LANGUAGE OF LANGUAGES

www.languageoflanguages.org
LANGUAGE OF LANGUAGES

www.languageoflanguages.org
Fortran
Lisp

Fortran
LANGUAGES TYPICALLY USED
LANGUAGES TYPICALLY USED
LANGUAGES TYPICALLY USED

Models
- Matlab
- Modelica
- Simulink
- ...

Tuesday, October 25, 2011
LANGUAGES TYPICALLY USED

Models
- Matlab
- Modelica
- Simulink
- ...

Architecture
- AADL
- SysML
- UML
- ...

Tuesday, October 25, 2011
LANGUAGES TYPICALLY USED

Models
- Matlab
- Modelica
- Simulink
- ...

Architecture
- AADL
- SysML
- UML
- ...

Software
- C/C++
- Fortran
- Java
- ...

Tuesday, October 25, 2011
LANGUAGES TYPICALLY USED

Models

Matlab  Modelica
Simulink  ...
SPECIALIZED DOMAINS

Models

Matlab  Modelica
Simulink  ...

Tuesday, October 25, 2011
SPECIALIZED DOMAINS

Models

Matlab  Modelica
Simulink  ...

Tuesday, October 25, 2011
SPECIALIZED DOMAINS

Models

Matlab  Modelica
Simulink  ...

Electrical Subsystem
SPECIALIZED DOMAINS

Models

Matlab
Modelica
Simulink
...
SPECIALIZED DOMAINS

Models

- Matlab
- Modelica
- Simulink
- ...
SPECIALIZED DOMAINS

Models

- Matlab
- Modelica
- Simulink
- ...
EXAMPLE MODEL
EXAMPLE MODEL

**IDLE**
- unlockDoor
- lockPanel

**ACTIVE**
- doorClosed

**WAITINGFOR DRAWER**
- lightOn
- drawerOpened

**WAITINGFOR LIGHT**
- lightOn

**UNLOCKEDPANEL**
- unlockDoor
- lockPanel

**PanelClosed**
EXAMPLE MODEL

- IDLE
  - unlockDoor
  - lockPanel
- ACTIVE
  - WAITINGFOR DRAWER
  - WAITINGFOR LIGHT
- UNLOCKED PANEL
  - unlockDoor
  - lockPanel

Transitions:
- doorClosed
- lightOn
- drawerOpened
- panelClosed

Node States:
- IDLE
- ACTIVE
- WAITINGFOR DRAWER
- WAITINGFOR LIGHT
- UNLOCKED PANEL
FOR THE PROGRAMMER
FOR THE PROGRAMMER

**Diagram:**

- **IDLE**
  - unlockDoor
  - lockPanel
  - doorClosed

- **ACTIVE**
  - lightOn
  - drawerOpened
  - panelClosed

- **WAITINGFOR DRAWER**
  - drawerOpened

- **WAITINGFOR LIGHT**
  - lightOn

- **UNLOCKEDPANEL**
  - unlockDoor
  - lockPanel
State idle = new State("idle");
State activeState = new State("active");
State waitingForLightState = new State("waitingForLight");
State waitingForDrawerState = new State("waitingForDrawer");
State unlockedPanelState = new State("unlockedPanel");
State idle = new State("idle");
State activeState = new State("active");
State waitingForLightState = new State("waitingForLight");
State waitingForDrawerState = new State("waitingForDrawer");
State unlockedPanelState = new State("unlockedPanel");
State idle = new State("idle");

State activeState = new State("active");

State waitingForLightState = new State("waitingForLight");

State waitingForDrawerState = new State("waitingForDrawer");

State unlockedPanelState = new State("unlockedPanel");
State idle = new State("idle");

State activeState = new State("active");

State waitingForLightState = new State("waitingForLight");

State waitingForDrawerState = new State("waitingForDrawer");

State unlockedPanelState = new State("unlockedPanel");

events
  doorClosed
drawerOpened
  ...
end

commands
  unlockPanel
lockPanel
  ...
end

state idle
  actions {unlockDoor lockPanel}
doorClosed => active
end
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");

State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");

events
doorClosed
drawerOpened

end

commands
unlockPanel
lockPanel

... end

state idle
actions {unlockDoor lockPanel}
doorClosed => active

end
State idle = new State("idle");

