VERIFIED SOFTWARE DEVELOPMENT WITH B-METHOD

Štefan Korečko

Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice

https://kpi.fei.tuke.sk/en/person/stefan-korecko

stefan.korecko@tuke.sk

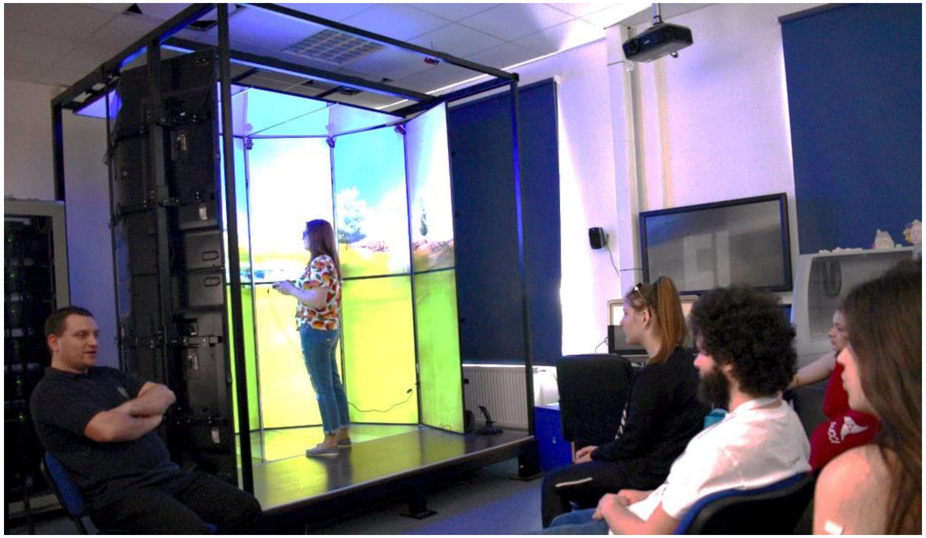
Contents

About

- DCI FEEI TU Košice
- This Course
- B-Method

DCI FEEI TU Košice

- Department of Computers and Informatics FEEI, Technical University of Košice
- https://kpi.fei.tuke.sk/en

This Course

- Verified Software Development with the B-Method
- A practical introduction to the formal software developments
  - specification,
  - verification,
  - refinement
- On an example from the railway domain

This Course :: Consultation's schedule

- All consultations are online, at https://tuke.webex.com/meet/stefan.korecko
- Friday, May 7: from 9:00 to 10:30
- Monday, May 10: from 9:00 to 10:30
- Tuesday, May 11: from 16:00 to 17:00
- Wednesday, May 12: from 15:30 till 16:30
- Friday, May 14: from 10:00 till 12:00

This Course :: Downloads

- Atelier B
  - https://www.atelierb.eu/en/download/
- Java JDK
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- BKPI Compiler
  - https://hron.fei.tuke.sk/~korecko/VVS/BKPICompiler.zip
- TS2JavaConn/TrainDirector Toolset
  - https://hron.fei.tuke.sk/~korecko/VVS/allInOneTDTS2J.zip
- Course package
  - https://hron.fei.tuke.sk/~korecko/VVS/coursePackage.zip

B-Method

- State based, model-oriented
- For software development
- Based on
  - Zermelo-Fraenkel set theory
  - Dijkstra's weakest precondition calculus
- Sophisticated development process
  - abstract formal specification -> implementation (code)
  - proof of correctness
- Specification components = B-machines (refinements, implementations)
  - Concept: close to class in OOP
  - state variables +
    invariant (restricts the variables) +
    operations (modify the variables) +
    constants, sets ...

