Introducing Control Theory in Industry: the case of V-model embedded software developers

Ubaldo Tiberi

The author is with Volvo Group Trucks Technology, Göteborg, Sweden.
Email: ubaldo.tiberi@volvo.com

Abstract: This paper presents an education course on Control Theory suitable for embedded software developers that are familiar with the V-model. The need of such a course is due to the specific audience and business needs, but it can be easily adapted and employed in other domains. The course is structured in terms of a Control theoretical workflow that resembles the development workflow provided by the V-model. Each phase represents an education module. The effectiveness of the course is evaluated through questionnaires and analysis of the software deliveries quality of some employees before and after having attended the course.

Keywords: Control Theory; Education; Pedagogy; V-model; Embedded software.

1. INTRODUCTION

The lack of technical competencies in a workplace has a twofold negative impact: the first is trivially connected to low quality deliveries whereas the second is related to workers health. In-fact, as Densten (2001) and Pines (1993) observed, failure in achieving important work-related goals could lead to potential consequences on the workers health and a shortage of technical competences is most likely to favor such a scenario. Embedded software Industry where Control Theory skills would be advantageous is not exempt. As reported by Samad (2017): ”Industry lacks staff with technical competency in advanced control that is required for high-impact applications”. One reason for such a competences shortage is due to that staff involved in Control Theory education in the past was typically formed by researchers that had to include teaching duties as a beside their research activities. Unfortunately, they had none or too little guidance on good practice in education and pedagogy as remarked in Rossiter et al. (2014). Moreover, we suspect that the increasing emphasis of Control Theory Research in applied math at the expense of minor applications focus over time, Samad (2017), further increased the gap over time between Control Theory education content and Industrial needs.

However, the importance of Control Theory education in Universities has been recently acknowledged. The creation of education committees under engineering institutions such as IFAC and IEEE boosted the collaboration among Universities for what concerns Control Theory education, Rossiter et al. (2018). It is also admirable how Control Theory is introduced even in different contexts like for example in Science, Technology, Engineering and Math (STEM) middle and high-schools, Abramovitch (2019). Nevertheless, most of the effort in improving Control Theory education is devoted to undergraduated programs but, to the best of the Author’s knowledge, there are no contributions devoted to an Industrial setting.

Although industry workers may fulfill their competence gap in Control Theory through University or Massive Open Online Courses (MOOC), this is often not possible due to Companies budget limitations and hard deadlines. Moreover, due to the universality of Control Theory, such courses are often too theoretical and they may fail in capturing Industrial needs. Hence, the necessity of tailoring an ad-hoc education course for industrial workers naturally emerged.

The proposed course is fairly practical and it is structured by connecting the V-model software development workflow (see e.g. Fowler and Silver (2015) or similar textbooks), with a Control theoretical development workflow. Each phase of the Control Theory workflow corresponds to a course module and it is connected to a V-model phase. The overall aim of the course is to offer trainees an organic knowledge and understanding underlying Control Theory. The effectiveness of the course is evaluated through questionnaires and through analysis of software quality improvements.

We wish to highlight that although the proposed approach includes jumps from one branch to the other of the V-model, and then the proposal of a modified version of such a development model would appear more natural, we have to consider the natural inertia with respect to change of humans. That is, the proposal of substantial changes to existing models would create an obstacle to the penetration of Control Theory in embedded software Industry where the V-model is typically well-established. Hence, we preferred to keep the V-model as is and to map the Control Theory workflow into it.

2. PROBLEM FORMULATION

There are two main challenges for the proposed course: the first is connected to the available budget and the second to the different background of the trainees.

The budget allocated for competence development in Industry is typically very low compared to the budget used for product development. This aspect, along with the prioritization of software development tasks, leave little room for self-learning and final course assessments are seldom performed. Accordingly, the idea of providing a complete education course where the

⋆ This work was financed by the Vinnova FFI project MutiMEC, ref. number 2014-06249.

trainees are expected to perform self-learning activities and to perform a final assessment is not feasible in most of the cases.

Next, the background of the trainees may be largely different and so is their approach in designing control strategies. Hence, the educator must found a common ground to allow all the trainees to develop knowledge. The vocabulary used shall also be carefully considered. This aspect is central since Control Theory and Software Engineering may attribute completely different meanings to the same words. For example, what is called ”system” in Control Theory (i.e. the plant or the process that needs to be controlled), in Software Engineering it becomes the electronic board with a running software; the inputs in Control Theory become the parameters in Software Engineering; the process/plant in Control Theory is referred as the ”environment” in Software Engineering. A final hurdle is that many of the trainees have the tendency to avoid mathematics since they often perceive it as an obstacle rather than as a supportive tool.

In the light of such challenges, we propose an education course in terms of a journey through the culture, methods and tools by showing only the surface of Control Theory. That is, instead of instructing the trainees on some specific method or tool as it would happen in a typical industrial training, we attempt at imparting knowledge, skill and judgment in a cultural sense. We start the proposed education course by explaining why Control Theory can help them in their daily job and what it is. Then, we show how to develop a embedded software products in a Control theoretical workflow. Such a workflow, that somehow goes along the line of the V-model, is presented as a set of sequential (but possibly overlapping) steps needed to develop the whole embedded software product. Such steps include system analysis, signal processing, system modeling, controller synthesis, calibration and test. The proposed course content is completed with the observers and with some additional notions that may be useful to understand the Control Theory literature.

