Language Shapes (Architectural) Thought

Markus Völter

voelter { ingenieurbüro für softwaretechnologie //

voelter@acm.org
www.voelter.de
@markusvoelter

Language

Shapes

Thought

Sapir–Whorf hypothesis
aka Whorfianism

The principle of linguistic relativity holds that the structure of a language affects the ways in which its respective speakers conceptualize their world, i.e. their world view, or otherwise influences their cognitive processes.

Sapir–Whorf hypothesis
aka Whorfianism

The principle of linguistic relativity holds that the structure of an **architecture modeling** language affects the ways in which its users conceptualize an **architecture**.

Sapir–Whorf hypothesis
aka Whorfianism

1

What is Software Architecture

... the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them.

Wikipedia

A collection of software [..] components, connections, and constraints.

A collection of system stakeholders' need statements.

A rationale which demonstrates that [the system fulfils the needs]

Boehm et. al

… is its style and method of design and construction.

Hayes-Roth

...fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

ISO/IEC/IEEE 42010

> … the set of design decisions which, if made incorrectly, may cause your project to be cancelled.

Eoin Woods

… everything that must be consistent throughout a software system.

“strategic design” – Eric Evans

Mine

[Examples]

Class Structure

Components, Subsystems, Layers

Deployment onto Hardware

Dataflow Architecture (ETL)

Threading/Timing Architecture

Locking Protocol

It‘s not about Granularity.

And it‘s not just structure.

It‘s about consistency.

2

Why would you model Architecture?

[Communication]

between people

„Diagramming“

Doesn‘t count :-)

[Communication]
[Analysis]

finding flaws early

predicting properties

Timing

Concurrency

Bus Utilization

[Communication]

[Analysis]

[Checking]

model expected characteristics
and check against implement’n

Architecture Analysis

Archteology

[Communication]

[Analysis]

[Checking]

[Synthesis]

not just class skeletons!

generate other artifacts,
typically implementation code

UML Code Gen

Many DSLs

AUTOSAR

Communication
Analysis
Checking
Synthesis

Model Purpose

Relevant for any Modeling Language

Drives Selection/Design of Language

Requires Tool Support!

Just pictures doesn‘t cut it.

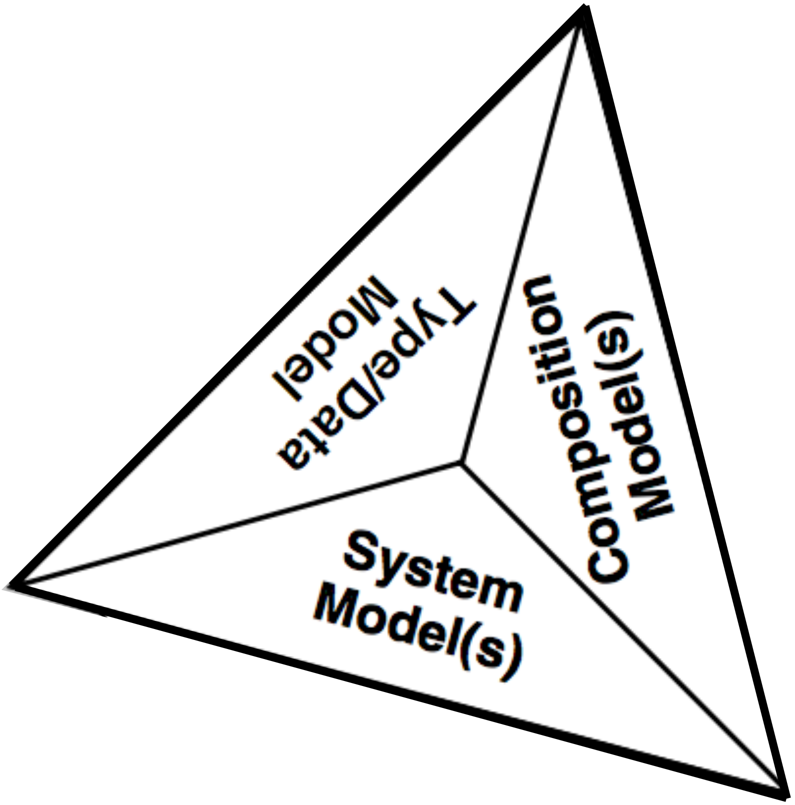
There‘s more than code gen.

Purpose determines Language!

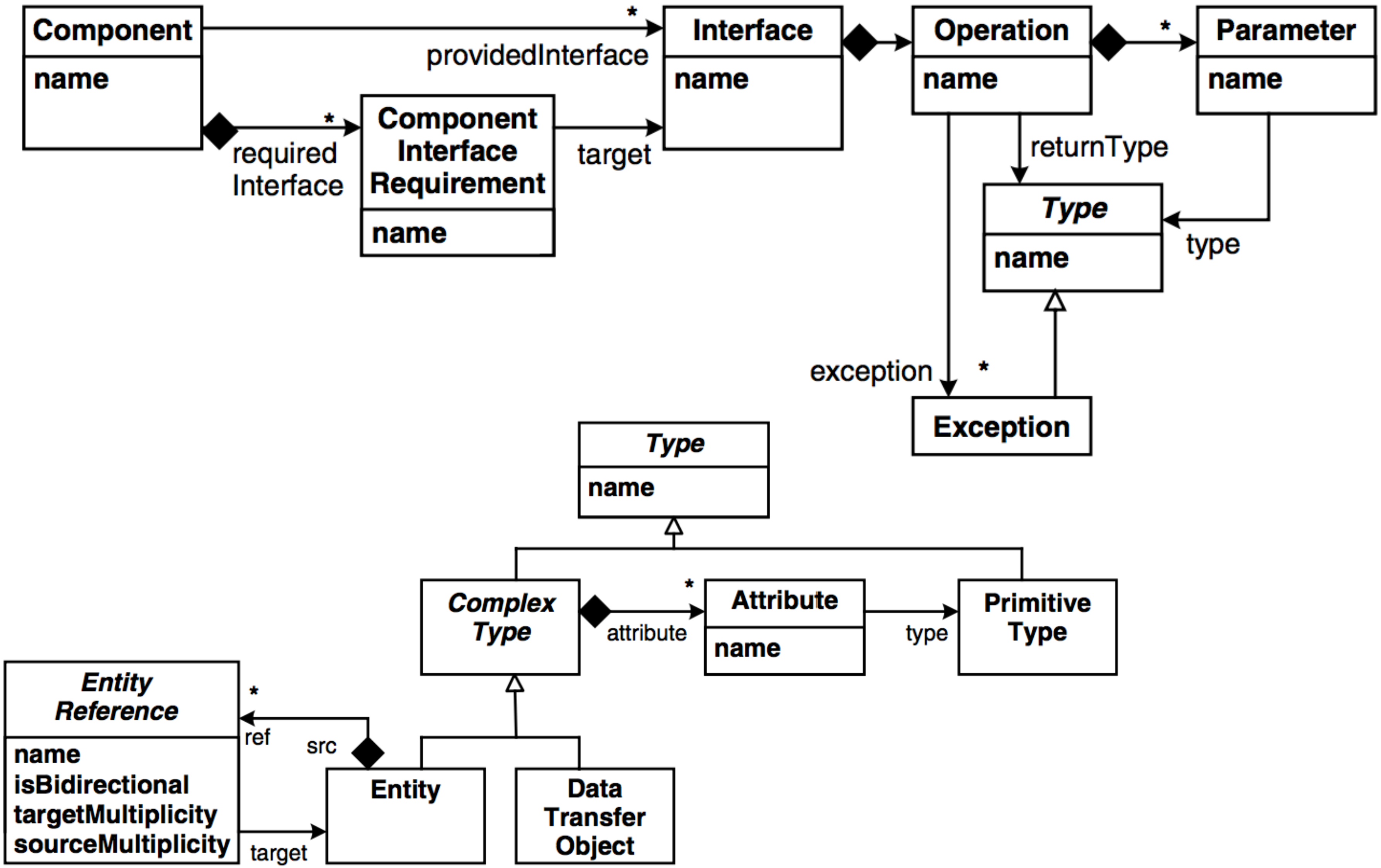
3

Separation Of Concerns

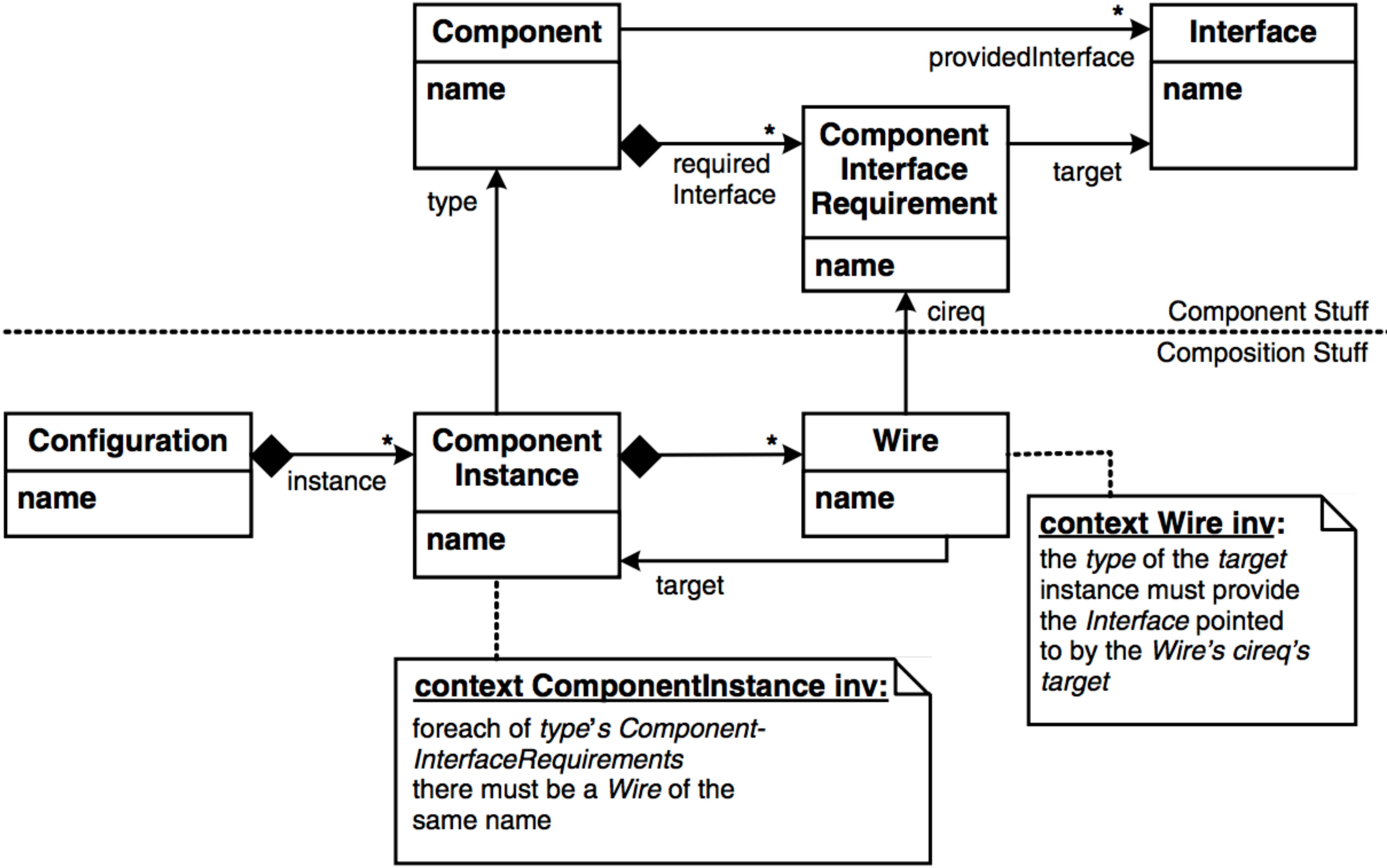
[Three Core Concerns]



[Types]


Component
name
*
providedInterface
Interface
name
Operation
name
*
Parameter
name
*
required Interface
Component Interface Requirement
name
target
returnType
Type
name
type
exception
*
Exception
Type
name
Complex Type
*
attribute
Attribute
name
type
Primitive Type
Entity Reference
name
isBidirectional
targetMultiplicity
sourceMultiplicity
*
ref
src
target
Entity
Data Transfer Object

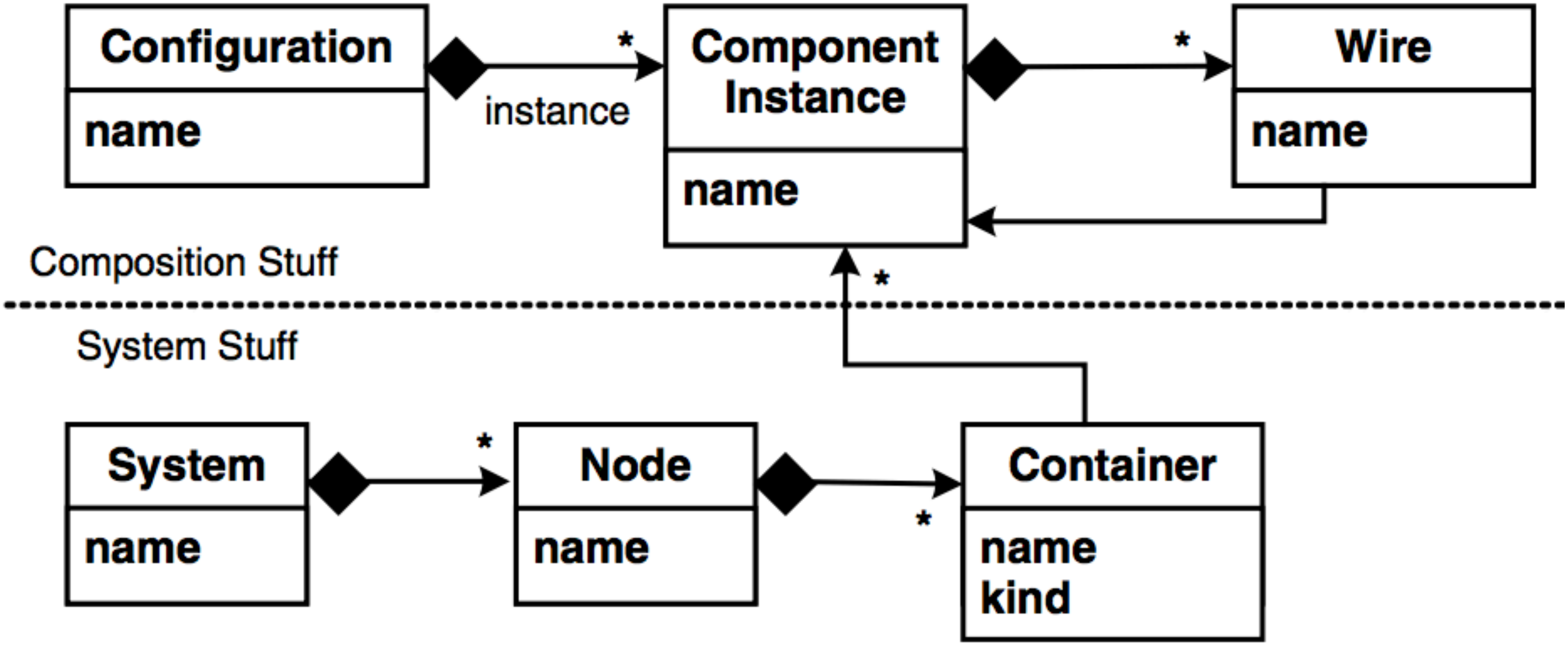
[Composition]



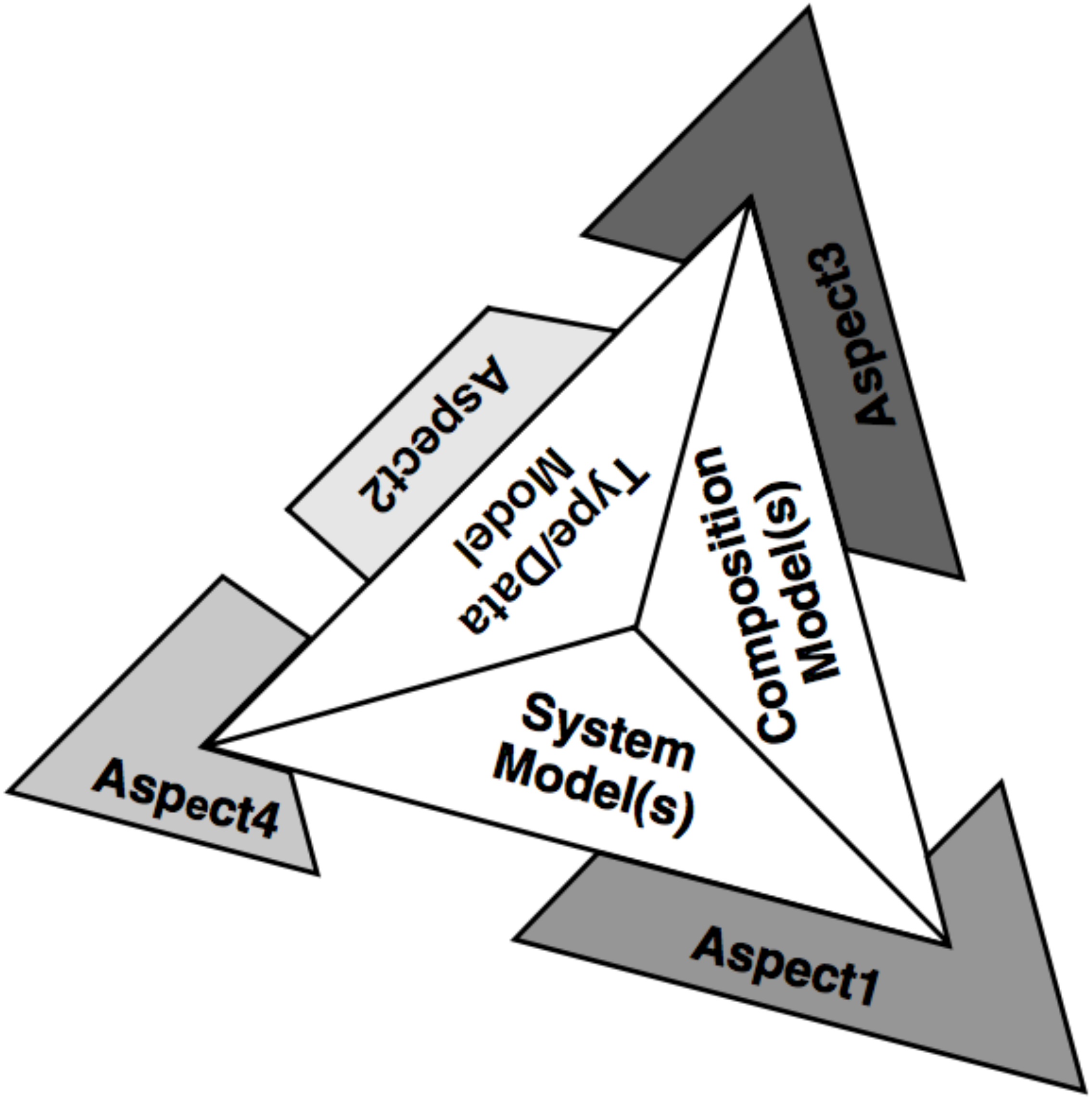
context Wire inv:
the *type* of the *target* instance must provide the *Interface* pointed to by the *Wire's cireq's target*

context ComponentInstance inv:
foreach of *type*'s *ComponentInterfaceRequirements*
there must be a *Wire* of the same name

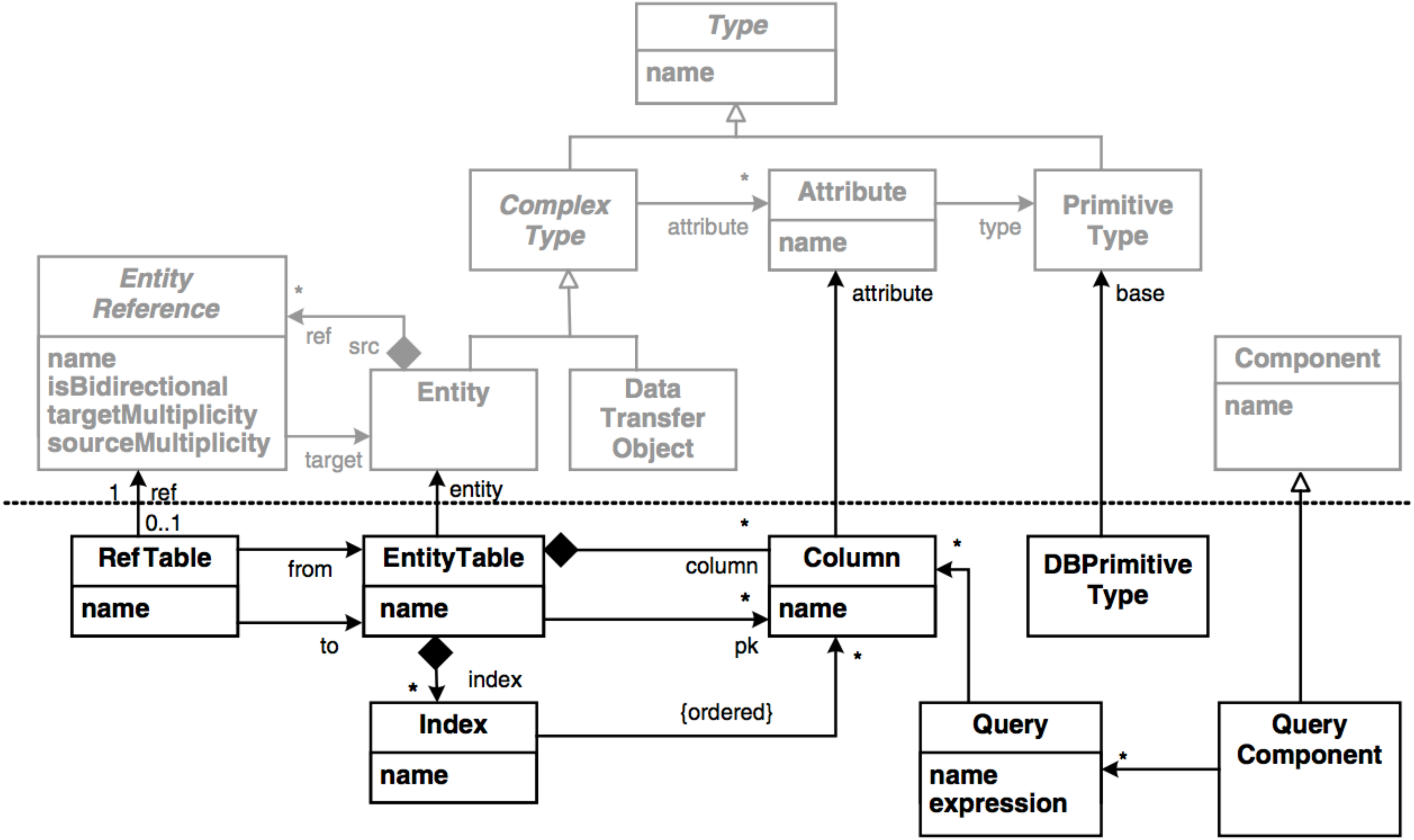
[Systems/Deployment]



[Additional Aspects*]

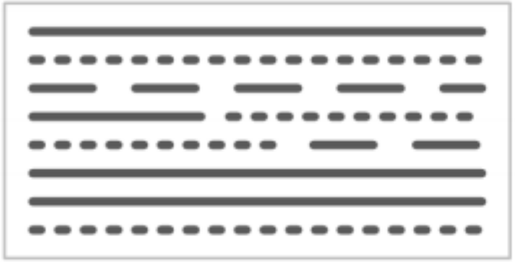


[Aspect: Persistence]

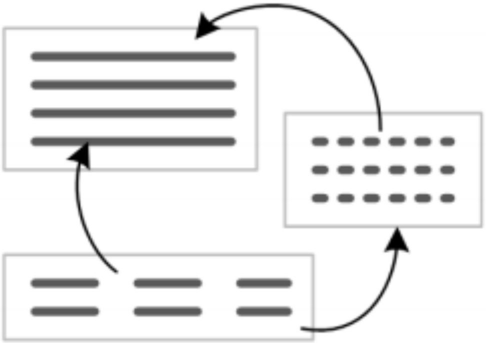


[Separation?]

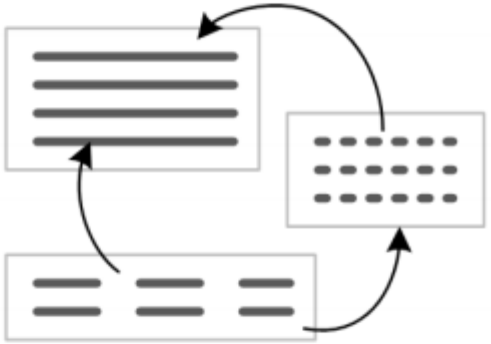
integrated into
one fragment



separated into
several fragments

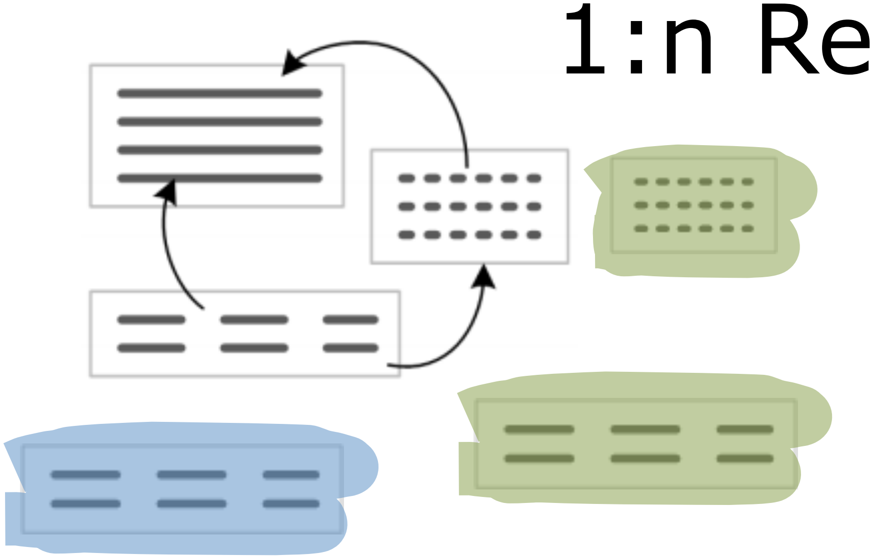


[Separation?]



Sufficiency
Different Stakeholders
Different Process Steps

1:n Relationships



Well-defined interfaces

Avoid Cycles

Avoid Synchronization

Think in terms of Concerns.

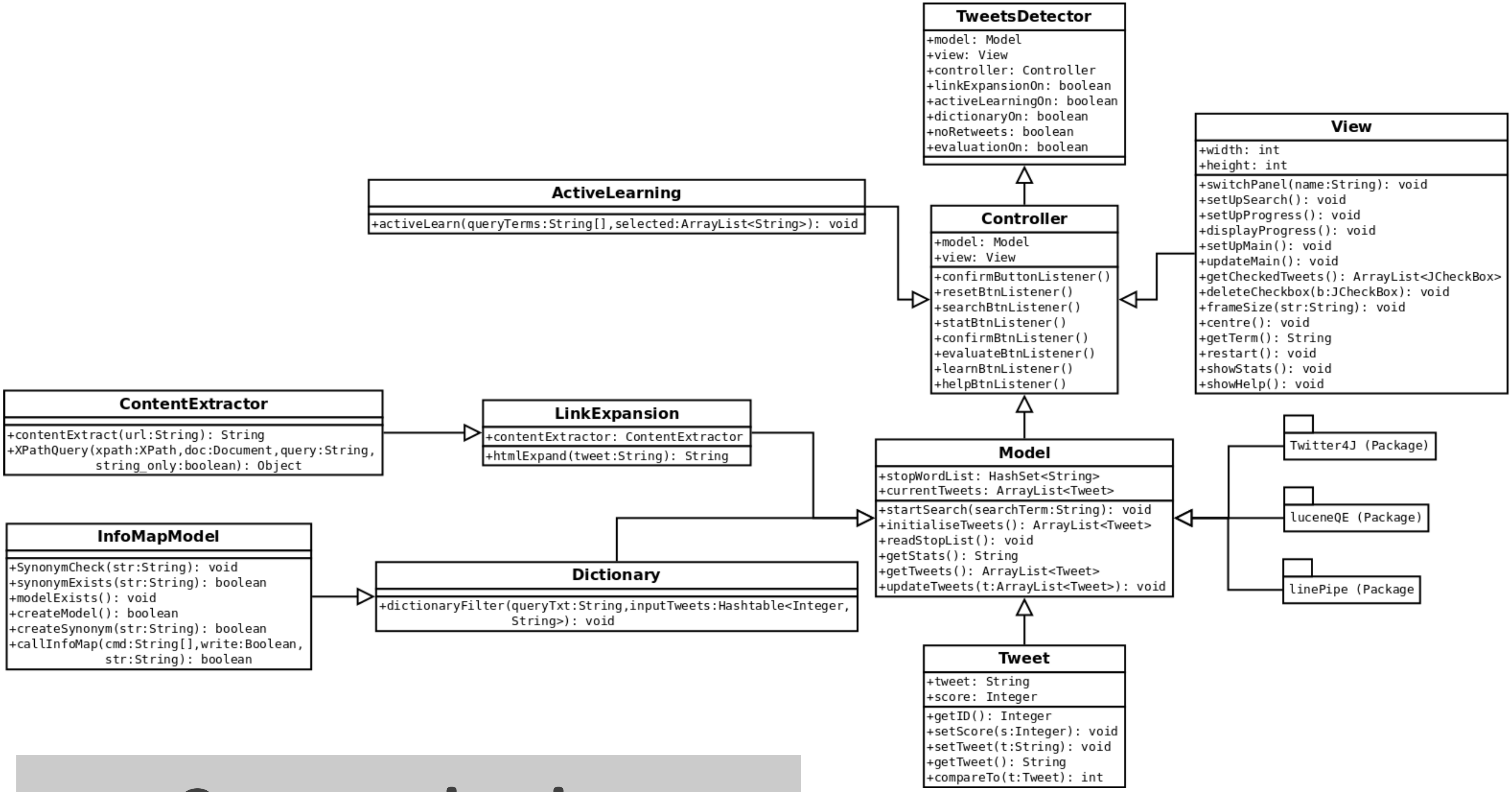
Separate them if necessary.

Also support integration!