TS2JavaConn/TrainDirector Toolset

Train Director

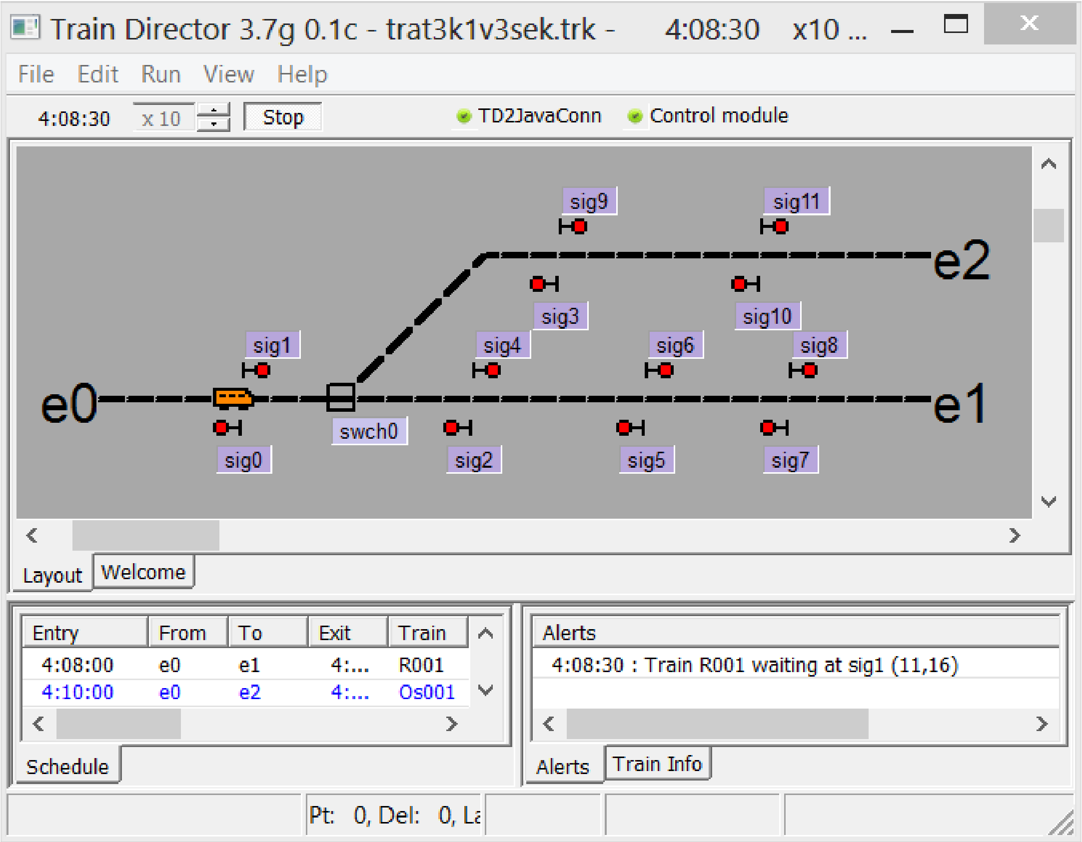
TS2JavaConn

How it works

Scenarios for the course

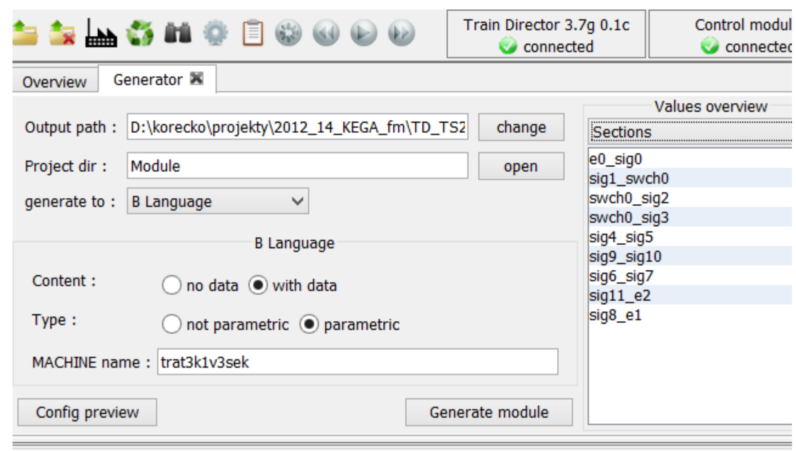
Train Director

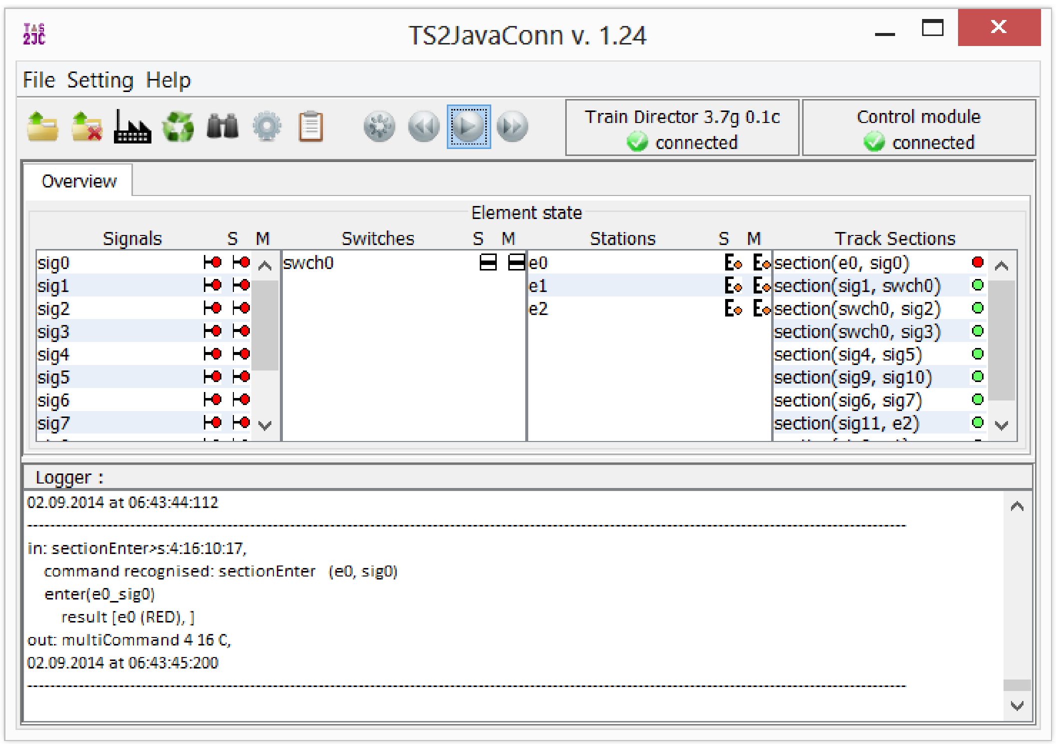
- Centralised traffic control simulator
- Modified by us
  - Enhanced communication interface
  - Internal logic disabled
- to create and simulate scenarios
  - layout + schedule



TS2JavaConn

- Communication between Train Director and control modules
- Generation of control module templates
  - Java, B language, Perfect