State activeState = new State("active");

State waitingForLightState = new State("waitingForLight");

State waitingForDrawerState = new State("waitingForDrawer");

State unlockedPanelState = new State("unlockedPanel");

events
  doorClosed
drawerOpened
  ...
end

commands
  unlockPanel
  lockPanel
  ...
end

state idle
  actions {unlockDoor lockPanel}
doorClosed => active
end
FOR THE SYSTEMS ANALYST

State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");

State events
    doorClosed
drawerOpened

end

State commands
    unlockPanel
    lockPanel

end

State state idle
    actions {unlockDoor lockPanel}
    doorClosed => active
end
State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");

events
doorClosed
drawerOpened
end

commands
unlockPanel
lockPanel
...end

state idle
actions {unlockDoor lockPanel}
doorClosed => active
end
State idle = new State("idle");

State activeState = new State("active");

State waitingForLightState = new State("waitingForLight");

State waitingForDrawerState = new State("waitingForDrawer");

State unlockedPanelState = new State("unlockedPanel");

State idle =
new State("idle");

State activeState =
new State("active");

State waitingForLightState =
new State("waitingForLight");

State waitingForDrawerState =
new State("waitingForDrawer");

State unlockedPanelState =
new State("unlockedPanel");

events
doorClosed
drawerOpened
end
commands unlockPanel lockPanel
end

state idle
actions {unlockDoor lockPanel}
doorClosed => active
end
State idle = new State("idle");
State activeState = new State("active");
State waitingForLightState = new State("waitingForLight");
State waitingForDrawerState = new State("waitingForDrawer");
State unlockedPanelState = new State("unlockedPanel");

events
doorClosed
drawerOpened
end
commands
unlockPanel
lockPanel
end
state idle
actions {unlockDoor lockPanel}
doorClosed => active
end

KEEP IN SYNC
LANGUAGE OF LANGUAGES
LANGUAGE OF LANGUAGES

Experimental language workbench that embraces the use of multiple notations (textual and graphical)
Experimental language workbench that embraces the use of multiple notations (textual and graphical) for flexible development.
LANGUAGE OF LANGUAGES
LANGUAGE OF LANGUAGES

Language Workbench: IDE for convenient language experimentation (creating, editing, translating)
LANGUAGE OF LANGUAGES

Language Workbench: IDE for convenient language experimentation (creating, editing, translating)
ROADMAP
ROADMAP

• Support a mix of textual and graphical languages.
• Support parsing as well as projecting
• Minimal paradigm with Language Elements, Concepts, Language Definitions.
• Support for outline, syntax coloring, code completion, etc
• Support for language debugging
• Web based app (like lively kernel)
• Community repository of Concepts for plug-n-play
• Fine-grained version control based on concepts
• Meta-circularity: LoLs is implemented in LoLs
ROADMAP

- Support a mix of textual and graphical languages.
- Support parsing as well as projecting
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for plug-n-play
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs
ROADMAP

- Support a mix of textual and graphical languages.
- Support parsing as well as projecting
- Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for plug-n-play
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs
ROADMAP

- Support a mix of textual and graphical languages.
- Support parsing as well as projecting
  - Support for outline, syntax coloring, code completion, etc
- Support for language debugging
- Web based app (like lively kernel)
- Community repository of Concepts for plug-n-play
- Fine-grained version control based on concepts
- Meta-circularity: LoLs is implemented in LoLs
ROADMAP

• Support a mix of textual and graphical languages.
• Support parsing as well as projecting
• Minimal paradigm with Language Elements, Concepts, Language Definitions.
  • Support for outline, syntax coloring, code completion, etc
• Support for language debugging
• Web based app (like lively kernel)
• Community repository of Concepts for plug-n-play
• Fine-grained version control based on concepts
• Meta-circularity: LoLs is implemented in LoLs
SMALL TASTE
SMALL TASTE
SMALL TASTE

• Calculation
SMALL TASTE

• Calculation
• Western Math
SMALL TASTE

- Calculation
- Western Math
- Roman Numerals
SMALL TASTE

- Calculation
- Western Math
- Roman Numerals
- Stack Machine
Concepts
CALCULATOR EXAMPLE

3 + 4
CALCULATOR EXAMPLE

1 Numbers

3 + 4
CALCULATOR EXAMPLE

① Numbers
② Addition

3 + 4
3 + 4
3 + 4
Parse!
TREE REPRESENTATION

Language Element Tree (LET)
TREE REPRESENTATION

Language Element Tree (LET)
ATTACH CONCEPTS
ATTACH DEFINITIONS
SWITCH DEFINITIONS
SWITCH DEFINITIONS
SWITCH DEFINITIONS
Addition Concept

Definitions

Western Math

Calculate

...