4

Established Formalisms

[Type|UML Class Diagram]



TweetsDetector
+model: Model
+view: View
+controller: Controller
+linkExpansionOn: boolean
+activeLearningOn: boolean
+dictionaryOn: boolean
+noRetweets: boolean
+evaluationOn: boolean

View
+width: int
+height: int
+switchPanel(name:String): void
+setUpSearch(): void
+setUpProgress(): void
+displayProgress(): void
+setUpMain(): void
+updateMain(): void
+getCheckedTweets(): ArrayList<JCheckBox>
+deleteCheckbox(b:JCheckBox): void
+frameSize(str:String): void
+centre(): void
+getTerm(): String
+restart(): void
+showStats(): void
+showHelp(): void

ActiveLearning
+activeLearn(queryTerms:String[],selected:ArrayList<String>): void

Controller
+model: Model
+view: View
+confirmButtonListener()
+resetBtnListener()
+searchBtnListener()
+statBtnListener()
+confirmBtnListener()
+evaluateBtnListener()
+learnBtnListener()
+helpBtnListener()

ContentExtractor
+contentExtract(url:String): String
+XPathQuery(xpath:XPath,doc:Document,query:String,
string_only:boolean): Object

LinkExpansion
+contentExtractor: ContentExtractor
+htmlExpand(tweet:String): String

Model
+stopWordList: HashSet<String>
+currentTweets: ArrayList<Tweet>
+startSearch(searchTerm:String): void
+initialiseTweets(): ArrayList<Tweet>
+readStopList(): void
+getStats(): String
+getTweets(): ArrayList<Tweet>
+updateTweets(t:ArrayList<Tweet>): void

Twitter4J (Package)

luceneQE (Package)

linePipe (Package

InfoMapModel
+SynonymCheck(str:String): void
+synonymExists(str:String): boolean
+modelExists(): void
+createModel(): boolean
+createSynonym(str:String): boolean
+callInfoMap(cmd:String[],write:Boolean,
str:String): boolean

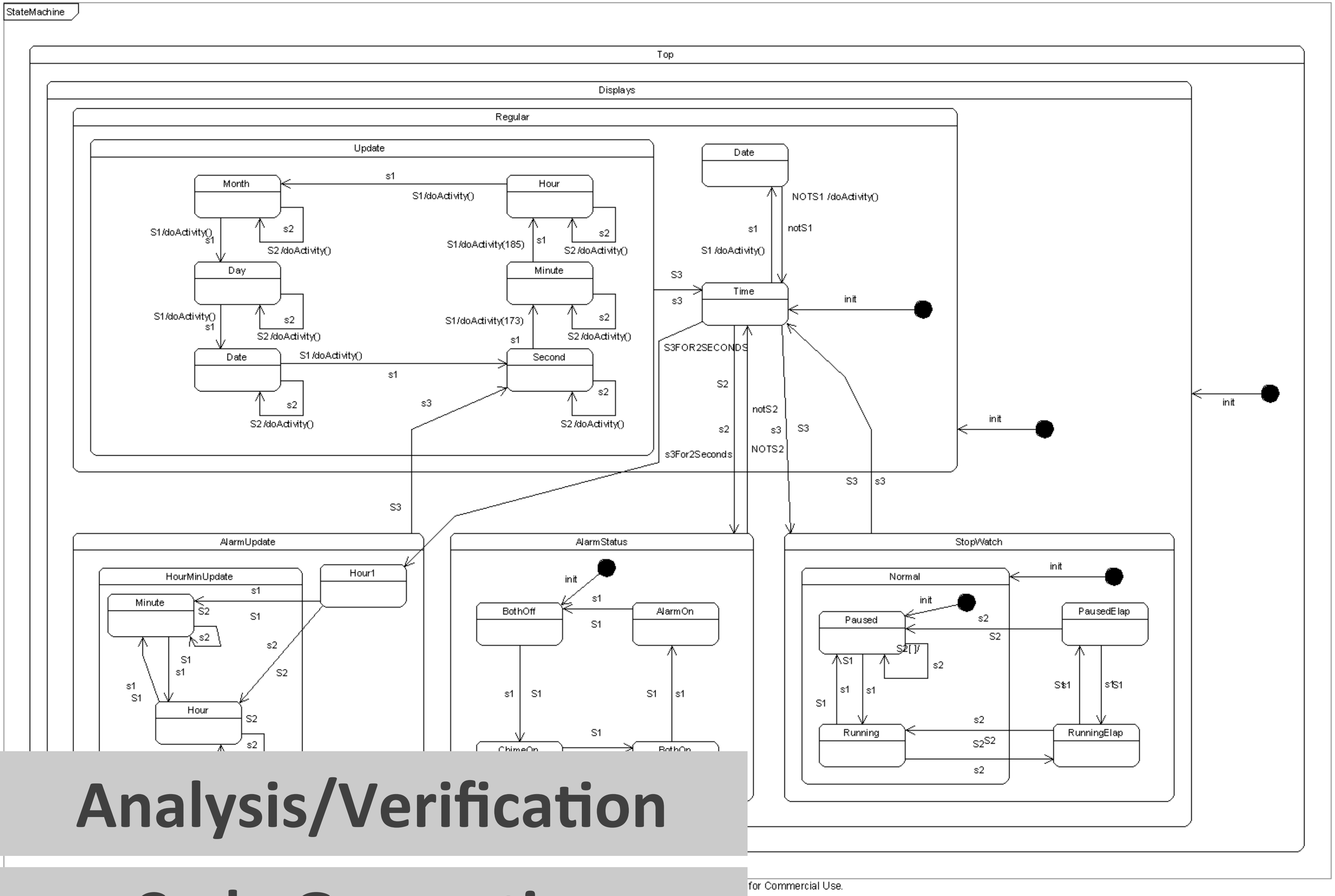
Dictionary
+dictionaryFilter(queryTxt:String,inputTweets:Hashtable<Integer,
String>): void

Tweet
+tweet: String
+score: Integer
+getID(): Integer
+setScore(s:Integer): void
+setTweet(t:String): void
+getTweet(): String
+compareTo(t:Tweet): int

Communication

Simple Code Generation

[Type|UML State Diagram]

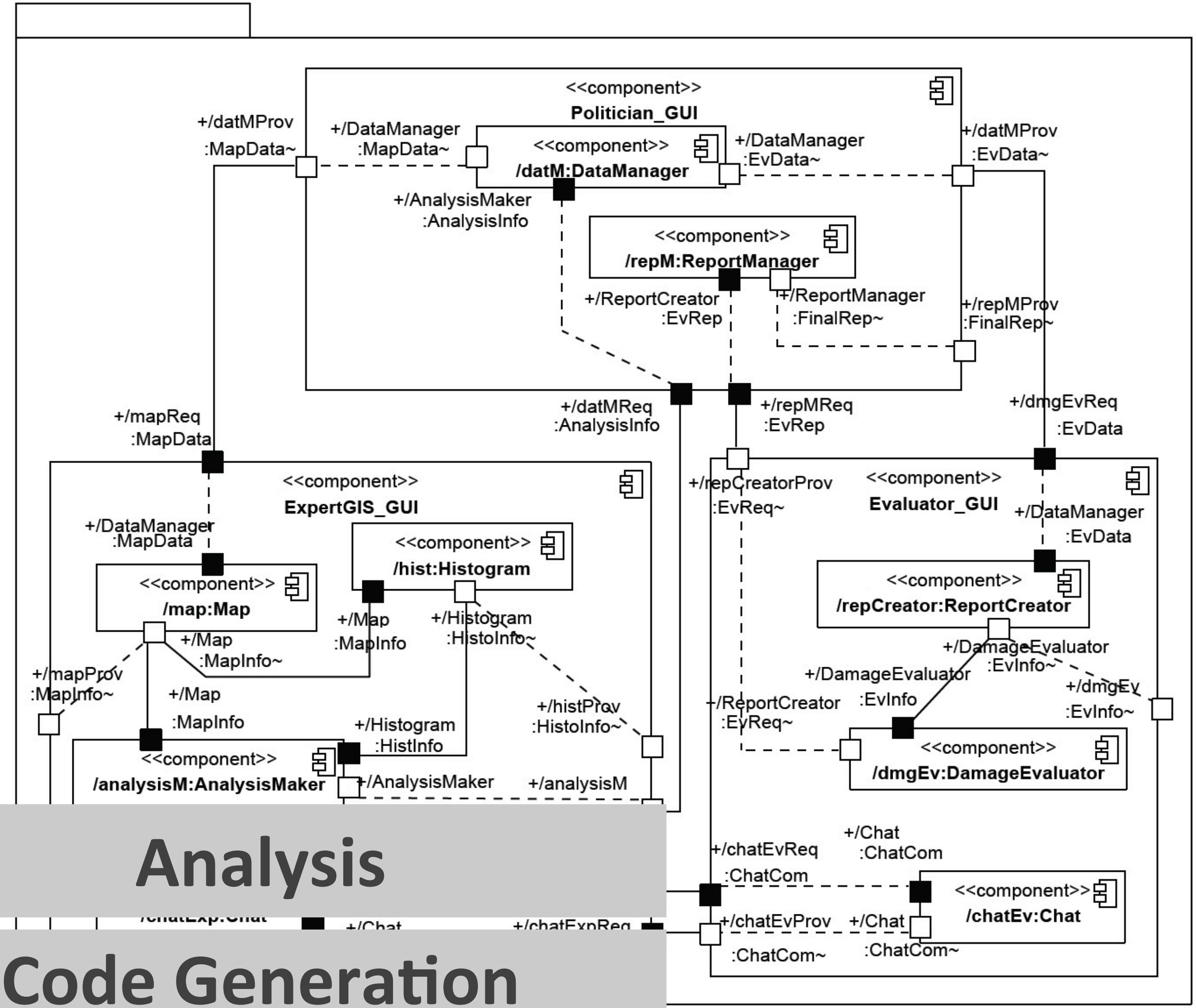


Analysis/Verification

Code Generation

http://www.primordion.com

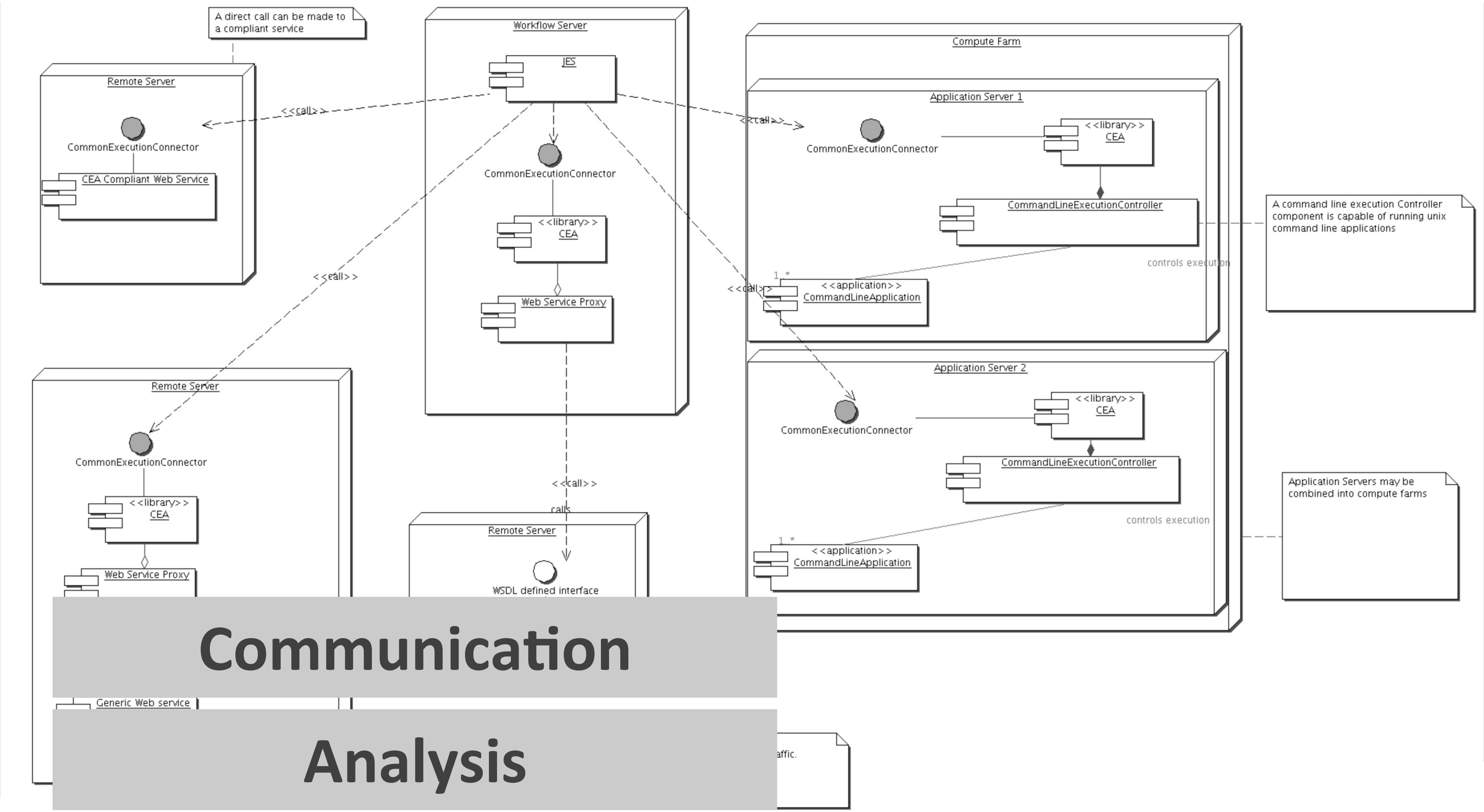
[Composition|UML Comp. Struct.]



Analysis

Code Generation

[System|UML Deployment]



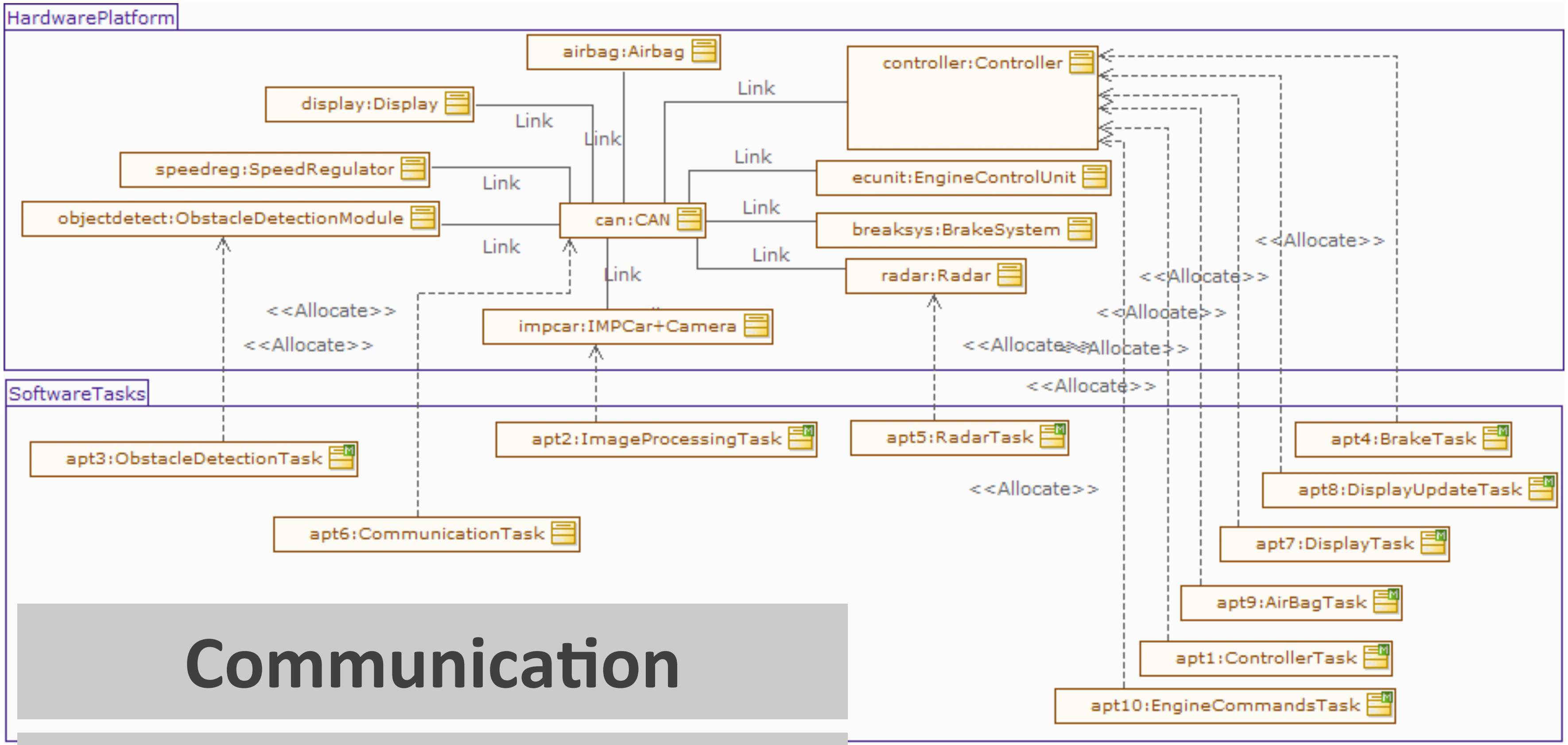
Communication

Analysis

Code/Script Generation

http://www.ivoa.net/

[System|Class Diagram + Profile]

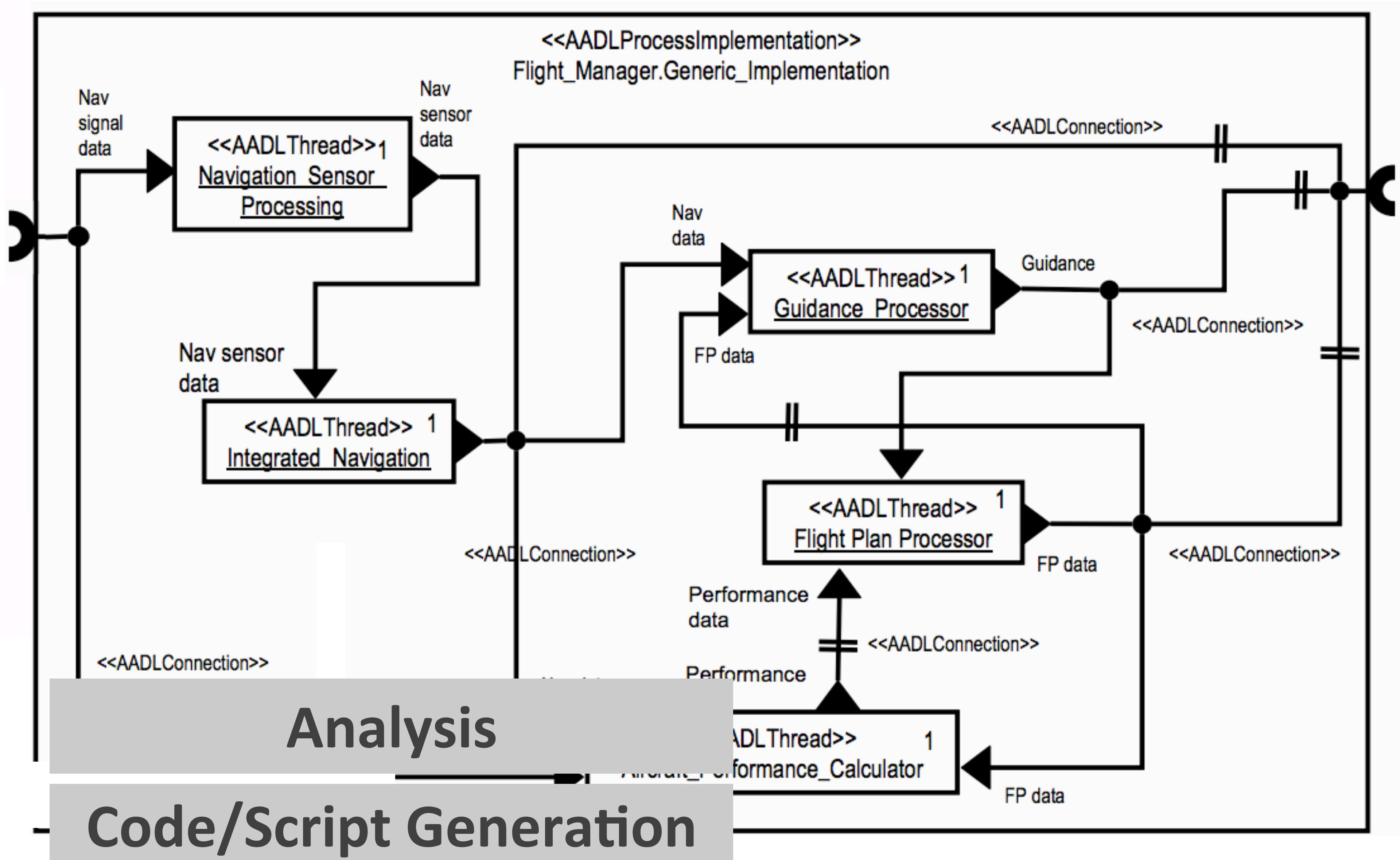


Communication

Analysis

Code/Script Generation

[Composition|AADL]

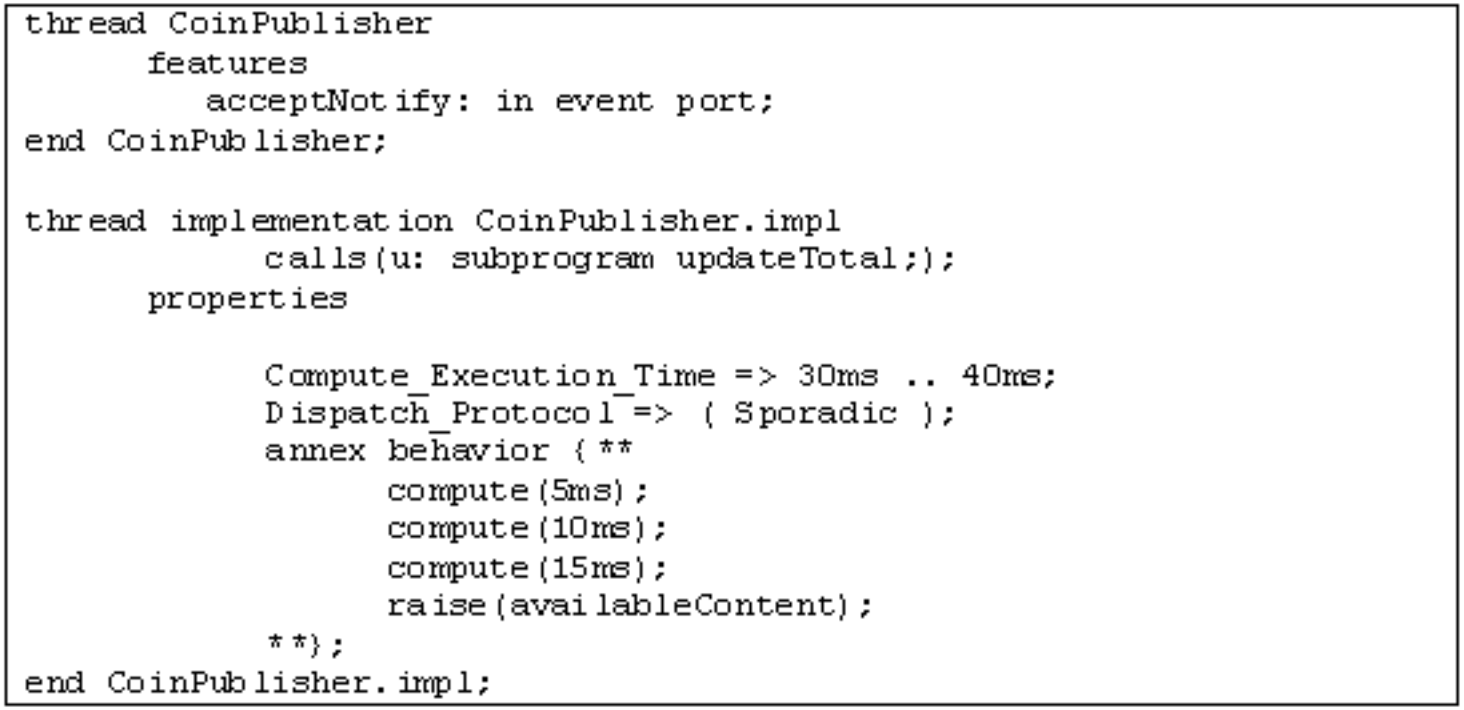


Analysis

Code/Script Generation

http://www.omg.org/news/meetings/workshops/MIC_2004_Manual/00-T2_Feiler_etal2.pdf

[Composition|AADL]



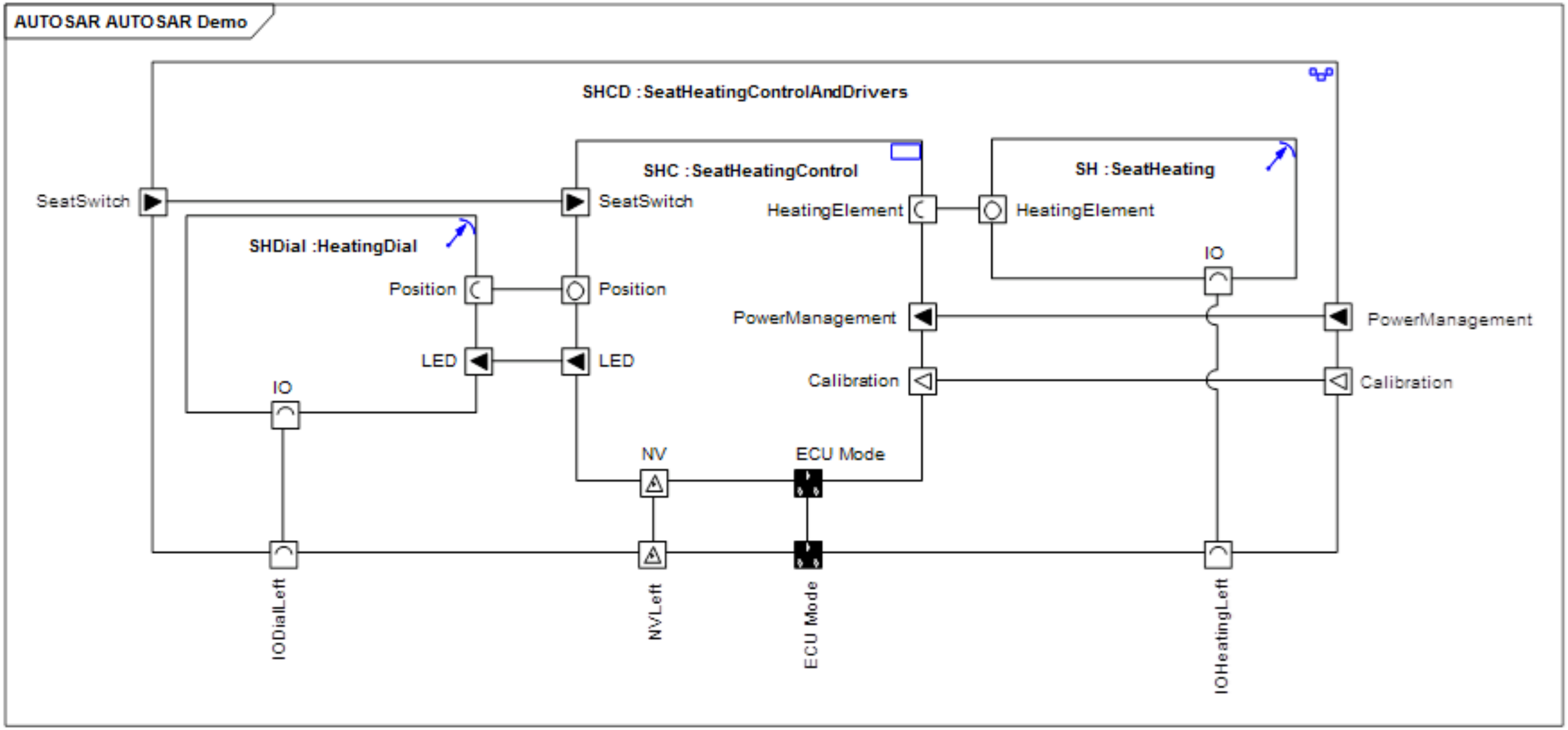
```
thread CoinPublisher
      features
         acceptNotify: in event port;
end CoinPublisher;

thread implementation CoinPublisher.impl
            calls(u: subprogram updateTotal;);
      properties

            Compute_Execution_Time => 30ms .. 40ms;
            Dispatch_Protocol => ( Sporadic );
            annex behavior {**
                  compute(5ms);
                  compute(10ms);
                  compute(15ms);
                  raise(availableContent);
            **};
end CoinPublisher.impl;
```

Analysis

Code/Script Generation

http://www.jot.fm/issues/issue_2009_01/column1/

[Composition|AUTOSAR]