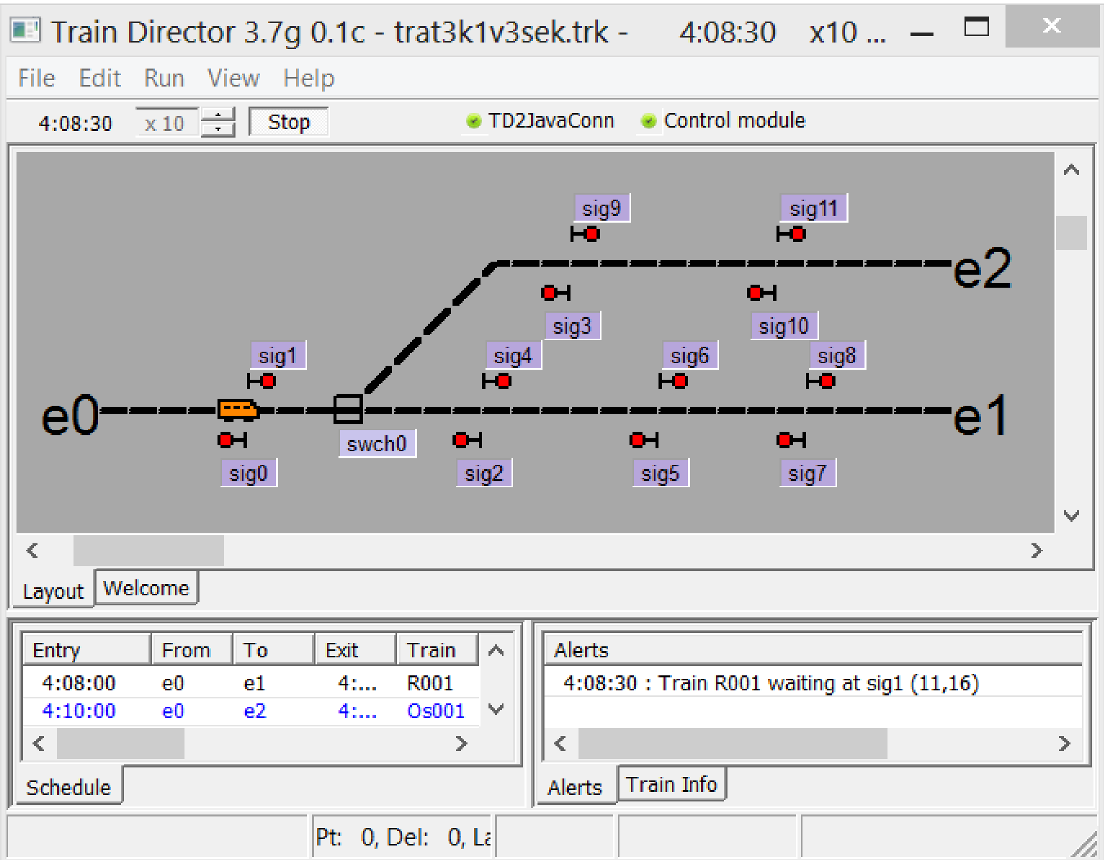
Control module

- reactive system
- Java application (developed using FM) + configuration file

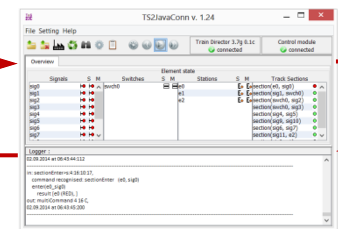
- getters
  - signals, switches, (sections)
- methods responding to events from TD
    - requestEnter
    - requestGreen
    - sectionLeave
    - sectionEnter

How it works

Train Director



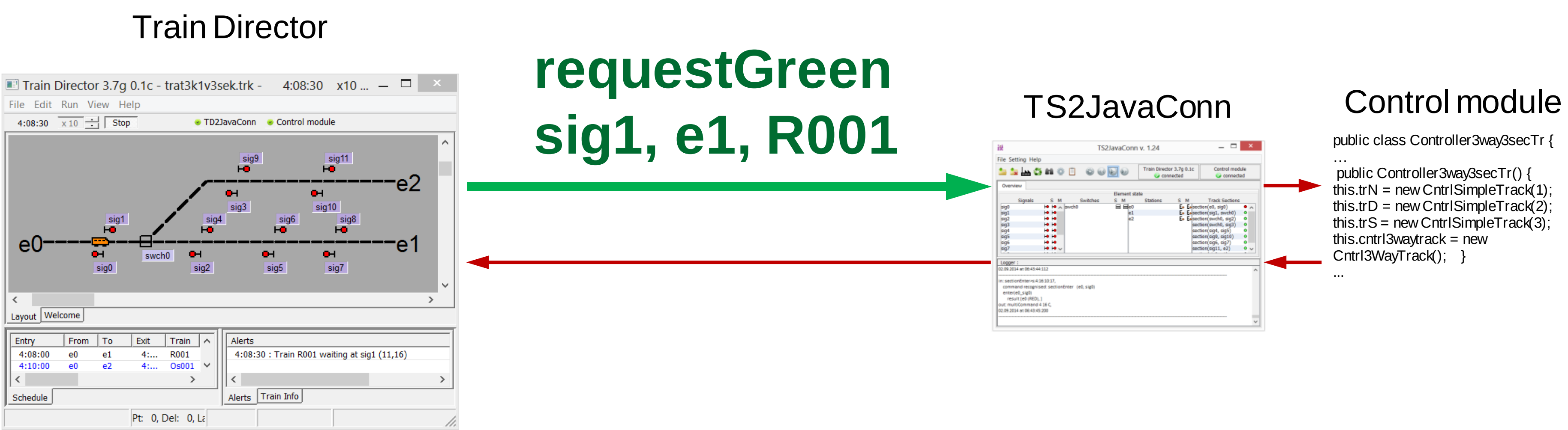
TS2JavaConn



Control module

```
public class Controller3way3secTr {
…
 public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new
Cntrl3WayTrack();   }
…
```

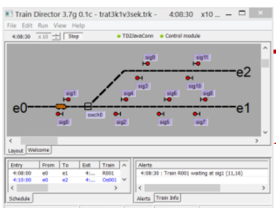
How it works


Train Director
requestGreen
sig1, e1, R001
TS2JavaConn
Control module
Train Director 3.7g 0.1c - trat3k1v3sek.trk - 4:08:30 x10 ...
File Edit Run View Help
4:08:30 x 10 Stop TD2JavaConn Control module
sig9 sig11 e2 sig3 sig10 sig1 sig4 sig6 sig8 e0 e1 swch0 sig0 sig2 sig5 sig7
Layout Welcome
Entry From To Exit Train
4:08:00 e0 e1 4:... R001
4:10:00 e0 e2 4:... Os001
Alerts
4:08:30 : Train R001 waiting at sig1 (11,16)
Schedule
Alerts Train Info
public class Controller3way3secTr {
…
public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new
Cntrl3WayTrack(); }
…