Number Concept

Definitions

Western Math

Roman Numerals

...

LANGUAGE DEFINITIONS
LANGUAGE DEFINITIONS
Parsing

Context Checking

Projecting

3+4
Addition Concept
Definitions
- Western Math
- Calculate
- ...

Number Concept
Definitions
- Western Math
- Roman Numerals
- ...

Parse
Context Checking
Projecting

3 + 4 →

LANGUAGE DEFINITIONS
Addition ← term:t ‘+’ factor:f \{t,f\}

3+4 →
Addition ← term:t ‘+’ factor:f {t,f}

^self an: [t := self apply: #term]
n: [self a: $+]
n: [f := self apply: #factor]
n: [self le: {t, f}]

3 + 4 →
**LANGUAGE DEFINITIONS**

**Addition Concept**
- Western Math
- Calculate
- ...

**Number Concept**
- Western Math
- Roman Numerals
- ...

**Steps**
1. Parsing
2. Context Checking
3. Projecting
LANGUAGE DEFINITIONS
LANGUAGE DEFINITIONS

Parsings
2 Context Checking
3 Projecting

Which definition to use?
LANGUAGE DEFINITIONS

Parsing

Context Checking

Projecting

Addition Concept

Definitions

Western Math

Calculate

... 

Number Concept

Definitions

Western Math

Roman Numerals

... 

^self language = 'Western Math'

Which definition to use?
Addition Concept

Definitions

Western Math  Calculate  ...

Number Concept

Definitions

Western Math  Roman Numerals  ...

Language Definitions

^self language = 'Western Math'
^self child1 value = 1

Which definition to use?
Addition Concept

Definitions

Western Math

Calculate

...
LANGUAGE DEFINITIONS
3 + 4

LANGUAGE DEFINITIONS
Addition Concept

Definitions

- Western Math
- Calculate
- ...

Number Concept

Definitions

- Western Math
- Roman Numerals
- ...

PROJECTING

Projectional pattern form

^\{\textbf{self} lp1. $+ \text{asText allBold. self lp2}\}

\[ 3 + 4 \]
QUICK TOUR
ACCOMMODATING CHANGE

100
ACCOMMODATING CHANGE
ACCOMMODATING CHANGE

100

Kilometers? Miles?
LANGUAGE OF LANGUAGES
APPROACH

100

Kilometers?

Miles?
LANGUAGE OF LANGUAGES
APPROACH

100

Kilometers? Miles?
LANGUAGE OF LANGUAGES
APPROACH

100

Number
Concept

Kilometers?
Miles?
LANGUAGE OF LANGUAGES
APPROACH

100

Number Concept +

Kilometers?

Miles?
LANGUAGES OF LANGUAGES

APPROACH

100

Number Concept + Units Concept

Kilometers? Miles?
UNITS DEMO
FUTURE WORK
FUTURE WORK

Handle different notations within one domain

Models

Matlab
Modelica
Simulink
...
The team then identified the set of analyses for each of those tiers, propagating and validating requirements and constraints across model levels and across multiple operational dimensions. In addition, the POC project was to demonstrate the feasibility of an architecture-centric model repository supporting the business process of an integrator/supplier interaction.

3.2 The Aircraft System Model

Figure 6 shows the drawing of the aircraft system provided to the POC team. It shows major physical subsystems, some providing aircraft capability, such as navigation or landing gear, and others providing physical resources to the subsystems, such as the electrical power, hydraulics, and fuel.
FUTURE WORK

**Models**
- Matlab
- Modelica
- Simulink
- ...

**Architecture**
- AADL
- SysML
- UML
- ...

**Software**
- C/C++
- Fortran
- Java
- ...

Tuesday, October 25, 2011
The team then identified the set of analyses for each of those tiers, propagating and validating requirements and constraints across model levels and across multiple operational dimensions.