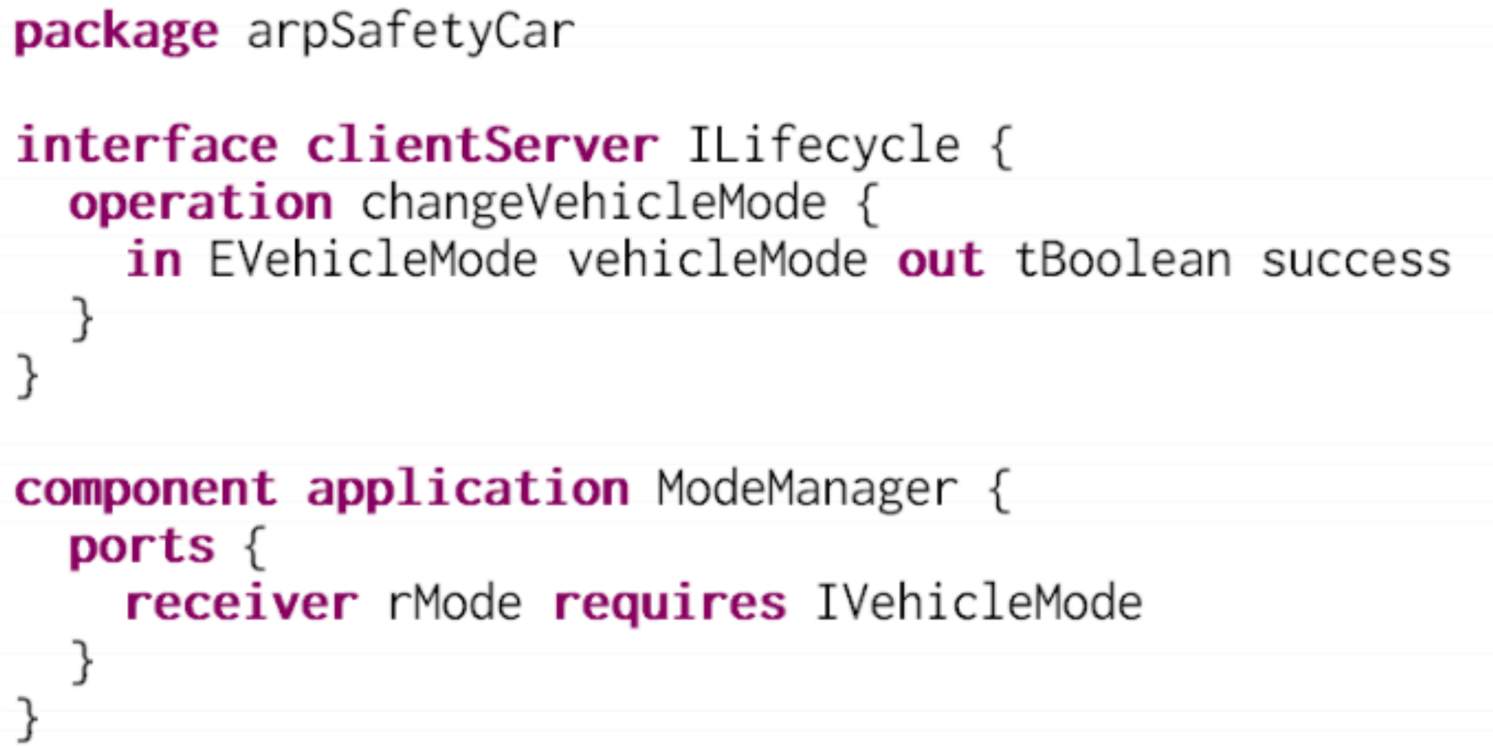


Analysis

Code/Script Generation

http://blog.lieberlieber.com/

[Type|Artext]



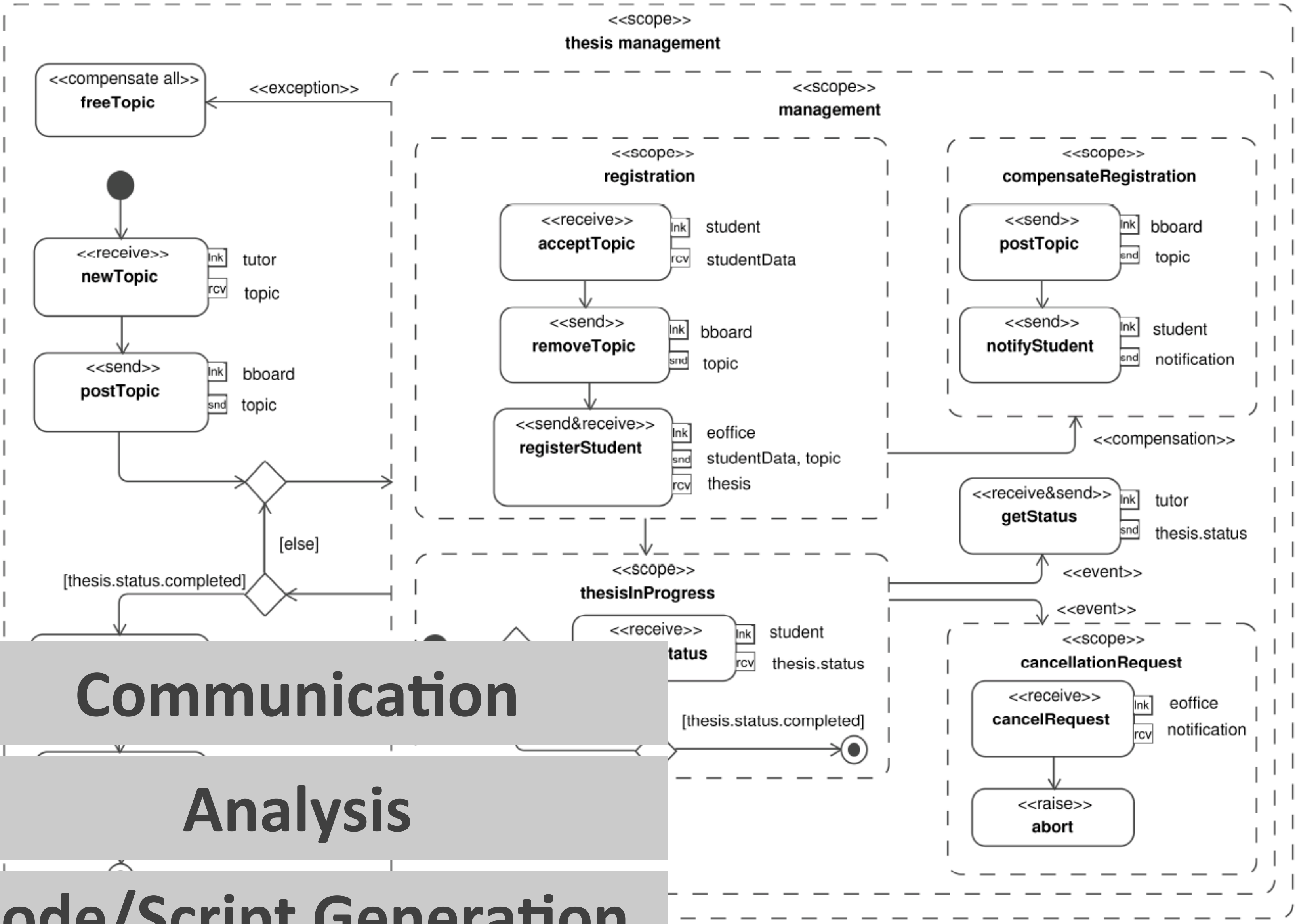
```
package arpSafetyCar

interface clientServer ILifecycle {
  operation changeVehicleMode {
    in EVehicleMode vehicleMode out tBoolean success
  }
}

component application ModeManager {
  ports {
    receiver rMode requires IVehicleMode
  }
}
```

Communication

Code Generation

http://www.bmw-carit.com/downloads/publications/
Interoperable%20AUTOSAR%20tooling%20with%20Artop.pdf

[Adaptation|UML Profiles]



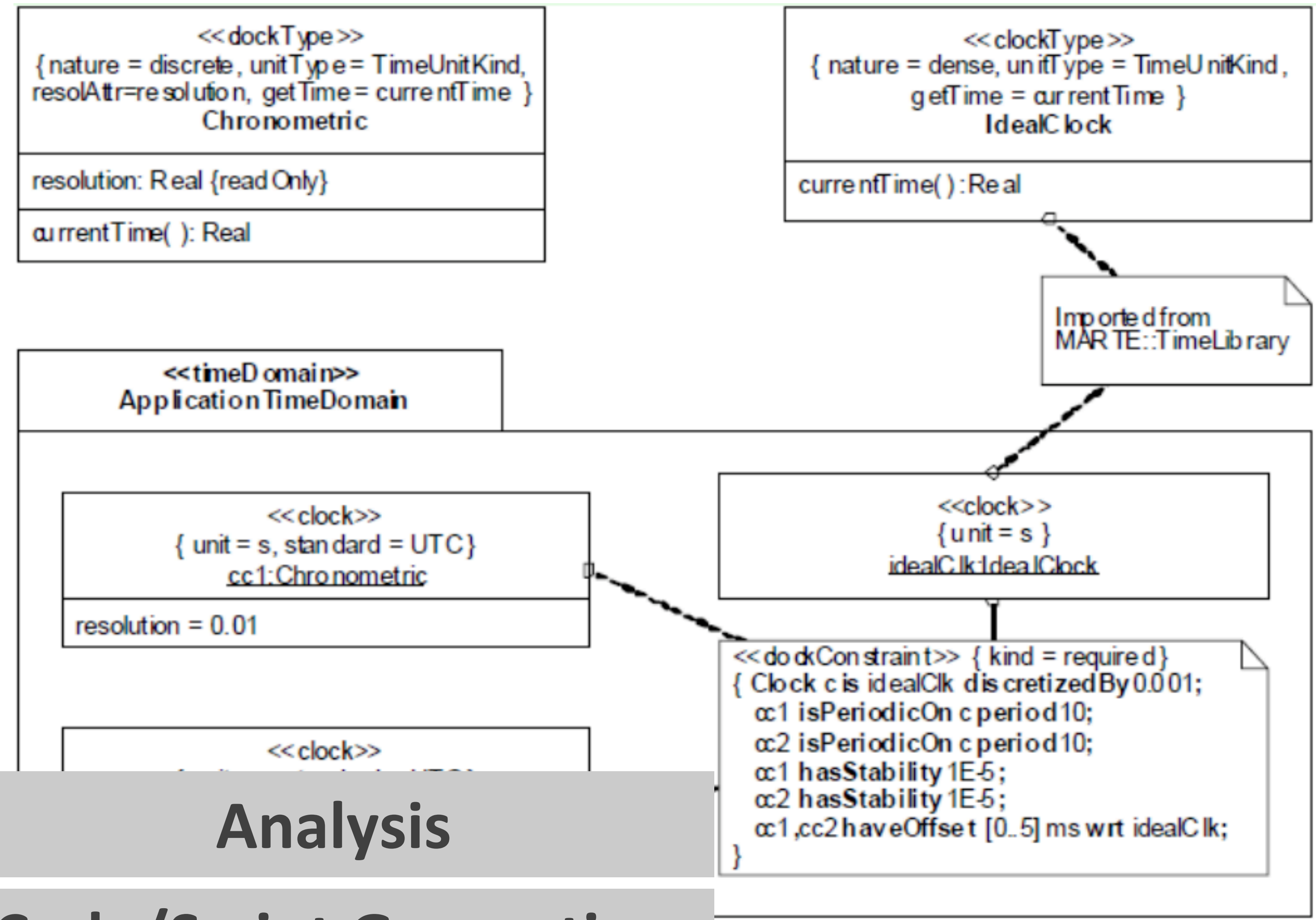
Communication

Analysis

Code/Script Generation

http://www.uml4soa.eu/

[Adaptation|UML Profiles]



<<clockType>>
{nature = discrete, unitType = TimeUnitKind,
resolAttr=resolution, getTime = currentTime }
Chronometric

resolution: Real {readOnly}

currentTime( ): Real

<<clockType>>
{ nature = dense, unitType = TimeUnitKind,
getTime = currentTime }
IdealClock

currentTime( ):Real

Imported from
MARTE::TimeLibrary

<<timeDomain>>
ApplicationTimeDomain

<<clock>>
{ unit = s, standard = UTC}
cc1:Chronometric

resolution = 0.01

<<clock>>
{ unit = s }
idealClk:IdealClock

<<clockConstraint>> { kind = required}
{ Clock c is idealClk discretizedBy 0.001;
cc1 isPeriodicOn c period 10;
cc2 isPeriodicOn c period 10;
cc1 hasStability 1E-5;
cc2 hasStability 1E-5;
cc1,cc2 haveOffset [0..5] ms wrt idealClk;
}

<<clock>>

Analysis

Code/Script Generation

http://doc.omg.org/formal/2009-11-02.pdf

Components + Ports Ubiquituous

Graphical + Textual

Concerns mostly Separated!

Some means of Adaptation.

5

Extension and Adaptation

The majority of our interviewees were very successful with MDE but all of them either **built their own** modeling tools, made **heavy adaptations** of off-the-shelf tools, or spent a lot of time finding ways to **work around** tools. The only accounts of easy-to-use, intuitive tools came from those who had developed tools themselves for bespoke purposes. Indeed, this suggests that current tools are a barrier to success rather than an enabler.

Jon Whittle, John Hutchinson, Mark Rouncefield, Hakan Burden, and Rogardt Heldal. Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS) 2013*. ACM, 2013.

Complexity problems are typically associated with off-the- shelf tools. Of particular note is **accidental complexity** – which can be introduced due to [..] [the] lack of flexibility to adapt the tools to a company’s own context [..]

Jon Whittle, John Hutchinson, Mark Rouncefield, Hakan Burden, and Rogardt Heldal. Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS) 2013*. ACM, 2013.

Our interviews point to a **strong need for tailoring** of some sort: either tailor the tool to the process, tailor the process to the tool, or build your own tool that naturally fits your own process. Based on our data, it seems that, on balance, it is currently much easier to do the latter.

Jon Whittle, John Hutchinson, Mark Rouncefield, Hakan Burden, and Rogardt Heldal. Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In *Proceedings of the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS) 2013*. ACM, 2013.

[Profiles are Hard]

not used right very often &
unnecessarily complicated models

[Profiles are Hard]

[May be misleading]

models mean something else than what they appear to mean

[Profiles are Hard]

[May be misleading]

[Models are not “intentional”]

> low-level abstractions make models hard to analyze

[Profiles are Hard]

[May be misleading]

[Models are not “intentional”]

[Unintended Features]

> …because profiles must limit existing functionality. Coverage!

[Profiles are Hard]

[May be misleading]

[Models are not “intentional”]

[Unintended Features]

[Hard to include textual Aspects]

There is no extensible way for textual syntax in UML

[Profiles are Hard]

[May be misleading]

[Models are not “intentional”]

[Unintended Features]

[Hard to include textual Aspects

[Many UML tools suck @ profiles]

Magicdraw is the only exception
I have seen so far!

[Profiles are Hard]

[May be misleading]

[Models are not “intentional”]

[Unintended Features]

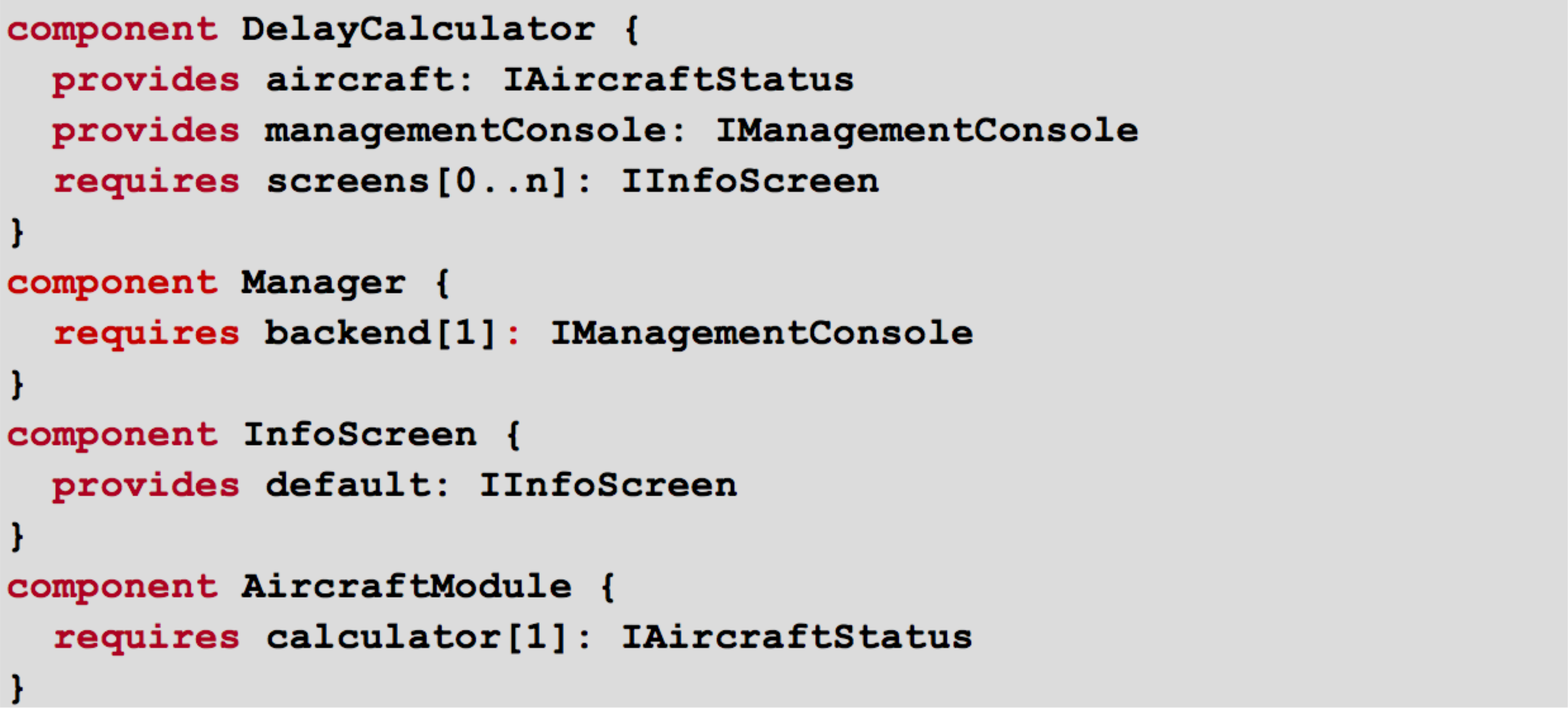
[Hard to include textual Aspects

[Many UML tools suck @ profiles]

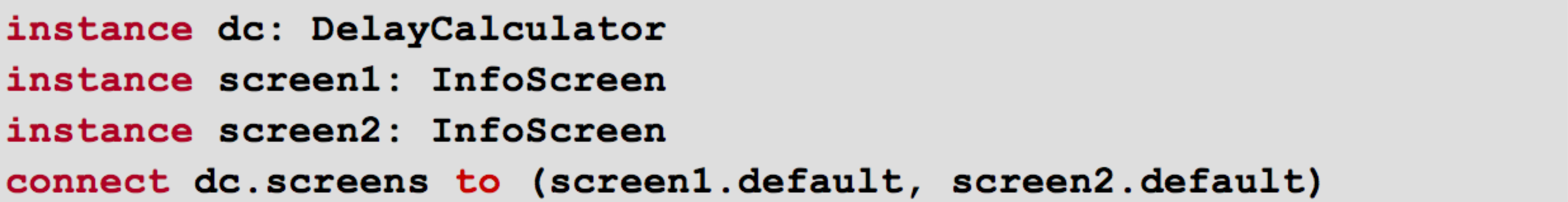
[Standard Profiles are Complex]

  MARTE is 600 pages – how much of that stuff do you really need?

[Type/Comp|Arch As Language]

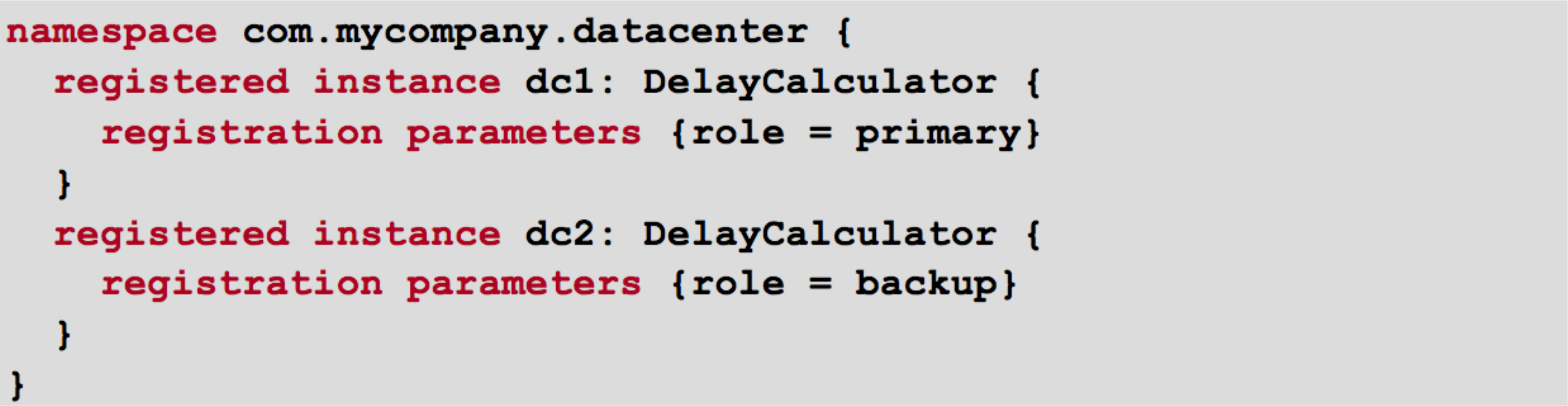


```
component DelayCalculator {
  provides aircraft: IAircraftStatus
  provides managementConsole: IManagementConsole
  requires screens[0..n]: IInfoScreen
}
component Manager {
  requires backend[1]: IManagementConsole
}
component InfoScreen {
  provides default: IInfoScreen
}
component AircraftModule {
  requires calculator[1]: IAircraftStatus
}
```

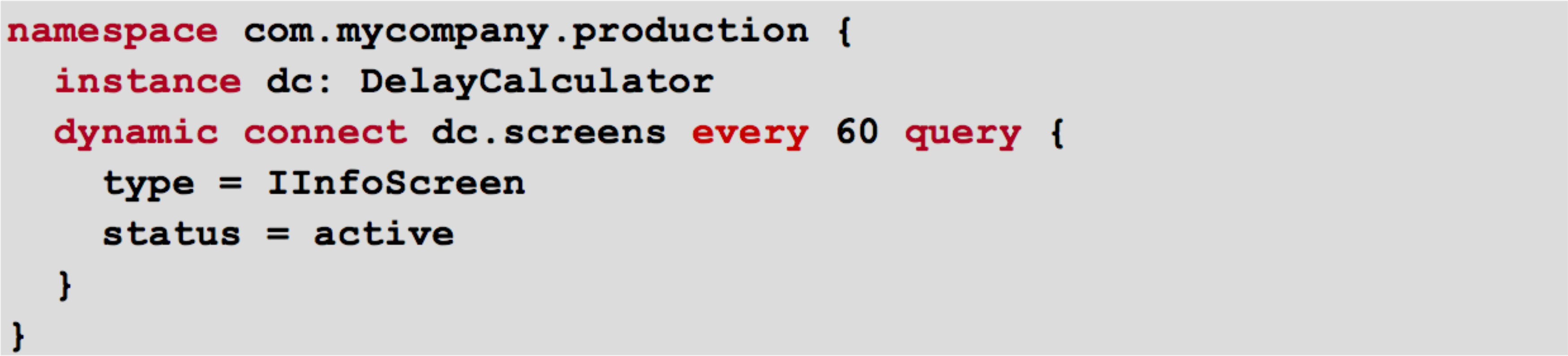


```
instance dc: DelayCalculator
instance screen1: InfoScreen
instance screen2: InfoScreen
connect dc.screens to (screen1.default, screen2.default)
```

[Type/Comp|Arch As Language]

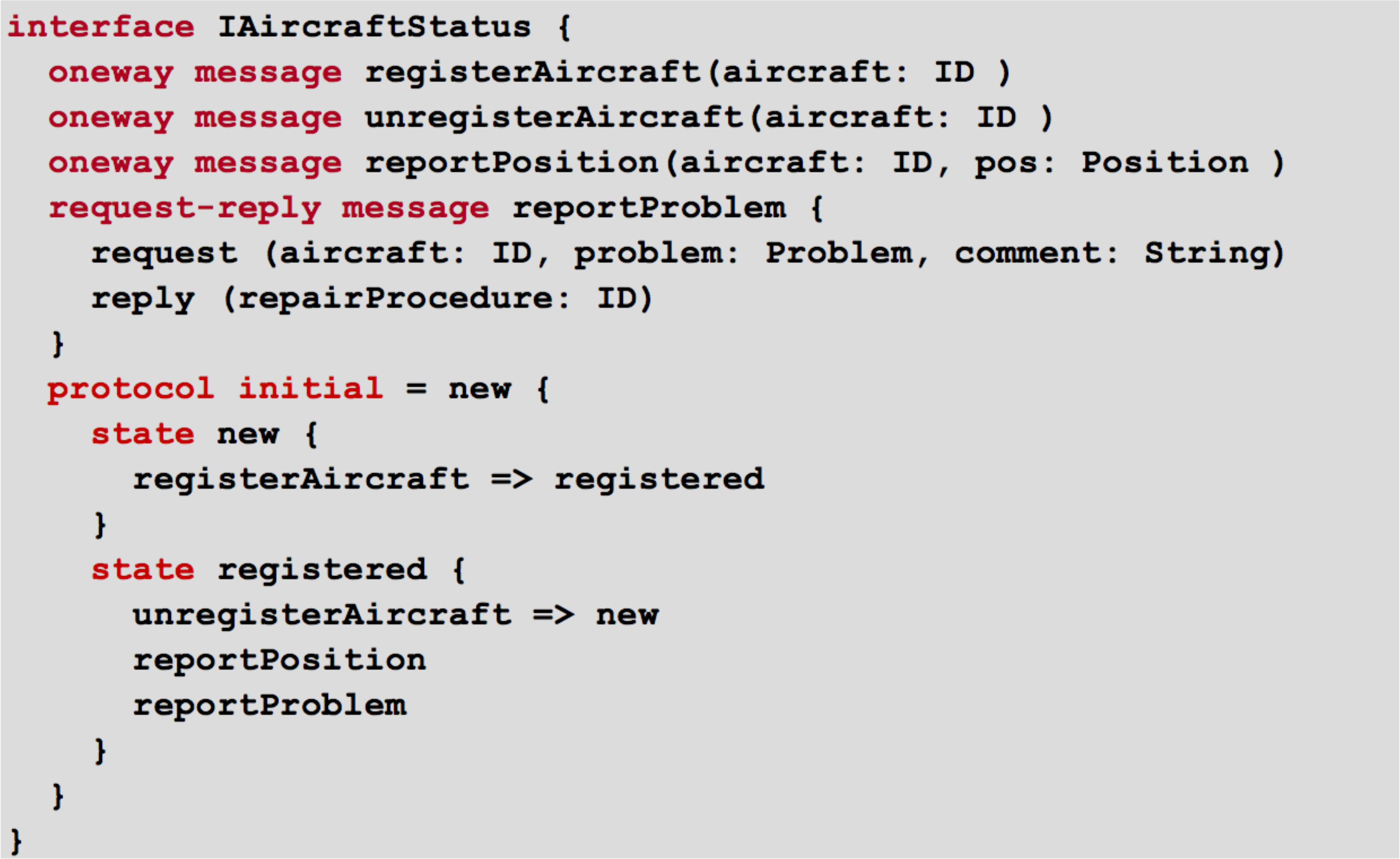


```
namespace com.mycompany.datacenter {
  registered instance dc1: DelayCalculator {
    registration parameters {role = primary}
  }
  registered instance dc2: DelayCalculator {
    registration parameters {role = backup}
  }
}
```



```
namespace com.mycompany.production {
  instance dc: DelayCalculator
  dynamic connect dc.screens every 60 query {
    type = IInfoScreen
    status = active
  }
}
```

[Type|Arch As Language]

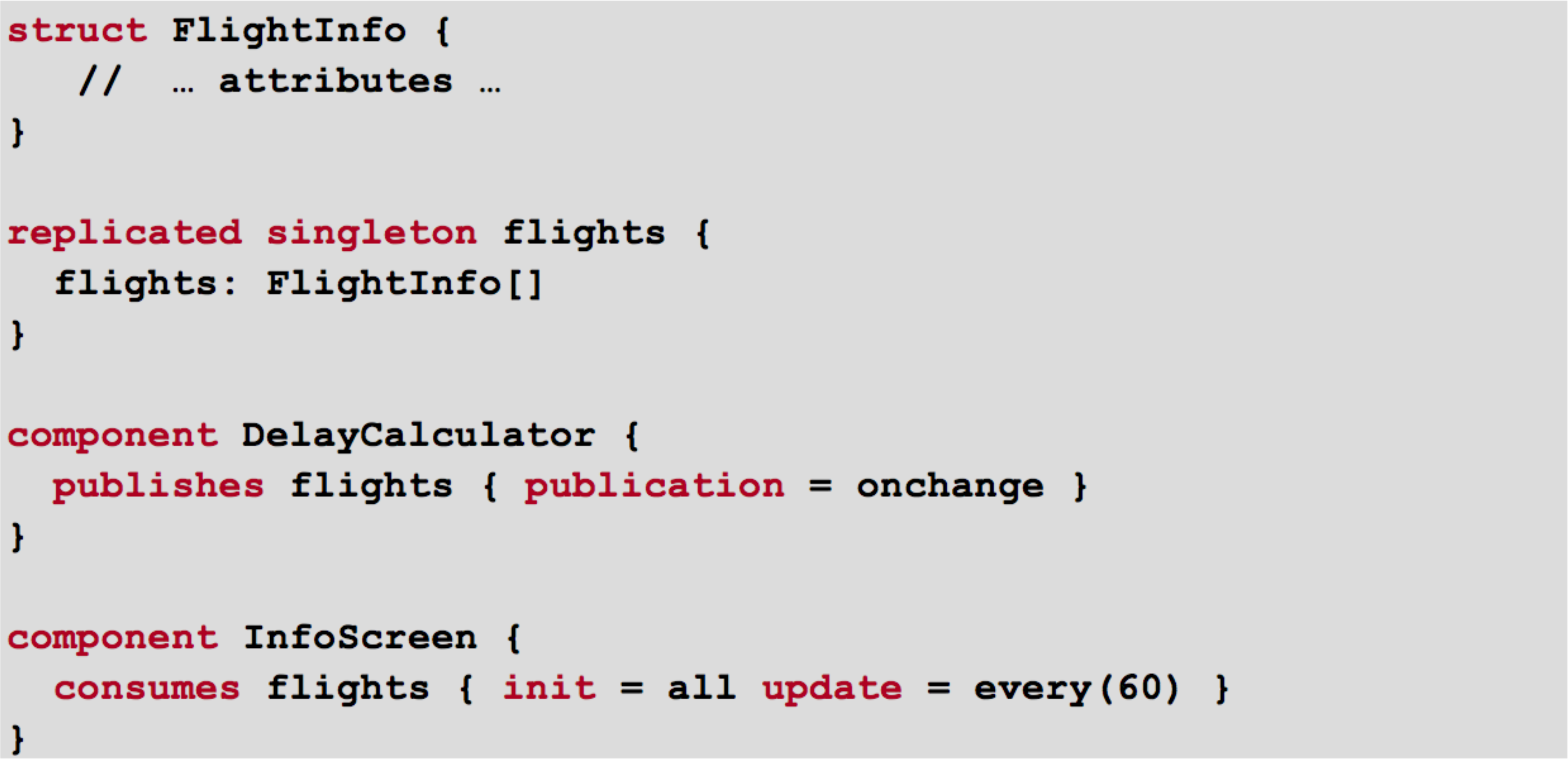


```
interface IAircraftStatus {
  oneway message registerAircraft(aircraft: ID )
  oneway message unregisterAircraft(aircraft: ID )
  oneway message reportPosition(aircraft: ID, pos: Position )
  request-reply message reportProblem {
    request (aircraft: ID, problem: Problem, comment: String)
    reply (repairProcedure: ID)
  }
  protocol initial = new {
    state new {
      registerAircraft => registered
    }
    state registered {
      unregisterAircraft => new
      reportPosition
      reportProblem
    }
  }
}
```

[Type|Arch As Language]

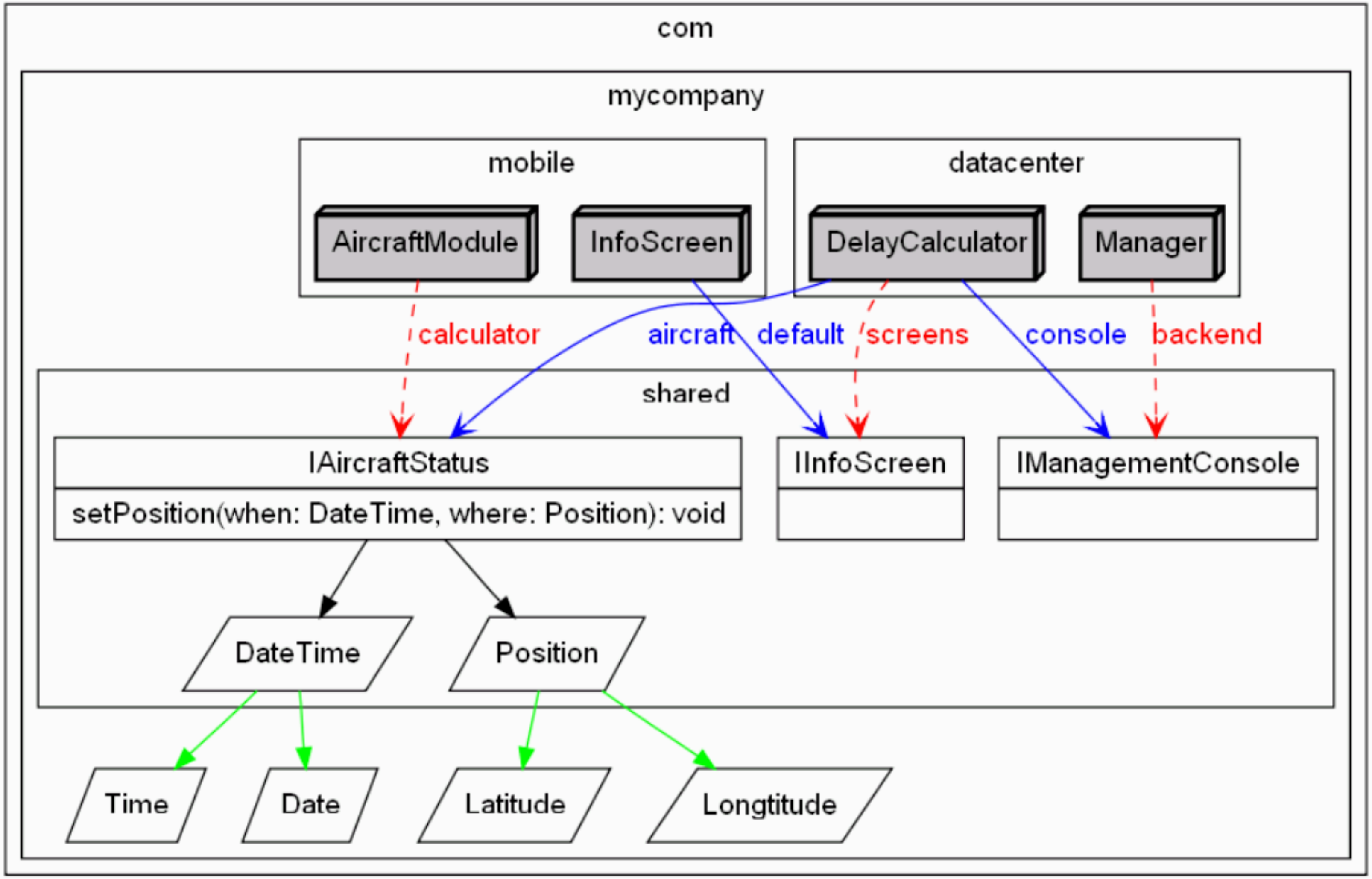
?!

