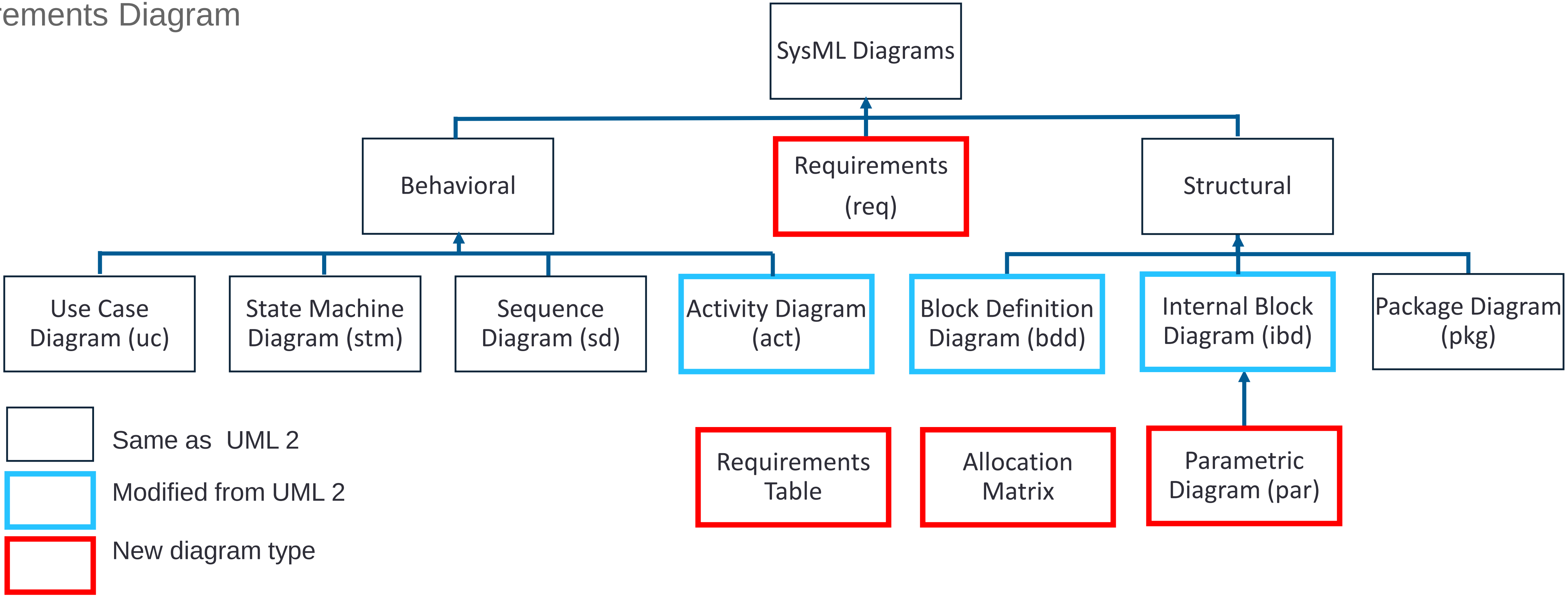
2021

Click here to learn about the latest release of SysML
https://www.omg.org/spec/SysML/About-SysML/

Nine SysML Views

The nine SysML diagrams are categorized as follows:

– Behavioral Diagrams - dynamic change of system **behavior** over time
– Structural Diagrams - static system **structure** diagrams
– Requirements Diagram



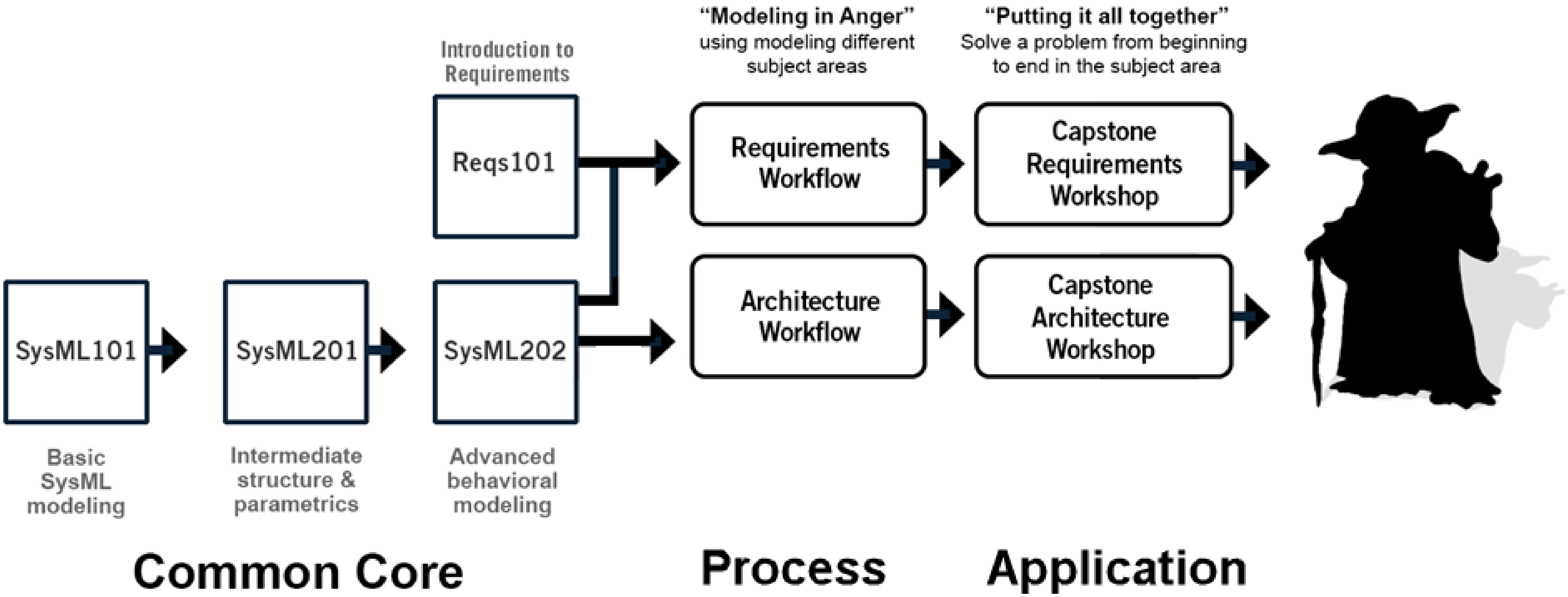
A Priori Systems
Real-Time Agile Systems and Software

