A Formal Model for the SCJ Level 2 Paradigm

Doctoral Symposium FM2015

Matthew Luckcuck

Supervisors:
Ana Cavalcanti and Andy Wellings

University of York

22nd June 2015

Outline

Outline

- Introduction
  - Problem
  - *Circus*
  - Safety-Critical Java Level 2
- Developing the Model
- Contributions
- Summary and Further Work

Introduction

Open Problem

- Safety-Critical Java (SCJ)...
  - Java profile
  - Interesting paradigm for high integrity programming
  - Three compliance levels: 0, 1, and 2
- SCJ does not directly address verification techniques
  - Existing results address verification for Levels 0 and 1
- SCJ Level 2 presents a bigger challenge

Introduction

Thesis Hypothesis

The paradigm embedded in SCJ Level 2 can be formally modelled using a language that captures state and behaviour, to show that neither the SCJ infrastructure nor a given SCJ program present undesirable program states such as deadlock, divergence, or nondeterminism.

Introduction

Aims

- Produce a model of the SCJ Level 2 paradigm in *Circus*
  - ✓
- Devise a formal translation strategy to convert SCJ Level 2 programs to this model
  - ...
- Show a model of a program can be used for proof of program properties
  - ...

Introduction

SCJ Level 2

- Little attention from academia or industry so far...
  - We examine the uses of Level 2's features and example applications[a]
  - Icelab HVM supports SCJ and provides an SDK
    - Now supports Level 2
- SCJ Standard does not cover verification...

[a] Luckcuck, Wellings, and Cavalcanti, "Safety-Critical Java Level 2: Motivations, Example Applications and Issues".

Introduction

SCJ Level 2

- Verification has been addressed...
  - SCJChecker[a]
  - SafeJML[b]
  - Refinement strategy using *Circus*[c]
- ...but not specifically for Level 2

---

[a] Tang, Plsek, and Vitek, "Static Checking of Safety Critical Java Annotations".

[b] Haddad, Hussain, and Leavens, "The Design of SafeJML, a Specification Language for SCJ with Support for WCET Specification".

[c] Cavalcanti et al., "Safety-Critical Java in Circus".

Circus Family

Why Use *Circus*?

- Previous work on *Circus* with Java and SCJ...
  - Existing model of SCJ Level 1[a]
- Refinement-based development
  - Refinement strategy...

[a] Zeyda et al., "Circus Models for Safety-Critical Java Programs".

Circus Family

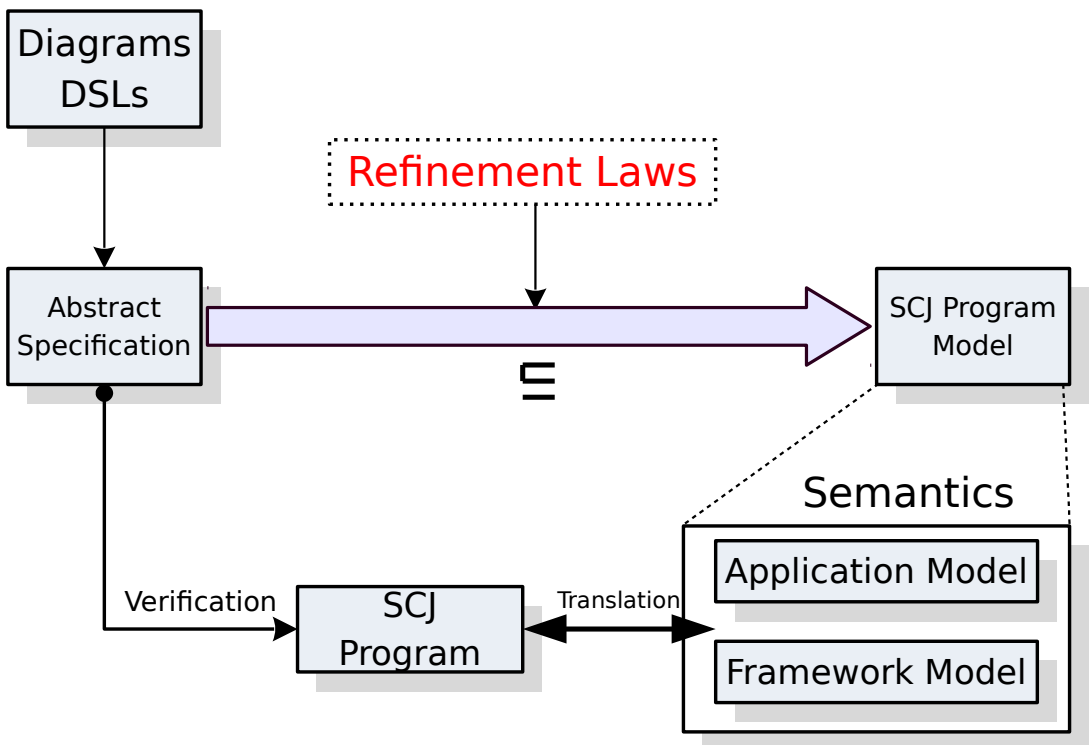
Circus Language

- Combination of **Z** and **CSP**
  - Captures both State and Behaviour
- Contains a notion of refinement
- Organised around Processes...
  - State component (**Z**) to hold variables
  - Actions (**Z** and **CSP**) to perform behaviours