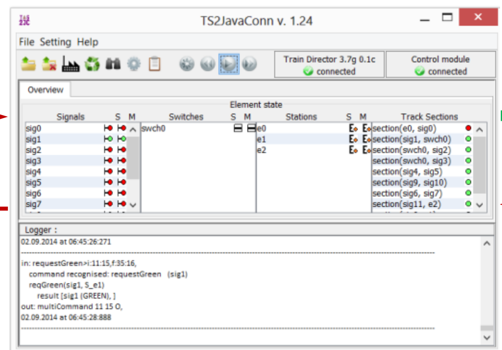
How it works

TS2JavaConn

Train Director



requestGreen sig1, e1, R001



reqGreen(sig1,S_e1)

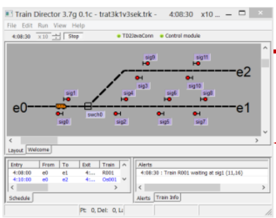
Control module

```
public class Controller3way3secTr {
…
 public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new
Cntrl3WayTrack();   }
…
```

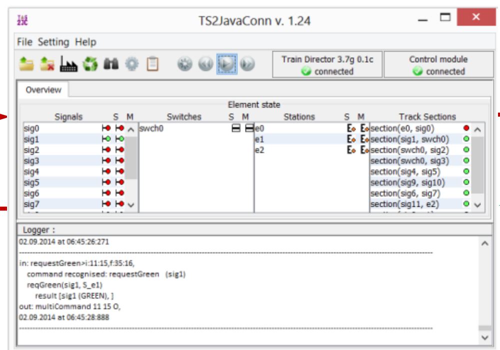
entryIndexStationName=S_%name%
requestGreen=reqGreen
requestGreenParams=%SIGNALS%,%STATIONS.f%

How it works

Train Director



TS2JavaConn



reqGreen(sig1,S_e1)

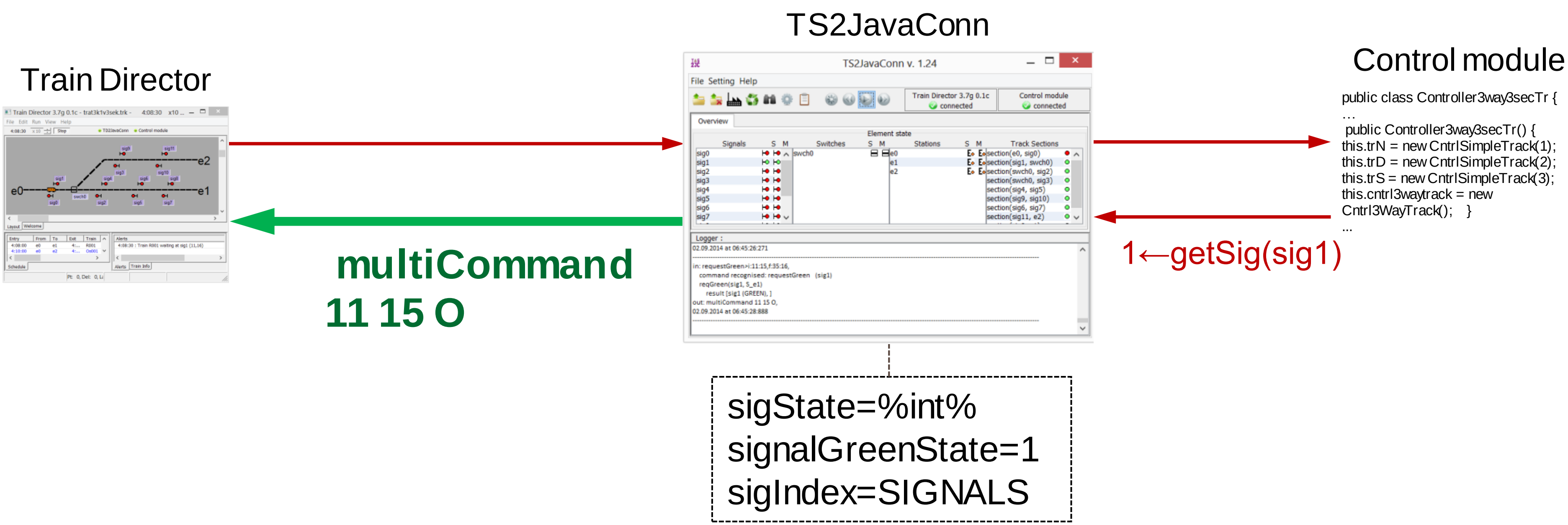
1←getSig(sig1)