Characteristics of Predefined SysML Views

View	Type	UML2 Analog	Lifecycle usage	Essential	Dynamic simulation	Computational	Supports code gen	Formal
Requirements Diagram (req)	Static Functionality	n/a	Requirements Specification; Functional Analysis					
Use Case Diagram (uc)	Static Functionality	Use case diagram	Requirements Specification; Functional Analysis	✓				
Activity Diagram (act)	Dynamic Behavior	Activity diagram – minor changes	All	✓	✓		✓	✓
Sequence Diagram (sd)	Interaction Behavior	Sequence Diagram	All	✓	✓			✓
State Diagram (stm)	Dynamic Behavior	State Diagram	All	✓	✓		✓	✓
Block Definition Diagram (bdd)	Static Structure	Class Diagram (moderate change)	Architecture; Design	✓			✓	✓
Internal Block Diagram (ibd)	Static Structure	Structure Diagram (moderate change)	Architecture; Design	✓			✓	✓
Parametric Diagram (par)	Static Functionality	n/a	All			✓		✓
Package Diagram (pkg)	Static Structure	Package diagram	All					
Requirements Table	Static Table	n/a	Requirements Specification; Functional Analysis					
Allocation Matrix	Static Matrix	n/a	All					

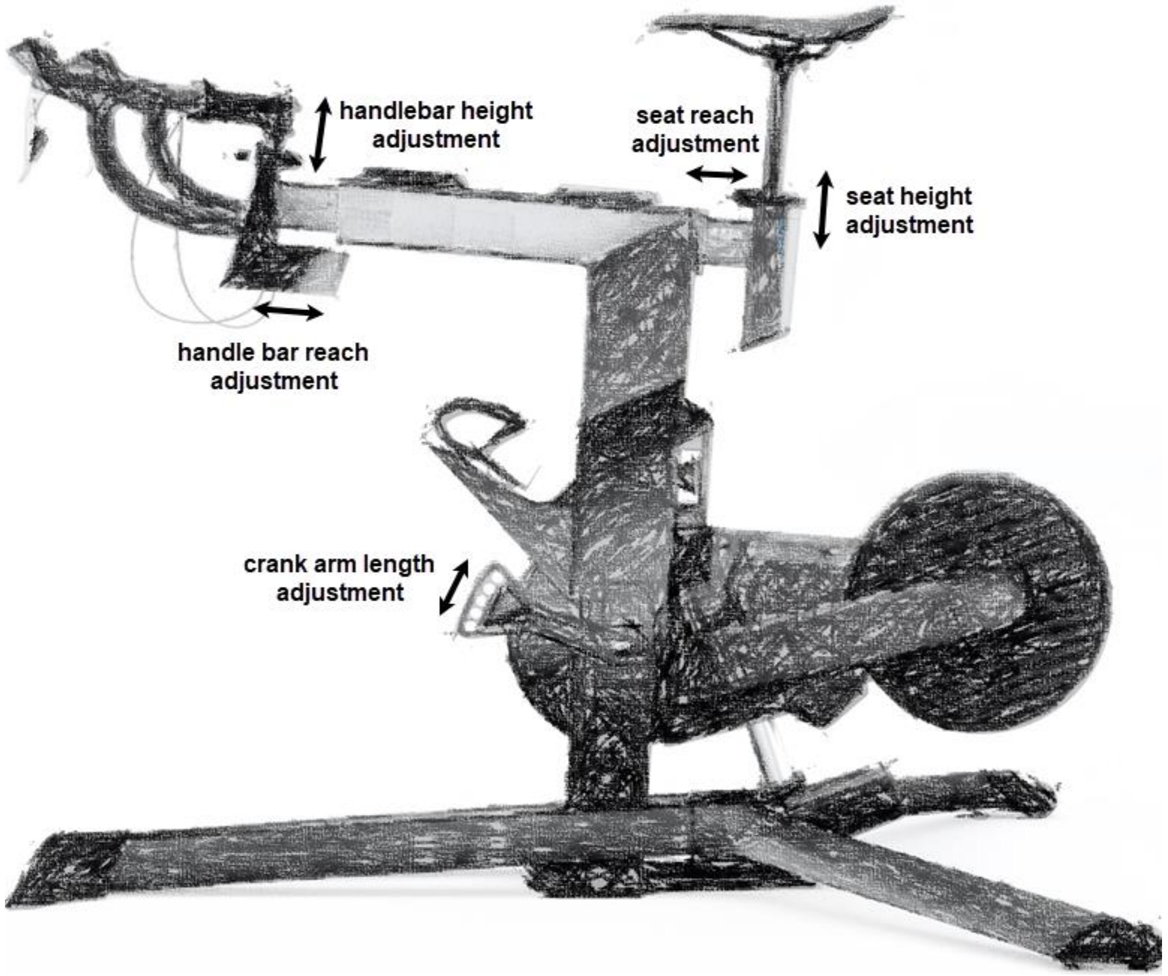
A Priori Systems
Real-Time Agile Systems and Software