Circus Variants

- *OhCircus*...
  - Classes based on Java's implementation of Object-Orientation
  - Inheritance
- *Circus* Time...
  - Notion of (relative) time

Circus Refinement Strategy



Safety-Critical Java

SCJ...

- Restriction of the Real-Time Specification for Java (RTSJ)
- Provides a Java-based language for systems that need to be certified...
  - Avionics: ED-12C/DO-178C at Level A
  - Failure would prevent continuous safe flight and landing
- Restricted programming and execution model

Centred Around Missions

- Activated in sequence by a Mission Sequencer
- Aim to perform a particular function
- Manage a set of real-time tasks...
  - Embodied in SCJ by Schedulable Objects

SCJ Compliance Levels

Compliance Levels

- SCJ is organised into three compliance levels
- Intends to ease certification efforts
- Each level has a set of unique features plus those from the level(s) below it
- Ascending complexity...
  - Level 0: Cyclic Programs
  - Level 1: Concurrent Tasks
  - Level 2: Concurrent Missions

SCJ Compliance Levels

Compliance Level 2

- Least restricted compliance level
- Complex structures due to concurrent missions. . .
  - Tasks from any active mission may preempt
- May use all four SCJ execution patterns: periodic, aperiodic, run-once after a time offset, and run-to-completion.
- Access to Java suspension
  - `wait()` and `notify()`

SCJ Level 2

Aircraft Example

- Three modes: Take Off, Cruise, Land
  - Each has its own specific Schedulable Objects
- There are also Schedulable Objects which run throughout all the modes...
  - Handling the controls
  - Monitoring the cabin pressure, fuel, etc.
- Adapted from an example in our paper

