# Model Transformations for Round-Trip Engineering in **Control Deployment Co-Design**

**Ken Vanherpen** University of Antwerp ken.vanherpen@uantwerpen.be

**Joachim Denil** University of Antwerp joachim.denil@uantwerpen.be

Hans Vangheluwe University of Antwerp McGill University hv@cs.mcgill.ca

# **Paul De Meulenaere** University of Antwerp paul.demeulenaere@ uantwerpen.be

51

52

53

54

55

56

57

58

59

60

61

62

64

65

66

67

68

71

74

75

76

#### ABSTRACT 1

When developing a control algorithm for a mechatronic sys-2 tem, its deployment on hardware is rarely taken into ac-3 count. Hardware properties such as execution performance, 4 memory consumption, communication delays, buffer sizes, 5 (un)reliability of the communication channel, etc. are often 6 not the first concern of the control engineer. However, these 7 properties may have important effects on the control loop be-8 haviour such that initial requirements may no longer be ful-9 10 filled. To tackle this issue, we propose a Round-Trip Engineering (RTE) method allowing for a semi-automatic integra-11 tion of hardware properties, corresponding to the deployment, 12 into the control model. The proposed RTE method combines 13 techniques of model transformations and model-based design 14 space exploration. The resulting method will enable an en-15 gineer to further enhance the control model based on imple-16 mentation properties such that the initial requirements are still 17 satisfied when deployed on the target hardware platform. 18

#### **Author Keywords** 19

Behavioural Modelling; Co-Design; Deployment 20

Optimization; Real-Time Embedded Systems; Round-Trip 21

Engineering (RTE) 22

#### **ACM Classification Keywords** 23

**B.8.2 PERFORMANCE AND RELIABILITY; C.4 PER-**24

- FORMANCE OF SYSTEMS; D.3.4 PROGRAMMING 25
- LANGUAGES: Processors-Code generation 26

#### INTRODUCTION 27

- Model Driven Engineering (MDE) and Model-Based Design 28 (MBD) are gaining more interest for the design and develop-
- 29 ment of software-intensive and cyber-physical systems [12].
- 30 This results in a development shift from hand-written code
- 31 to models from which implementation code is automatically
- 32
- generated through model-to-text transformations. Further-33
- more, various disciplines are involved in designing a cyber-34
- physical system such as mechanical engineering, control en-35
- gineering, software engineering, integration engineering, etc. 36

TMS/DEVS 2015 April 12-15, 2015, Alexandria, VA, USA Copyright © 2015 Society for Modeling & Simulation International (SCS)

This interdisciplinary design involves a multitude of views 37 on the system under design. For example, a control engi-38 neer designs a control algorithm for a certain problem and is 39 concerned about the control performance and stability of the 40 algorithm. Afterwards, a software and integration engineer 41 have to deploy this control model on a set of networked em-42 bedded systems. They use the constraints given by the con-43 trol engineer to deploy the algorithm but are also concerned 44 about resource constraints, timing constraints, schedulability 45 of software tasks, etc. 46

Because control engineers have limited aids to estimate the impact of the deployment process on their (created) control 48 algorithms, they typically do not take any resource constraints into consideration during control design. Software and inte-50 gration engineers face similar difficulties when deploying a control algorithm on hardware. The control engineer specifies timing constraints on the periodic behaviour of the algorithm for numerical stability. However, the software and integration engineer still have to decide the optimal control-loop timing for the algorithm based on control performance and resource constraints [7]. It is clear that an optimised deployment of control models onto an electronic control unit (ECU) or a set of networked ECUs remains a huge challenge.

To mend these problems, this work focuses on controldeployment co-design by introducing a Round-Trip Engineering (RTE) approach [19] as globally represented in Figure 1. It allows both the control engineer and integration en-63 gineer to assess the impact of the deployment on the control algorithm. Our RTE approach is implemented in a common design process for software-intensive and cyber-physical systems: (a) Control Design: Based on a set of requirements, a control engineer designs an algorithm to control (a part of) the system. This can be done by using Simulink<sup>®</sup> block di-69 agrams, currently the de facto standard for control engineer-70 ing. Once the simulation results conform to the given require-72 ments, the model of the system is handed over to the software engineer. (b) Deployment: The software engineer receives 73 the control model and prepares the model for deployment, for example modularisation of the control model. From the prepared control model, code is generated. Afterwards, an