Learning SysML: The *A Priori* Curriculum



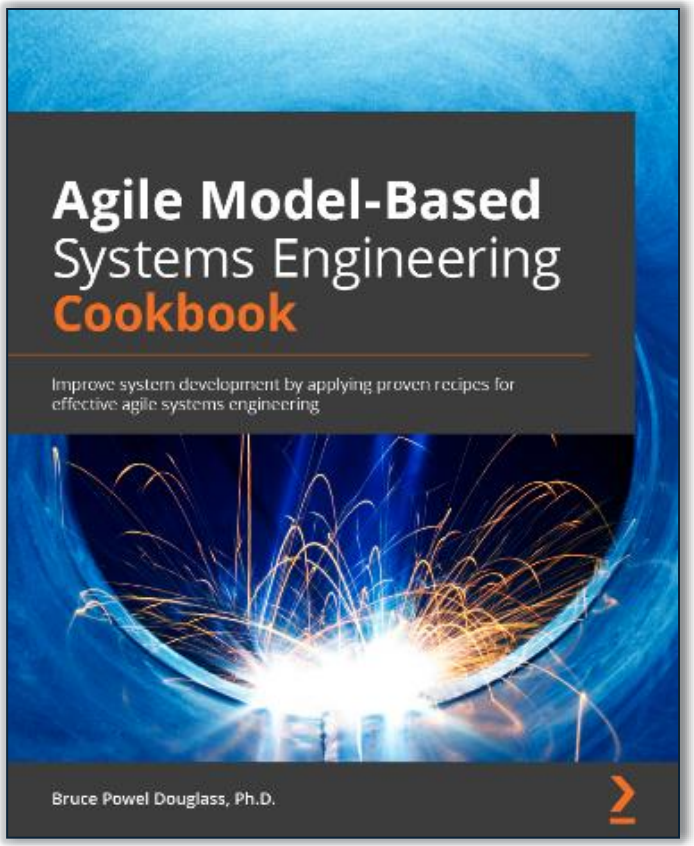
Example

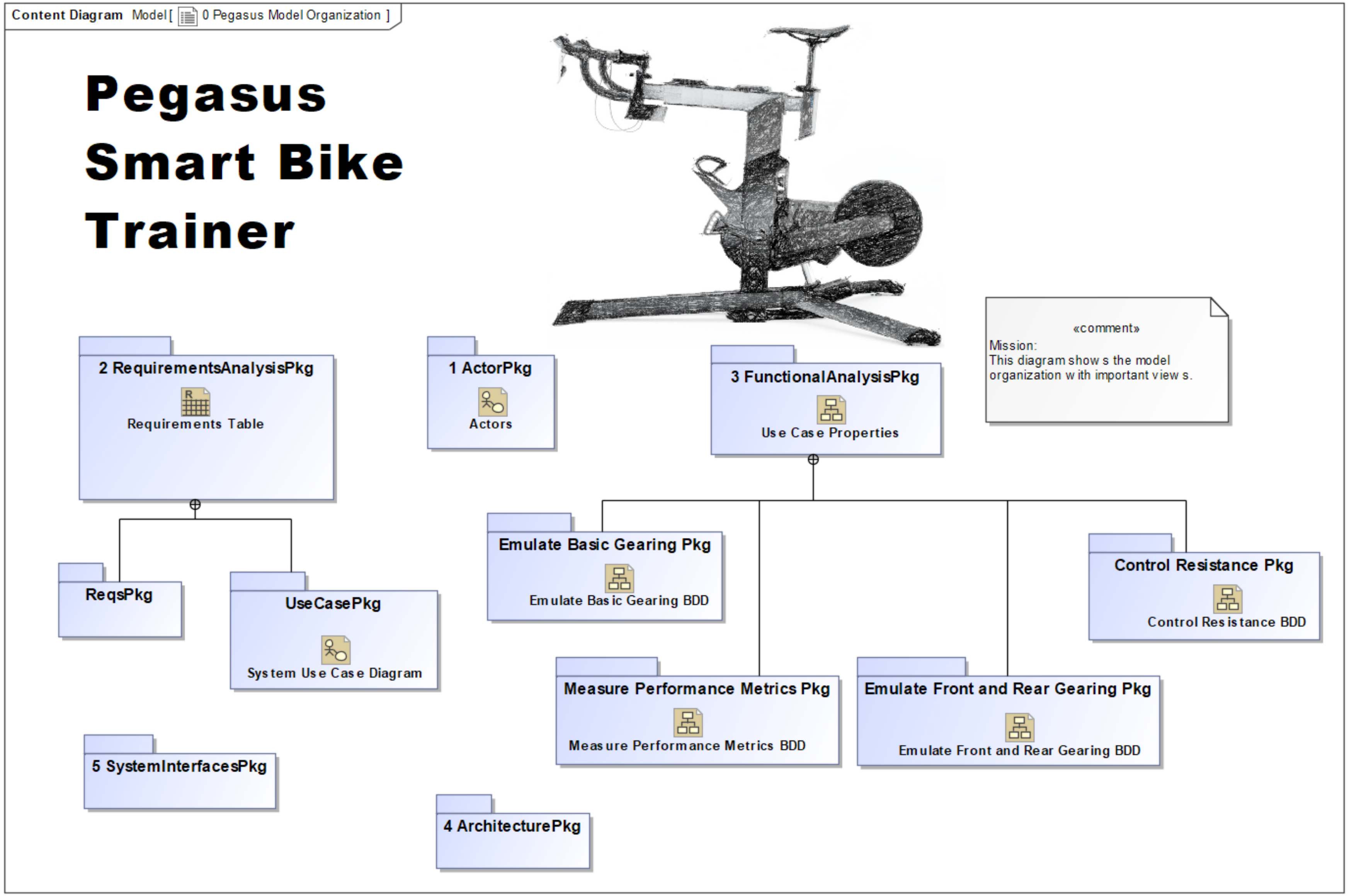
A quick look at the Pegasus Smart Bike Trainer



