Language Design
Cross-fertilizing with other areas

Charles Consel
Phoenix Research Group
Inria / University of Bordeaux
phoenix.inria.fr
Language Design: Central Activity

Usage Areas
- Telephony
- Game Design
- Home Automation

Programming Language related
- Language Definition
- Types
- Program Analysis
- Program Transformation

Areas in Core Computer Science
- Operating System
- Networking
- Software Engineering
Collaborating with Areas in Core Computer Science

- Domain-Specific Languages
  - Domain analysis
  - DS properties
  - DS abstractions
  - Program analysis
  - Compilation
  - Examples: Networking, Operating systems, Multimedia, etc.
Collaborating with Areas in Core Computer Science

• Benefits
  • Interacting with researchers from other CS communities
  • Building a complementary expertise (more job opportunities)
  • Strengthening research hypotheses
  • Validating research results
  • Widening impact of research results
  • Opening new research avenues
  • Finding new research problems
  • Creating more funding opportunities (government, industry)
  • Attracting students
Collaborating with Usage Areas

• User programming
  • Domain analysis
  • Domain process
  • Language engineering
  • Evaluation: usability, productivity, verifiability, etc.
• Tools
• Examples: Telephony, Assisted living, etc.
Collaborating with Usage Areas

- Benefits
  - Interacting with non-computer scientists (multidisciplinary)
  - Widening research hypotheses
  - Widening research avenues
  - Finding new research problems
  - Creating more funding opportunities (government, industry)
  - Attracting students
The DiaSuite Project *(ongoing)*
diasuite.inria.fr

---

**Usage Areas**
- Avionics
- Assisted Living
- Home Automation

**Programming Language related**
- DS Language design
- Verification
- Generative Programming
- Runtime

**Areas in Core Computer Science**
- Software Engineering
- Networking
- Security
The DiaSuite Project

A design language to specify a software system
A paradigm-specific language
A compiler to process a specification to
• guide the implementation
• verify safety properties
• ensure conformance

Paradigm: Sense/Compute/Control software systems
Sense/Compute/Control (SCC) Software System
Sense/Compute/Control (SCC)
Software System

- Sense
- Compute
- Control

GPS, flight plan → Sense → Compute → Control
aileron, engine
The SCC Architectural Style

Control

orders

actuators
actions

Environment

Sense

raw data

sources
sensors

compute

refined information

context
operators

time

control
operators
Compiling a Design

Design
Language

abstract

Design

concrete

Design
Compiling a Design

Design Language

abstract

defeats guidance and verification

Concrete

Compiler

generated

Software system

Design

Design

Language Design

Charles Consel - Phoenix Research Group - Language Design

January 2012
Compiling a Design

- **Design Language**
  - abstract
  - concrete

- Design
  - mixes design and implementation
  - defeats guidance and verification

- Compiler
  - generated
  - Software system

- January 2012
- Charles Consel - Phoenix Research Group - Language Design
Our Approach
Our Approach
Our Approach

Design

Compiler

verification by construction

Developer’s code

Programming framework

January 2012
Charles Consel - Phoenix Research Group - Language Design
Flight Guidance System

DiaSpec – Taxonomy

```plaintext
action Control {
    incline(targetRoll as Float);
}

device Aileron {
    action Control;
}

device NavigationMMI {
    source targetHeading as Float;
    action DisableMode;
}

device IRU {
    source heading as Float;
    source position as Coordinates;
    action Deactivate;
}
```
Aileron

Aileron Controller

Target Roll

Intermediate Heading

Navigation MMI

IRU

controller AileronController {
  context TargetRoll;
  action Control on Aileron;
}

class TargetRoll as Float {
  source roll from IRU;
  context IntermediateHeading;
}

class IntermediateHeading as Float {
  source heading from IRU;
  source targetHeading from NavigationMMI;
}
controller AileronController {
  context TargetRoll;
  action Control on Aileron;
  interaction {
    when provided TargetRoll;
    do Control on Aileron;
  }
}

context TargetRoll as Float {
  source roll from IRU;
  context IntermediateHeading;
  interaction {
    when provided IntermediateHeading;
    get roll from IRU;
    always publish;
  }
}

context IntermediateHeading as Float {
  source heading from IRU;
  source targetHeading from NavigationMMI;
  interaction {
    when provided heading from IRU;
    get targetHeading from NavigationMMI;
    always publish;
  }
}
Flight Guidance System

DiaSpec – Non-Functional Specification
[OOPSLA’11, FASE’11]

```plaintext
action Control {
    incline(targetRoll as Float);
}

device Aileron {
    action Control;
}

device NavigationMMI {
    source targetHeading as Float;
    action DisableMode;
}

device IRU {
    source heading as Float [frequency 200ms];
    source position as Coordinates;
    action Deactivate;
    raises FailureException;
}
```
Flight Guidance System

DiaSpec – Non-Functional Specification
[OOPSLA’11, FASE’11]

context TargetRoll as Float {
    source roll from IRU;
    context IntermediateHeading;
    interaction {
        when provided IntermediateHeading;
        get roll from IRU in 100ms
        [skipped catch];
        always publish;
    }
}

class IntermediateHeading as Float {
    source heading from IRU;
    source targetHeading from NavigationMMI;
    interaction {
        when provided heading from IRU;
        get targetHeading from NavigationMMI
        in 100ms [mandatory catch];
        always publish;
    }
}
Flight Guidance System

DiaSpec – Generative Programming [GPCE’09]

context IntermediateHeading as Float {
  source heading from IRU;
  source targetHeading from NavigationMMI;
  interaction {
    when provided heading from IRU;
    get targetHeading from NavigationMMI;
    always publish;
  }
}

public abstract class IntermediateHeading extends AbstractIntermediateHeading {
  private PIDController controller;

  @Override
  public Float onHeadingFromIRU(Float heading, Binding binding, GetContext getContext) {
    NavigationMMI mmi = binding.navigationMMI();
    Float targetHeading = mmi.getTargetHeading();
    return controller.update(Config.PERIOD, targetHeading, heading.value(), 0);
  }
}

Software Designer

Generation

Implementation

Application Developer
The DiaSuite Project

• Funding
  • Avionics
  • Home automation
  • Assisted living

• Collaborations
  • Avionics engineers
  • Researchers in Cognitive Sc.
  • Caregivers

• Research problems
  • Software design
  • Software reliability
  • Quality of service, …

• New directions
  • Security
  • User interaction
  • Certification, …