Figure 1. Overview of RTE approach.

integration engineer decides how the model is deployed on a 77 set of networked ECUs by setting the operating system pa-78 135 rameters, setting bus priority messages, etc. (c) Round-Trip 79 loop: The results of the deployment process are used to create 80 new behavioural diagrams by updating the control model with 81 extra blocks. These blocks represent the effects of the deploy-82 138 ment at the level of the control engineer. As a result, the con-83 139 trol engineer can use his/her appropriate view and techniques 140 84 to evaluate the behaviour after deployment. 85 142

The rest of this paper is structured as follow. Related work is 86 elaborated in Section "Related Work". Section "Approach" 87 explains in detail our contributions, whereas Section "Case 88 Study: Power Window" applies our theory on a case study. 89 A discussion about our proposed approach and some future 90 work which needs to be tackled is described in Section "Dis-91 147 cussion & Future Work". Finally, Section "Conclusions" con-92 cludes our contributions. 93

#### **RELATED WORK** 94

The literature describes multiple scientific contributions to 152 95 introduce real-time execution behaviour when modelling a 153 96 Cyber-Physical System (CPS). Eidson et al. [6] presents the 97 PTIDES design environment as an extension to the Ptolemy 155 98 II framework. It allows a control designer to add a no-156 99 157 tion of physical time without actually deploying the system. 100 158 101 Therefore, PTIDES extends discrete-event systems with a relationship between model time and physical time at sen-102 sors, actuators and network interfaces. Another approach 159 103 is presented by Guerra et al. [10] where triple graph trans-160 104 formations are used to back-annotate original models with 161 105 analysis results. In [15] Naderlinger demonstrates how to 162 106 manipulate the Zero Execution Time (ZET) simulation be- 163 107 haviour of MATLAB/Simulink<sup>®</sup> models to real-time execu-108 tion behaviour by introducing building blocks consuming a 165 109 finite amount of simulation time. In addition, a more gen- 166 110 eral overview of integrating real-time execution behaviour at 167 111 model level is given by Derler et al. [5] where a framework of 168 112

design contracts is proposed to facilitate interaction between control and embedded integration engineers designing CPS.

However, the former mentioned methods all perform horizontal model-to-model transformations meaning they operate at a same level of abstraction. By contrast, our approach operates vertically in terms of levels of abstraction and thus performing transformations from model-to-text and vice versa. To this end, Ciccozzi et al. describe in [2,3] an approach which is similar to ours. Their round-trip solution consists out of three steps: (1) the generation of code from a source model, (2) monitoring of extra-functional properties at system level, and (3) back-annotation of the source model. Nevertheless, their back-annotation consists out of a textual description with implementation related properties meaning the system developer needs to be aware of these specific technical terms in order to optimise the deployment. In this respect, Morelli and Di Natale present the T-Res framework in [13] allowing for a co-simulation of the software model and the hardware execution platform. Inspired by TrueTime [1, 11], they introduce kernel and task blocks into the Simulink<sup>®</sup> software model (i.e. the control model which is adapted for implementation). Although they graphically back-annotate the control model with deployment information, the T-Res framework, in our opinion, aims for a collaborative design between software and integration engineer.

Our work differs in the sense that we aim for a co-design between control and integration engineer. To this end, we focus on changing the behaviour of the source models with (deployment) timing information at the level of abstraction of the control engineer, i.e. by introducing rather simple Simulink<sup>®</sup> blocks and eliminating excessive deployment information.

### APPROACH

130

131

132

136

141

143

144

145

146

148

149

150

151

Nowadays, many different disciplines are involved in the development of a control algorithm, each with their own view, skills and concerns regarding the algorithm to be developed. However, some of these views may conflict with each other and may affect the overall performance of the algorithm under development. Moreover, due to the introduction of modelbased development the integration of some of those views is postponed to a later development phase. In our case a precedence relation [17] exist between the control design and the integration engineering. This results in a late detection of conflicting views which in turn results in multiple iterations to deploy a single control model. In this paper we focus on a method to facilitate this co-design, in particular between a control engineer and an integration engineer.