A Priori Systems
Real-Time Agile Systems and Software

Model Overview Diagram





Adapted to Cameo Magic Draw from the Rhapsody models within the book

A Priori Systems
Real-Time Agile Systems and Software

Requirements Table

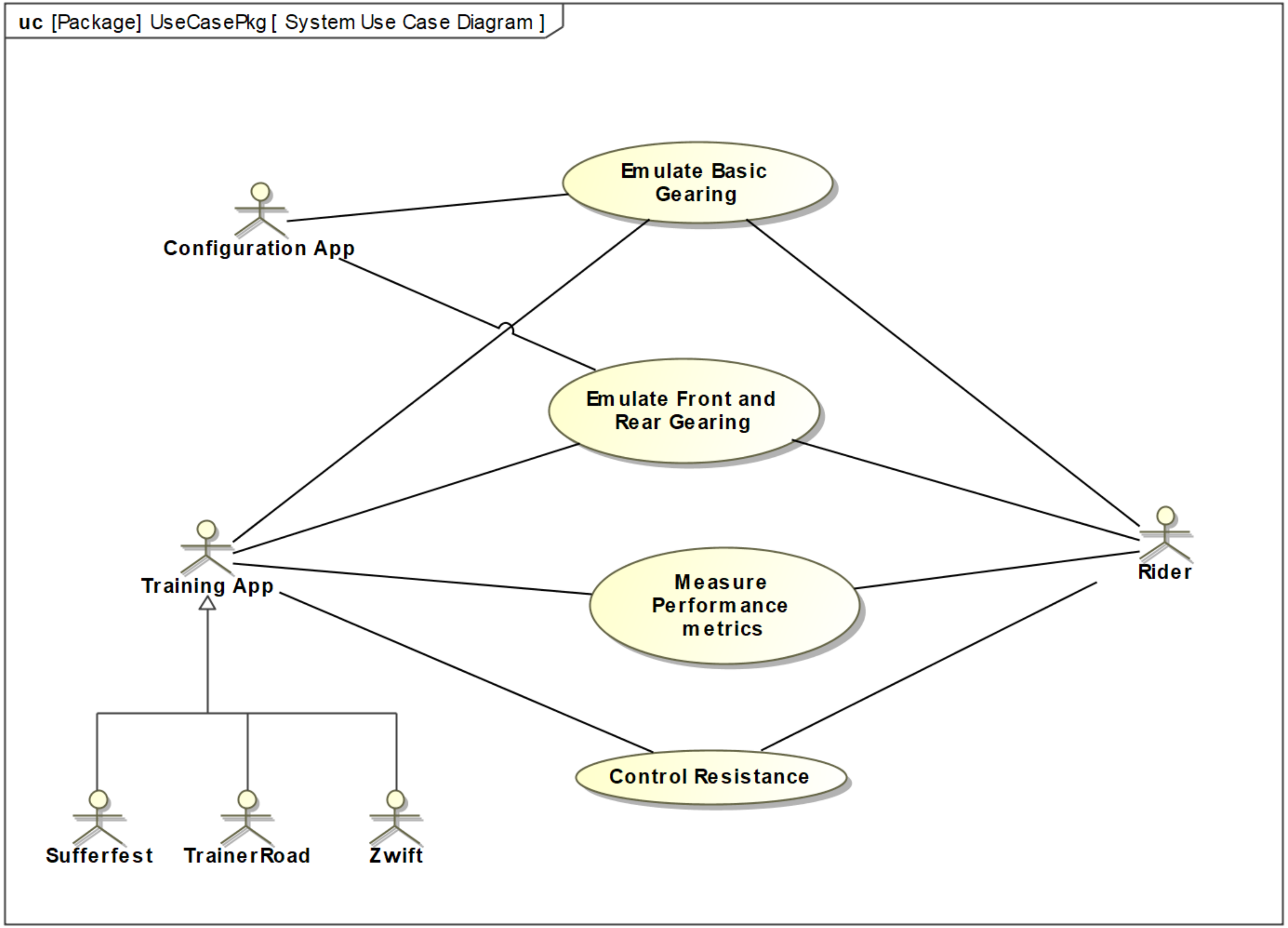
Criteria

Scope (optional): ReqsPkg ... Filter:

#	Name	Text
1	R 63 DIReg01	In DI Shifting mode, if the UP button is pressed and the system is already in the highest possible gear, then the system shall audibly beep and keep the current gearing.
2	R 64 DIReg02	The system shall provide shifting with a DI Shifting mode that enables the DI shifting buttons and disables the shifting levers.
3	R 65 DIReg03	In DI Shifting mode, the UP button shall shift into the next highest possible gearing from the selected gear set, as measured in gear inches.
4	R 66 DIReg04	In DI Shifting mode, the UP button shall shift into the next highest possible gearing from the selected gear set, as measured in gear inches.
5	R 67 DIReg05	In DI Shifting mode, if the DOWN button is pressed and the system is already in the lowest possible gear, then the system shall audibly beep and keep the current gearing.
6	R 68 DIReg06	In DI Shifting mode, when an upshift requires changing the chain ring, the system shall progress to the next largest gearing, as measured by gear inches.
7	R 69 DIReg07	In DI Shifting mode, when a downshift requires changing the chain ring, the system shall progress to the next smallest gearing, as measured by gear inches.
8	R 70 DIReq10	The system shall enter DI Shifting Mode by selecting that option in the Configuration App.
9	R 71 DIReq11	Once DI Shifting mode is selected, this selection shall persist across resets, power resets, and software updates.
10	R 72 DIReq12	Mechanical shifting shall be the default on initial start up or after a factory-settings reset.
11	R 73 DIReq13	The system shall leave DI Shifting mode when the user selects the Mechanical Shifting option in the Configuration App.
12	R 74 efarg01	The system shall notify the rider of the current number of chain rings and cassette rings on start up.
13	R 75 efarg02	The system shall accept a rider command to enter a mode to configure the gearing.
14	R 76 efarg03	The system shall accept a rider command to set up from 1 to 3 front chain rings, inclusive.
15	R 77 efarg04	The default number of chain rings shall be 2.
16	R 78 efarg05	The rider shall be able to decrement the cassette ring from a higher (smaller number of teeth) to the next lower (larger number of teeth) gear until the largest cassette ring is reached.
17	R 79 efarg06	The system shall accept a rider command to set up from 10-12 cassette rings, inclusive.
18	R 80 efarg07	The default number of cassette rings shall be 12.
19	R 81 efarg08	The system shall accept a rider command to set any chain ring to have from 20 to 70 teeth.
20	R 82 efarg09	The system shall accept a rider command to set up any cassette ring to have from 10 to 50.
21	R 83 efarg10	The default number of teeth for 1 chain ring shall be 48.
22	R 84 efarg11	The default number of teeth for 2 chain rings shall be 34 and 53.

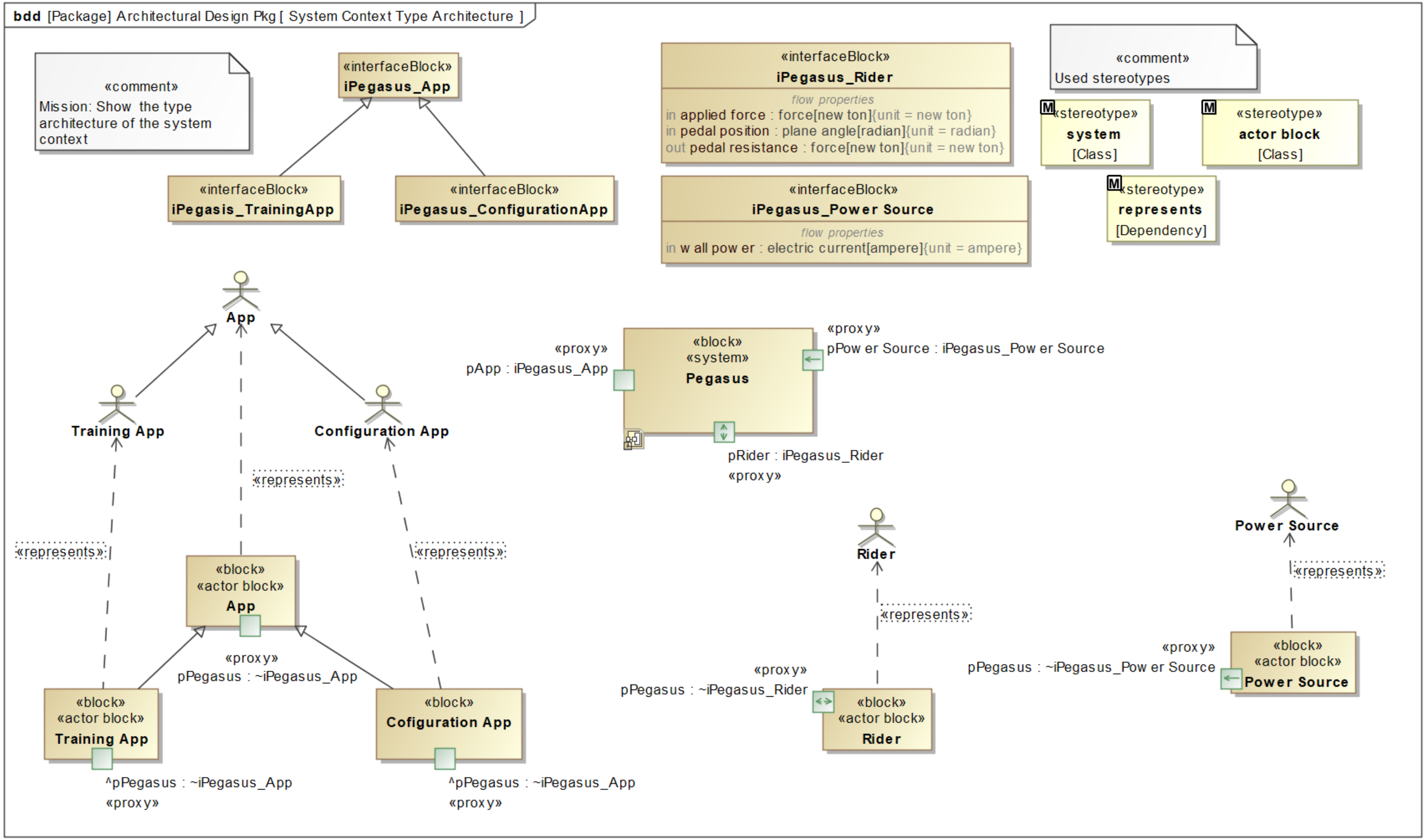
A Priori Systems
Real-Time Agile Systems and Software