[Type|Arch As Language]



```
struct FlightInfo {
   //  … attributes …
}

replicated singleton flights {
  flights: FlightInfo[]
}

component DelayCalculator {
  publishes flights { publication = onchange }
}

component InfoScreen {
  consumes flights { init = all update = every(60)  }
}
```

[Type|Arch As Language]



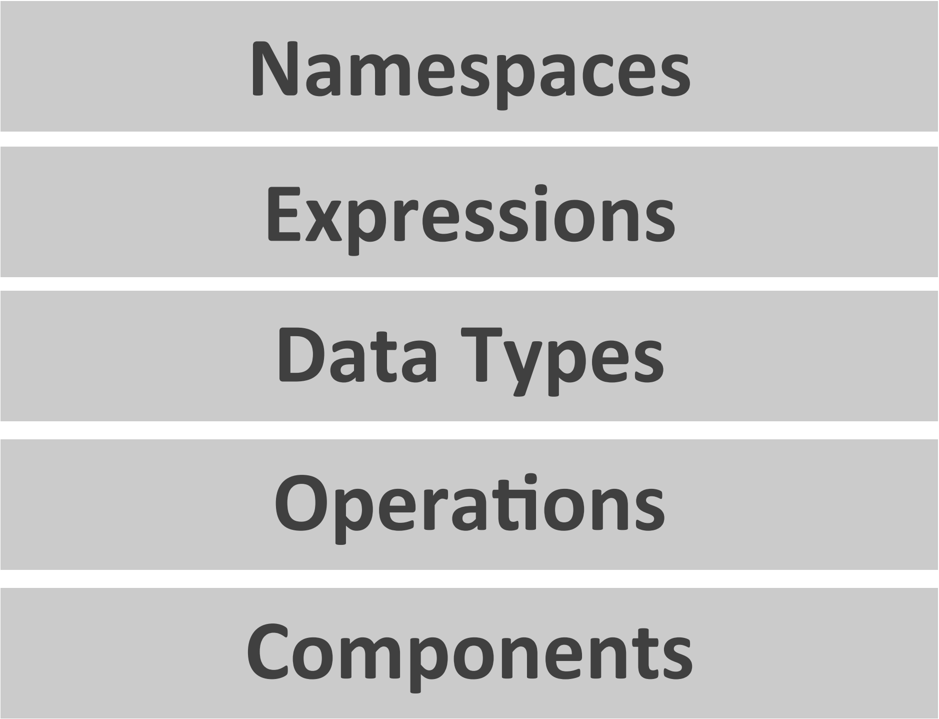
A DSL per Architecture/Platform

Really fits the A exactly.

But what about Effort?

What can we reuse? DSL-PLE?

[Candidates for Reuse]



But still: extension, restriction & adaptation is required!

[More Candidates for Reuse]

- Tracing to Requirements
- Architecture Decisions
- Variability Support
- Documentation

Of Course: extension, restriction & adaptation is required!

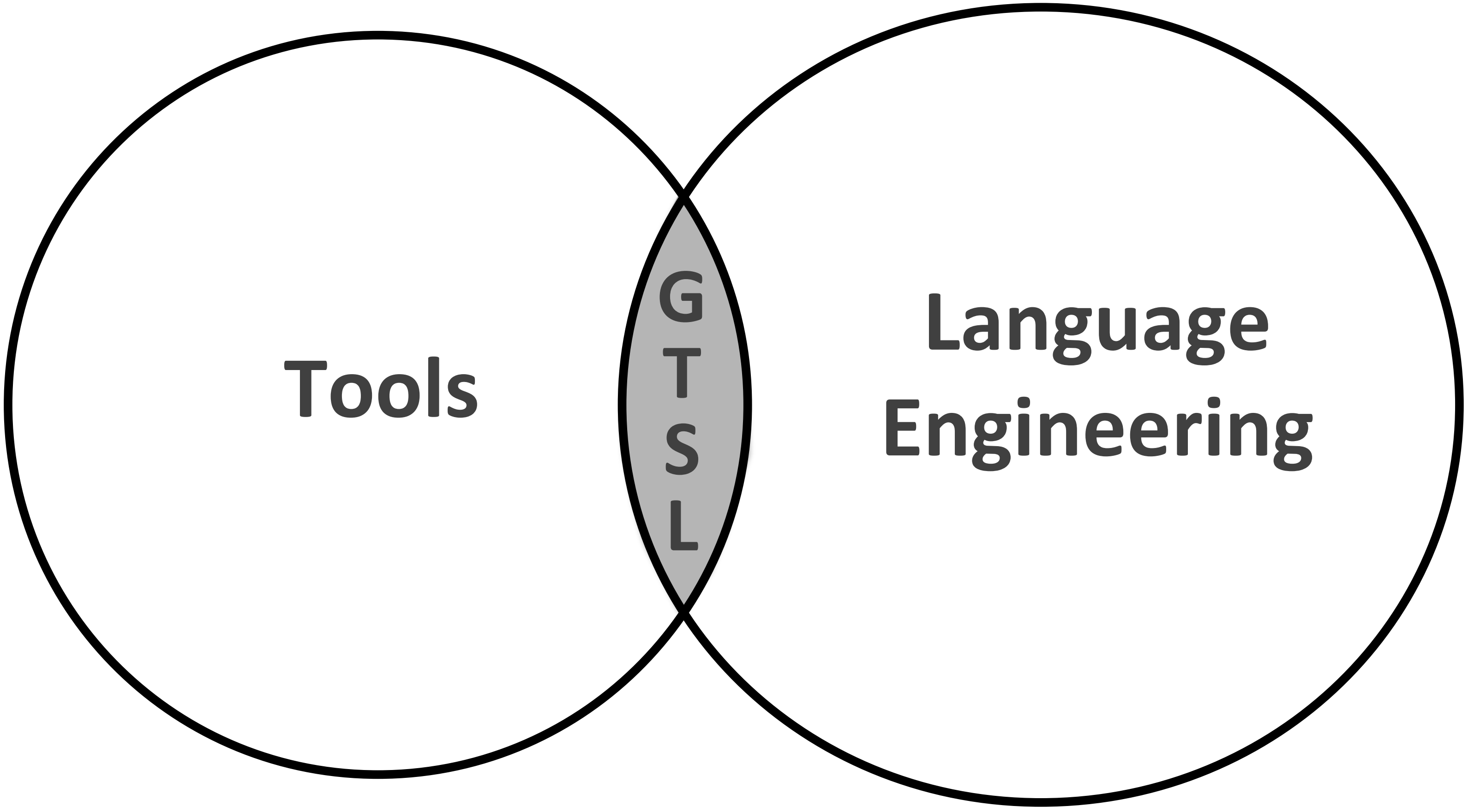
Fine-grained Reuse as in OO

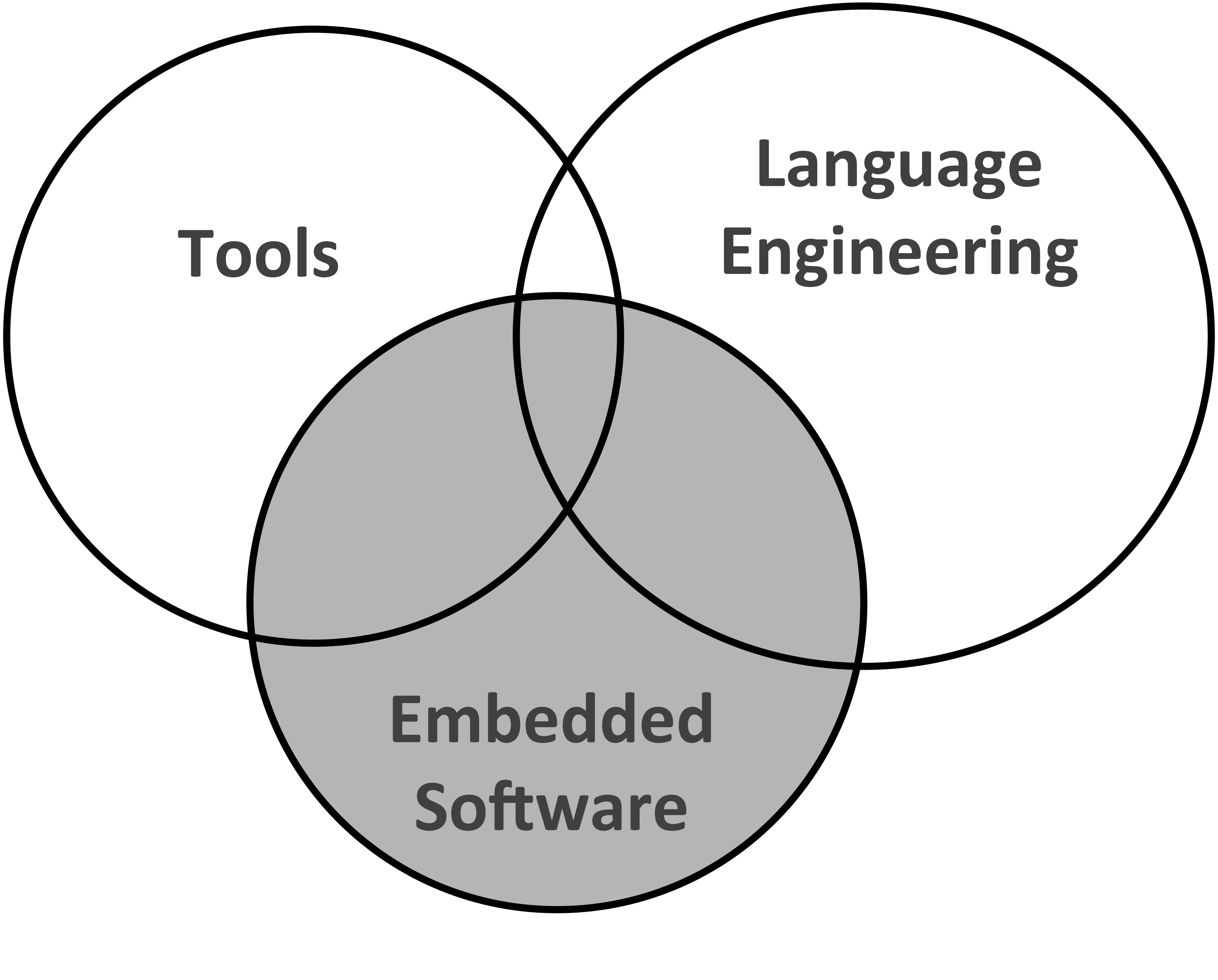
Handle Crosscuts

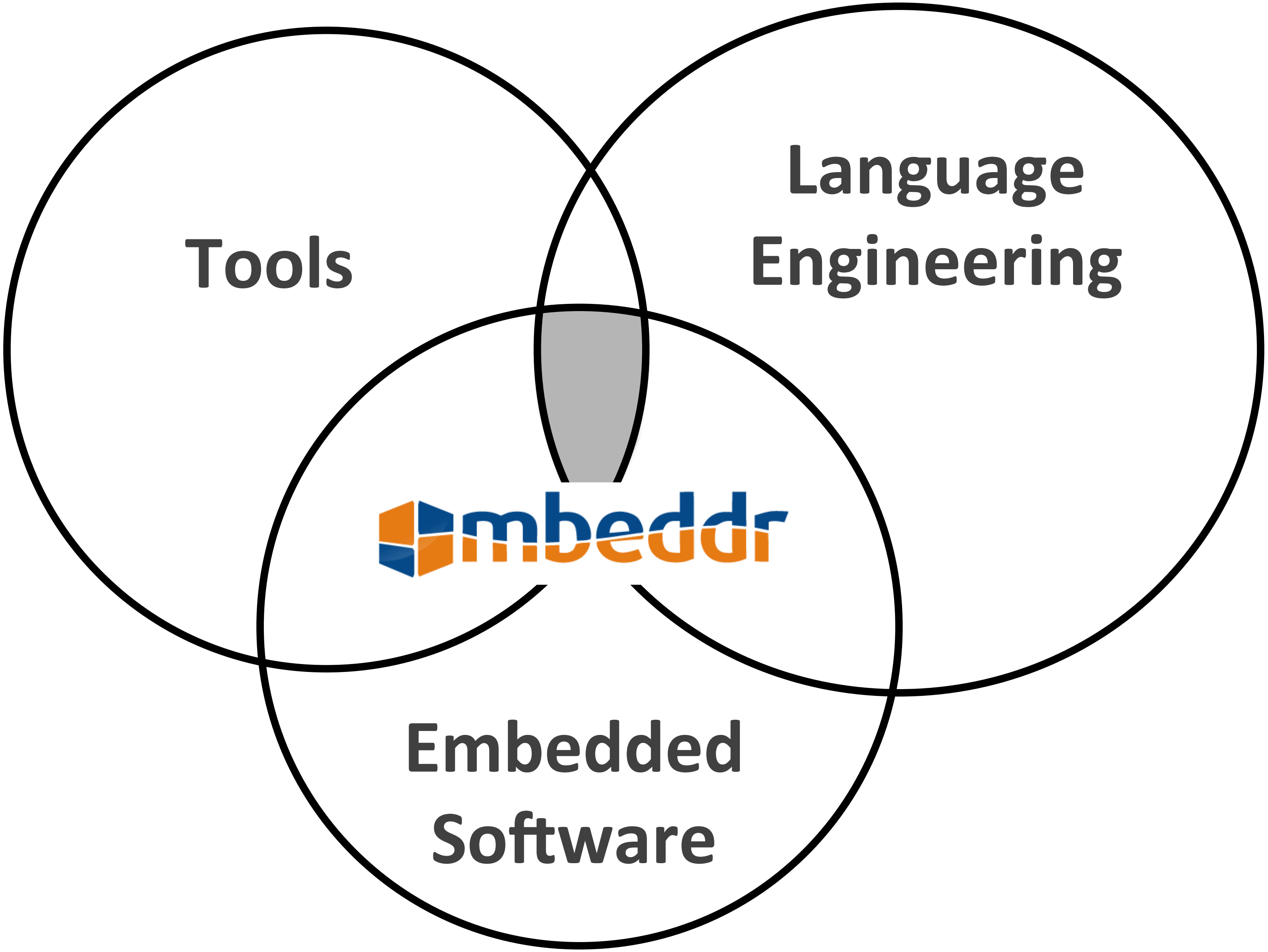
IDE Support

6

mbeddr

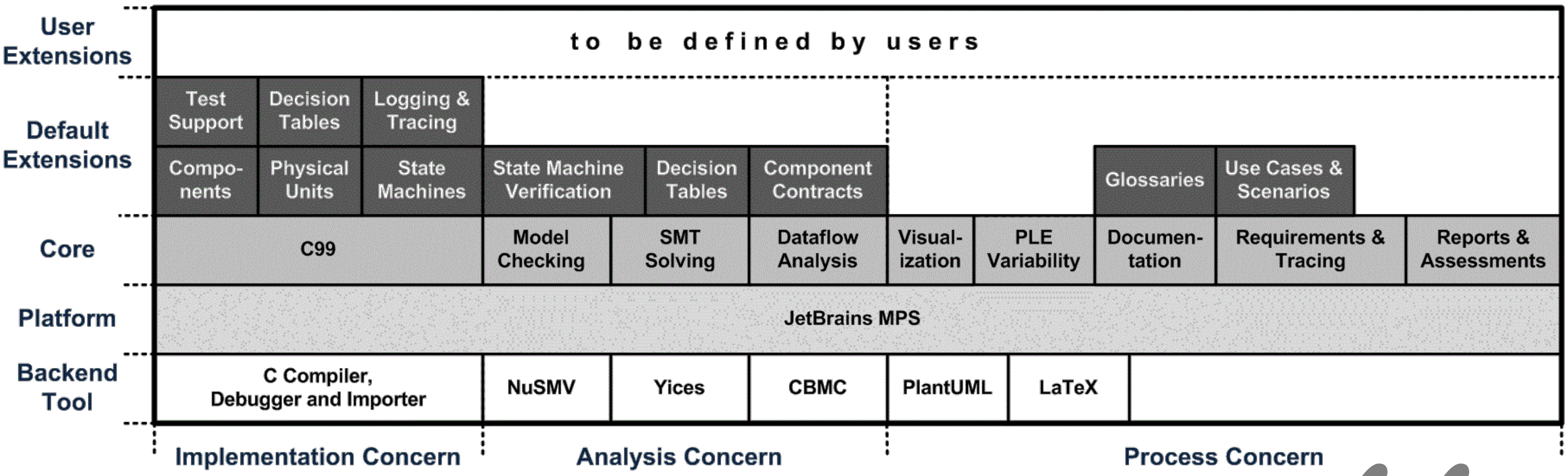

Tools
G
T
S
L
Language
Engineering


Tools
Language Engineering
Embedded Software


Tools
Language Engineering
mbeddr
Embedded Software

mbeddr

An extensible set of integrated languages for embedded software engineering.



User Extensions: to be defined by users

Default Extensions: Test Support, Decision Tables, Logging & Tracing; Compo-nents, Physical Units, State Machines, State Machine Verification, Decision Tables, Component Contracts, Glossaries, Use Cases & Scenarios

Core: C99, Model Checking, SMT Solving, Dataflow Analysis, Visual-ization, PLE Variability, Documen-tation, Requirements & Tracing, Reports & Assessments

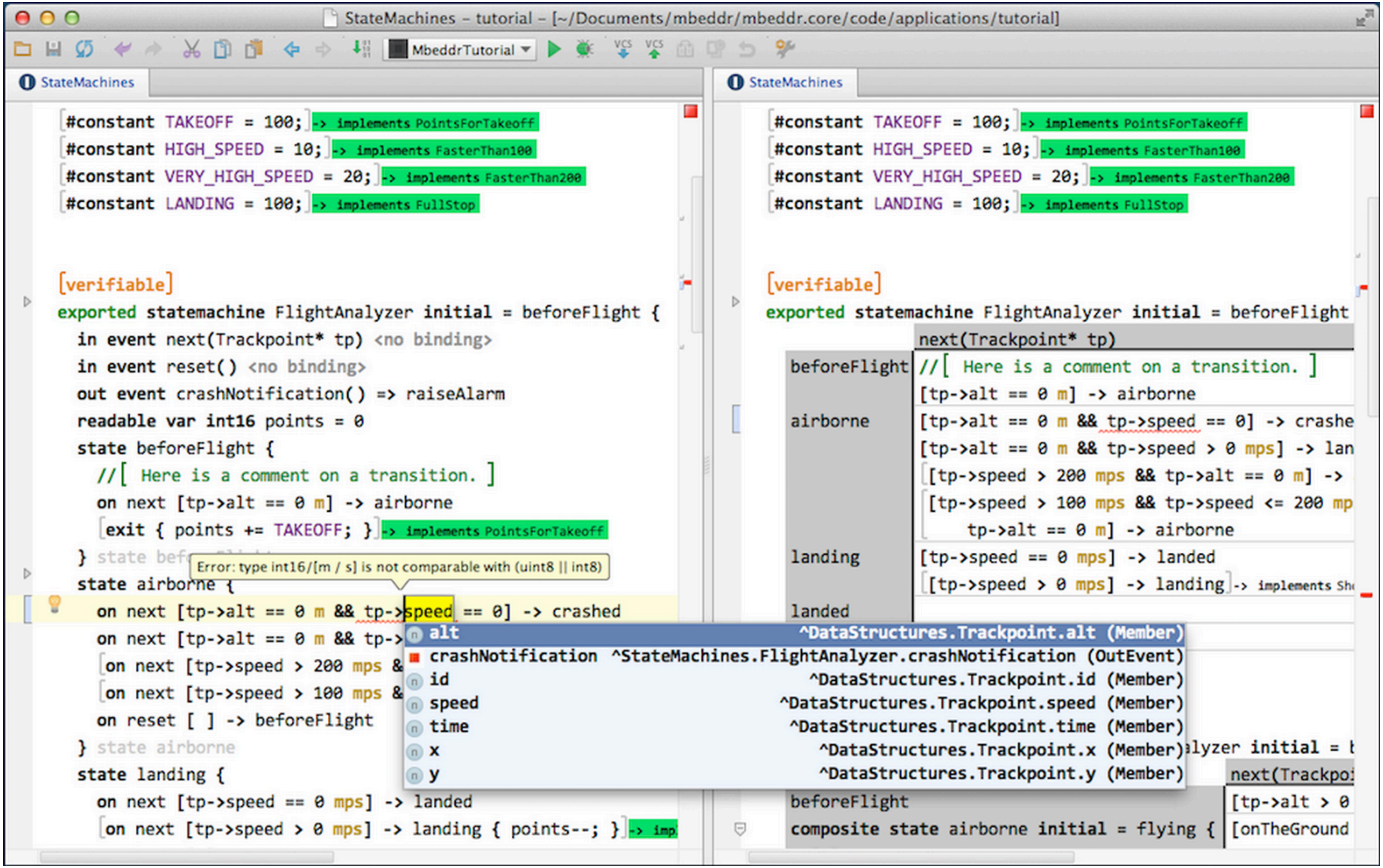
Platform: JetBrains MPS

Backend Tool: C Compiler, Debugger and Importer, NuSMV, Yices, CBMC, PlantUML, LaTeX

Implementation Concern | Analysis Concern | Process Concern

„Specific Languages"

mbeddr



StateMachines - tutorial - [~/Documents/mbeddr/mbeddr.core/code/applications/tutorial]

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop


[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight {
  in event next(Trackpoint* tp) <no binding>
  in event reset() <no binding>
  out event crashNotification() => raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    //[ Here is a comment on a transition. ]
    on next [tp->alt == 0 m] -> airborne
    exit { points += TAKEOFF; } -> implements PointsForTakeoff
  } state bef
  state airborne {
    on next [tp->alt == 0 m && tp->speed == 0] -> crashed
    on next [tp->alt == 0 m && tp->
    on next [tp->speed > 200 mps &
    on next [tp->speed > 100 mps &
    on reset [ ] -> beforeFlight
  } state airborne
  state landing {
    on next [tp->speed == 0 mps] -> landed
    on next [tp->speed > 0 mps] -> landing { points--; } -> imp
```

Error: type int16/[m / s] is not comparable with (uint8 || int8)

```
alt                ^DataStructures.Trackpoint.alt (Member)
crashNotification  ^StateMachines.FlightAnalyzer.crashNotification (OutEvent)
id                 ^DataStructures.Trackpoint.id (Member)
speed              ^DataStructures.Trackpoint.speed (Member)
time               ^DataStructures.Trackpoint.time (Member)
x                  ^DataStructures.Trackpoint.x (Member)
y                  ^DataStructures.Trackpoint.y (Member)
```

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop


[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight
                          next(Trackpoint* tp)
             beforeFlight //[ Here is a comment on a transition. ]
                          [tp->alt == 0 m] -> airborne
             airborne     [tp->alt == 0 m && tp->speed == 0] -> crashe
                          [tp->alt == 0 m && tp->speed > 0 mps] -> lan
                          [tp->speed > 200 mps && tp->alt == 0 m] ->
                          [tp->speed > 100 mps && tp->speed <= 200 mp
                              tp->alt == 0 m] -> airborne
             landing      [tp->speed == 0 mps] -> landed
                          [tp->speed > 0 mps] -> landing -> implements Sh
             landed

                                             lyzer initial = 
                                                       next(Trackpo
             beforeFlight                              [tp->alt > 0
             composite state airborne initial = flying { [onTheGround
```

Open Source @ eclipse.org

Eclipse Public License 1.0

http://mbeddr.com

mbeddr

itemis France: Smart Meter

First significant mbeddr project
ca. 100,000 LoC
about to be finished
great modularity due to components
uses physical units extensively
great test coverage due to special extensions

mbeddr

ACCEnT Control.Lab

LMS INTERNATIONAL
Researchpark Haasrode 1237 | Interleuvenlaan 68 | B-3001 Leuven [Belgium]
T +32 16 384 200 | F +32 16 384 350 | info@lmsintl.com | www.lmsintl.com

Worldwide
For the address of your local representative, please visit www.lmsintl.com/lmsworldwide

LMS is a leading provider of test and mechatronic simulation software and engineering services in the automotive, aerospace and other advanced manufacturing industries. As a business segment within Siemens PLM Software, LMS provides a unique portfolio of products and services for manufacturing companies to manage the complexities of tomorrow's product development by incorporating model-based mechatronic simulation and advanced testing in the product development process. LMS tunes into mission-critical engineering attributes, ranging from system dynamics, structural integrity and sound quality to durability, safety and power consumption. With multi-domain and mechatronic simulation solutions, LMS addresses the complex engineering challenges associated with intelligent system design and model-based systems engineering. Thanks to its technology and more than 1250 dedicated people, LMS has become the partner of choice of more than 5000 manufacturing companies worldwide. LMS operates in more than 30 key locations around the world.

Siemens PLM Software

SIEMENS

LMS®

A Siemens Business

mbeddr

20+ Projects in various stages

by various “Big Name” companies.