### **Overview**

We graphically represent our Round-Trip Engineering (RTE) method by the Formalism Transformation Graph and Process Model (FTG+PM) shown in Figure 2. The left side of the FTG+PM declares all the involved formalisms (boxes) and all the model transformations (circles) between these different formalisms. The right side shows the process with the involved models (boxes), transformed by a model transformation (round-tangled boxes). Note that a model in the PM part is an instance of a formalism declared in the FTG part.

Furthermore, complex data-flow (dashed line) and control- 224 169 flow (full line) relations can exist in the process part of the 225 170

- FTG+PM. 171
- Our RTE method is typed by three distinguished phases: (1) 227 172

Control Design, (2) Deployment and (3) Round-Trip Loop. 228 173 Note that each of these phases respectively correspond to 229 174 one column in Figure 2. Furthermore, an Architecture De-230 175 176 scription Language (ADL) is used to store design information 232

- while executing each phase and to maintain traceability. For 177
- this purpose, formalisms such as MARTE can be used which 233 178 is a UML profile for the Modeling and Analysis of Real-Time 234 179
- and Embedded Systems [8]. 180

236 (1) During the first phase a control algorithm is created and 181 modelled by the control engineer. A software engineer re-182 238 ceiving the control model applies some other actions such as 183 the modularisation of the control model (i.e. the subdivision 184 239 of the control model in to several subsystems). Simultane-185 240 ously, the ADL model which holds the requirements of the 186 241 system is updated by adding the different subsystems to the 187 242 component level. Traceability links link the ADL model to 188 243 the behavioural models. From each subsystem, source code 189 244 is generated by performing a model-to-text transformation. 190 245

(2) From the source code, several analyses are executed in or-246 191 der to find a feasible and optimised deployment onto an Elec-247 192 tronic Control Unit (ECU) or a set of networked ECUs. These 193 248 analyses include timing analysis to obtain the Worst Case Ex-194 249 ecution Time (WCET) of the source code and schedulability 195 250 analysis to obtain the Worst Case Response Time (WRT) for 196 251 each subsystem. These extra functional properties are added 197 252 to the ADL model while maintaining traceability. Further-198 253 more, parameters related to the mapping of software compo-199 254 nents to tasks, operating systems parameters, parameters of 200 255 the tasks and messages on the bus, etc. are stored in the ADL. 201 256

(3) By using parametrised model transformation templates in 257 202 our Round-Trip Loop, a set of model transformations, based <sup>258</sup> 203 on the result of the deployment process, are created to up- 259 204 date the control design by inserting extra blocks. Note that <sup>260</sup> 205 the results of the deployment process are retrieved from the <sup>261</sup> 206 ADL model. The control engineer receives this updated con-207 trol model enabling him/her to evaluate its behaviour after 208 deployment. 209 264

#### **Round-Trip Engineering Method** 210

For each of the described phases of our RTE method, we elab-211 orate on the involved methods and tools that are used. This 268 212 further clarifies the presented FTG+PM of our RTE method. 213

#### Control Design 214

Based on a set of requirements, which are stored in an ADL, 272 215 a control engineer creates an algorithm to control (part of) 273 216 the system. A common way to specify control logic is by 274 217 using the Causal Block Diagram formalism, usually referred 275 218 to as Simulink<sup>(R)</sup> diagrams. Control engineers connect plant 276 219 models to the control models to verify the behaviour of the 277 220 designed algorithms in the context of the system with respect 278 221 to the requirements of the system. The created control models 279 222 are prepared for deployment by the control engineers. This 280 223

involves the discretisation of a continuous-time model into a discrete-time model.

If the output of the control model still meets the predetermined requirements, the model is handed over to the software engineer. He applies some other actions such as the modularisation of the control model with respect to the hardware configuration while maintaining traceability. To this end, the software engineer adds the different components to the component model of the ADL and models the interactions between the new components and the rest of the system. Traceability links link the ADL model to the behavioural models. From the different subsystems, source code is automatically generated for deployment onto the ECUs. Widely available tools like Simulink Embedded Coder® are used for this purpose.

### Deployment

226

235

265

266

267

269

270

271

The source code generated from each subsystem, called software components, need to be feasibly and optimally mapped to an ECU or a set of networked ECUs. Each software component is allocated to an operating system task and task related parameters are set. Signals originating in the software components are packed into bus messages for communication between networked ECUs. To this end, parameters such as message priority are set.