Use Case Diagram



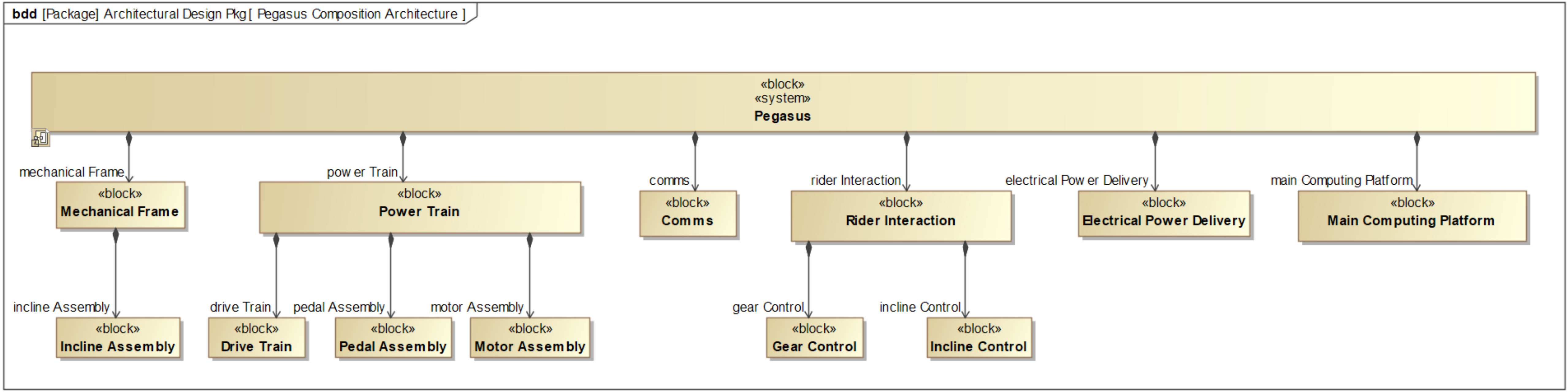
A Priori Systems
Real-Time Agile Systems and Software

Type Context



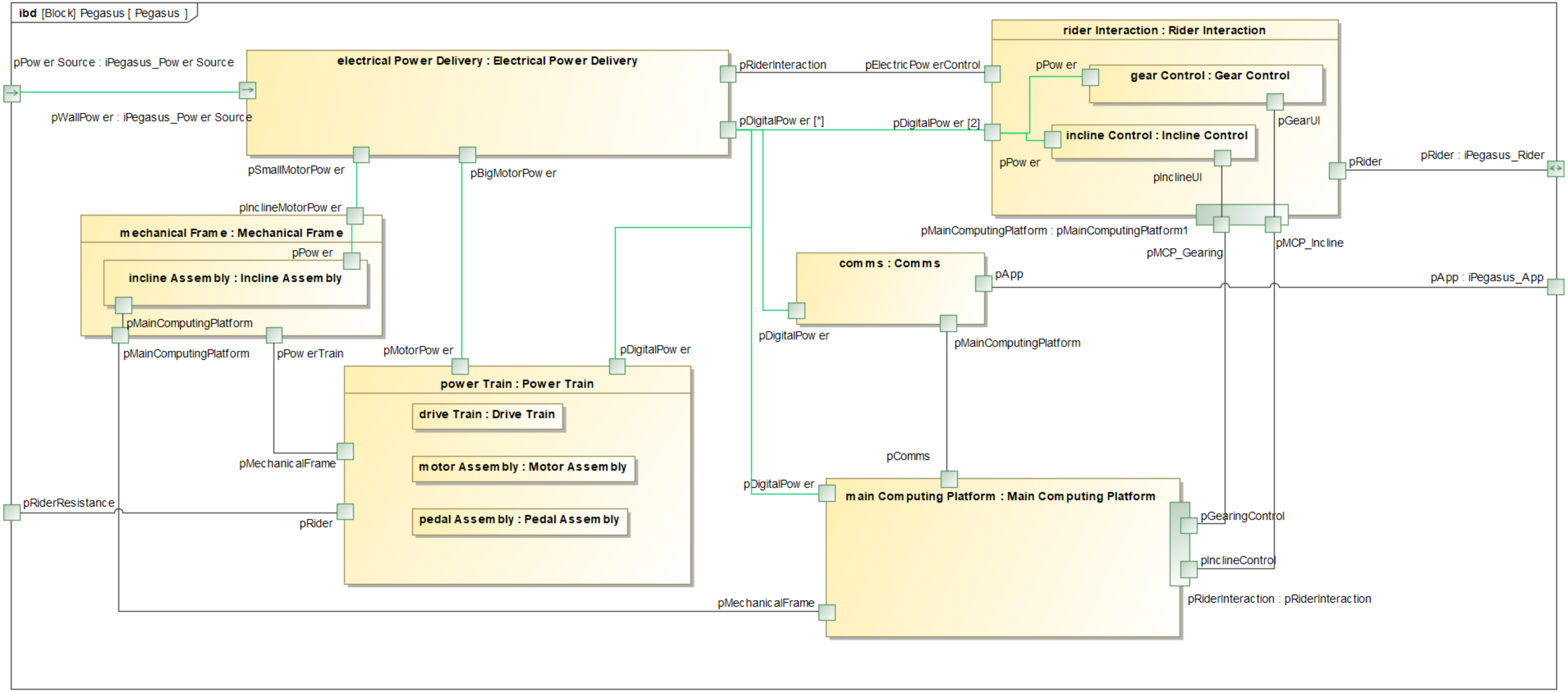
A Priori Systems
Real-Time Agile Systems and Software