sigState=%int%
sigIndex=SIGNALS
getSignalNames=getSig
getSignalParams=%SIGNALS

Control module

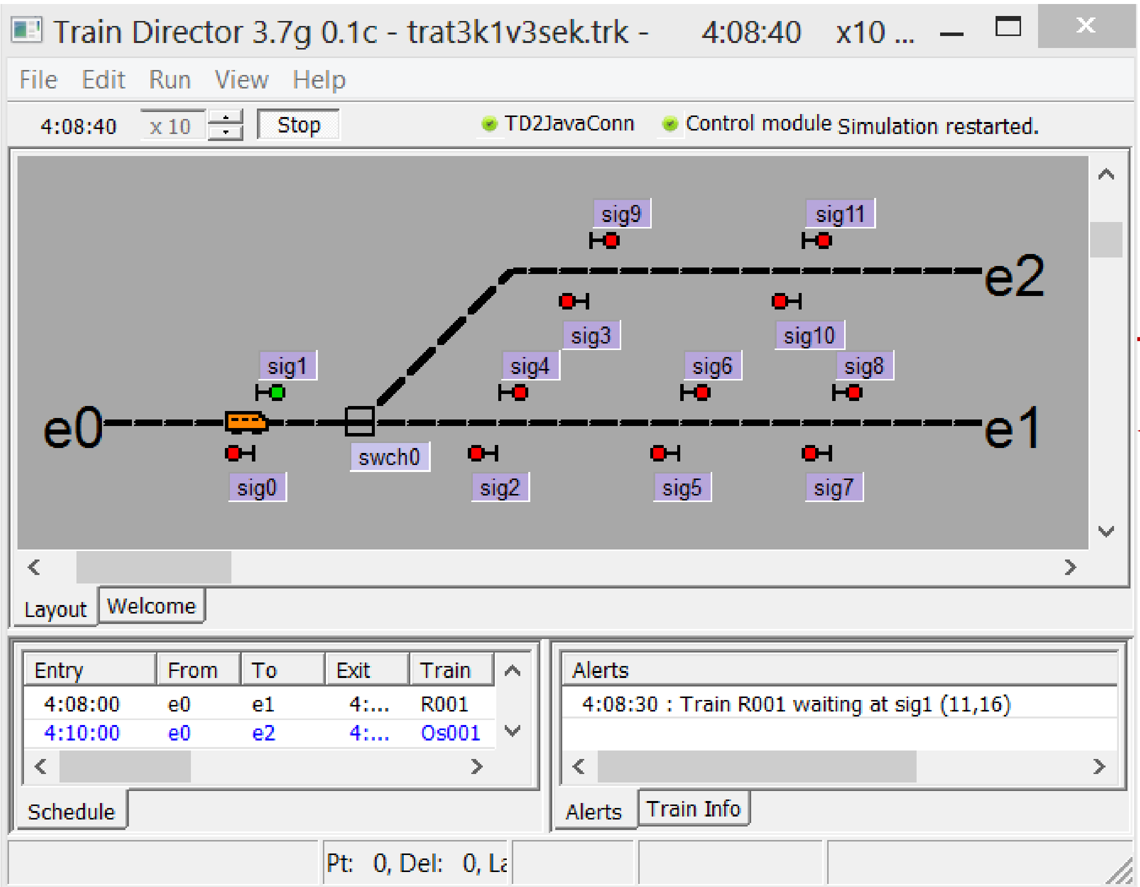
public class Controller3way3secTr {
…
public Integer **getSig**(Controller3way3secTr.SIGNALS sg)
…
public Integer getSwch(Controller3way3secTr.SWITCHES sw)
…
public void reqGreen(Controller3way3secTr.SIGNALS sg_data, Controller3way3secTr.STATIONS st) {

How it works

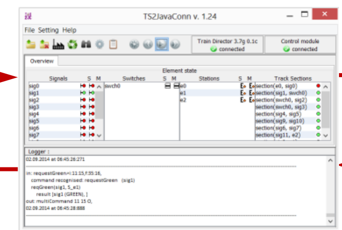

TS2JavaConn
Train Director
Control module
public class Controller3way3secTr {
…
public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new Cntrl3WayTrack(); }
…
multiCommand
11 15 O
1←getSig(sig1)
sigState=%int%
signalGreenState=1
sigIndex=SIGNALS

How it works

Train Director



TS2JavaConn

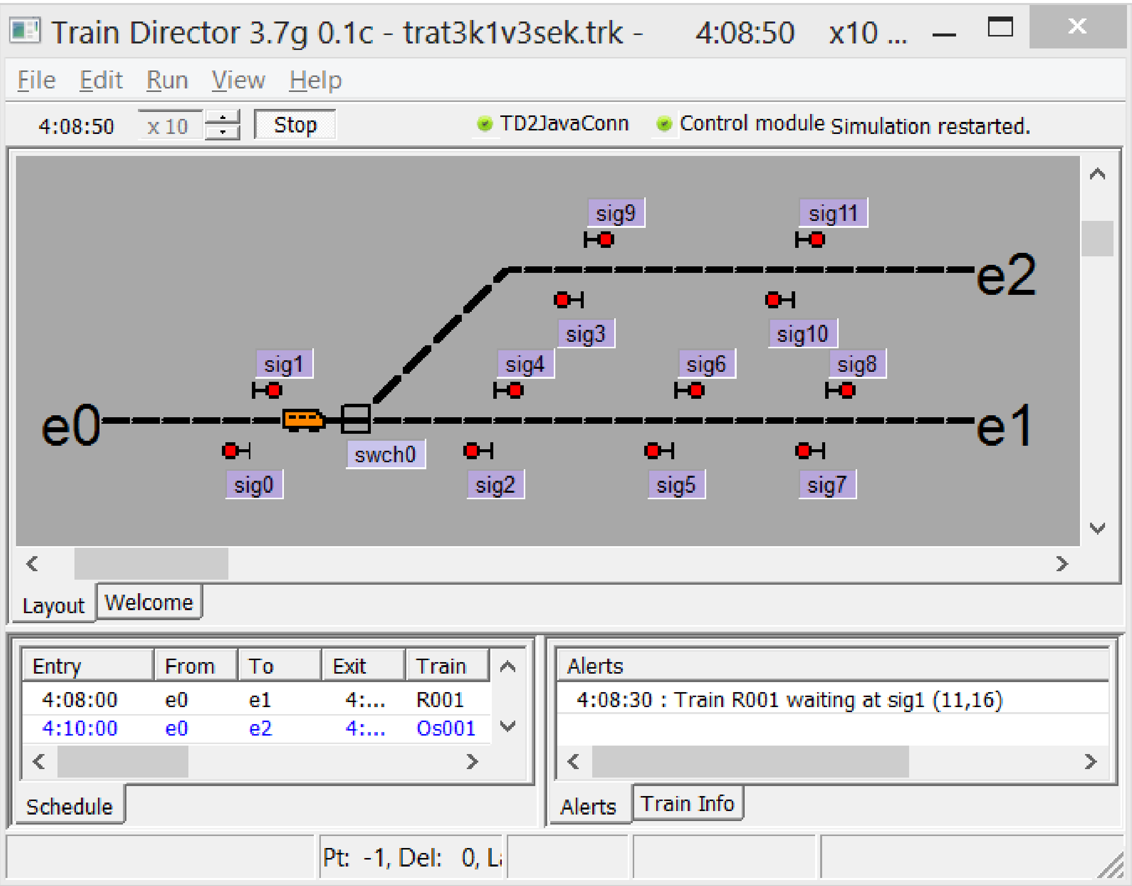


Control module

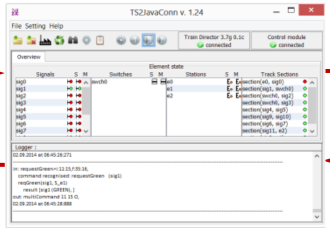
```
public class Controller3way3secTr {
…
 public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new
Cntrl3WayTrack();   }
…
```