To check whether a configuration is feasible and optimal, the integration engineer starts by determining the performance of each software component by executing a timing analysis. For this purpose, two different methods exist: static and measurement-based method. The former method makes use of the generated code and a model of the target hardware to analyse the set of different possible control flow paths. The latter method executes the generated code on the target hardware or on a low-level simulation model to measure the execution time given a set of inputs. Both methods lead to the determination of the Worst Case Execution Time (WCET) indicating the upper bound on execution times of the software components running on the target hardware. A detailed overview of the tools and methods involved in obtaining the WCET can be found in [21].

The results of the timing analysis are added to the ADL model, from which a Real-Time Task Model can be derived. As a result, this model contains software related information (e.g. WCET) and information about the target hardware (e.g. number of ECUs), as well as the information related to tasks on the operating system and messages on the bus. Based on this model, an integration engineer executes a schedulability analysis. Different techniques such as the one described by Tindell and Clark [20] or Palencia and Harbour [16] can be used. As a result, the schedulability analysis provides the integration engineer a trace containing the Worst Case Response Time (WRT) for each subsystem. As its name implies, the WRT indicates the maximum bound at which the subsystem produces a signal on (one of) its outputs. Nowadays, several tools can be invoked to perform a schedulability analysis resulting in a trace containing the WRT. In the scope of this paper, we are using an analysis tool called MAST [9] for this purpose.



Figure 2. Formalism Transformation Graph (left) + Process Model (right) of our Round Trip Engineering Method.

317

323

324

325

336

337

338

339

340

341

342

One should notice that the former mentioned deployment pro- 314 281 cess is typically an iterative process. This is not taken into ac- 315 282 count in the FTG+PM shown in Figure 2, but is demonstrated 283 316

in the work of Mustafiz et al. [14]. 284

The results of this deployment are fed back to the ADL 318 285 model. However, the choices made by the integration engi-286 319 neer when deploying the system onto the hardware affect the 287 320 performance of the designed control loop. For example, a 288 321 signal can be delayed due its transmission via a bus or a soft-289 ware component may encounter a longer execution time due 322 290

to task related parameters. 291

#### Round-Trip Loop 292

- Finally, the results of the deployment process are used to cre-293 ate new behavioural models by updating the Simulink<sup>®</sup> con-327 294 trol model with extra blocks. These blocks introduce the ef-295 328 fects of the deployment at the level of the control engineer. 296 329 The control engineer can use his/her appropriate view and 297 330 techniques to evaluate the behaviour after deployment. 298 331 Therefore, we rely on a model transformations for, and in 332 299
- Simulink [4] as can be seen in the third column of the 333 300 FTG+PM. Our technique creates a parametrised rule-based 301 334 model transformation based on the original control model and 302 335 a self-defined (set of) rule(s). It serves as a template to create 303 a set of rule-based model transformations whereby the pa-304 rameters are replaced by parsing the information stored in the 305
- ADL. 306
- Updating the control model is supported by introducing delay 307
- blocks. These blocks delay signals in the Simulink model to 308 reflect the WRT as a result of the schedulability analysis.
- 309

#### CASE STUDY: POWER WINDOW 310

- In this section, the former mentioned RTE method is applied 343 311 to an automotive case study. At the time of writing, we imple- 344 312
- mented a mock-up of an ADL prior to the selection of an ap- 345 313

propriate ADL which covers all our current and future needs. As a result, however, traceability is manually maintained.

As an automotive case study, we select the power window case consisting out of four windows and typed by the following requirements/specifications [18]:

- 1. The power window consists out of four windows which can be separately operated.
- 2. All three passenger windows can be globally operated by the driver.
- 3. The operations of the driver have priority in case a passenger window is simultaneously operated by the driver and a passenger.
- 4. A window shall start moving within 200 ms after a command is issued.
- 5. A window shall automatically move to a final position when the up or down command is issued for less than 500 ms.
- 6. A window shall be fully opened or closed within 4.5 s.
- 7. When closing a window, a force of no more than 100 N may be present.
- 8. The detection of an object when closing a window should result in lowering the window by approximately 10 cm.