Type Composition Architecture



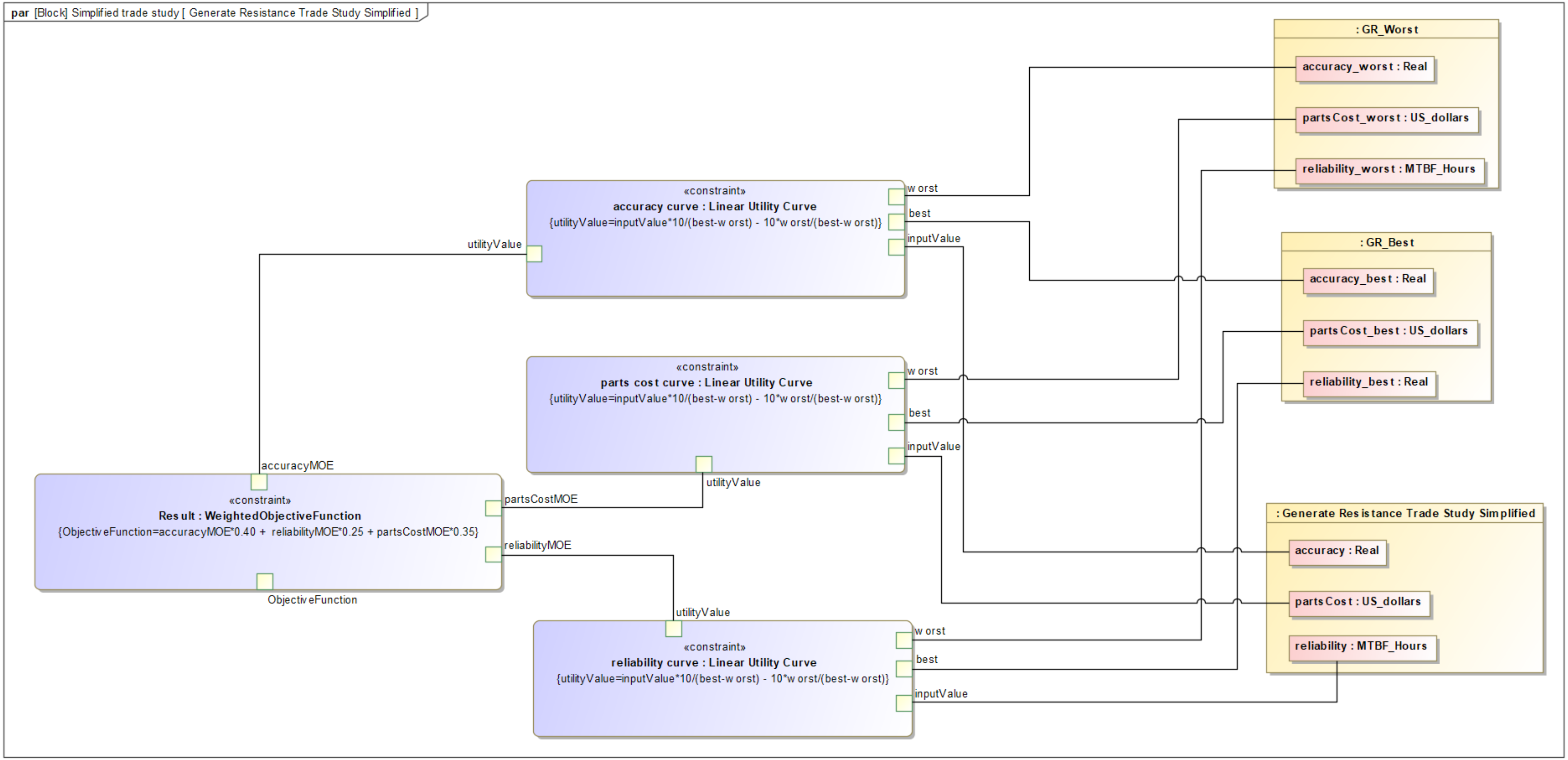
A Priori Systems
Real-Time Agile Systems and Software