Branching into other domains

insurance, financial, tax

7

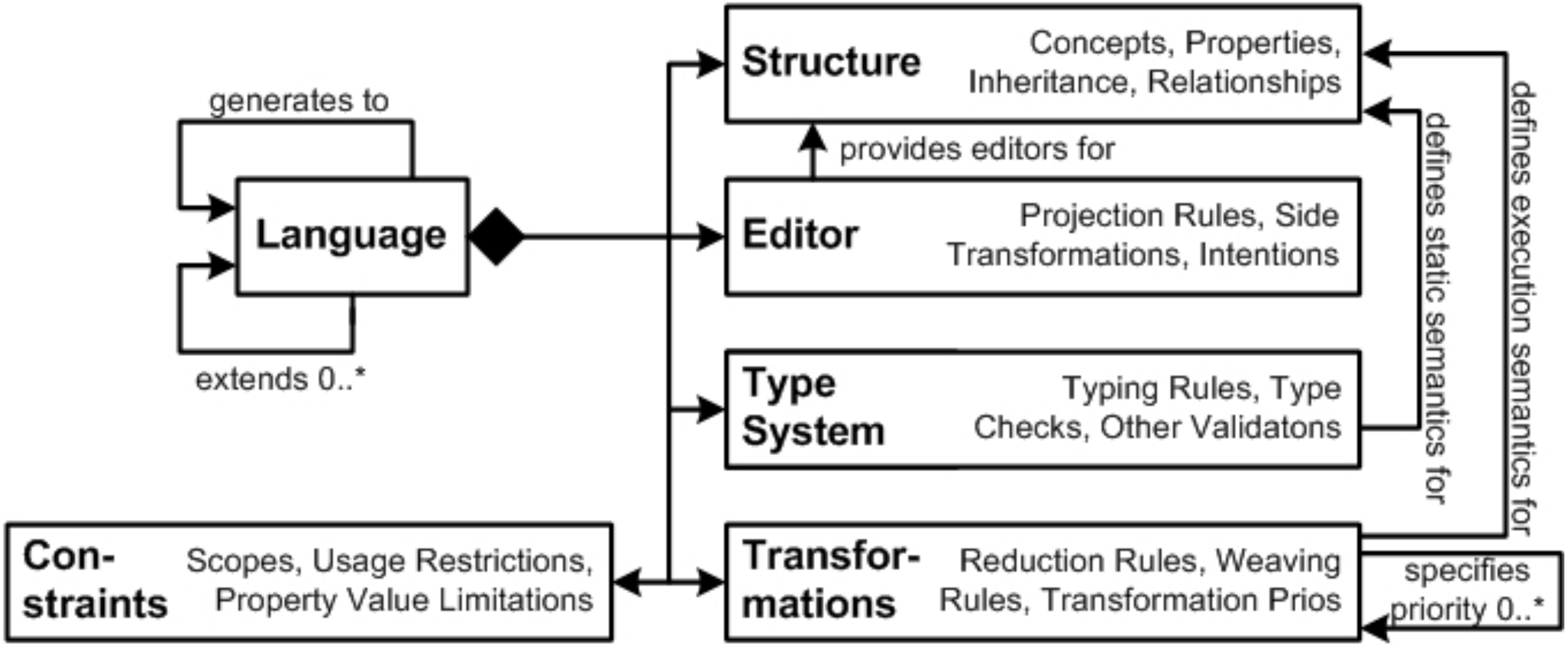
The Language Workbench

Open Source

Apache 2.0

http://jetbrains.com/mps

[Language Workbench]

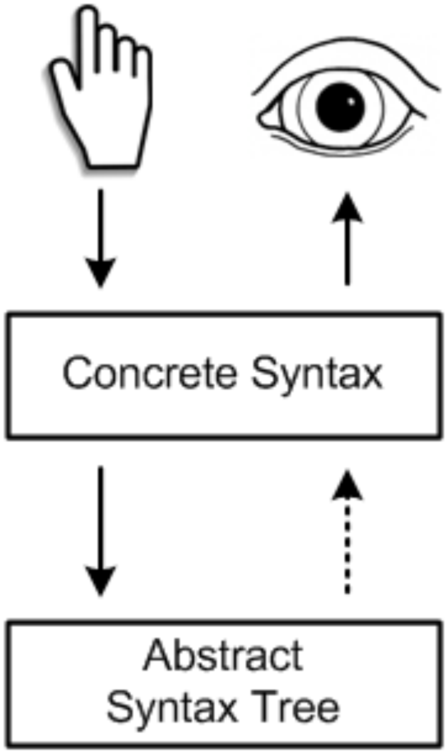

generates to
Language
extends 0..*
Structure
Concepts, Properties, Inheritance, Relationships
provides editors for
Editor
Projection Rules, Side Transformations, Intentions
Type System
Typing Rules, Type Checks, Other Validatons
Con-straints
Scopes, Usage Restrictions, Property Value Limitations
Transfor-mations
Reduction Rules, Weaving Rules, Transformation Prios
defines static semantics for
defines execution semantics for
specifies priority 0..*

+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...

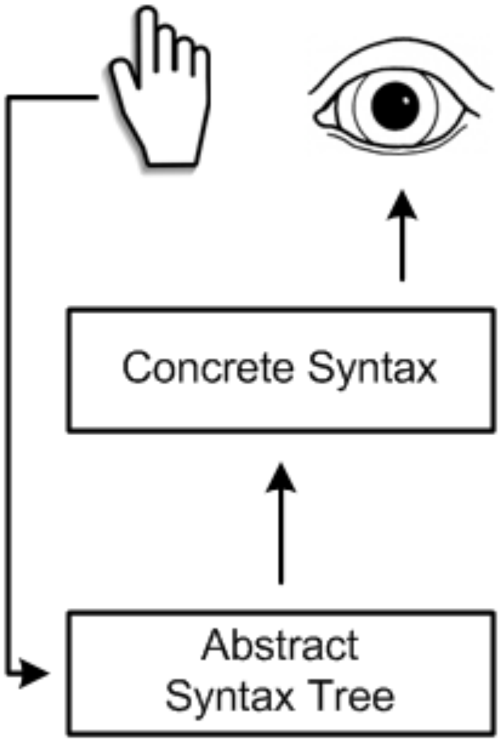
Projectional Editing

[Projectional Editing]

Parsing



Projectional Editing



[Projectional Editing]

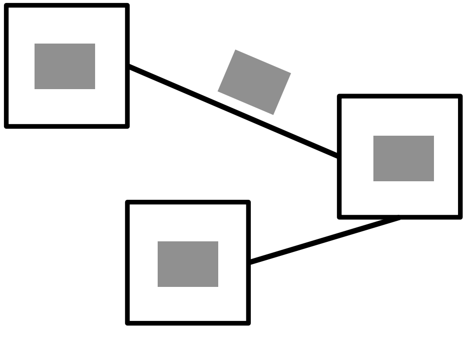
Syntactic Flexibility

Regular Code/Text

Mathematical

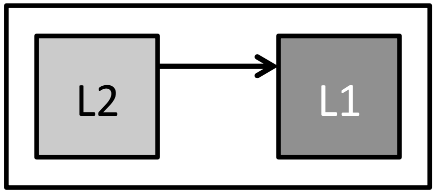
Tables

Graphical



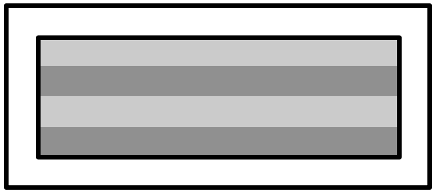
[Projectional Editing]

Language Composition



Separate Files

Type System
Transformation
Constraints



In One File

Type System
Transformation
Constraints
Syntax
IDE

LWBs make Languages Easier

Multiple (Mixed) Notations

Language Extension and Composition

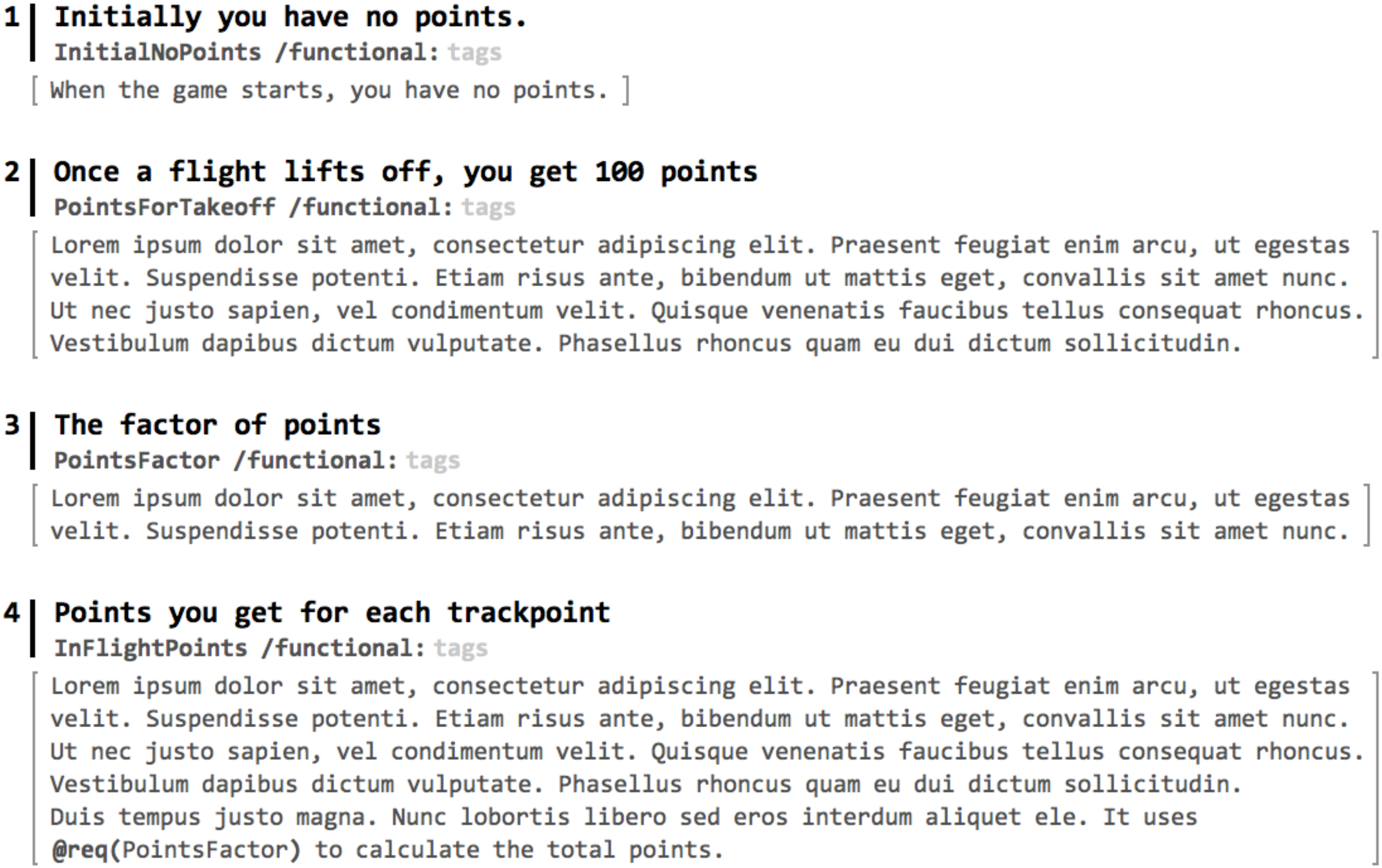
MPS works, but not the only one.

8

Demo

[Requirements]

DEMO



1 Initially you have no points.

InitialNoPoints /functional: tags

[When the game starts, you have no points.]

2 Once a flight lifts off, you get 100 points

PointsForTakeoff /functional: tags

[Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.]

3 The factor of points

PointsFactor /functional: tags

[Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc.]

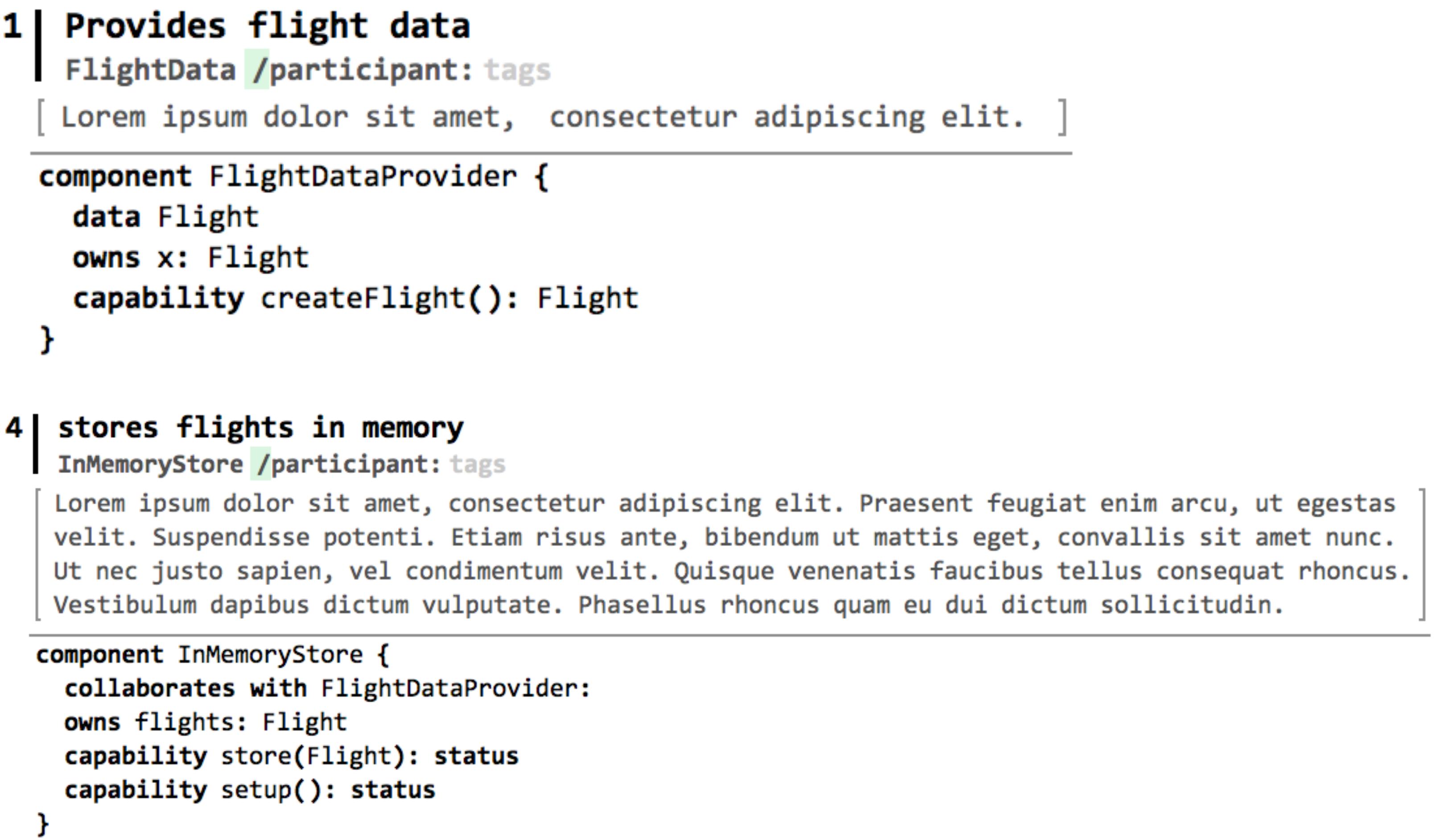
4 Points you get for each trackpoint

InFlightPoints /functional: tags

[Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.
Duis tempus justo magna. Nunc lobortis libero sed eros interdum aliquet ele. It uses **@req**(PointsFactor) to calculate the total points.]

[Requirements + Components]

DEMO



1 **Provides flight data**

FlightData /participant: tags

[Lorem ipsum dolor sit amet, consectetur adipiscing elit.]

```
component FlightDataProvider {
  data Flight
  owns x: Flight
  capability createFlight(): Flight
}
```

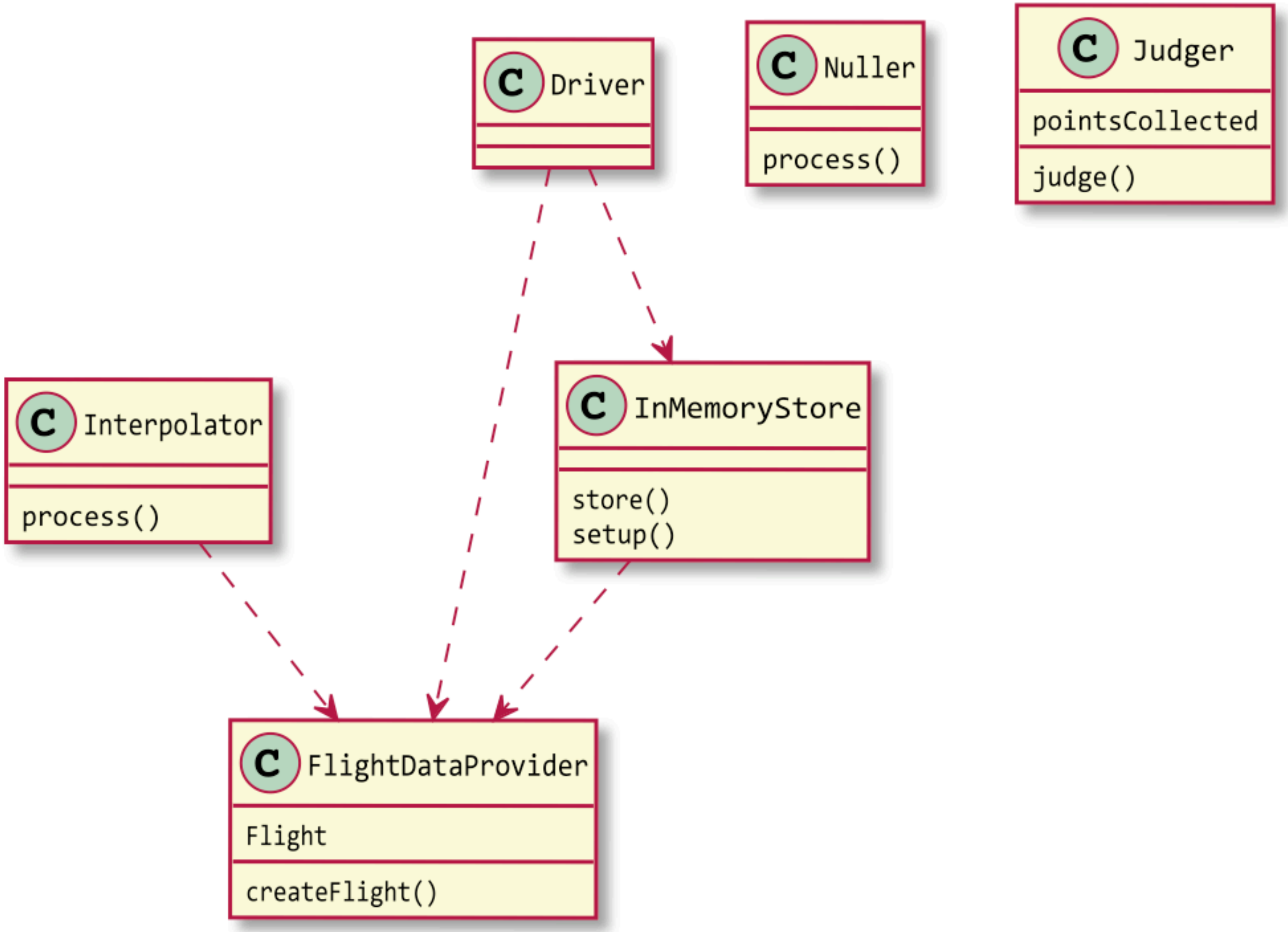
4 **stores flights in memory**

InMemoryStore /participant: tags

[Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin.]

```
component InMemoryStore {
  collaborates with FlightDataProvider:
  owns flights: Flight
  capability store(Flight): status
  capability setup(): status
}
```

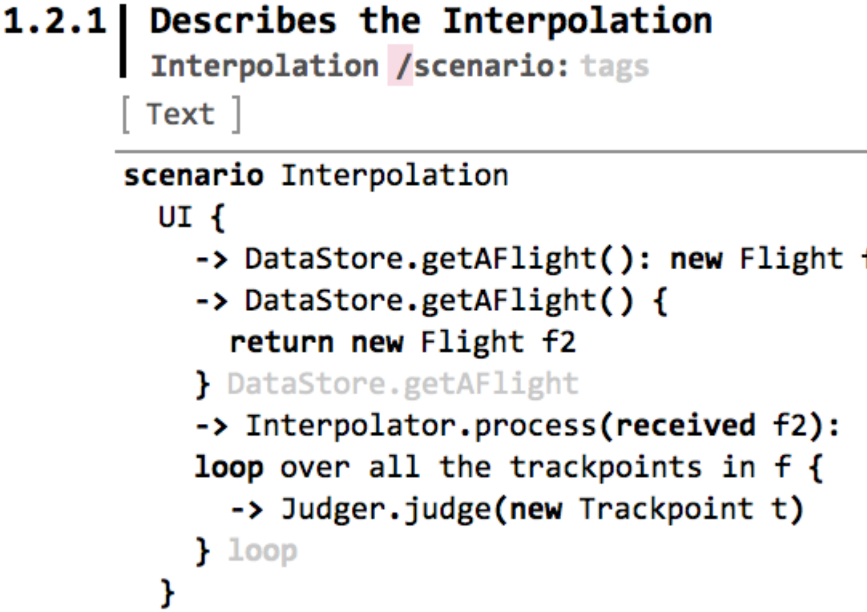
[Requirements + Components]



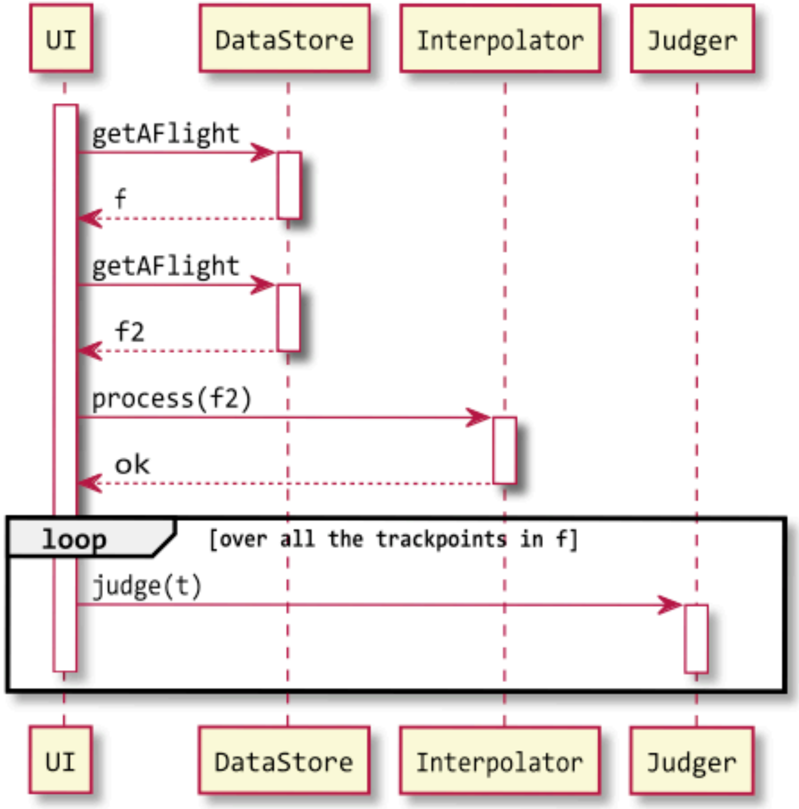
DEMO

[Collaborations and Scenarios]

DEMO

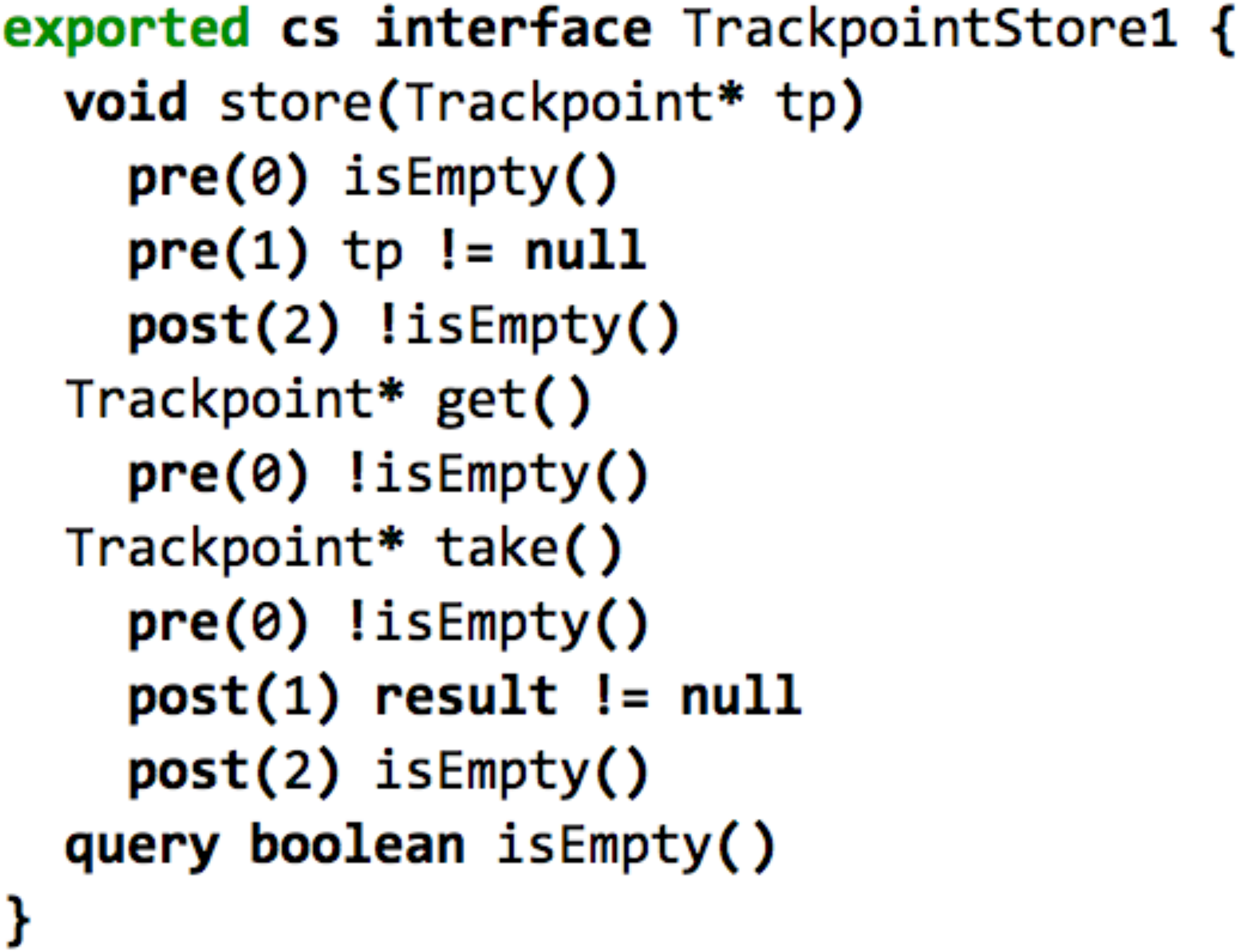


```
1.2.1 | Describes the Interpolation
      | Interpolation /scenario: tags
      [ Text ]
      ----------------------------------------------
      scenario Interpolation
        UI {
          -> DataStore.getAFlight(): new Flight f
          -> DataStore.getAFlight() {
            return new Flight f2
          } DataStore.getAFlight
          -> Interpolator.process(received f2):
          loop over all the trackpoints in f {
            -> Judger.judge(new Trackpoint t)
          } loop
        }
```



[Interfaces with Contracts]

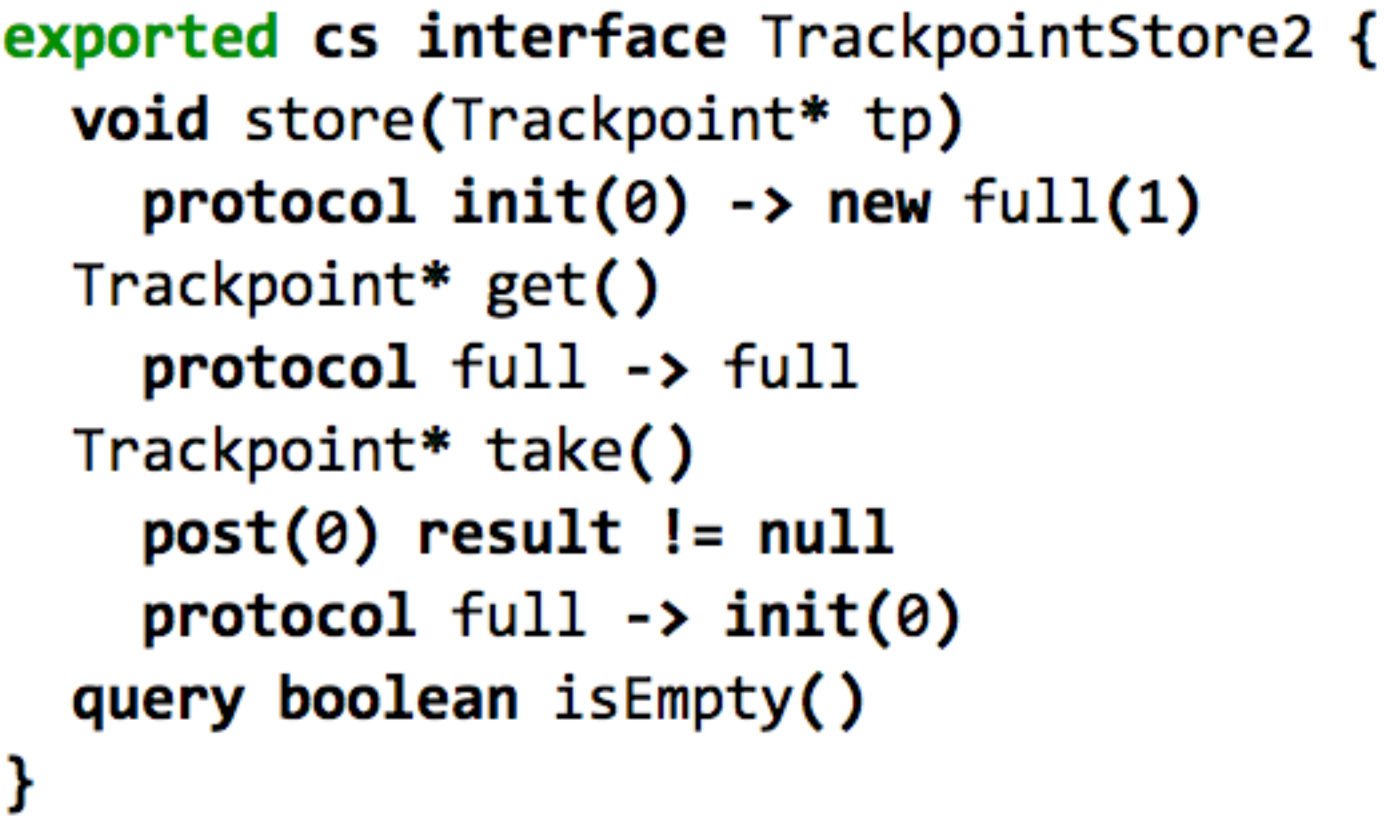
DEMO



```
exported cs interface TrackpointStore1 {
  void store(Trackpoint* tp)
    pre(0) isEmpty()
    pre(1) tp != null
    post(2) !isEmpty()
  Trackpoint* get()
    pre(0) !isEmpty()
  Trackpoint* take()
    pre(0) !isEmpty()
    post(1) result != null
    post(2) isEmpty()
  query boolean isEmpty()
}
```

[Interfaces with Protocols]