### **Control Design**

We create a Simulink<sup>®</sup> model based on the presented requirements consisting out of five sequential connected main parts (further called subsystems): (1) Control signals simulating the actions of driver and passengers, (2) signal debouncing, (3) control exclusion for driver priority, (4) main control design, (5) plant model to simulate the environment. We briefly describe their implementation details. Furthermore, subsystems (2) to (4) are deployed on several ECUs. This will be further elaborated in Subsection Deployment.

(1) The control signals which simulate the actions of driver  $_{401}$ 346 402

and passengers are built by using Simulink Signal Builder<sup>( $\mathbb{R}$ )</sup>. 347

For each driver and passenger a set of up and down signals 403 348 are generated. At some points in time, a simultaneous action 404

349 from driver and passenger is generated to simulate the control 405 350

exclusion. 351

407 (2) Signal debouncing is modelled by the use of Simulink 352 408 Stateflow<sup>®</sup>. The implementation of the debounce circuit is 353 409 trivial: A signal has to be in its new state for at least 30 ms 354 410 before it is forwarded. 355 411

(3) Likewise, the implementation of the control exclusion 412 356 circuit is straightforward. By using some basic logic gates, 413 357 driver priority is obtained when a driver and passenger oper- 414 358 ate the same window simultaneously. 359

(4) Our controller is based on the work of Prabhu and Moster- 416 360 man [18] which can also be find as a Simulink<sup> $\mathbb{R}$ </sup> tutorial. It 417 361 includes an implementation for most of the aforementioned 418 362 requirements as a Simulink Stateflow<sup>®</sup> diagram. 419 363

420 (5) As a last subsystem of our control design, the plant model 364 represents the environment of our power window. This in-365 cludes mainly the behaviour of the motor and the window 366 mechanism. Their properties are explicitly modelled by us-367 424 ing control theory. Note that the external pinch force is part 368 425 of the environment. Therefore, a feed back loop from plant to 369 426 control model is present. 370 427

The simulation result of this ideal control design is shown 428 371 as a solid blue curve in Figure 5 and more detailed in the 429 372 upper part of Figure 6, where the window behaviour of the 430 373 front passenger is shown. At time stamp 1s the driver ini-431 374 tiates an up-command, whereafter the window closes within 432 375 50 ms. During this movement, a force of 100 N is detected 376 at time stamp 3.15 s. This results in a revert movement of 377 the window 35 ms after pinch detection. The driver sends 378 a down-command at time stamp 8 s for a time period longer 379 than 500 ms, resulting in lowering the window 52 ms after the 380 command is initiated. At time stamp 10 s both driver and pas-381 senger issue a window command. However, their commands 382 conflict with each other giving it priority to the driver. This 383 results in a completely closed window within a time period 384 of 4.41 s. At the remainder time stamps, some other require-385 ments are tested which are irrelevant to the further course of 386 this paper. Bottom line of these simulation results is that the 387 aforementioned requirements are met. 388

#### Deployment 389

The modularisation of the subsystem is made on the basis 390 of their specific action. Each subsystem (i.e. parts (2) to 391 (4) since the other ones are a simulation of the environment) 392 needs to be deployed onto an ECU. Therefore, code is gener-393 ated for each of those subsystems by performing a model-394 to-code transformation. Since we have modelled the con-395 trol design in Matlab Simulink<sup>®</sup> we are using the Embedded 396 Coder<sup>®</sup> for this purpose. Furthermore, the ADL is updated 397 such that it contains the names of all the subsystems. 398

Once the different subsystems are transformed to C-code, a 440 399 timing analysis is executed. The Worst Case Execution Time 441 400

(WCET) is derived from this trace. To this purpose, each code-segment is separately deployed on a DVK90CAN1 development board which holds an 8-bit AT90CAN128 microcontroller. After a repeated execution of all possible functionalities of the different subsystems, our timing analysis results in the WCET shown in column 'C' of Table 1. Along with the timing periods ('T') and deadlines ('D') shown in Table 1 this information is stored in the ADL. Note that the task names refer to the different subsystems of our control design, where 'Deb', 'PW' and 'CE' are abbreviations for 'Debounce', 'Power Window' (i.e. the controller which holds the stateflow of the power window) and 'Control Exclusion' respectively. For the rear passenger windows the same results are obtained as for the front passenger.