How it works

Train Director



TS2JavaConn

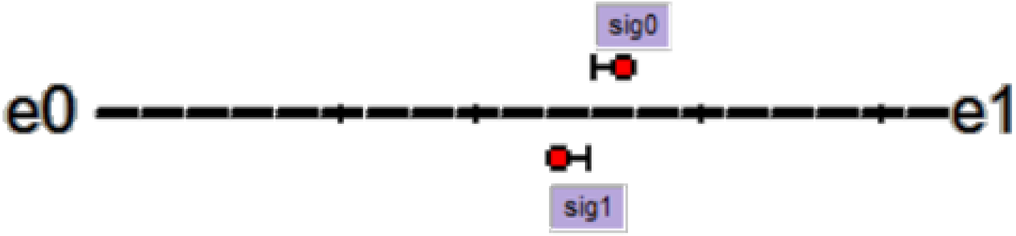


Control module

```
public class Controller3way3secTr {
…
 public Controller3way3secTr() {
this.trN = new CntrlSimpleTrack(1);
this.trD = new CntrlSimpleTrack(2);
this.trS = new CntrlSimpleTrack(3);
this.cntrl3waytrack = new
Cntrl3WayTrack();   }
…
```

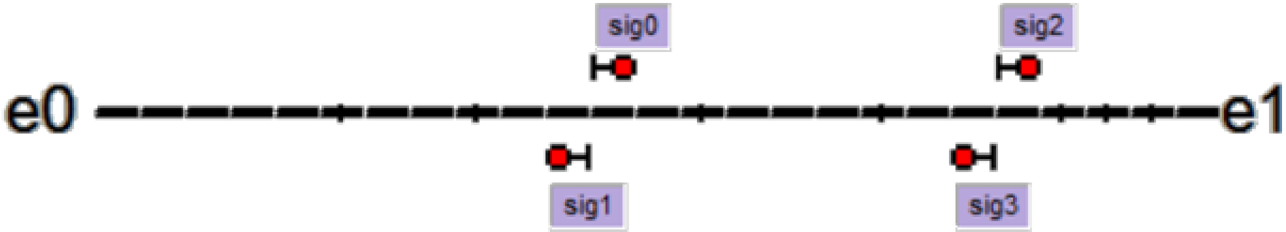
Our scenarios

- route2sec/route2sec_deadlck



|-- e0_sig1 ------||- sig0_e1 --|

- route3sec/route3sec_deadlck



- |-- e0_sig1 -------||- sig0_sig3-||-sig2_e1-|

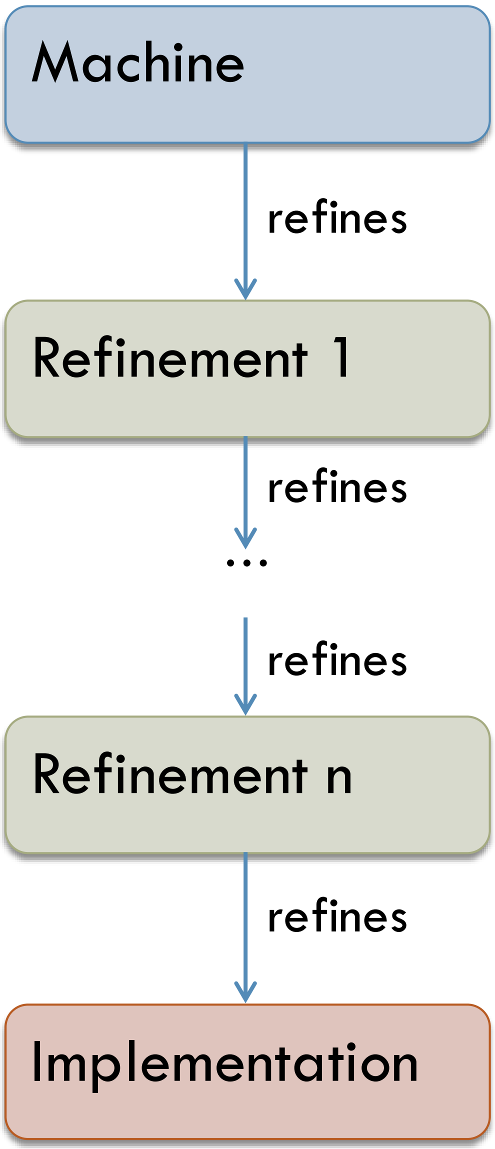
Formal Specification and Verification in B-Method

Development process

B-language

B-machine

Development in B-method



B-language

- Expressions and Predicates
  - mathematical notation
  - based on first-order logic and Zermelo-Fraenkel set theory
- Operations
  - Generalized Substitution Language (GSL)
  - based on E.W. Dijkstra's Guarded commands and Weakest Precondition Calculus

- Documentation supplied with Atelier-B
  - **B Language Reference** Manual
    - Detailed description
  - B Lanugage Keywords and Operators (**B Symbols**)
    - ASCII vs. mathematical notation

GS :: Syntax

GS	meaning of GS
skip	Empty GS (do nothing).
x := e	Assignment of value of expression e to variable x.
S_1 ; S_2	Sequential composition (do GS S_1, then GS S_2).
S_1 || S_2	Parallel composition (do S_1 and S_2 at once).
PRE E THEN S_1 END	If predicate E holds, do S_1. Otherwise, do anything.
SELECT E THEN S_1 END	If E holds, do S_1. Otherwise, do not execute.
IF E THEN S_1 ELSE S_2 END	If E holds, do S_1. Otherwise, do S_2.