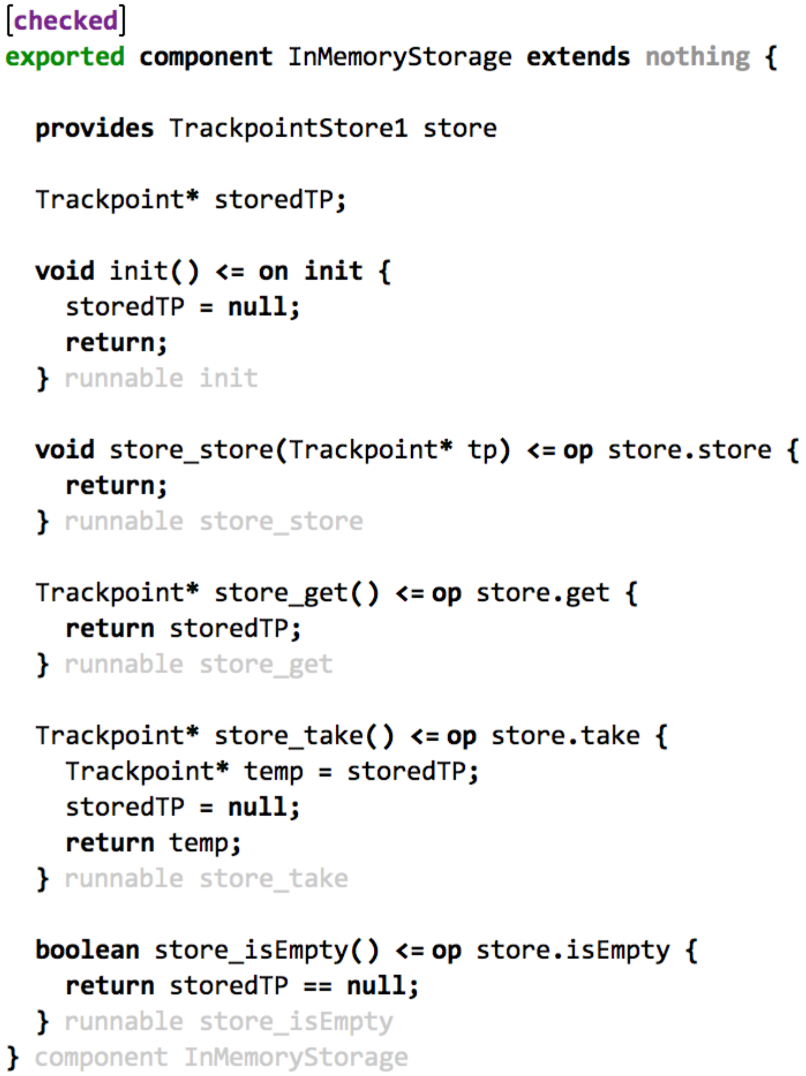
DEMO



```
exported cs interface TrackpointStore2 {
  void store(Trackpoint* tp)
    protocol init(0) -> new full(1)
  Trackpoint* get()
    protocol full -> full
  Trackpoint* take()
    post(0) result != null
    protocol full -> init(0)
  query boolean isEmpty()
}
```

[Components]

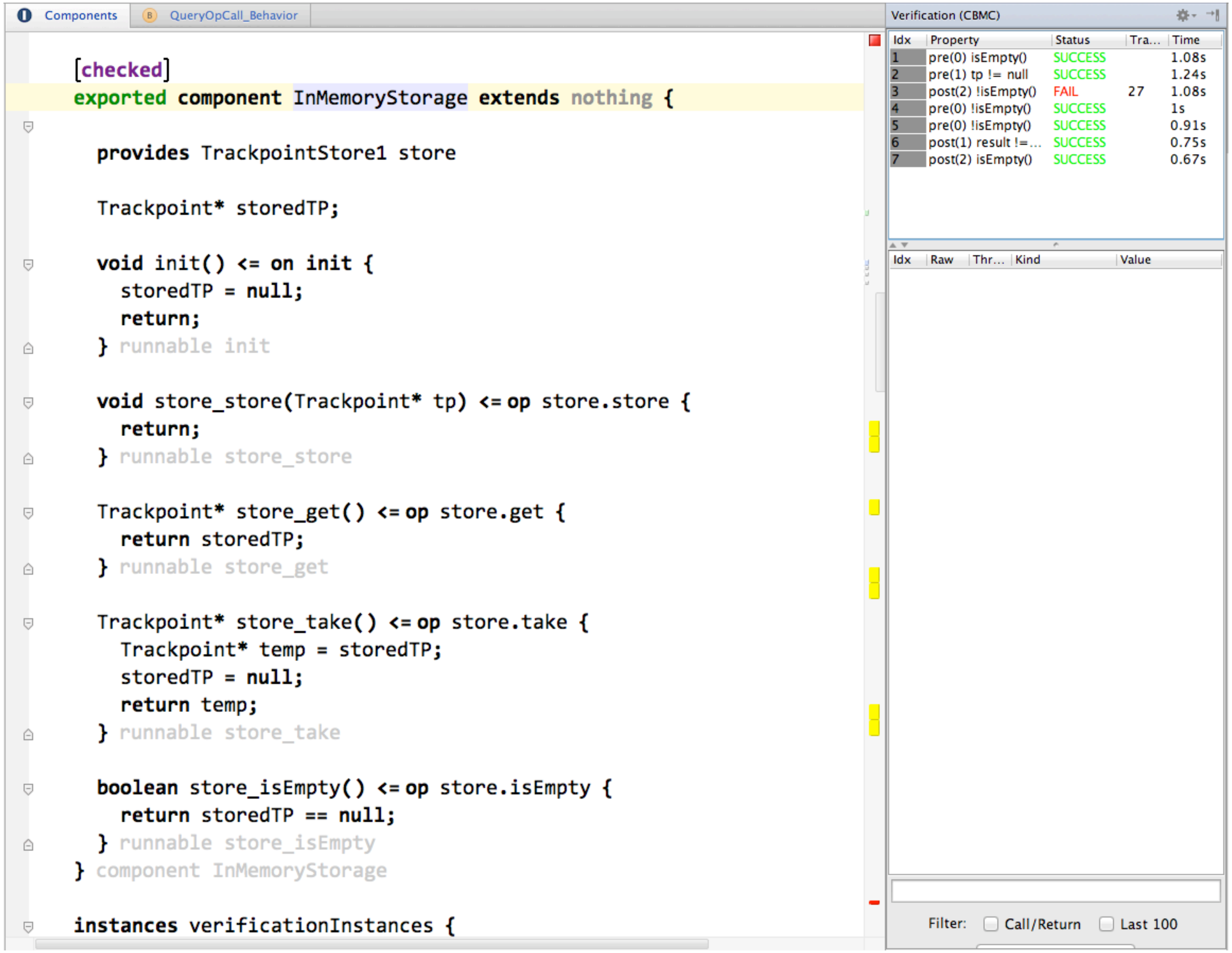
DEMO



```
[checked]
exported component InMemoryStorage extends nothing {

  provides TrackpointStore1 store

  Trackpoint* storedTP;

  void init() <= on init {
    storedTP = null;
    return;
  } runnable init

  void store_store(Trackpoint* tp) <= op store.store {
    return;
  } runnable store_store

  Trackpoint* store_get() <= op store.get {
    return storedTP;
  } runnable store_get

  Trackpoint* store_take() <= op store.take {
    Trackpoint* temp = storedTP;
    storedTP = null;
    return temp;
  } runnable store_take

  boolean store_isEmpty() <= op store.isEmpty {
    return storedTP == null;
  } runnable store_isEmpty
} component InMemoryStorage
```

[Component Verification]

DEMO



```
[checked]
exported component InMemoryStorage extends nothing {

  provides TrackpointStore1 store

  Trackpoint* storedTP;

  void init() <= on init {
    storedTP = null;
    return;
  } runnable init

  void store_store(Trackpoint* tp) <= op store.store {
    return;
  } runnable store_store

  Trackpoint* store_get() <= op store.get {
    return storedTP;
  } runnable store_get

  Trackpoint* store_take() <= op store.take {
    Trackpoint* temp = storedTP;
    storedTP = null;
    return temp;
  } runnable store_take

  boolean store_isEmpty() <= op store.isEmpty {
    return storedTP == null;
  } runnable store_isEmpty
} component InMemoryStorage

instances verificationInstances {
```

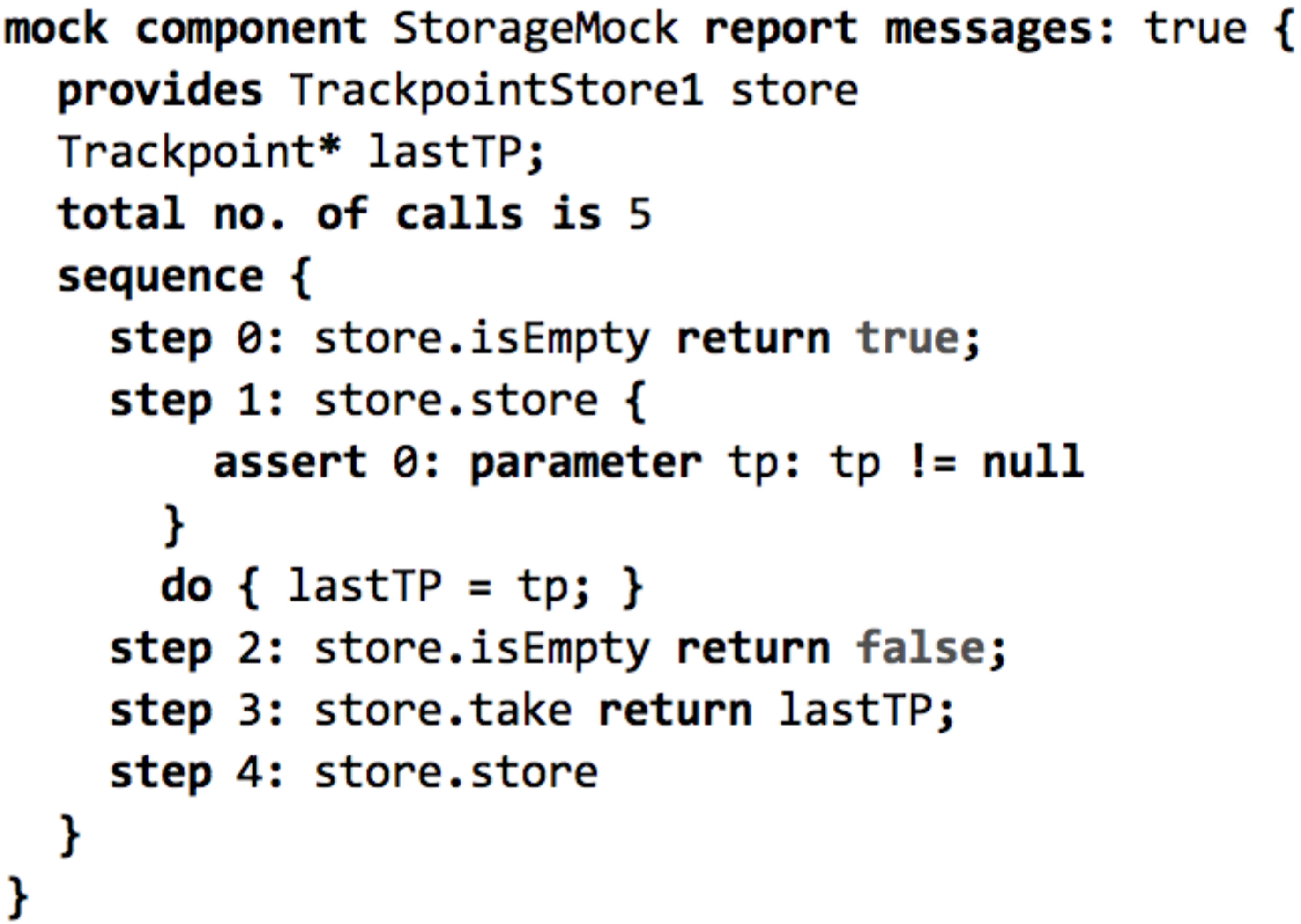
Verification (CBMC)

Idx	Property	Status	Tra...	Time
1	pre(0) isEmpty()	SUCCESS		1.08s
2	pre(1) tp != null	SUCCESS		1.24s
3	post(2) !isEmpty()	FAIL	27	1.08s
4	pre(0) !isEmpty()	SUCCESS		1s
5	pre(0) !isEmpty()	SUCCESS		0.91s
6	post(1) result !=...	SUCCESS		0.75s
7	post(2) isEmpty()	SUCCESS		0.67s

Idx	Raw	Thr...	Kind	Value

Filter: □ Call/Return □ Last 100

[Mocks for Testing]

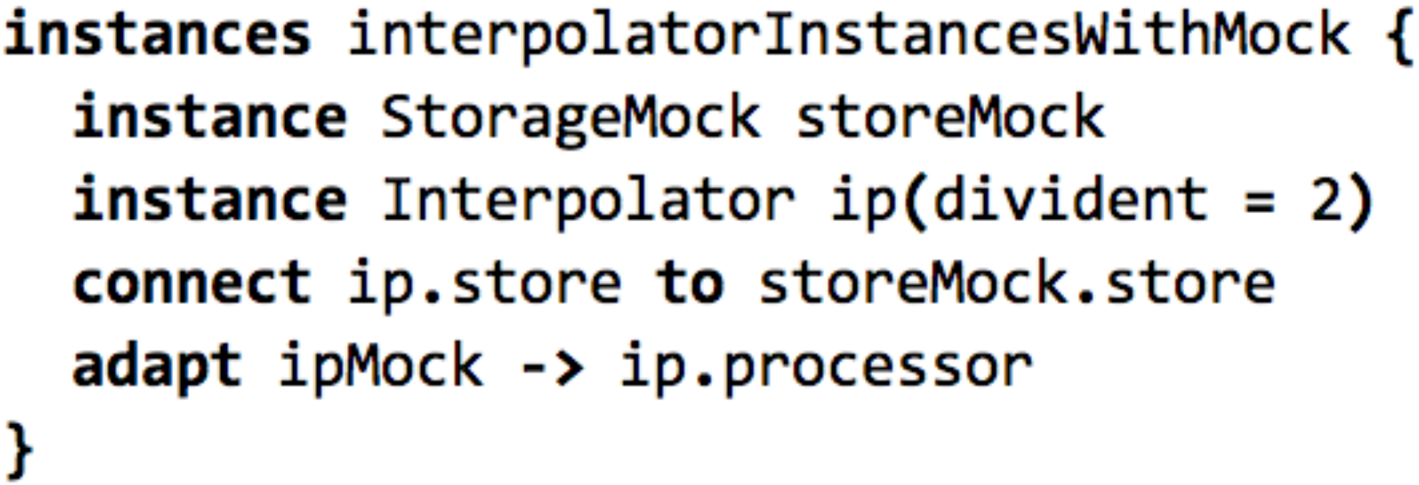


```
mock component StorageMock report messages: true {
  provides TrackpointStore1 store
  Trackpoint* lastTP;
  total no. of calls is 5
  sequence {
    step 0: store.isEmpty return true;
    step 1: store.store {
        assert 0: parameter tp: tp != null
      }
      do { lastTP = tp; }
    step 2: store.isEmpty return false;
    step 3: store.take return lastTP;
    step 4: store.store
  }
}
```

DEMO

[Instantiation]

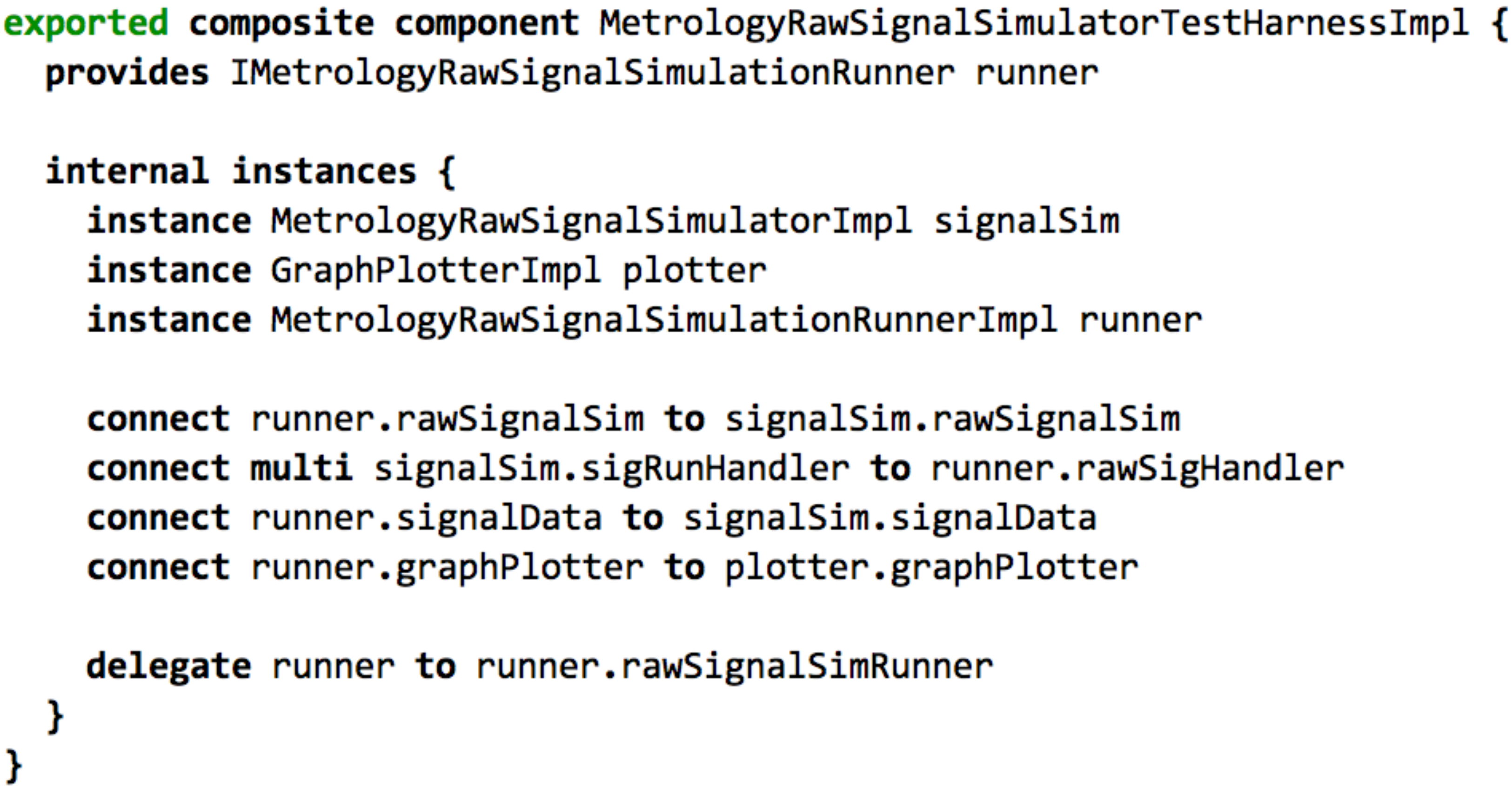
DEMO



```
instances interpolatorInstancesWithMock {
  instance StorageMock storeMock
  instance Interpolator ip(divident = 2)
  connect ip.store to storeMock.store
  adapt ipMock -> ip.processor
}
```

[Composite Components]

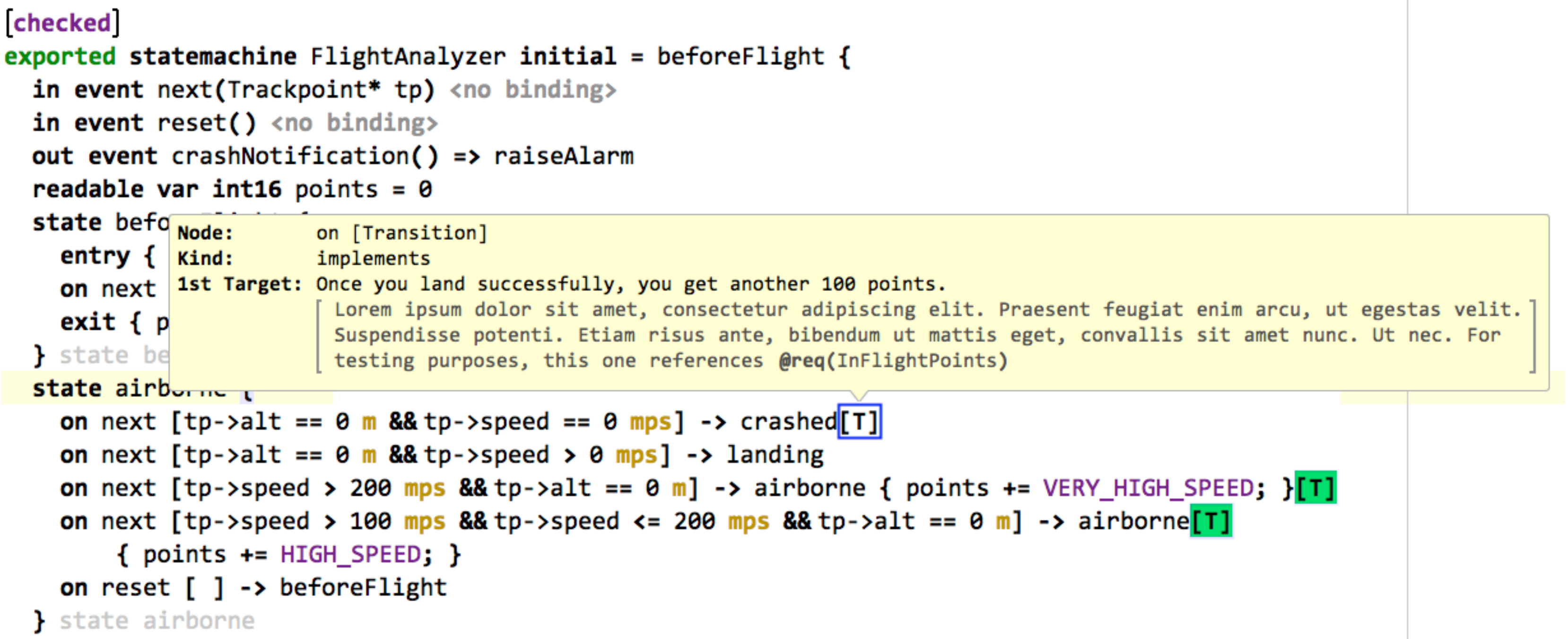
DEMO



```
exported composite component MetrologyRawSignalSimulatorTestHarnessImpl {
  provides IMetrologyRawSignalSimulationRunner runner

  internal instances {
    instance MetrologyRawSignalSimulatorImpl signalSim
    instance GraphPlotterImpl plotter
    instance MetrologyRawSignalSimulationRunnerImpl runner

    connect runner.rawSignalSim to signalSim.rawSignalSim
    connect multi signalSim.sigRunHandler to runner.rawSigHandler
    connect runner.signalData to signalSim.signalData
    connect runner.graphPlotter to plotter.graphPlotter

    delegate runner to runner.rawSignalSimRunner
  }
}
```

[Tracing from Code]

DEMO

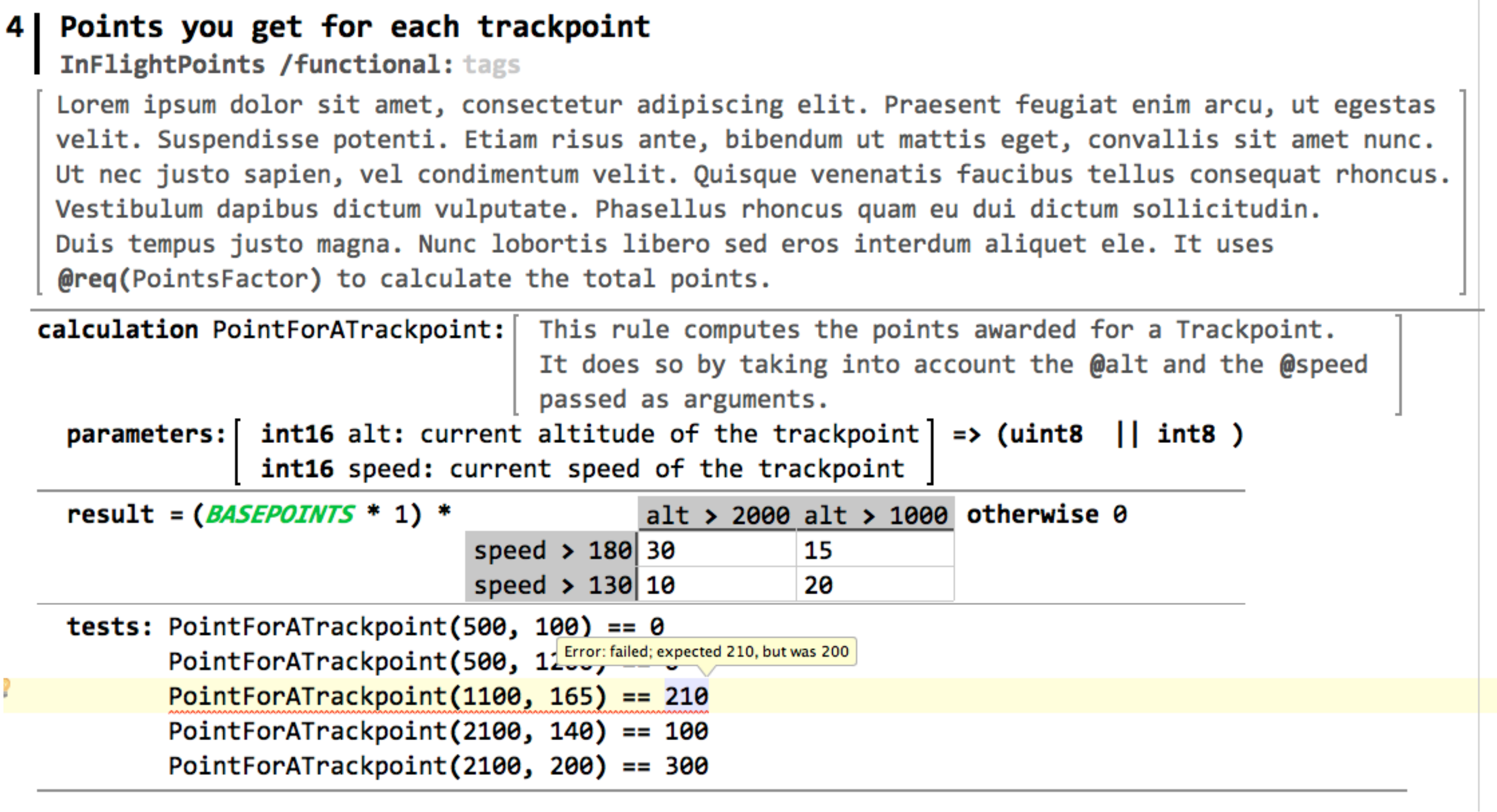


```
[checked]
exported statemachine FlightAnalyzer initial = beforeFlight {
  in event next(Trackpoint* tp) <no binding>
  in event reset() <no binding>
  out event crashNotification() => raiseAlarm
  readable var int16 points = 0
  state befo
    entry {
    on next
    exit { p
  } state be
  state airb
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed[T]
    on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
    on next [tp->speed > 200 mps && tp->alt == 0 m] -> airborne { points += VERY_HIGH_SPEED; }[T]
    on next [tp->speed > 100 mps && tp->speed <= 200 mps && tp->alt == 0 m] -> airborne[T]
        { points += HIGH_SPEED; }
    on reset [ ] -> beforeFlight
  } state airborne
```

```
Node:        on [Transition]
Kind:        implements
1st Target:  Once you land successfully, you get another 100 points.
             [ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. ]
             [ Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec. For ]
             [ testing purposes, this one references @req(InFlightPoints)                                          ]
```

[Formal, Testable Req.]

DEMO



4 **Points you get for each trackpoint**

InFlightPoints /functional: tags

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent feugiat enim arcu, ut egestas velit. Suspendisse potenti. Etiam risus ante, bibendum ut mattis eget, convallis sit amet nunc. Ut nec justo sapien, vel condimentum velit. Quisque venenatis faucibus tellus consequat rhoncus. Vestibulum dapibus dictum vulputate. Phasellus rhoncus quam eu dui dictum sollicitudin. Duis tempus justo magna. Nunc lobortis libero sed eros interdum aliquet ele. It uses **@req**(PointsFactor) to calculate the total points.

```
calculation PointForATrackpoint: This rule computes the points awarded for a Trackpoint.
                                 It does so by taking into account the @alt and the @speed
                                 passed as arguments.
  parameters: int16 alt: current altitude of the trackpoint  => (uint8  || int8 )
              int16 speed: current speed of the trackpoint

  result = (BASEPOINTS * 1) *
```

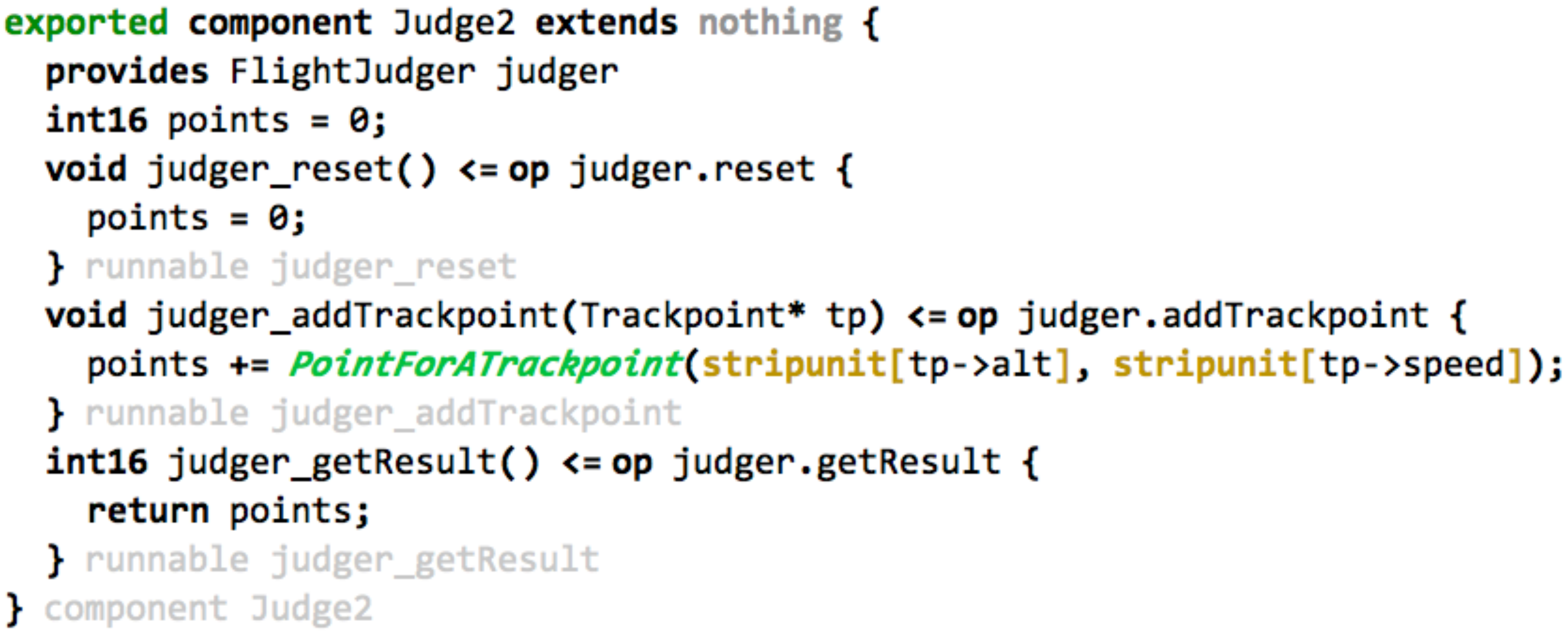
	alt > 2000	alt > 1000	otherwise 0
speed > 180	30	15	
speed > 130	10	20	

```
  tests: PointForATrackpoint(500, 100) == 0
         PointForATrackpoint(500, 12
         PointForATrackpoint(1100, 165) == 210
         PointForATrackpoint(2100, 140) == 100
         PointForATrackpoint(2100, 200) == 300
```

Error: failed; expected 210, but was 200

[Using „Req Code“ in Comp.]