SCJ Level 2

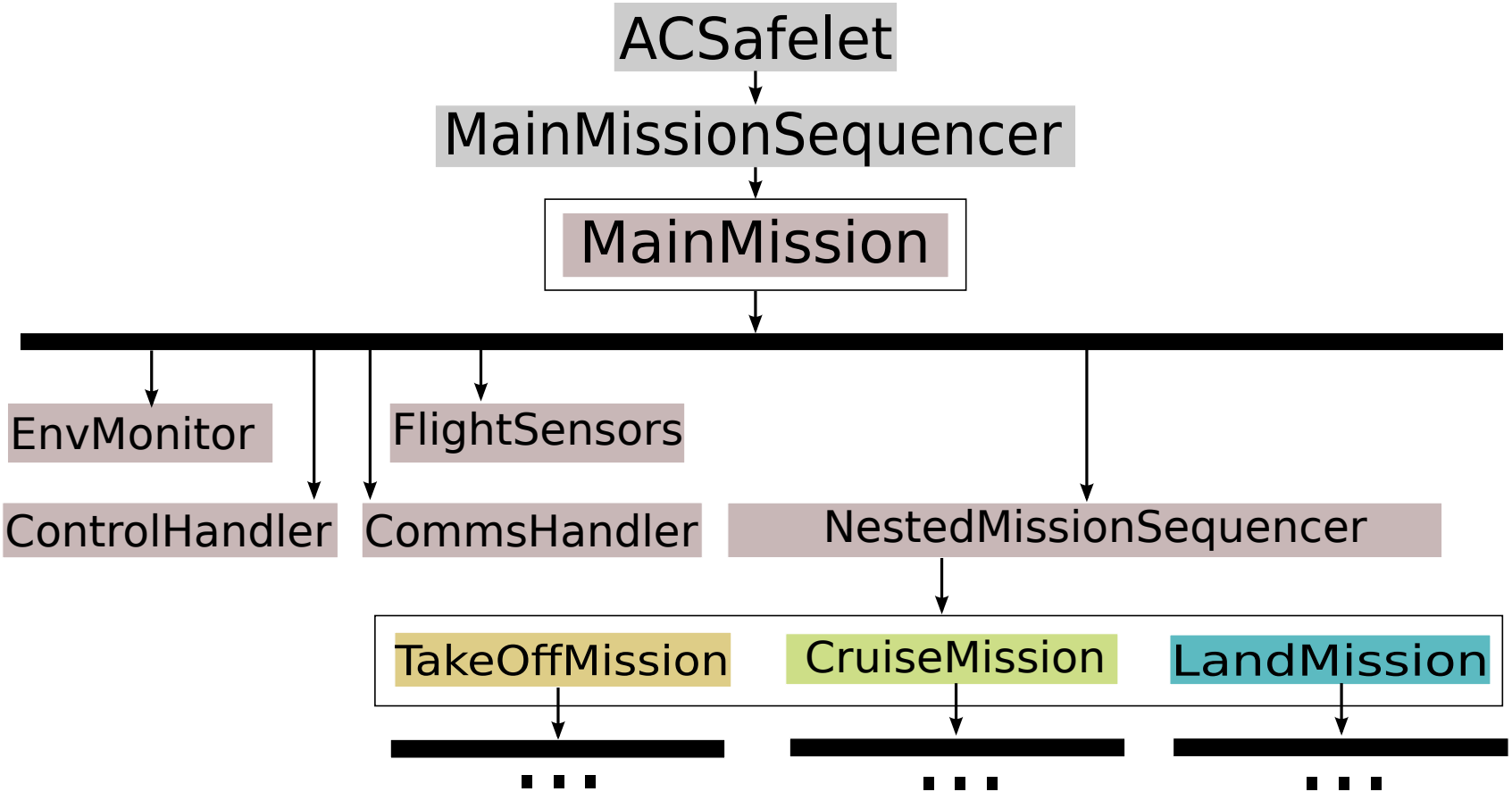


Figure 2 : Simplified Object Diagram of the Aircraft example application

Developing the Model

Modelling Approach

- Captures SCJ program as...
  - **Framework**: API behaviour
  - **Application**: application-specific behaviour
- Translation strategy captures the application-specific information
  - Simplifies translation strategy