(selected)

GS :: Semantics

[skip]P	=	P
[x := e]P	=	P[x:=e]
[PRE E THEN S END]P	=	$E \wedge [S]P$
[SELECT E THEN S END]P	=	$E \Rightarrow [S]P$
$[S_1 ; S_2]P$	=	$[S_1][S_2]P$
[IF E THEN S_1 ELSE S_2 END]P	=	$(E \Rightarrow [S_1]P) \wedge (\neg E \Rightarrow [S_2]P)$

(selected)

GS :: Semantics of Simultaneous Substitution

skip $\parallel$ T	= T	
x := E $\parallel$ y := F	= x, y := E, F	
(PRE P THEN S END) $\parallel$ T	= PRE P THEN S $\parallel$ T END	
(SELECT E THEN S END) $\parallel$ T	= SELECT E THEN S $\parallel$ T END	(*)
(IF E THEN S_1 ELSE S_2 END) $\parallel$ T	= IF E THEN $S_1 \parallel$ T ELSE $S_2 \parallel$ T END	(**)

(*) holds if S always terminates.
(*) holds if S_1 and S_2 always terminate.

(selected)

B-machine :: Syntax

```
MACHINE M(p)
CONSTRAINTS C
SETS St
CONSTANTS k
PROPERTIES Bh
VARIABLES v
DEFINITIONS D
INVARIANT I
INITIALISATION T
OPERATIONS
  y←op(x) =
   PRE P  THEN S END
  …
END
```

- VARIABLES is ABSTRACT_VARIABLES
  - we can also have CONCRETE_VARIABLES
- CONSTANTS is CONCRETE_CONSTANTS
  - we can also have ABSTRACT_CONSTANTS

Task 1: Machine Specification

1. Get familiar with the B-machine `route2sec.mch`
   - in `\controllers\route2sec\B` in the course package

route2sec.mch

2. Create a new B-machine `route3sec.mch` for the scenario `route3sec`
   - in `scenarios` in the course package

route3sec.mch

- Do the steps in separate Atelier B projects.
- Don´t forget about the **safety properties.**

B-machine :: Proof Obligations

```
MACHINE M(p)
CONSTRAINTS C
SETS St
CONSTANTS k
PROPERTIES Bh
VARIABLES v
DEFINITIONS D
INVARIANT I
INITIALISATION T
OPERATIONS
  y←op(x) =
   PRE P THEN S END
  ...
END
```

- Initialisation establishes the invariant
  - $(C \wedge Bh) \Rightarrow [T]\, I$
- Each operation preserves the invariant
  - $(C \wedge Bh \wedge I \wedge P) \Rightarrow [S]\, I$

(selected, simplified)

Task 2: Machine Verification

1. Prove both `route2sec.mch` and `route3sec.mch` in Atelier B.

2. Do the proof obligation for one of the `reqGreen` operations manually.

route2sec.mch

route3sec.mch

Verified Refinement

- Implementation
- GS allowed in components
- Refinement proof obligations

From machine to Implementation

```
MACHINE M(p)
CONSTRAINTS C
SETS St
CONSTANTS k
PROPERTIES Bh
VARIABLES v
DEFINITIONS D
INVARIANT I
INITIALISATION T
OPERATIONS
  y←op(x) =
   PRE P  THEN S END
  …
END
```

```
IMPLEMENTATION M_i(p)
REFINES M
SETS St1
CONCRETE_CONSTANTS k1
VALUES Vk1
PROPERTIES Bh1
VARIABLES w
DEFINITIONS D1
INVARIANT J
INITIALISATION T1
OPERATIONS
  y←op(x) =
   BEGIN S1  END
  …
END
```

GS in components

GS	Machine	Refinement	Implementation
skip	yes	yes	yes
x := e	yes	yes	yes
S_1 ; S_2	no	yes	yes
S_1 || S_2	yes	yes	no
PRE E THEN S_1 END	yes	yes	no
SELECT E THEN S_1 END	yes	yes	no
IF E THEN S_1 ELSE S_2 END	yes	yes	yes
WHILE E DO S … END	no	no	yes

(selected)

Refinement Proof Obligations

MACHINE M(p)
CONSTRAINTS C
PROPERTIES Bh
VARIABLES v
INVARIANT I
INITIALISATION T
OPERATIONS
  y←op(x) =
    PRE P THEN S END
END

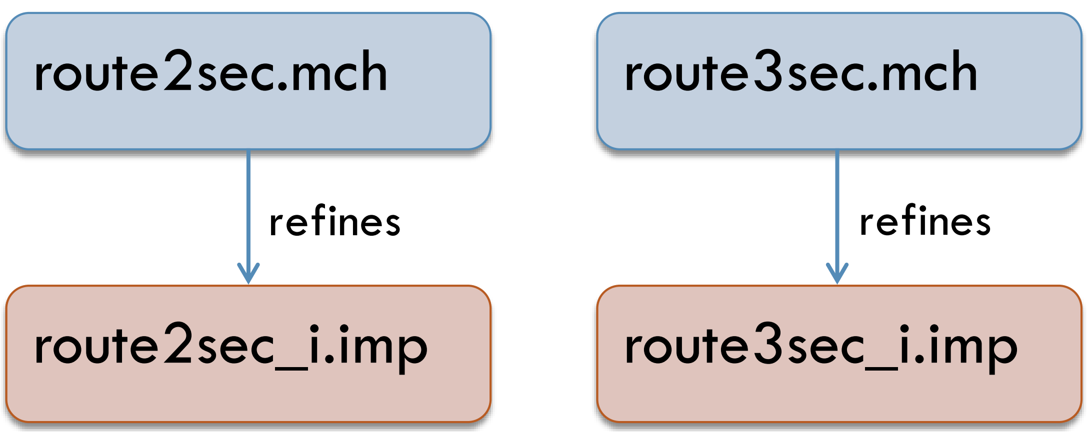
IMPLEMENTATION M_i(p)
PROPERTIES Bh1
VARIABLES w
INVARIANT J
INITIALISATION T1
OPERATIONS
  y←op(x) =
    BEGIN S1 END
  …
END