Connected Architecture



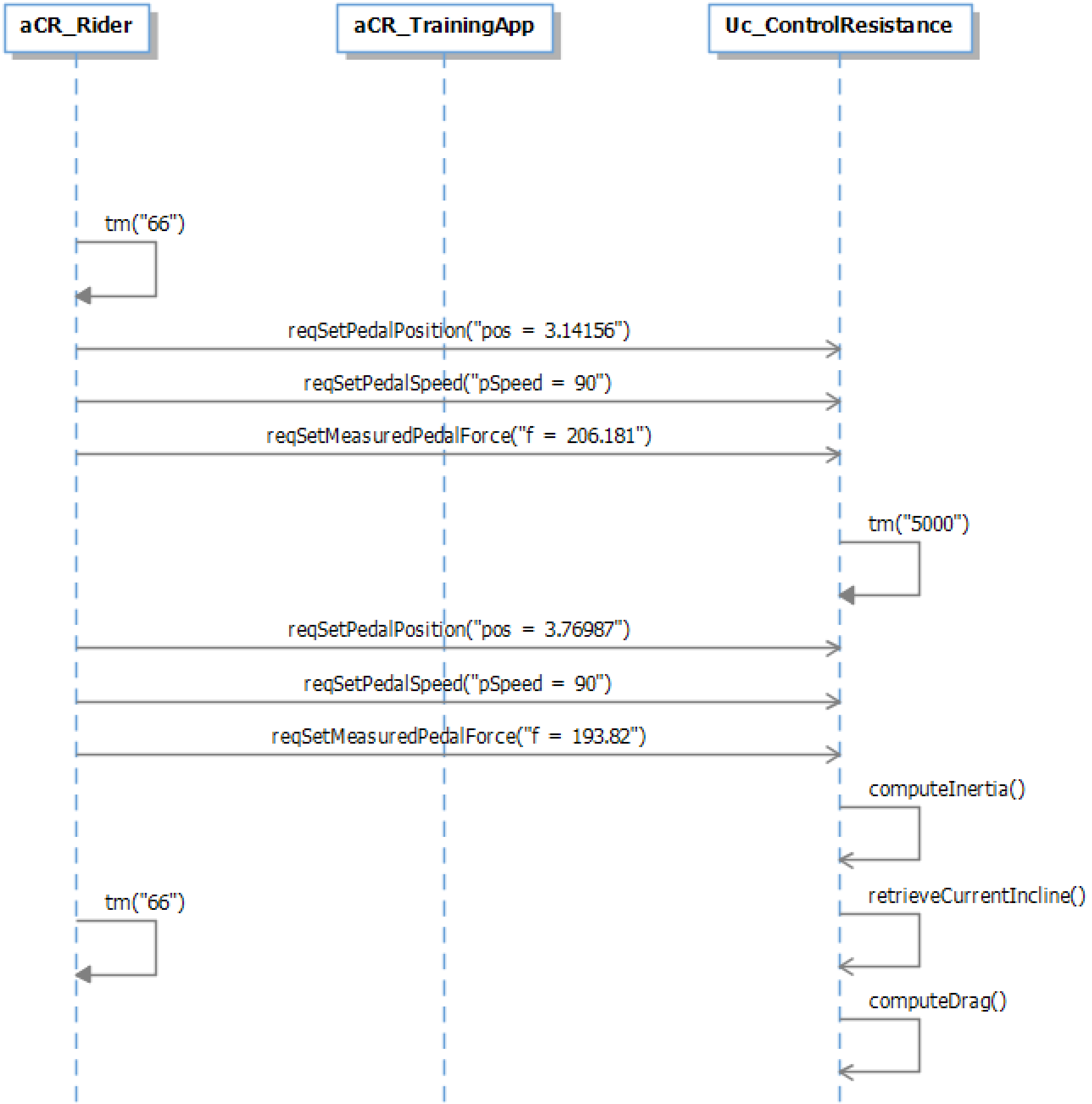
A Priori Systems
Real-Time Agile Systems and Software

Motor Selection Trade Study



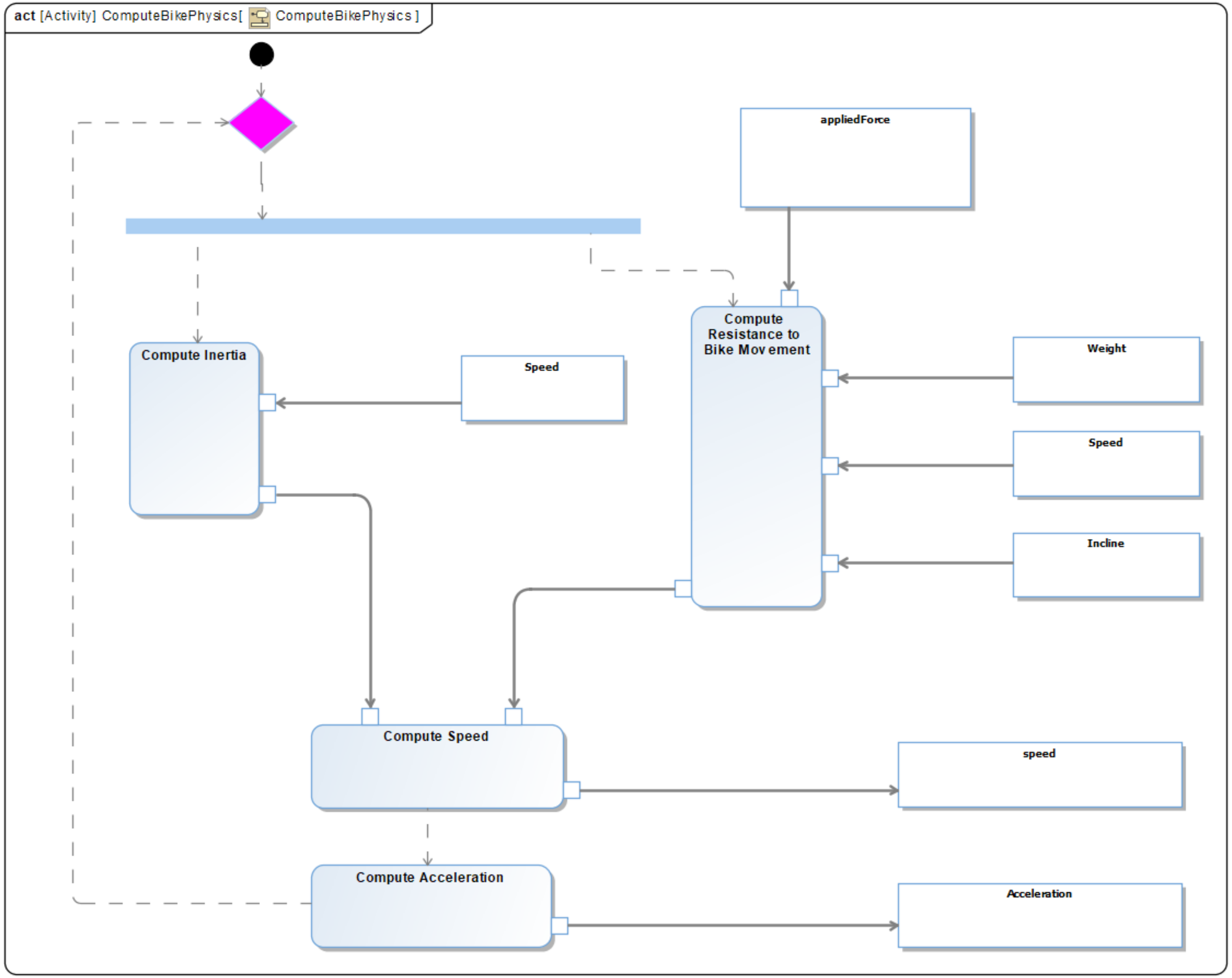
A Priori Systems
Real-Time Agile Systems and Software