406

In addition to the information related to the subsystems, the ADL is further extended by the deployment engineer with a description of the hardware architecture (i.e. the number of ECUs, type of scheduler, etc.). At this point, the ADL stores al the information needed by schedulability algorithms to determine the deployment of the subsystems onto the different ECUs. For this purpose we use the MAST tool [9] and select the 'Offset Based Approximate Analysis' technique [16]. An optimal system deployment is found by using two ECUs for each power window. One holds the debounce circuit, while the other one holds the control exclusion circuit and the power window control logic. However, the window control parts are not the only tasks allocated to the ECUs. Other miscellaneous tasks ('Misc'), which for example control the wing mirrors ('WM') or the door lock system ('DL'), interfere with the execution of the window control parts because of operating system mechanisms like pre-emption. This is taken into account in our schedulability analysis as can be seen in Table 1.

ECU	Tasks					
Name	Name	C	Т	D	Р	W
DRV_1	WM_DRV	21	85	85	8	21.5
DRV_1	DL_DRV	32	95	95	7	54
DRV_1	Deb_DRV	30	100	100	5	84.5
DRV_2	PW_DRV	42	50	50	5	42.5
Front_1	Misc1_Front	21	85	85	8	21.5
Front_1	Misc2_Front	32	95	95	7	54
Front_1	Deb_Front	30	100	100	5	84.5
Front_2	PW_Front	42	50	50	5	42.5
Front_2	CE_Front	0.25	50	50	4	43.25

Table 1. ECU Mapping (C: WCET in [ms], T: Time Interval in [ms], D: Deadline in [ms], P: Priority, W: WRT in [ms]).

### **Round-Trip Loop**

433

434

435

436

437

438

439

The schedulability trace holding the Worst Case Response Time (WRT) is added to the ADL and serves as an input for our Round-Trip Loop. As already mentioned, the WRT indicates the maximum bound at which the subsystem produces a signal on (one of) its outputs. In other words, how much an output signal is delayed compared to an ideal simulation. For example, a debounce circuit takes up to 84.5 ms to produce an output. In more detail, we can deduce that this time



(b) Right Hand Side (RHS).

Figure 3. Parametrised Model Transformation Rule.

is composed out of 30 ms computation time ('C') and 54.5 ms 442 scheduling related time. This is valuable information for the 443 control engineer because these delays might affect the over-444 all behaviour of the designed control loop (which was done 445 by the availability of unlimited resources). Moreover, by de-446 tailing the WRT a control engineer is able to focus on the 447 optimization of the computation time since this is related to 448 the control design. 449

To this end, the Simulink<sup> $\mathbb{R}$ </sup> model of the control design is 450 updated with extra delay blocks. Therefore, a parametrised 451 rule-based model transformation is created as depicted in 452 Figure 3. At the Left Hand Side (LHS) of the rule an 453 original output of a subsystem is modelled, which needs 454 to be transformed to an output followed by a two delay 455 blocks as modelled at the Right Hand Side (RHS) of the rule. 456 This parametrised template will be used by our Round-Trip 457 Loop to create a set of model transformations based on the 458 traces of the schedulability analysis which are stored in 459 the ADL. Therefore, parameters characterised by asterisks 460 will be replaced by values derived from the schedulability 461 trace. For example, Figure 4 shows the result of a model 462 transformation applied on the debounce subsystem of the 463 front passenger. Executing all model transformations based 464 on the schedulability trace results in a new control design 465 allowing a control engineer to evaluate its behaviour after 466 467 deployment.

# 468

### 469 Results

When evaluating the simulation result of the new behavioural 492 470 493 diagram, shown as a dashdotted red curve in Figures 5 and 6. 471 494 we see remarkable differences compared to the ideal simu-472 lation result (solid blue curve). While closing the window, 495 473 a force of 100 N was detected at time stamp 3.15 s. Ideally 474 it only took 35 ms to reverse the movement of the window. 496 475 When taking into account the delays after deployment, one 497 476 can notice the timespan between detection and action is in- 498 477 creased to 83 ms. Due to this slow response time, the window 499 478 closes for another 3.9 mm before an action takes place. Al- 500 479 though requirements don't mention any response time when 501 480



Figure 4. Result of a model transformation.