- Initialisation establishes the invariant
  - $(C \wedge Bh \wedge Bh1) \Rightarrow [T1] (\neg([T] \neg J))$
- Each operation preserves the invariant
  - $(C \wedge Bh \wedge Bh1 \wedge I \wedge J \wedge P) \Rightarrow [S1'](\neg([S] \neg (J \wedge y'=y)))$

(selected, simplified)

Task 3: Machine Refinement

1. Get familiar with the implementation `route2sec_i.imp`.
   - in `\controllers\route2sec\B` in the course package
2. Refine the B-machine `route3sec.mch` for the scenario `route3sec` in a similar way.
   - in `scenarios` in the course package
3. Verify the implementation.
4. Compile the implementation with BKPI Compiler and try with TS2JavaConn/TrainDirector

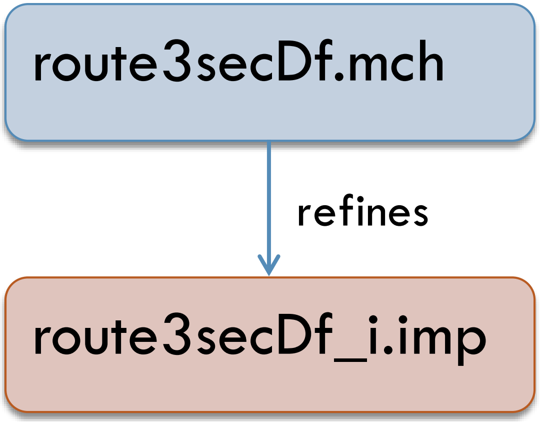


Task 4: Deadlock Free Controller

1. Specify and verify a deadlock free version of the controller for the scenarios `route3sec` and `route3sec_deadlck`.

2. Refine it to an implementation and verify the implementation.

3. Compile the implementation with BKPI Compiler and try with TS2JavaConn/TrainDirector



Modular Specification

- …

Composition Mechanisms

- To access one component from another
- Only machines can be accessed

Mech.	Access type
SEES	Read only to data elements. Seen variables cannot be used in I of the seeing component. Getter operations can be called.
USES	Read only but the used variables are allowed in the using invariant
INCLUDES	Read/white. *Promoted* operations become operations of the including component
EXTENDS	INCLUDES with all operations promoted.
IMPORTS	As INCLUDES, but in implementations

Composition Mechanisms Usability

Mechanism	Machine	Refinement	Implementation
SEES	yes	yes	yes
USES	yes	no	no
INCLUDES	yes	yes	no
EXTENDS	yes	yes	no
IMPORTS	no	no	yes

Task 5: Universal Controller

1. Specify, refine and verify a universal version controller component for a straight track of n sections.
   - It should use
     - `1` instead of `green`
     - `0` instead of `red`
     - `0` instead of `free`
     - `1` instead of `occup`
   - Number of sections is given as its parameter `length`
   - partial specification is in `\controllers\route3secComp\B` in the course package

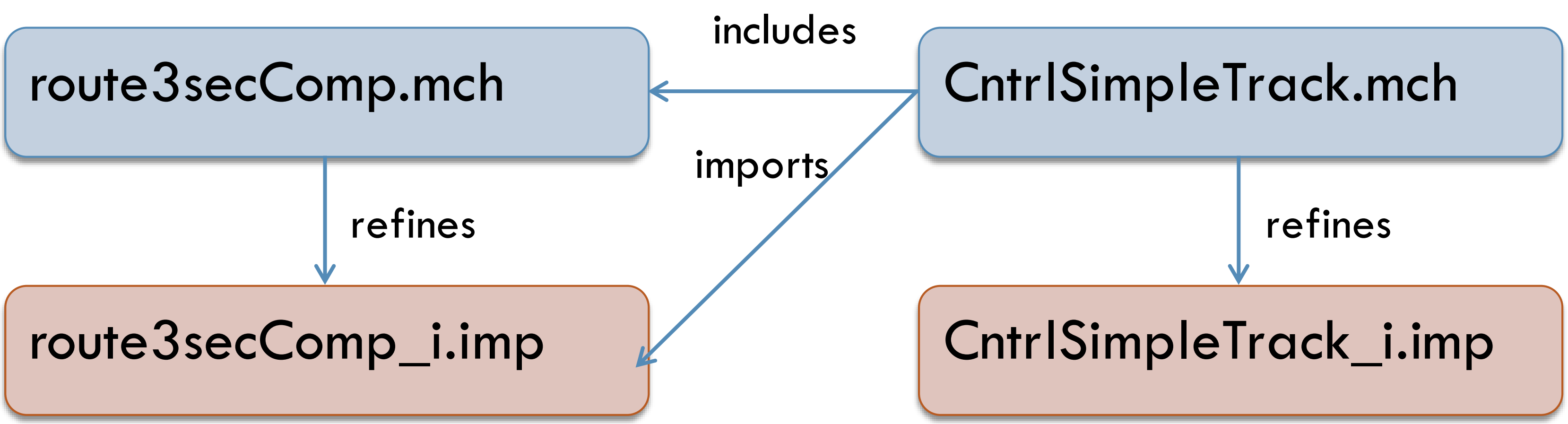
CntrlSimpleTrack.mch

refines

CntrlSimpleTrack_i.imp

Task 6: Universal Controller for route3sec

1. Use the developed universal controller to control the scenario `route3sec`.



On TS2JavaConn / TD / OR

- Korečko, Š., Sorád, J. (2014). Using simulation games in teaching formal methods for software development. *Innovative Teaching Strategies and New Learning Paradigms in Computer Programming*, in Innovative Teaching Strategies and New Learning Paradigms in Computer Programming, R. Queirós, Ed., IGI Global, 2015, pp. 106–130. (draft version available here).

- Korečko, Š., Sorád, J., Dudláková, Z., Sobota, B. (2014, September). A toolset for support of teaching formal software development. In *International Conference on Software Engineering and Formal Methods* (pp. 278-283). Springer

- Korečko, Š., Sobota, B. (2017). Computer Games as Virtual Environments for Safety-Critical Software Validation. *Journal of Information and Organizational Sciences*, *41*(2), 197-212.