An interaction



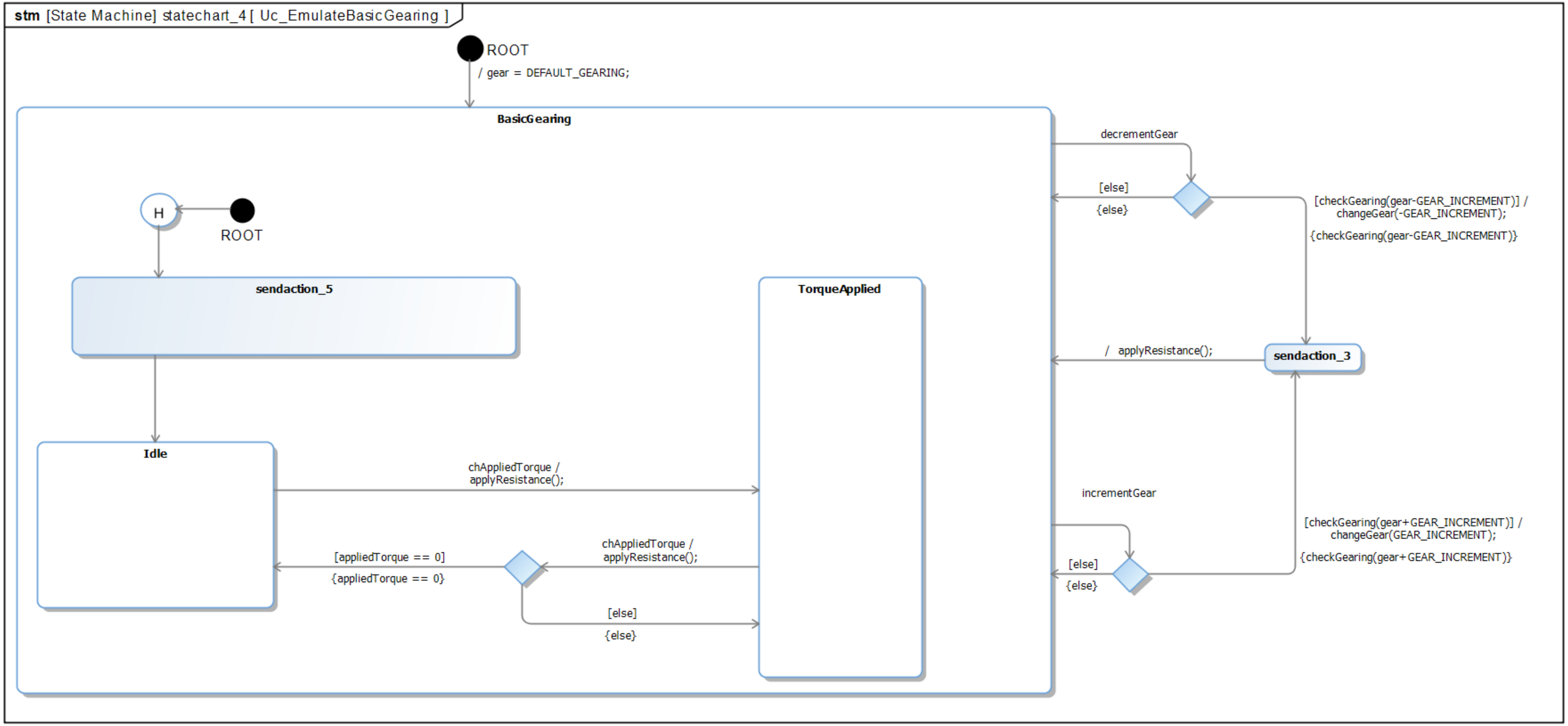
A Priori Systems
Real-Time Agile Systems and Software

Flow of control behavior



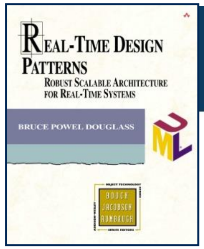
A Priori Systems
Real-Time Agile Systems and Software

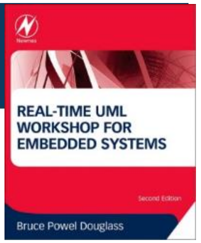
Some state behavior

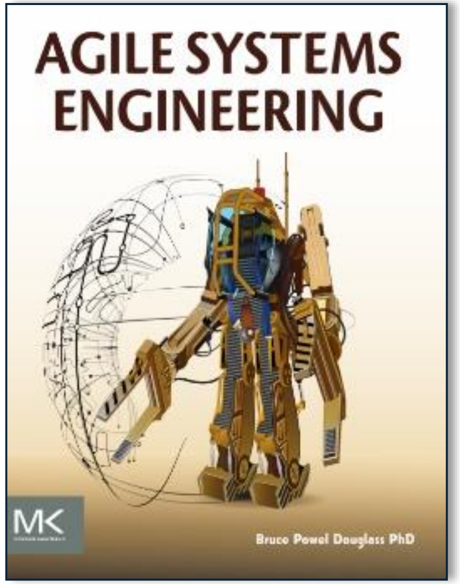


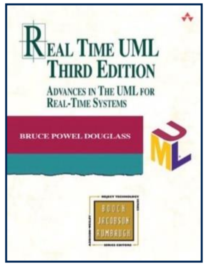
A Priori Systems
Real-Time Agile Systems and Software

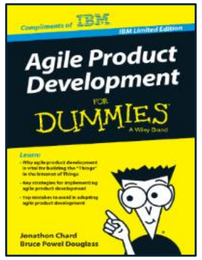
For more information

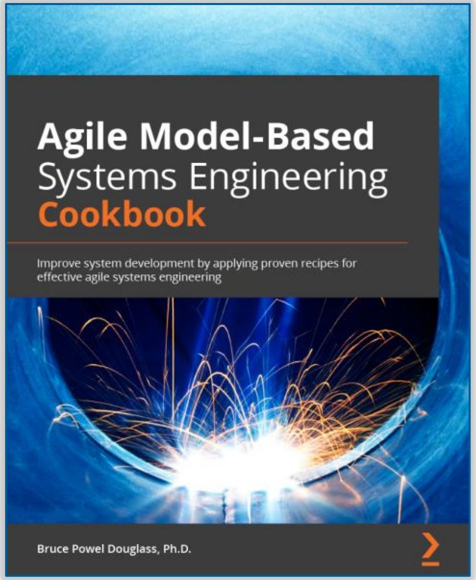


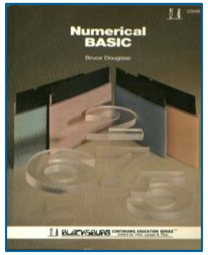


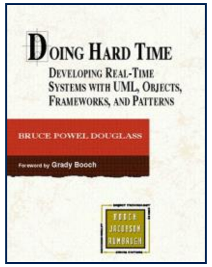


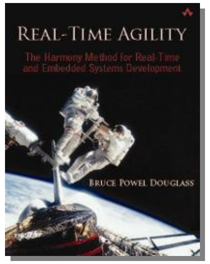


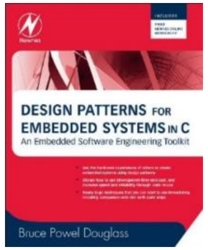












A Priori Systems
Real-Time Agile Systems and Software