Figure 5. Simulation results - Front passenger.

an object is detected, it is clear that the slower response time may lead to safety issues compared to the ideal situation where the window closes for only 1.4 mm after detection. A serious violation of the fourth requirement can be found when comparing the reaction times after a command is issued. For example, at time stamp 8 s the driver issues a lower-command resulting in an ideal reaction time of 52 ms. However, after deployment the reaction time appears to be 221 ms which is 10% higher than required. Likewise, a violation of the sixth requirement can be identified because it now takes 4.7 s to fully close the window.

The simulation results show that our parametrised model transformations do add essential information to the control model in order to evaluate the control performance for a given deployment.

### **DISCUSSION & FUTURE WORK**

482

485

487

488

489

490

491

Applying our Round-Trip Engineering method to a rather simple case study shows how to inject the behaviour of the deployment to a higher level of abstraction. This enables a control engineer to evaluate the updated control design using his/her appropriate view and techniques. Updating the con-



Figure 6. Detail of simulation results - Upper: Ideal simulation; Lower: After in-place transformation.

trol design consists of placing delay blocks representing the 536 502 Worst Case Response Time of the prior subsystems. Future 537 503 work consists out of the implementation of an appropriate Ar-538 504 chitecture Description Language (ADL) which replaces the 539 505 mock-up used so far. This will enable us to maintain trace-506 540 ability (semi-)automatically by adding traceability links be-541 507 tween the ADL model and the behavioural models, similarly 542 508 as in [2]. Naturally, the software and integration engineers 543 509 receiving, subdividing and deploying the control model will 544 510 model their actions to the component model of the selected 545 511 ADL. 512 546

In order to obtain a control design which is deployable onto a 513 set of ECUs, we believe our proposed method needs to be ex-514 549 ecuted iteratively. After the first Round-Trip Loop, a control 515 550 engineer can modify the control design such that the require-516 551 ments are again fulfilled whereafter the process starts over. 517 During these successive iterations, the available ADL can be 518 552 used to enhance the deployment process. In other words, the 519 553 Design-Space Exploration (DSE) process for optimal deploy-520 554 ment can be influenced by taking into account information 521 555 available in the ADL. 522 556

These future design optimizations should result in con-523 557 tracts to facilitate interaction between control and soft-524 558 ware/integration engineers in the design of Cyber-Physical 525 559 Systems [5]. In the end, the design contracts are used to in-526 560 fluence the design process in a structured way to allow for 527 561 control deployment co-design. 528 562

### 529 CONCLUSIONS

In this paper we described a Round-Trip Engineering method
 which allows control engineers to evaluate the implications
 of their deployed control design. Our contribution focused on
 making this information available into the behavioural mod this method enables a control engineer to evaluate the
 control design using his/her appropriate view and techniques

without having knowledge of lower level specifications (e.g. buffer usage). The Round-Trip Engineering method proposed in this paper makes use of a set model-to-model and modelto-text transformation to schedule the control design onto a set of ECUs. Therefore, trace information originating from timing analysis is used. By using a parametrised model transformation, a set of model transformations are created based on the traces of schedulability analysis. This enables us to update the control design by introducing delay blocks conform the Worst Case Response Times (WRTs) of the deployed system. The usefulness of this method is illustrated by a power window case study. We showed how the deployment of the power window control design affected the overall behaviour. Moreover, we have illustrated how initial requirements are no longer met resulting in unsafe conditions. This is of importance when dealing with safety critical systems.

### Acknowledgements

This work has been carried out within the framework of the MBSE4Mechatronics project (grant nr. 130013) of the agency for Innovation by Science and Technology in Flanders (IWT-Vlaanderen).

## REFERENCES

563

- 1. Cervin, A., Henriksson, D., Lincoln, B., Eker, J., and Arzen, K.-E. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems 23*, 3 (June 2003), 16–30.
- Ciccozzi, F., Cicchetti, A., and Sjödin, M. Round-trip support for extra-functional property management in model-driven engineering of embedded systems. *Information and Software Technology 55*, 6 (June 2013), 1085–1100.
- Ciccozzi, F., Saadatmand, M., Cicchetti, A., and Sjödin, M. An automated round-trip support towards