Developing the Model

$$Framework \llbracket AppSync \rrbracket Application$$

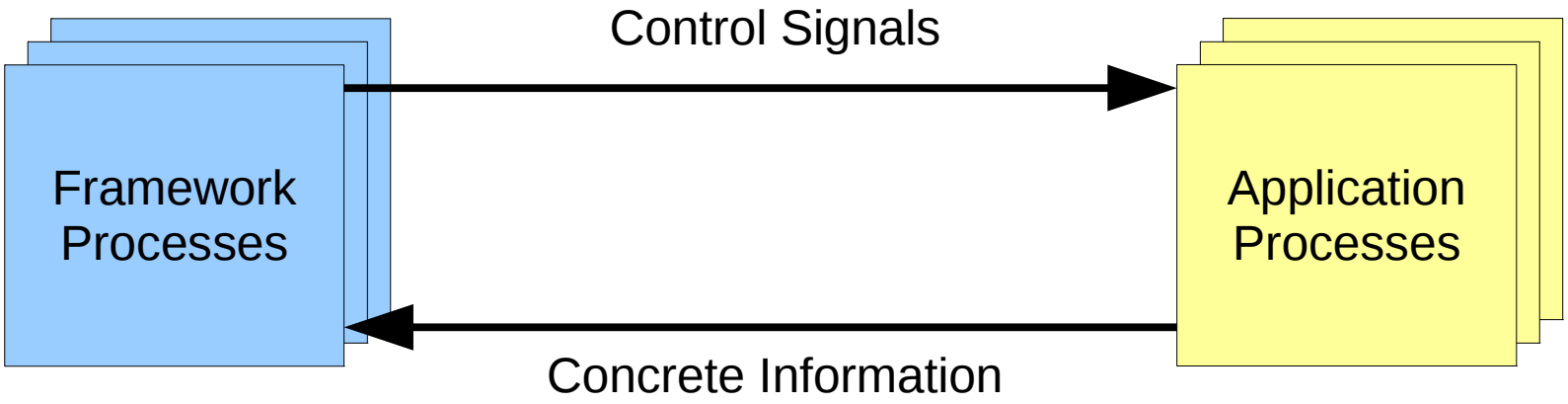


Figure 3 : High Level Model

SCJ Level 2

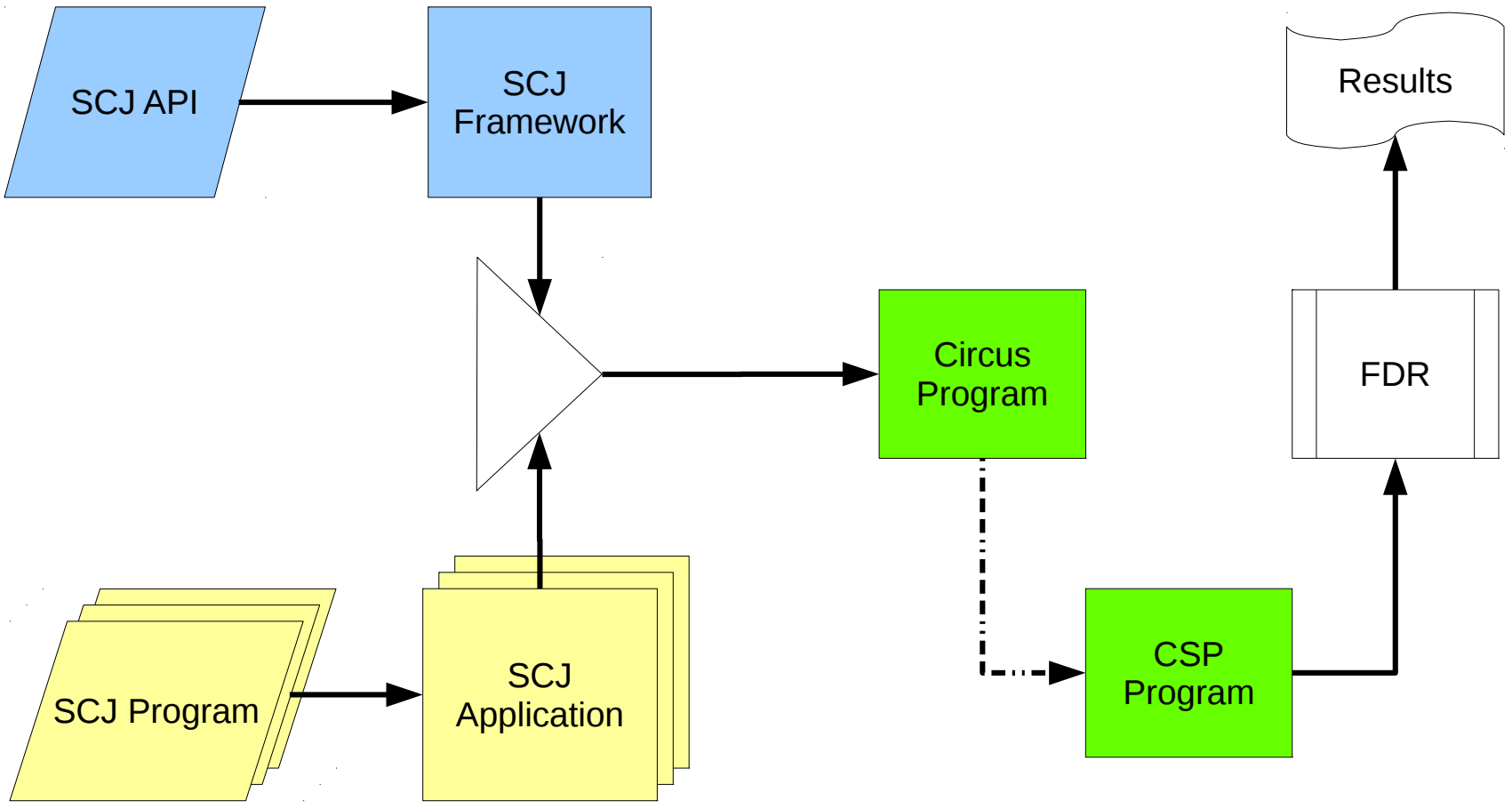


Figure 4 : Solution Flow Chart

Developing the Model

Coverage

- Model Captures...
  - Behaviour and state of objects
  - Suspension
  - Limited treatment of some exceptions
- Exceptions only captured when they indicate a misuse of the SCJ paradigm...
  - Null parameter exceptions not covered, for example
  - Represented by $\mathbf{Chaos}$ in the specification
- Model abstracts away from ...
  - Scheduling
  - Resources (E.g. Memory)

Contributions

Bottom-Up

The model and translation strategy...

1. Verification of SCJ programs
2. Verification of SCJ API
   - Modelled separately

Top-Down

The model in *Circus*...

3. Target for refinement-based development of SCJ Level 2 programs...
   - Refinement from abstract specifications...
   - ... to concrete specifications that capture the SCJ paradigm

Modelling Challenges

SCJ Challenges

- Changing or untested language specification...
  - Though this has stabilised more recently
- Complexity of the unique features of Level 2
- Lack of wide experience of Level 2 ...
  - Only recently acquired a Level 2 implementation

Circus Challenges

- Model checker still in development so convert to CSP...
  - Different feature set to *Circus*
  - Modelling state becomes complicated
    - Large state process to model variables
    - Smart translations needed for efficient implementation in FDR

Summary and Further Work

Summary

- Provided the first examination of Level 2 features and described example applications
- Model SCJ Level 2 paradigm as **Framework** and **Application** combination
- Model of SCJ Level 2 contributes to ...
  - **Bottom-up** development as verification tool
  - **Top-down** development as a refinement target

Further Work

- Devise translation strategy
  - Tool to automate translation
- Translate programs to validate model