In addition, the POC project was to demonstrate the feasibility of an architecture-centric model repository supporting the business process of an integrator/supplier interaction.

### 3.2 The Aircraft System Model

Figure 6 shows the drawing of the aircraft system provided to the POC team. It shows major physical subsystems, some providing aircraft capability, such as navigation or landing gear, and others providing physical resources to the subsystems, such as the electrical power, hydraulics, and fuel.

**Figure 6**: Aircraft System Drawing

**Figure 7**: AADL Model of Aircraft System
The team then identified the set of analyses for each of those tiers, propagating and validating requirements and constraints across model levels and across multiple operational dimensions. In addition, the POC project was to demonstrate the feasibility of an architecture-centric model repository supporting the business process of a framersupplier interaction.

3.2 The Aircraft System Model

Figure 6 shows the drawing of the aircraft system provided to the POC team. It shows major physical subsystems, some providing aircraft capability, such as navigation or landing gear, and others providing physical resources to the subsystems, such as the electrical power, hydraulics, and fuel.

Figure 6: Aircraft System Drawing

Figure 7: AADL Model of Aircraft System
We have elaborated the flight guidance system (FGS) of this Tier IMA platform. This elaboration is not a separate model, but a refinement of the FGS system model into a Tier 1 model and instantiate both for analysis from a single source.

In the model, we have represented a distributed computer platform (physical view) of the IMA subsystem (logical view) of the IMA computer platform.

AADL buses. Each aircraft subsystem is represented by a separate AADL system type whose characteristics (e.g., mass) are static and controlled in the physical view.

Using the AADL specification, we have captured the requirements of the subsystem. A system is represented by a separate AADL system type, and the embedded application subsystems by separate AADL system types.

The Tier 1 model has been used to produce a Tier 2 model also elaborating the electrical power distribution by a power subsystem that receives its supply from the main power system and provides it to the various computer hardware processing units and communication units. The symmetry reflects the dual redundant nature of the physical system, buses to represent networks such as ARINC429, and systems to represent the physical connection between subsystems and their resources. The bus types and access connections represent the physical connection between subsystems and their resources. The bus types and access connections also have mechanisms. Because of this refinement, we can now specify a Tier 1 variant and a Tier 2 variant of the aircraft that is separate from the physical view in the AADL model. The bus type has a resource capacity property, which is captured in a graphical view separate from the physical view in the AADL model. This elaboration is not a separate model, but a refinement of the Tier 1 model into a Tier 2 model and instantiation of the resource bus features (connection points) have resource supply properties, such as the engine contributing electrical power to the electrical power resource, and resource bus capacity selection.

The Tier 2 model also elaborates the electrical power distribution by a power subsystem that receives its supply from the main power system and provides it to the various computer hardware processing units and communication units. The symmetry reflects the dual redundant nature of the physical system, buses to represent networks such as ARINC429, and systems to represent the physical connection between subsystems and their resources. The bus types and access connections represent the physical connection between subsystems and their resources. The bus types and access connections also have mechanisms. Because of this refinement, we can now specify a Tier 1 variant and a Tier 2 variant of the aircraft that is separate from the physical view in the AADL model. This elaboration is not a separate model, but a refinement of the Tier 1 model into a Tier 2 model and instantiation of the resource bus features (connection points) have resource supply properties, such as the engine contributing electrical power to the electrical power resource, and resource bus capacity selection.

FUTURE WORK

- Matlab
- Modelica
- Fortran
- Java
- UML
- SysML
- AADL
- C/C++
- Java
The Tier 1 IMA platform.

We have elaborated the flight guidance system (FGS) of this Tier 2 model into a Tier 2 variant and a Tier 1 model and instantiated the distributed computer platform, which is captured in a graphical view separate from the physical view in the Tier 2 model also elaborates the electrical power distribution by a power subsystem that rec...
FUTURE WORK

Models
- Matlab
- Modelica
- Simulink
- ...

Architecture
- AADL
- SysML
- UML
- ...

Software
- C/C++
- Fortran
- Java
- ...

Tuesday, October 25, 2011
FUTURE WORK

Models
- Matlab
- Modelica
- Simulink
- ...

