Can robots ever be safe?
Software Engineering of Robots

Ana Cavalcanti
Jon Timmis, Jim Woodcock
Wei Li, Alvaro Miyazawa, Pedro Ribeiro
EPSRC RoboCalc
www.cs.york.ac.uk/circus/RoboCalc/

University of York

BCS-FACS Evening Seminar – September 2016
Software Engineering of Robots: why are we interested?

- One of UK eight great technologies: robotics and autonomous systems.
- £13 billion global market predicted for 2025
- Safety: numerous applications of concern
- Autonomous vehicles
- Home automation
- Full verification is beyond the state of the art
- Among other concerns: verification of controller software
Can robots ever be safe?

Software Engineering of Robots

- ASV: Unmanned Marine Systems, Rich Daltry
- Blue Bear Systems, Yoge Patel
- Bristol Robotics Laboratory, Alan Winfield
- Centre for Autonomous Systems Technology, Michael Fisher
- D-RisQ, Nick Tudor
- Flightworks, Matt Pilmoor
- IBM Ireland, Patrick O’Sullivan
- Tekever, Mark Baxter
Outline

- Current approach to development
- What do we want to do?
- How do we want to do?
- RoboChart: core notation
- Semantics
- RoboTool
- Timed RoboChart
- Simulations
- Conclusions
Current approach to development

1st phase: Abstract model

2nd phase: Simulation

3rd phase: Implementation

- state machine
- controller code
- hardware simulation
- discrete environment simulation
- low-level code
- robot
- environment
What do we want to do?

1st phase: Abstract model

- timed state machine

2nd phase: Simulation

- controller code
- hardware simulation
- discrete environment simulation

3rd phase: Implementation

- low-level code
- robot
- environment
What do we want to do?

1st phase: Abstract model
- probabilistic
- timed
- state machine

2nd phase: Simulation
- controller code
- hardware simulation
- discrete environment simulation

3rd phase: Implementation
- low-level code
- robot
- environment
What do we want to do?

1st phase: Abstract model

probabilistic
timed
state machine

2nd phase:
Simulation

code
hardware
discrete
simulation

3rd phase: Implementation

low-level
robot
environment
What do we want to do?

1st phase: Abstract model
- probabilistic
- timed
- state machine
- hardware
- +
- environment
- model

2nd phase: Simulation
- controller
- code
- hardware simulation
- discrete environment simulation

3rd phase: Implementation
- low-level code
+ +
- robot
- +
- environment
What do we want to do?

1st phase: Abstract model
- probabilistic
- timed
- state machine

Hardware + environment model

2nd phase: Simulation
- controller code
- hardware simulation
- discrete environment simulation

3rd phase: Implementation
- low-level code + robot + environment
Can robots ever be safe?

What do we want to do?

1st phase: Abstract model

- probabilistic
- timed
- state machine
- hardware
- environment model

2nd phase: Simulation

- controller code
- hardware simulation
- discrete environment simulation

3rd phase: Implementation

- low-level code
- robot
- environment
How do we want to do it?

RoboChart

UML profile
How do we want to do it?

- RoboChart
- Timed RoboChart
- UML profile
- budgets
- deadlines
How do we want to do it?

RoboChart

Timed RoboChart

Probabilistic RoboChart

UML profile

budgets deadlines

like in UML extension
How do we want to do it?

RoboChart

Timed RoboChart

Probabilistic RoboChart

Timed Probabilistic RoboChart

UML profile

budgets
deadlines

like in UML extension
Can robots ever be safe?

How do we want to do it?

- RoboChart
- Timed RoboChart
- Probabilistic RoboChart
- Timed Probabilistic RoboChart

Hybrid UML profile

Budgets deadlines

Like in UML extension
Behind the scenes: it’s a *Circus*
Can robots ever be safe?

Brought to you by the UTP
RoboChart: why a new notation?

Requirements from robotics

- **Architecture**
  Specific architectural pattern adopted in robotic systems

- **System**
  Clear identification of system

- **API**
  Capture common operations for common functions and kinds of equipment

- **Time and probability**
  Primitives to specify time budgets, deadlines, and probabilities
RoboChart: why a new notation?

Requirements from verification

- **Constraints** Constrained usage to simplify semantics and enable efficient verification
- **Compositional** Encourage component-based modelling to foster compositional reasoning
- **Language** Well defined language constructs with a fixed syntax and semantics
- **Refinement** Refinement-based semantics to support proof of correctness of simulations
Overall behaviour

- Search for chemical spills
- Approach
- Drop flag
- Continue

Video
Module

Identifies a robotic system

- Models a single robot
- One Robotic Platform
- One or more Controllers
- Communication
  - Synchronous
  - Asynchronous
- Robotic Platform may provide shared variables
Can robots ever be safe?

Chemical Detector: Module

- Links controller DetectAndFlagC and LightController to Rover
- Rover records assumptions about the hardware
- DetectAndFlagC and LightController interact asynchronously
Robotic Platform

- Records assumptions about the hardware
  - which events the hardware provides
  - which events the hardware accepts
  - which operations the hardware supports
  - which variables are available

- Independent of controller and state-machines
- Defines a module when composed with one or more controllers
- Single point of interaction with environment
Controller

- Models a specific behaviour
- Contains:
  - Behavioural state-machines
  - Operations
  - Variables
  - Events
- Supports multiple behavioural state-machines
- Communication between state-machines is synchronous
Can robots ever be safe?