DEMO



```
exported component Judge2 extends nothing {
  provides FlightJudger judger
  int16 points = 0;
  void judger_reset() <= op judger.reset {
    points = 0;
  } runnable judger_reset
  void judger_addTrackpoint(Trackpoint* tp) <= op judger.addTrackpoint {
    points += PointForATrackpoint(stripunit[tp->alt], stripunit[tp->speed]);
  } runnable judger_addTrackpoint
  int16 judger_getResult() <= op judger.getResult {
    return points;
  } runnable judger_getResult
} component Judge2
```

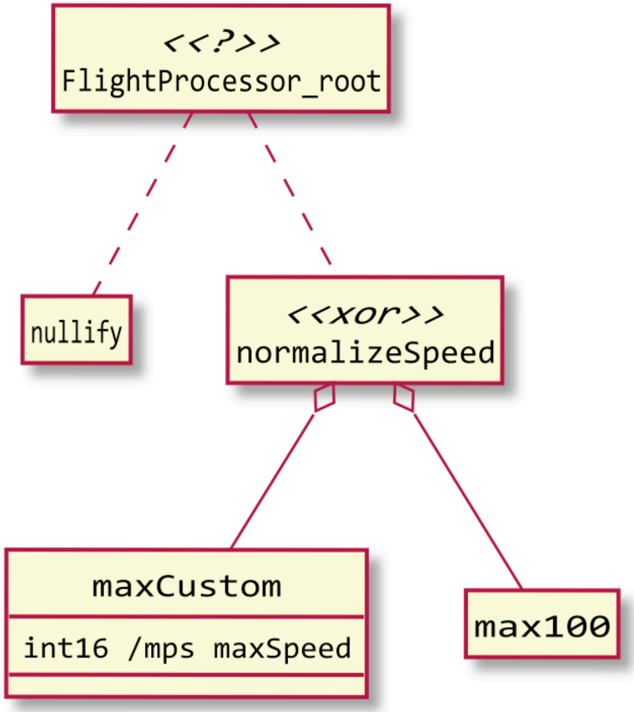
[PLE Variability]

DEMO

```
feature model FlightProcessor
root ? {
  nullify
  normalizeSpeed xor {
    maxCustom [int16/mps/ maxSpeed]
    max100
  }
}
```

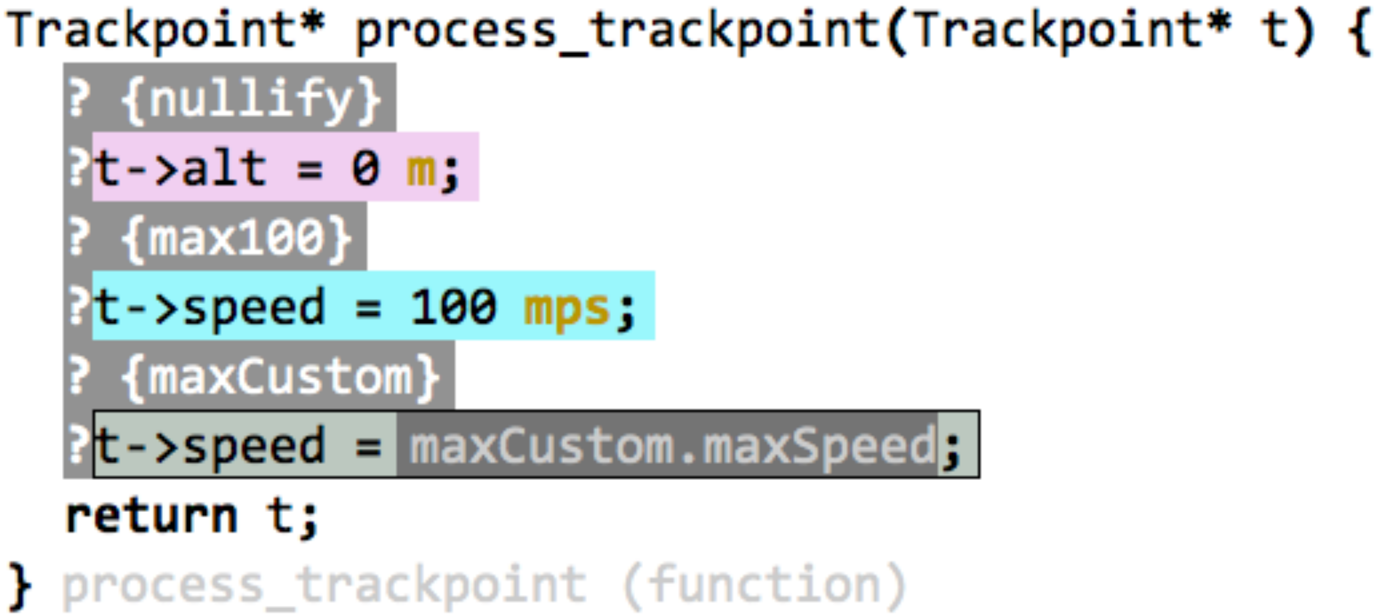
```
derived features
```

```
configuration model cfgDoNothing configures FlightProcessor
  FlightProcessor_root {

  }

configuration model cfgNullifyOnly configures FlightProcessor
  FlightProcessor_root {
    nullify
  }

configuration model cfgNullifyMaxAt200 configures FlightProcessor
  FlightProcessor_root {
    nullify
    normalizeSpeed {
      maxCustom [maxSpeed = 200 mps]
    }
  }
```

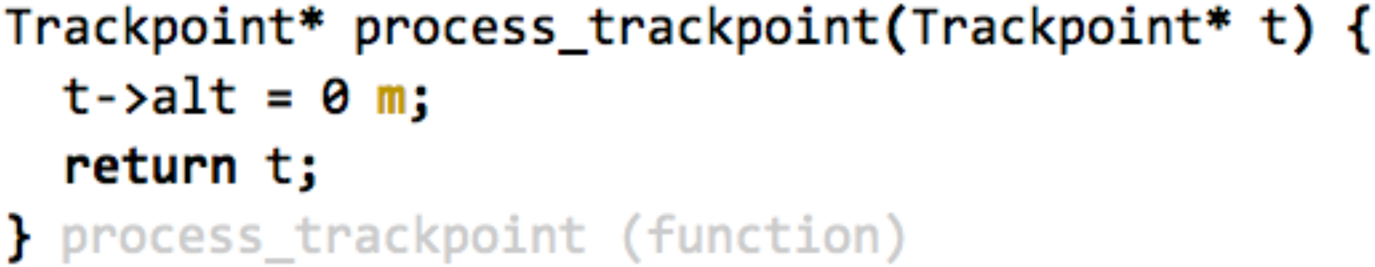


[PLE Variability]

DEMO



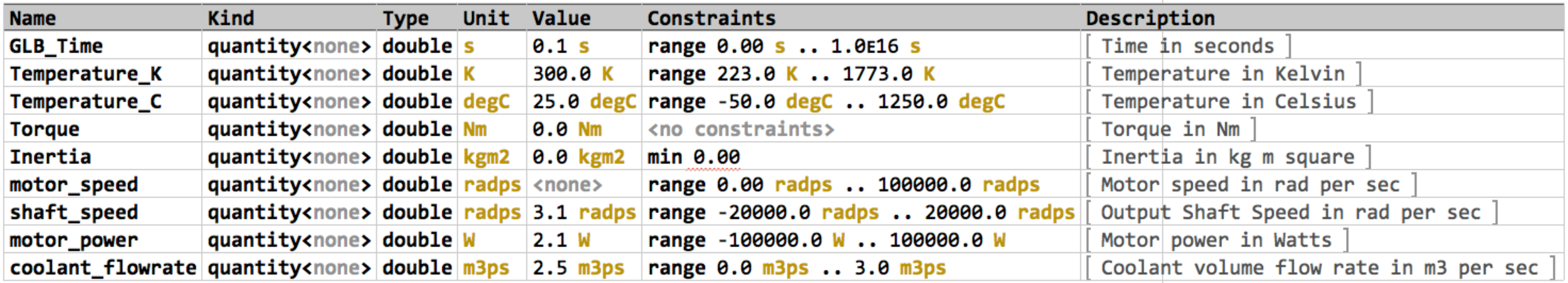
```
Trackpoint* process_trackpoint(Trackpoint* t) {
  ? {nullify}
  ?t->alt = 0 m;
  ? {max100}
  ?t->speed = 100 mps;
  ? {maxCustom}
  ?t->speed = maxCustom.maxSpeed;
  return t;
} process_trackpoint (function)
```



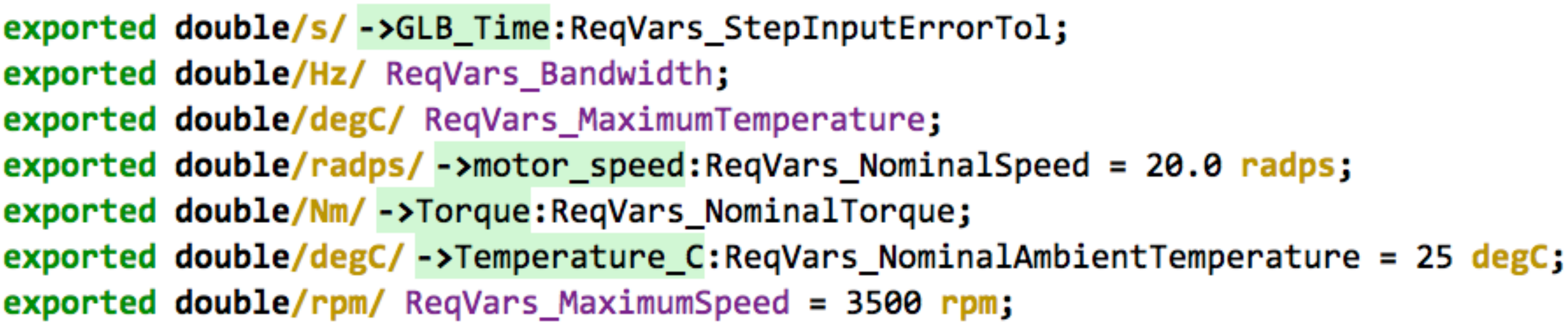
```
Trackpoint* process_trackpoint(Trackpoint* t) {
  t->alt = 0 m;
  return t;
} process_trackpoint (function)
```

[Controlled Names]

DEMO



Name	Kind	Type	Unit	Value	Constraints	Description
GLB_Time	quantity<none>	double	s	0.1 s	range 0.00 s .. 1.0E16 s	[Time in seconds]
Temperature_K	quantity<none>	double	K	300.0 K	range 223.0 K .. 1773.0 K	[Temperature in Kelvin]
Temperature_C	quantity<none>	double	degC	25.0 degC	range -50.0 degC .. 1250.0 degC	[Temperature in Celsius]
Torque	quantity<none>	double	Nm	0.0 Nm	<no constraints>	[Torque in Nm]
Inertia	quantity<none>	double	kgm2	0.0 kgm2	min 0.00	[Inertia in kg m square]
motor_speed	quantity<none>	double	radps	<none>	range 0.00 radps .. 100000.0 radps	[Motor speed in rad per sec]
shaft_speed	quantity<none>	double	radps	3.1 radps	range -20000.0 radps .. 20000.0 radps	[Output Shaft Speed in rad per sec]
motor_power	quantity<none>	double	W	2.1 W	range -100000.0 W .. 100000.0 W	[Motor power in Watts]
coolant_flowrate	quantity<none>	double	m3ps	2.5 m3ps	range 0.0 m3ps .. 3.0 m3ps	[Coolant volume flow rate in m3 per sec]



```
exported double/s/ ->GLB_Time:ReqVars_StepInputErrorTol;
exported double/Hz/ ReqVars_Bandwidth;
exported double/degC/ ReqVars_MaximumTemperature;
exported double/radps/ ->motor_speed:ReqVars_NominalSpeed = 20.0 radps;
exported double/Nm/ ->Torque:ReqVars_NominalTorque;
exported double/degC/ ->Temperature_C:ReqVars_NominalAmbientTemperature = 25 degC;
exported double/rpm/ ReqVars_MaximumSpeed = 3500 rpm;
```

Architectural Abstractions first class

Code-integrated where useful

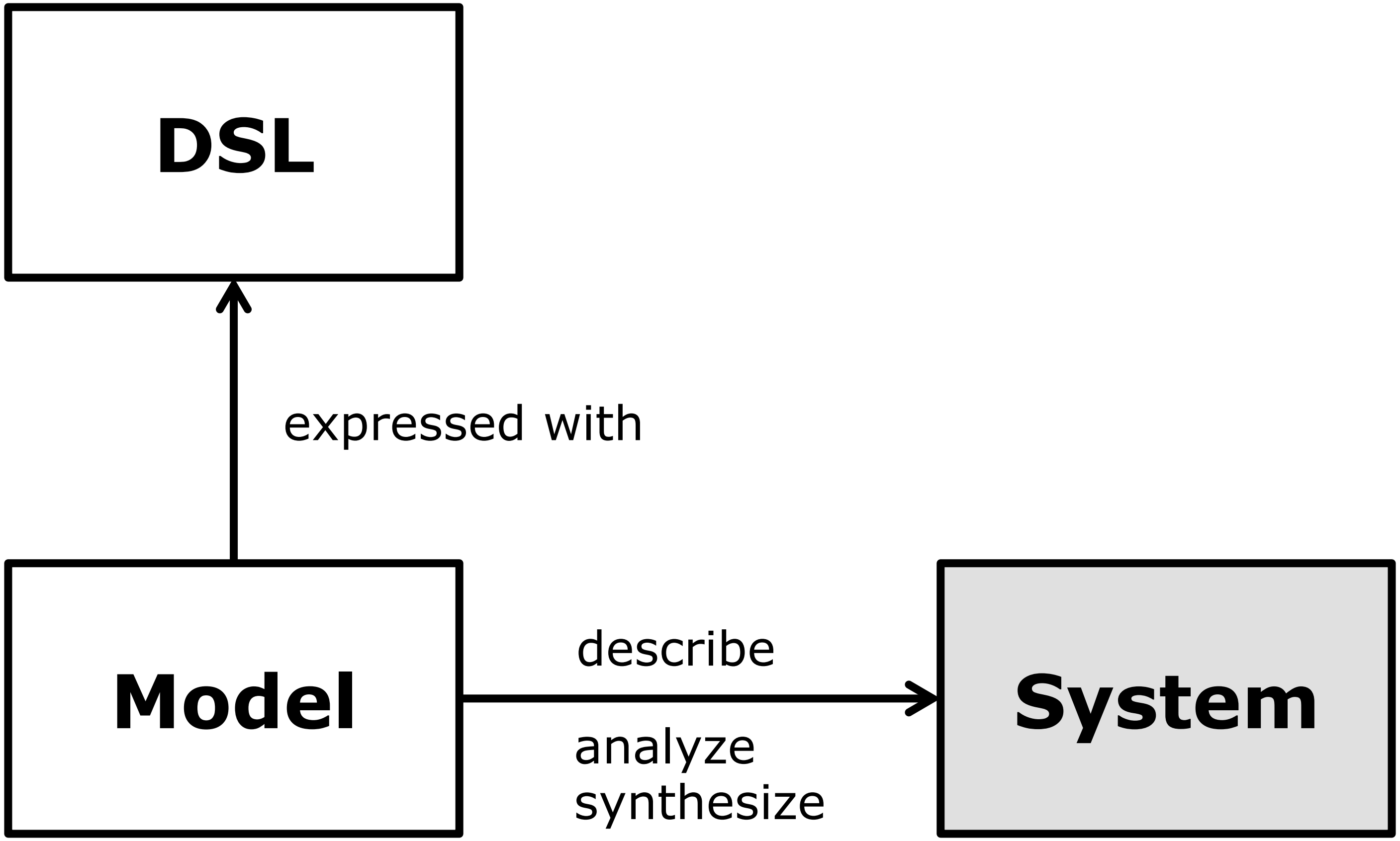
Analysis & Synthesis

Support Cross-Cutting Concerns

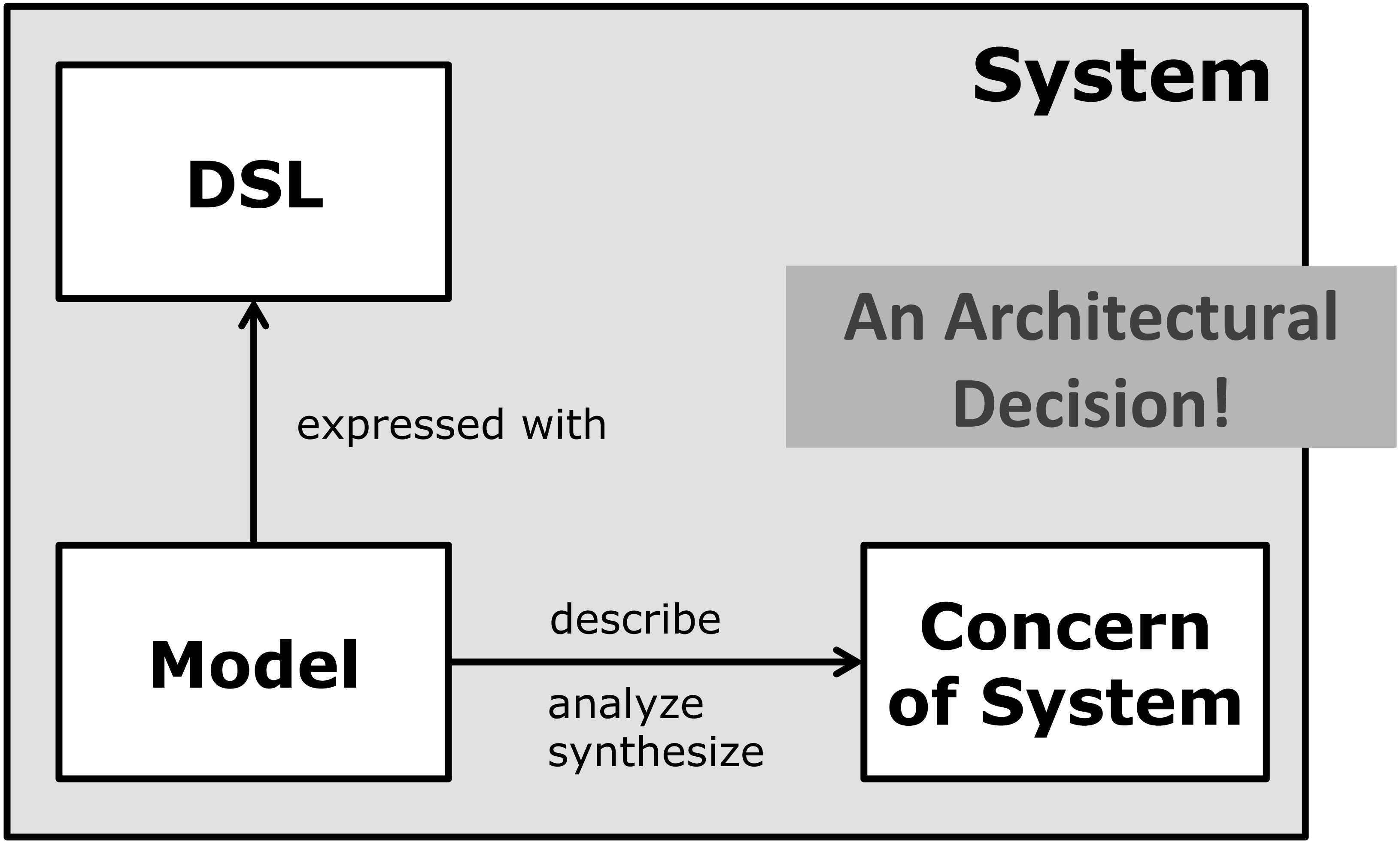
9

A different Perspective

[DSLs for Describing Architecture]

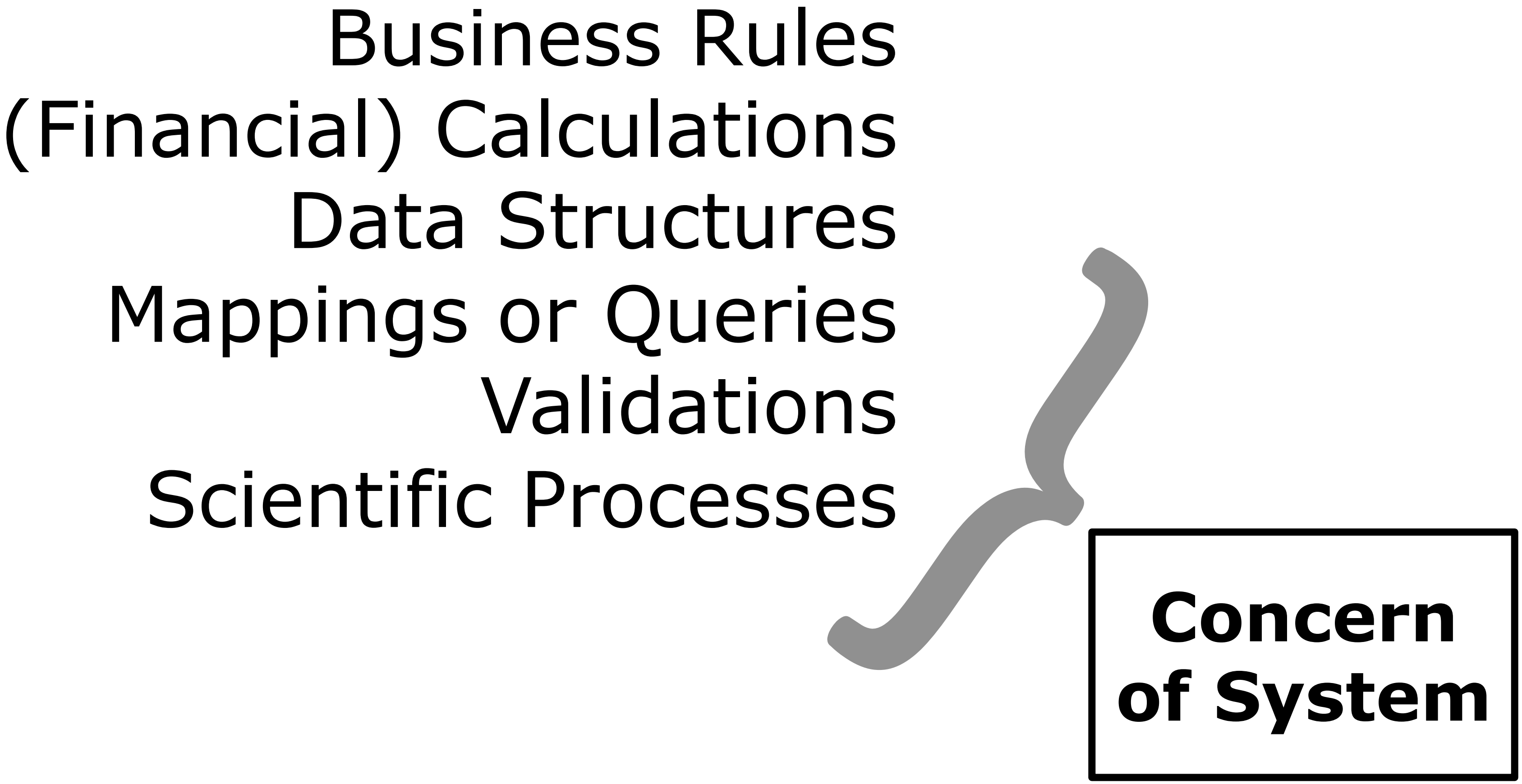


[DSLs as part of Systems]



An Architectural Decision!

[DSLs as part of Systems]



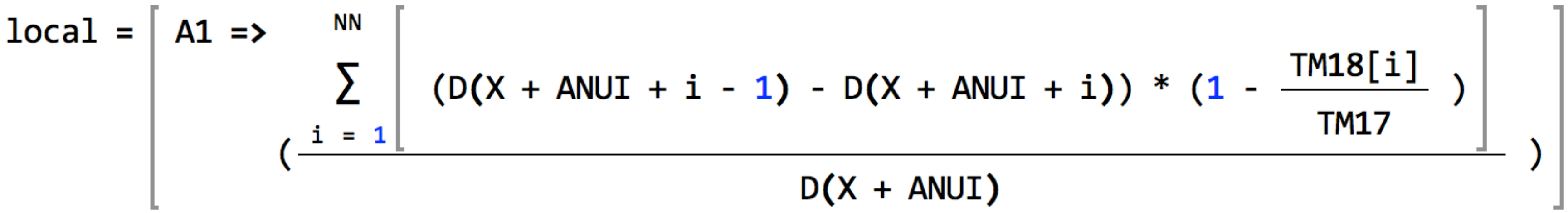
Business Rules
(Financial) Calculations
Data Structures
Mappings or Queries
Validations
Scientific Processes

Concern of System

[Examples]

❶

Insurance rules and products



$$\text{local} = \left[\text{A1} \Rightarrow \left(\frac{\sum_{i=1}^{NN} \left[(D(X + ANUI + i - 1) - D(X + ANUI + i)) * \left(1 - \frac{TM18[i]}{TM17}\right) \right]}{D(X + ANUI)} \right) \right]$$

❷

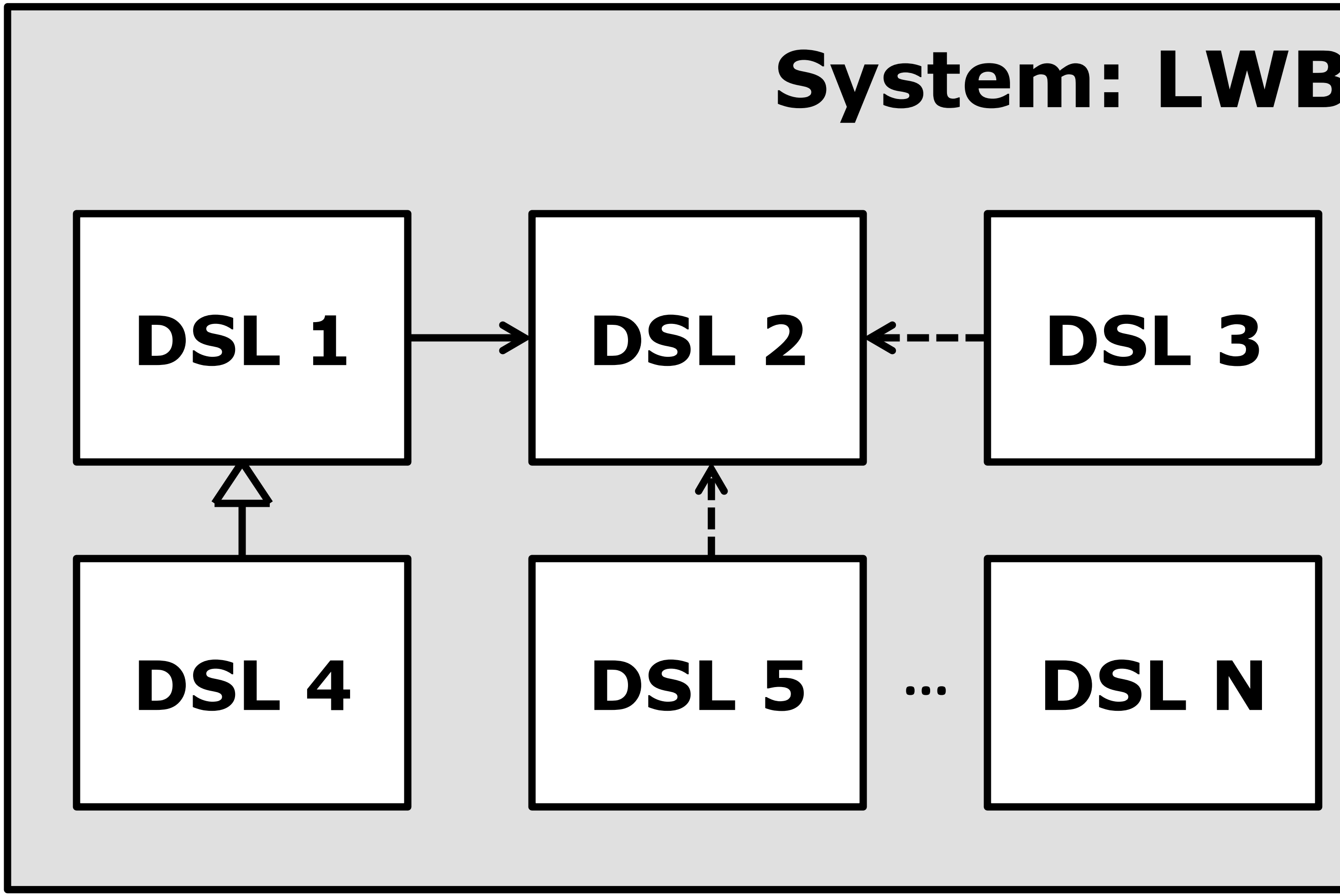
Tax/Benfits Rules (DTA Toeslagen)

Using DSLs is an Architectural Decision

Language Workbenches are the basis

Language Engineering is Efficient.