- deployment assessment in component-based embedded
- systems. In Proceedings of the 16th International ACM
   Sigsoft symposium on Component-based software
- <sup>573</sup> engineering CBSE '13, ACM Press (2013), 179–188.
- 4. Denil, J., Mosterman, P. J., and Vangheluwe, H.
  Rule-Based Model Transformation For, and In
  Simulink. In *DEVS '14 Proceedings of the Symposium* on Theory of Modeling & Simulation (2014).
- 578 5. Derler, P., Lee, E. a., Tripakis, S., and Törngren, M.
  Cyber-physical system design contracts. In *Proceedings* of the ACM/IEEE 4th International Conference on Cyber-Physical Systems - ICCPS '13, ACM Press
  (2013), 109.
- 6. Eidson, J. C., Lee, E. A., Matic, S., Seshia, S. A., and
  Zou, J. Distributed Real-Time Software for
  Cyber-Physical Systems. *Proceedings of the IEEE 100*,
  1 (Jan. 2012), 45–59.
- <sup>587</sup> 7. Ernst, R. Codesign of embedded systems: status and
  <sup>588</sup> trends. *IEEE Design & Test of Computers 15*, 2 (1998),
  <sup>589</sup> 45–54.
- 8. Faugere, M., Bourbeau, T., Simone, R. D., and Gerard,
  S. MARTE: Also an UML Profile for Modeling AADL
  Applications. In *12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS*2007), no. Iceccs, IEEE (2007), 359–364.
- Gonzalez Harbour, M., Gutierrez Garcia, J., Palencia
   Gutierrez, J., and Drake Moyano, J. MAST: Modeling
   and analysis suite for real time applications. In
   *Proceedings 13th Euromicro Conference on Real-Time Systems*, IEEE Comput. Soc (2001), 125–134.
- Guerra, E., Sanz, D., Diaz, P., and Aedo, I. A
  Transformation-Driven Approach to the Verification of
  Security Policies in Web Designs. *ICWE'07 4607*(2007), 269–284.
- Henriksson, D., Cervin, A., and Arzén, K.-E. TrueTime : Real-time Control System Simulation with MATLAB /
  Simulink. In *Proceedings of the Nordic MATLAB Conference* (2003).
- Liggesmeyer, P., and Trapp, M. Trends in Embedded
  Software Engineering. *IEEE Software 26*, 3 (May 2009),
  19–25.
- Morelli, M., and Di Natale, M. Control and Scheduling
   Co-design for a Simulated Quadcopter Robot : A
   Model-Driven Approach. In *SIMPAR 2014* (2014),
- <sup>614</sup> 49–61.
- Mustafiz, S., Denil, J., Levi, L., and Vangheluwe, H. The
   FTG + PM Framework for Multi-Paradigm Modelling :
- <sup>617</sup> An Automotive Case Study. In *Proceeding MPM* '12
- Proceedings of the 6th International Workshop on
   Multi-Paradigm Modeling (2012), 13–18.
- 15. Naderlinger, A. Multiple Real-Time Semantics on top of
   Synchronous Block Diagrams. In *DEVS 13 Proceedings* of the Symposium on Theory of Modeling & Simulation
   (2013).

16. Palencia, J., and Gonzalez Harbour, M. Schedulability analysis for tasks with static and dynamic offsets. *Proceedings 19th IEEE Real-Time Systems Symposium* (*Cat. No.98CB36279*) (1998), 26–37.

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

- Persson, M., Törngren, M., Qamar, A., Westman, J., Biehl, M., Tripakis, S., Vangheluwe, H., and Denil, J. A Characterization of Integrated Multi-View Modeling in the Context of Embedded and Cyber-Physical Systems. In *Proceedings of the Eleventh ACM International Conference on Embedded Software*, IEEE Press (2013), 10:1–10:10.
- Prabhu, S. M., and Mosterman, P. J. Model-Based Design of a Power Window System: Modeling, Simulation, and Validation. In *Society for Experimental Machines IMAC Conference* (2004).
- 19. Sendall, S., and Küster, J. Taming Model Round-Trip Engineering. In Proceedings of Workshop on BestPractices for Model-Driven Software Development (part of 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications) (2004).
- 20. Tindell, K., and Clark, J. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming 40*, 2-3 (Apr. 1994), 117–134.
- 21. Wilhelm, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., Stenström, P., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., and Heckmann, R. The worst-case execution-time problem-overview of methods and survey of tools. ACM Transactions on Embedded Computing Systems 7, 3 (Apr. 2008), 1–53.