Architecture
- AADL
- SysML
- UML
- ...

Software
- C/C++
- Fortran
- Java
- ...

Violation of data stream assumptions
- Stream miss rates, Mismatched data representation, Latency jitter & age
- Partitions as Isolation Regions
  - Space, time, and bandwidth partitioning
  - Isolation not guaranteed due to undocumented resource sharing
    - fault containment, security levels, safety levels, distribution
  - Virtualization of time & resources
    - Logical vs. physical redundancy

Inconsistent System States & Interactions
- Modal systems with modal components
- Concurrency & redundancy management
- Application level interaction protocols

Performance impedance mismatches
- Processor, memory & network resources
- Compositional & replacement performance mismatches
- Unmanaged computer system resources
FUTURE WORK

Systems Architecture Virtual Integration

Models
- Matlab
- Modelica
- Simulink
- ...

Architecture
- AADL
- SysML
- UML
- ...

Software
- C/C++
- Fortran
- Java
- ...

Violation of data stream assumptions
- Stream miss rates, Mismatched data representation, Latency jitter & age
- Partitions as Isolation Regions
- Space, time, and bandwidth partitioning
- Isolation not guaranteed due to undocumented resource sharing
- fault containment, security levels, safety levels, distribution
- Virtualization of time & resources
- Logical vs. physical redundancy
- Time stamping of data & asynchronous systems
- Inconsistent System States & Interactions
- Modal systems with modal components
- Concurrency & redundancy management
- Application level interaction protocols
- Performance impedance mismatches
- Processor, memory & network resources
- Compositional & replacement performance mismatches
- Unmanaged computer system resources

© Texas Engineering Experiment Station
FAA SW & AEH Conference | San Jose, CA
Tuesday, October 25, 2011
STATE MACHINE EXAMPLE
STATE MACHINE EXAMPLE

- **IDLE**
  - Unlock Door
  - Lock Panel

- **ACTIVE**
  - Waiting For Door
    - Door Closed
  - Waiting For Light
    - Light On
    - Drawer Opened

- **WAITING FOR DRAWER**
  - Drawer Opened

- **UNLOCKED PANEL**
  - Unlock Door
  - Lock Panel
  - Light On
DECOMPOSE INTO CONCEPTS

State
- IDLE
- ACTIVE
- WAITINGFOR DRAWER
- WAITINGFOR LIGHT
- UNLOCKEDPANEL
DECOMPOSE INTO CONCEPTS

**State**

**Actions**
STATEMENTS INTO CONCEPTS

1. State
2. Actions
3. Transitions

- IDLE
  - unlockDoor
  - lockPanel
- ACTIVE
  - doorClosed
- WAITINGFOR DRAWER
  - lightOn
  - drawerOpened
- WAITINGFOR LIGHT
  - drawerOpened
  - lightOn
- UNLOCKED PANEL
  - unlockDoor
  - lockPanel
UNIFY CONCEPTS
UNIFY CONCEPTS

- IDLE
  - unlockDoor
  - lockPanel
- ACTIVE
- WAITINGFOR DRAWER
  - drawerOpened
  - lightOn
- WAITINGFOR LIGHT
  - drawerOpened
  - lightOn
- UNLOCKEDPANEL
  - unlockDoor
  - lockPanel
- doorClosed
- panelClosed
UNIFY CONCEPTS
CONCLUSIONS
CONCLUSIONS

• Language of Languages: An experimental language workbench that unifies concepts expressed across different notations
CONCLUSIONS

• Language of Languages: An experimental language workbench that unifies concepts expressed across different notations

• Implemented using ideas of Language Elements, Concepts and Language Definitions
CONCLUSIONS

• Language of Languages: An experimental language workbench that unifies concepts expressed across different notations

• Implemented using ideas of Language Elements, Concepts and Language Definitions

• Available now from www.languageoflanguages.org
CONCLUSIONS

• Language of Languages: An experimental language workbench that unifies concepts expressed across different notations

• Implemented using ideas of Language Elements, Concepts and Language Definitions

• Available now from www.languageoflanguages.org

• Contact Jamie Douglass (jamie.douglass@boeing.com) or Nicholas Chen (nchen@illinois.edu)