[Language-Oriented Applications]



If you have to build a tool, consider using an LWB as the foundation,

and recasting the „application“ as a set of languages.

If you have to build a tool, consider using an LWB as

Generic Tools

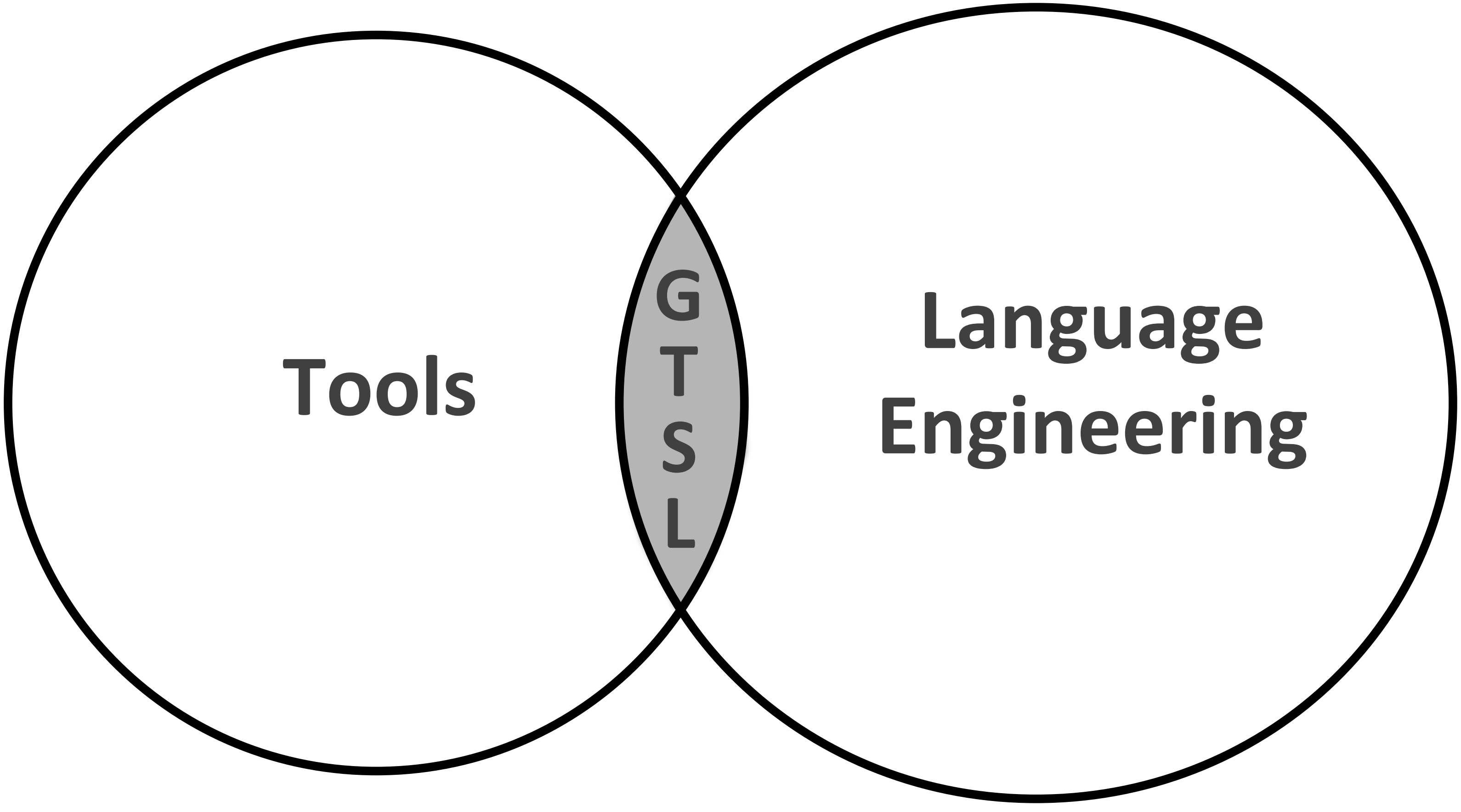
the foundation,

and recasting the „application“ as a set of languages.

Specific Languages

Generic Tools

Specific Languages


Tools
G T S L
Language Engineering

Tools are ways to work with Data.

work {

author
read
analyze
process

Data

Formats are

almost

Languages.

almost

[almost]

Structure, Constraints, Semantics

Data Format

[almost]

Structure, Constraints, Semantics

Data Format + Syntax + IDE

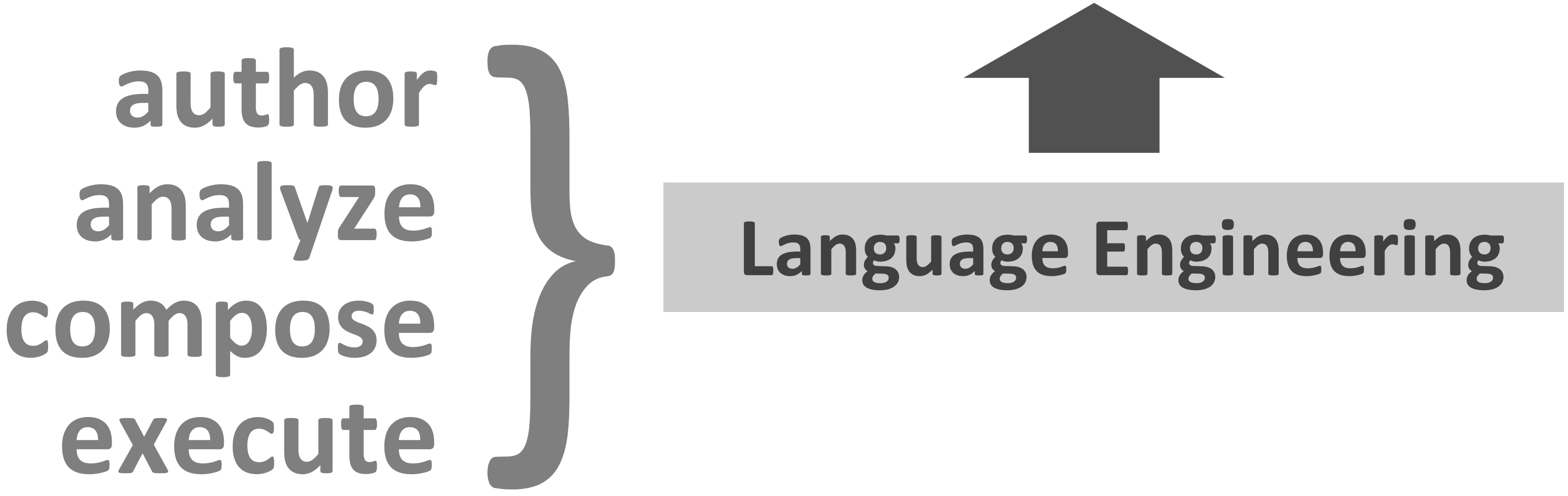
Language

[almost]

Structure, Constraints, Semantics

Data Format + Syntax + IDE

Language



author
analyze
compose
execute

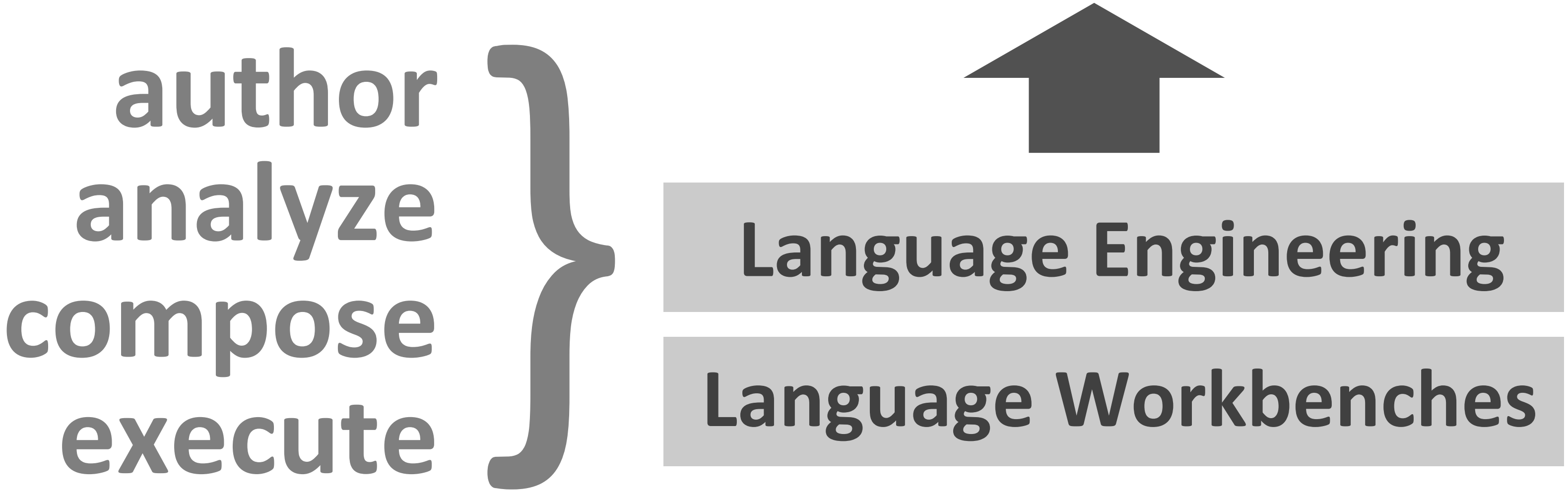
Language Engineering

[almost]

Structure, Constraints, Semantics

Data Format + Syntax + IDE

Language

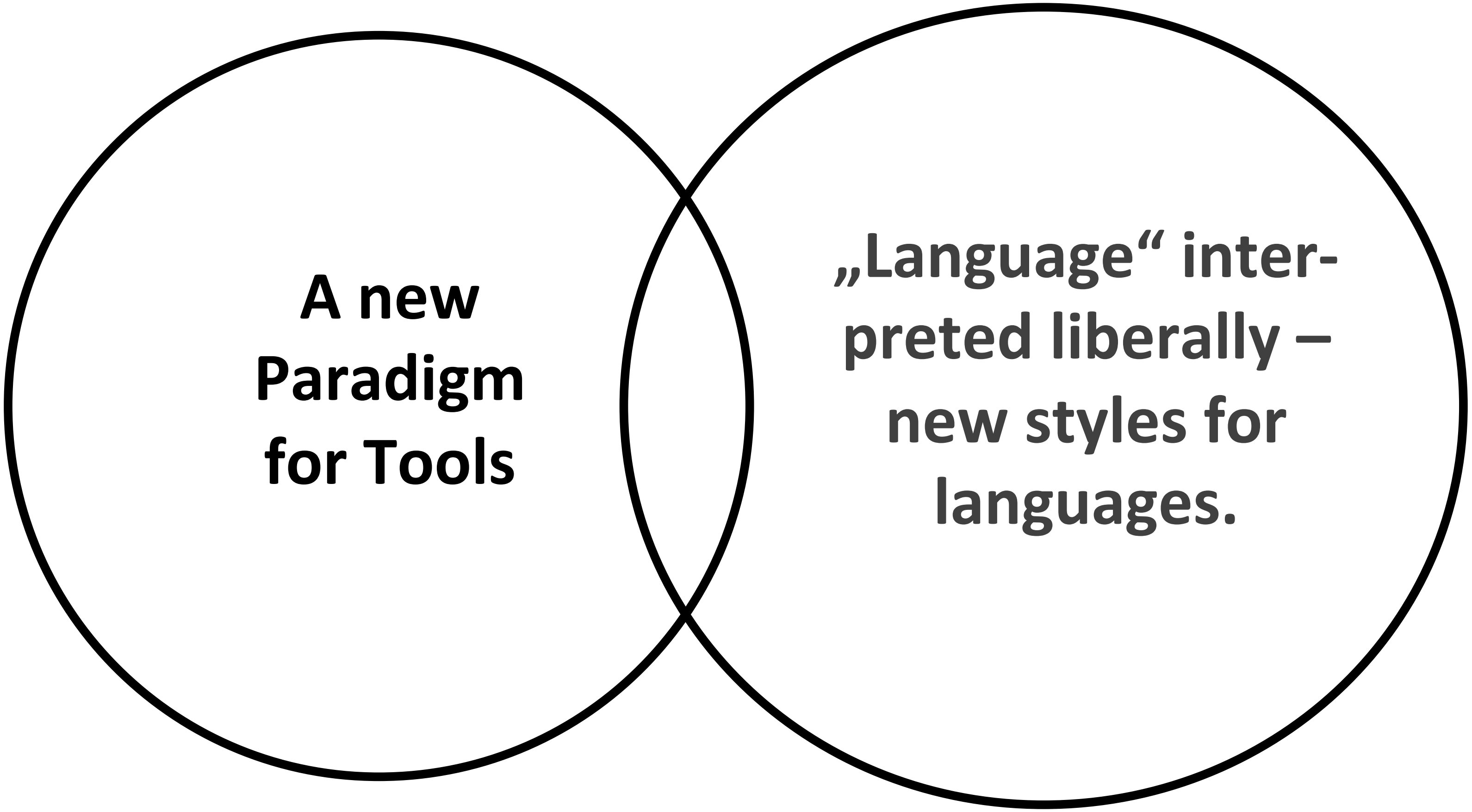


author
analyze
compose
execute

Language Engineering

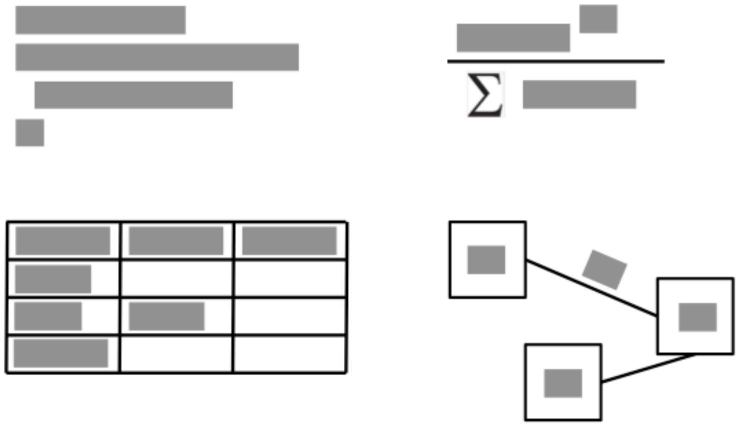
Language Workbenches

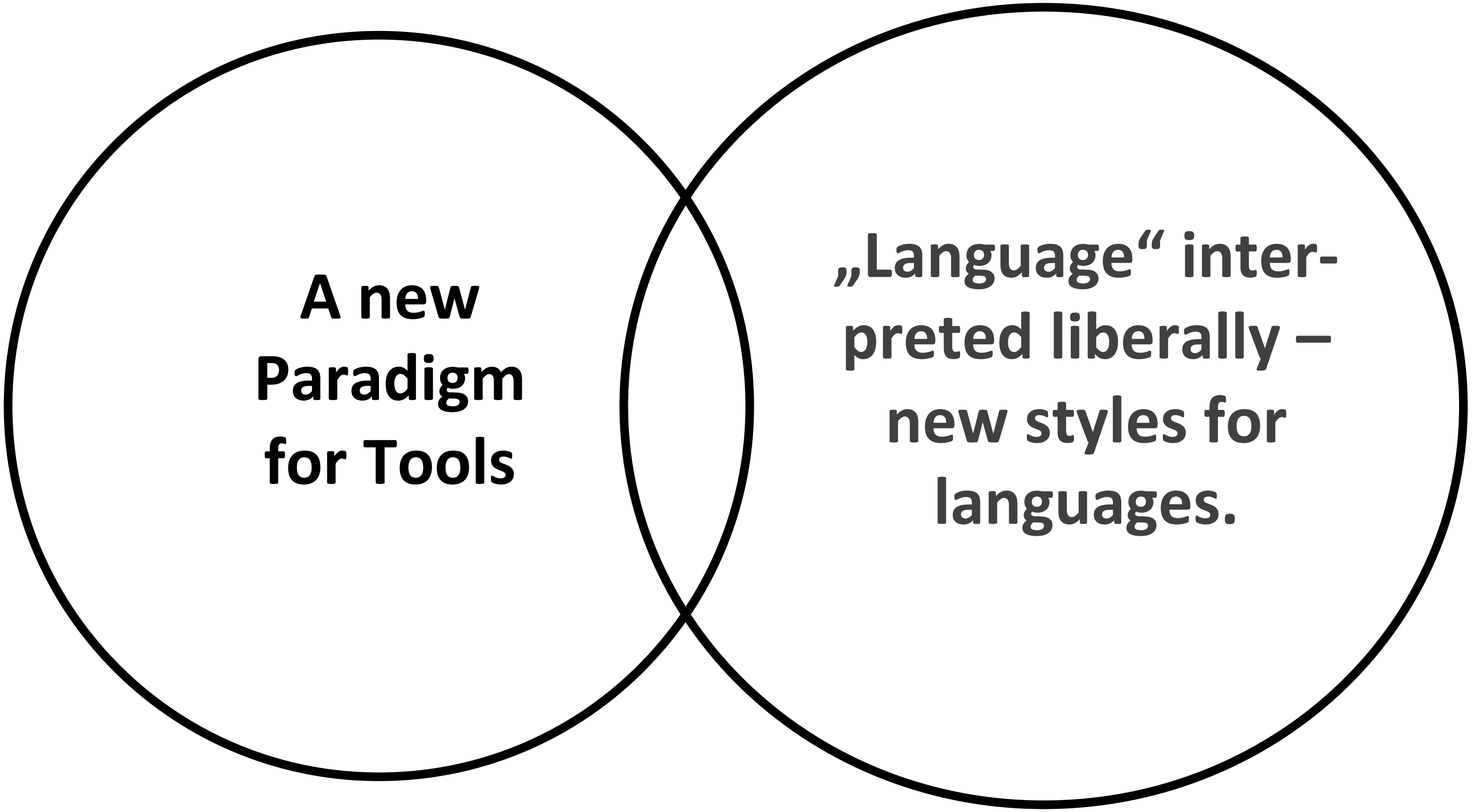
„Generic Tools“



A new Paradigm for Tools

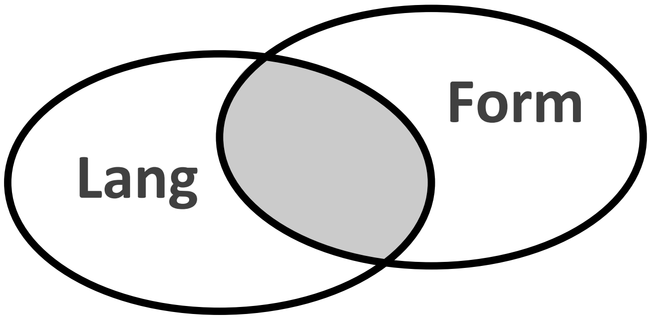
„Language“ interpreted liberally – new styles for languages.





A new Paradigm for Tools

„Language“ interpreted liberally – new styles for languages.



Lang

Form

Language

Expressions
„Code“
Code Completion
Error Highlighting
Version Control
Refactoring
Debugging

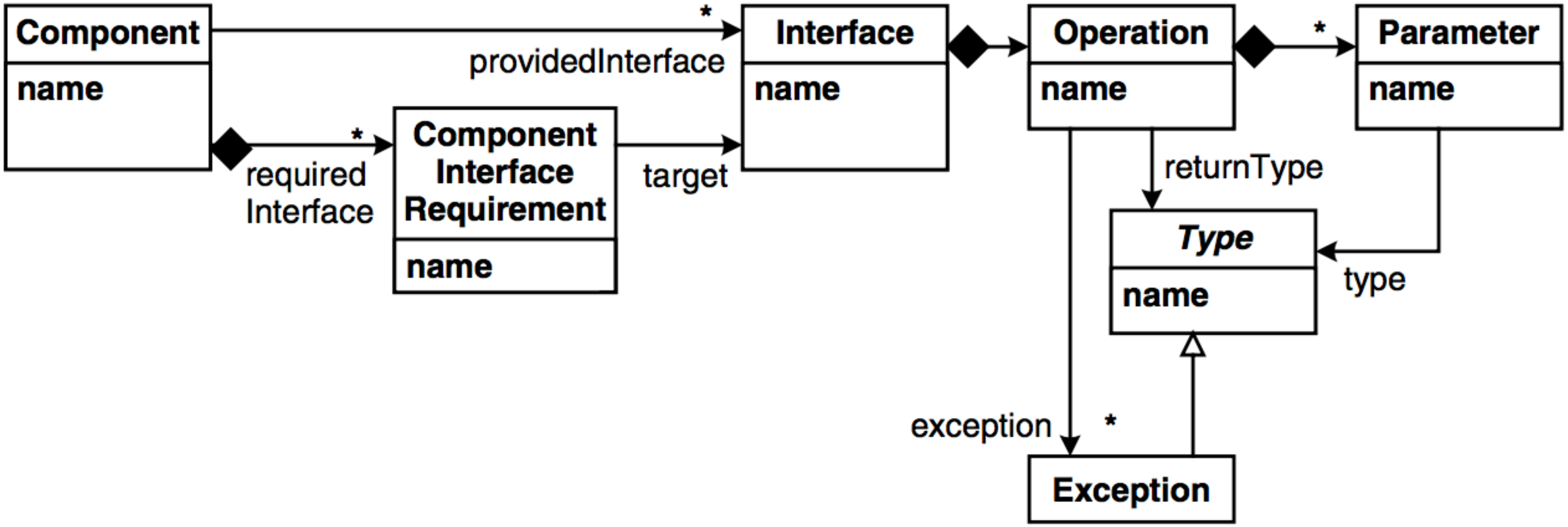
Form

Helper Buttons
Tables
Rigid Structures
Tree Views
Visualizations
Live Interpretation
Math Notation
Graphical
Prose + Code

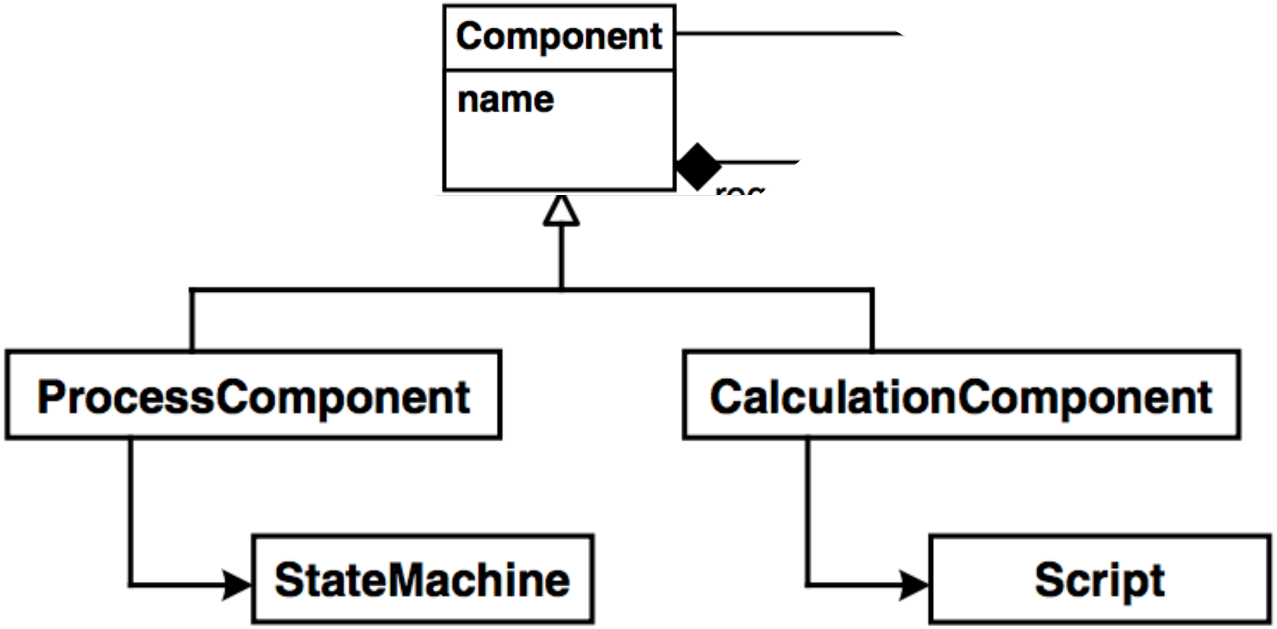
Generic Tools

Specific Languages

[Component Implementation]



[Component Implementation]



1. Structural Element
2. Classification
3. Behavior Specification Formalism

10

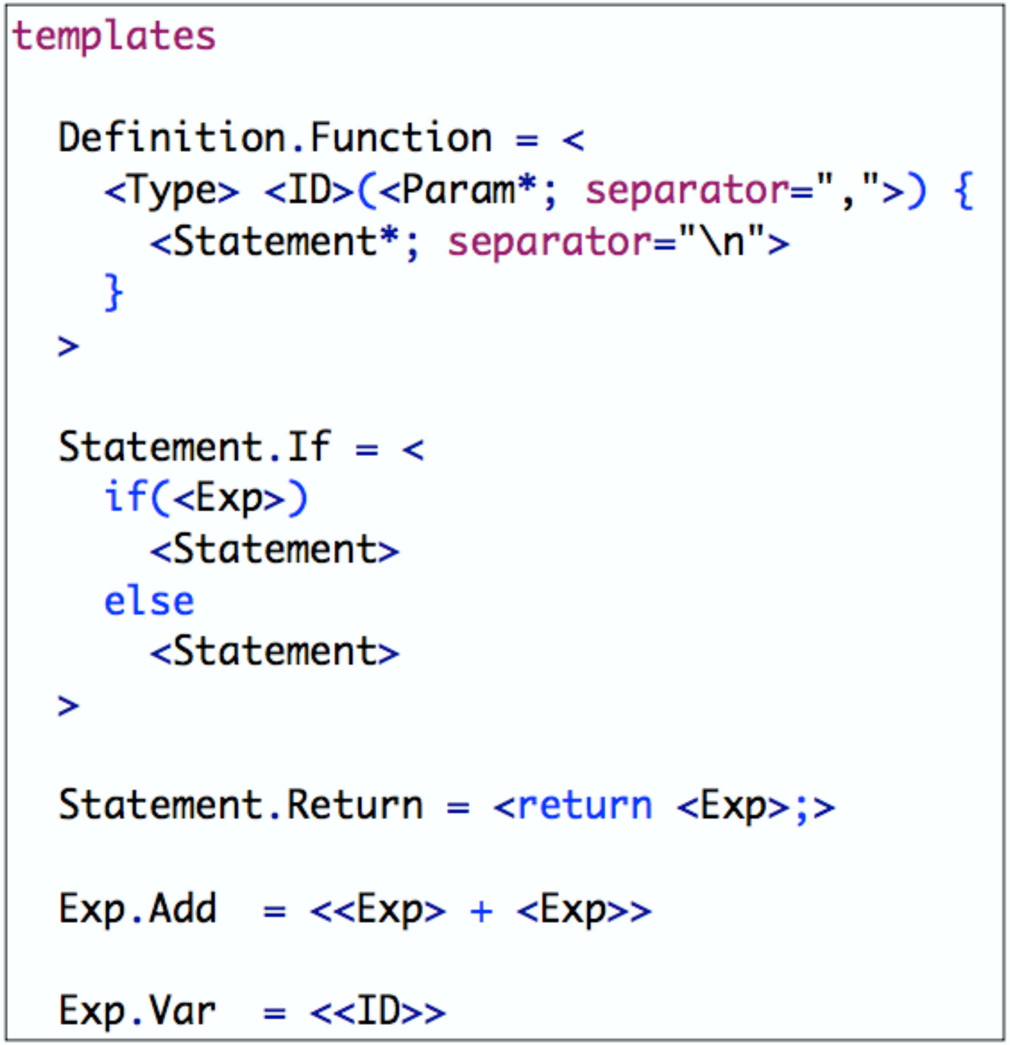
Outlook

A few more editing improvments in MPS.

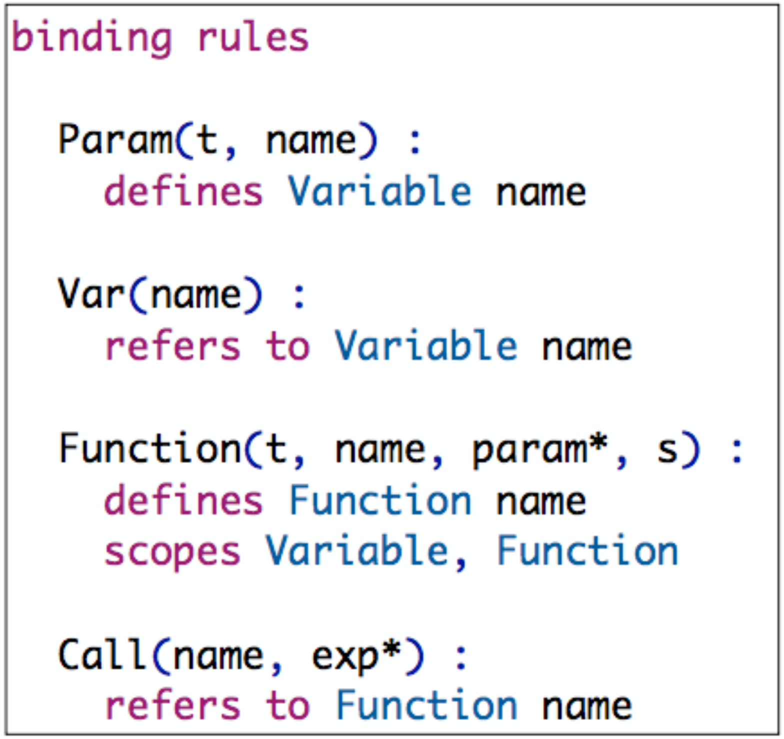
More declarative languages to specify languages.

http://eelcovisser.org/wiki/projects/ldwb

Eelco‘s Language Designer‘s Workbench



```
templates

  Definition.Function = <
    <Type> <ID>(<Param*; separator=",">) {
      <Statement*; separator="\n">
    }
  >

  Statement.If = <
    if(<Exp>)
      <Statement>
    else
      <Statement>
  >

  Statement.Return = <return <Exp>;>

  Exp.Add  = <<Exp> + <Exp>>

  Exp.Var  = <<ID>>
```



```
binding rules

  Param(t, name) :
    defines Variable name

  Var(name) :
    refers to Variable name

  Function(t, name, param*, s) :
    defines Function name
    scopes Variable, Function

  Call(name, exp*) :
    refers to Function name
```

More Competiton.
More good LWBs.

http://languageworkbenches.net

Thank you!

voelter@acm.org
www.voelter.de
@markusvoelter