Chemical Detector: DetectAndFlagC

- Three state-machines
  1. Operation Definition – DropFlag()
  2. Operation Reference – ref RandomWalk()
  3. Behaviour Definition – DetectAndFlag

- All communication is synchronous
- Interface DF_I records assumptions:
  - input events – found, right and left
  - output events – flagged
  - available operations – move, LoadFlag, ReleaseFlag

- Behaviour state-machine records:
  - position of detected chemical spill
  - status of approach action
Can robots ever be safe?

Chemical Detector: DetectAndFlagC
State Machines

Main behavioural specification constructs

- Simple, composite and final states
- Initial and junction nodes
- Actions: entry, during, exit, transition
- Local variables
- Action language: assignments, events, operation calls, sequential composition

Exclusions

- No interlevel transitions
- No history junctions
- No parallel regions
- No inner transitions
Extra constructs

- Types based on Z Mathematical Toolkit
- Interfaces: grouping variables, events, operations
- API
  - Common operations
  - State machines
  - Pre and postconditions
  - Grouped in packages
  - Default simulation
Core notation

- Formalised in CSP, for now, for the core notation
- *Circus* and UTP in the long term
- *Semantics for refinement*
- Module = CSP Process
  - Parallel composition of controllers
  - Connections define synchronisation sets
  - Asynchronous communication modelled through buffers
- Controller = CSP Process
  - Parallel composition of state machines
  - Connections define collaborations via events
- State machine = CSP process
  - Parallel composition of states
  - Connections define flow of transitions
Can robots ever be safe?

Semantics: Overview

Challenges

- Simplicity
- Compositionality

Our compromise

- Transitions are part of the source states
- Junctions are part of the incoming transition
- Initial nodes and final states are part of the parent state
- States interact with each other to enter and exit
- States synchronise on transition triggers to support top-down interruption
- State components isolated in memory process due to sharing
Can robots ever be safe?

RoboTool

- Eclipse plugins
- Code generator for subset of the semantics
- Validation rules

Validation

- Chemical Detector and other examples
- Generated semantics used for verification using FDR3
- Large state-space for simple state-machines
- *FDR3 compression functions highly effective*
Can robots ever be safe?

RoboTool: short demonstration
“A group of e-puck robots transporting an object (blue box) towards a goal (red cylinder).”

Timed Models

Requirements
- Reasoning about time
- Time budgets
- Time deadline

Main design decisions
- Operations take 0 time
- Budget: wait(t)
- Deadline: S <{d}
- Simple clocks based on states and transitions.
Can robots ever be safe?

Timed Language
Can robots ever be safe?

Timed Language

[Diagram of a timed language system with states such as Searching, Evading, and ClosingInOnObject, transitions and events like enableObjectWatch and objectSeen?distance #T.]
Timed semantics

Current status

Conservative discrete-time extension of the untimed semantics.

- Specified using constructs of Timed CSP/CircusTime
- Translated to tock-CSP for model checking of interesting properties
- Translation to UPPAAL also of interest
Timed semantics

Current assumptions

- Conjunctive conditions.
- No program variables compared with \(\text{since}(C)\) or \(\text{sinceEntry}(S)\).
- No more than one clock compared in the same expression.

These can likely be relaxed, however, the semantic model becomes more complicated, and potentially less compositional.
Can robots ever be safe?

And now to simulations and programs

RoboSim

- General: high-level and tool independent
- For use with a variety of tools
  - simulating different kinds of robots
  - including different scenarios

RoboSim

- Automatically generated
- Guaranteed to be sound
Can robots ever be safe?

But how can we handle the robot and the environment?

- **state machine**
- **controller simulation**
  - annotated with time restrictions
- **robot simulation**
- **environment simulation**
- **model transformation (traceable)**
- **hybrid**
- **code generation**
- **verified library**

OhCircus Time
Co-simulation

*Technique that deals with the increased complexity via the coordinated use of heterogeneous models and tools. An industry standard, FMI, supports orchestration.*

**SysML Profile**

RoboChart models with other notations:
- Simulink
- Modelica
- VDM
- ...
Can robots ever be safe?

Co-simulation: Architecture Structure Diagram

Diagram showing the architecture of a chemical detection system with components such as Arena, Rover, Chemical Detector, MobilityHw, WallSensor, and ChemHw, with various attributes and relationships.
Can robots ever be safe?

Co-simulation: Connection Diagram
Co-simulation: A Simulink Diagram
Co-simulation: A UTP-based FMI semantics

- We have a CSP semantics for FMI.
- Only one cyber component: with RoboChart semantics
- We need a timed simulation semantics
- Variables become channels: output ports
- Operations are hidden
- Specification for FMI simulations
  - Verification of master algorithms
  - Hybrid reasoning
- Extension to FMI: treatment of events
So, can robots be safe?

A lot to do

- Computer vision, artificial intelligence, human-robot interaction, ethics, ...
- Software Engineering
- Theory: UTP
- Practice: new languages (formal, diagrammatic, API)
- Verification: compositional, scalable, traceable

Our distinctive vision

- Notations akin to those already used
- Sound integration
- Full life cycle

The theory is that of cyber-physical systems.