Finally, from an education strategy standpoint, we observed that it is more pedagogic to follow an inductive rather than a deductive teaching approach. That is, for most of the modules we present some practical example - preferably belonging to the audience application domain - and then we let the trainees to discuss, collaborate and suggest possible solutions to that specific problem. We continuously place questions around the proposed solution in a colloquial way with the aim of stimulating, adjusting and facilitating their reasoning until they reach a correct conclusion autonomously. The correct conclusions are then used as a lever for introducing the target Control Theory theoretical argument. We further provide the audience with a large number of ad-hoc prepared simulation models at the end of each module where they can practice what they learned in addition to numerous reference to literature.

3. WHY DO WE NEED CONTROL THEORY?

One major challenge in educational programs is to capture the trainees interest and to keep it at high level throughout all the educational process. Sinek (2009) observed that an effective technique to capture audience interest is to firstly explain why a certain subject is worth of attention, further providing details on how and what only later on. With this in mind, we typically start our education course by depicting some unwanted situations that are common in embedded software development workplaces and such that the trainees can readily recognize. Our goal here is to alert the trainees that we are well-aware of the hurdles they face daily at work. Then, we firmly state that Control Theory provides culture, methods and tools to cope with such hurdles. The recognition of the exposed problems on one hand and the suggestion of a possible solution on the other hand typically create a bond between the trainees and the trainer that will hopefully be kept throughout the whole educational process. To prove that Control Theory can cope with the situations depicted so far, we provide a short history lesson by heavily relying on the paper of Bennet (1996) where we underline two evidences: the first is that engineering problems faced nowadays are exactly the same addressed at the beginning of the XX Century; the second is that many of these old problems have been successfully solved through the development and exploitation of Control Theory and therefore there are good chances that we can solve many current problems by following the same approach used in the past.

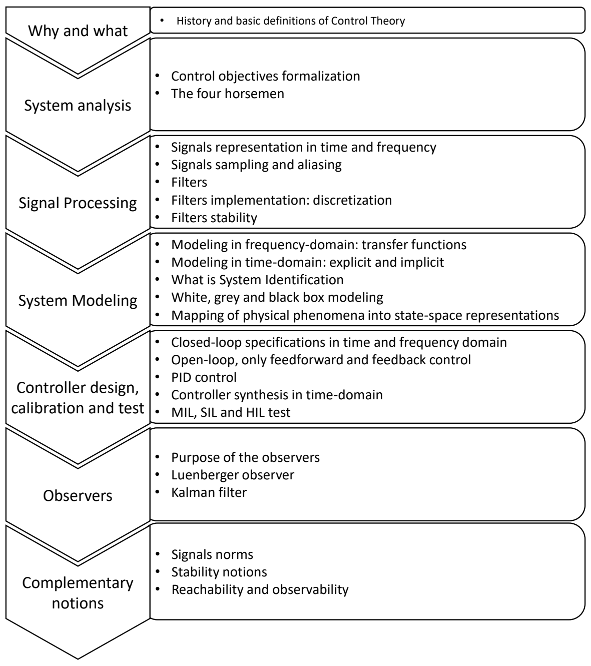
Why and what
History and basic definitions of Control Theory
System analysis
Control objectives formalization
The four horsemen
Signal Processing
Signals representation in time and frequency
Signals sampling and aliasing
Filters
Filters implementation: discretization
Filters stability
System Modeling
Modeling in frequency-domain: transfer functions
Modeling in time-domain: explicit and implicit
What is System Identification
White, grey and black box modeling
Mapping of physical phenomena into state-space representations
Controller design, calibration and test
Closed-loop specifications in time and frequency domain
Open-loop, only feedforward and feedback control
PID control
Controller synthesis in time-domain
MIL, SIL and HIL test
Observers
Purpose of the observers
Luenberger observer
Kalman filter
Complementary notions
Signals norms
Stability notions
Reachability and observability

Fig. 1. Schematic representation of the proposed education workflow.

We experienced that the above explanation is sound enough to provide an answer to the question ”Why do we need Control Theory?”. Such soundness is measured by observing the audience that typically displays a mixed expression between attention, curiosity and skepticism.

4. WHAT IS CONTROL THEORY: BASIC DEFINITIONS

We attempt at defining Control Theory by claiming that it studies decision making methods to achieve a given goal through exploitation of available information. Despite the given definition may sound rather incomplete and questionable, we experienced that it is a good start for introducing the subject. Nevertheless, the given definition may appear fairly abstract at first glance. We clarify it by stressing that it encompasses three ingredients: decision making, available information and goals. We further provide a number of extremely simple examples showing how control happens in our everyday life. Just to give some, we say that we take the umbrella (decision making) because outside is raining (available information) and we don’t want to get wet (goal); we push the pedal brake while driving (decision making) because there is a pedestrian on the zebra crossing (available information) and that don't want to hit him (goal) and so on. Then, we then challenge the audience to construct some example that contains all the three ingredients in the given definition. The outcome is often a pleasant experience and the audience seems to have fun with this exercise.

5. HOW CONTROL THEORY WORKS: THE PROPOSED WORKFLOW

After having defined why we need Control Theory and what it is, we describe how it works. Due to the huge amount of information that trainees have to process daily, we observed that they prefer to have guidelines and clear to-do list where items can be marked off. With this in mind, we try to establish an embedded product development workflow organized in sequential (and possibly overlapping) steps which include system analysis, signal processing, system modeling, controller synthesis, calibration and test. The education course content is depicted in Figure 1.

5.1 System Analysis: control objectives and the four horsemen

The education goal of this module includes two items: the formalization of control objectives and the identification of the "four horsemen", namely the state, the input, the output and the disturbance of the process that we want to control.

Regarding the first item, we suggest to initiate the design process by firstly figuring out the desired closed-loop system behavior through user-requirements analysis. In Software Engineering vocabulary, user-requirements describe "how the overall system shall behave when it is embedded into the relevant environment". Such requirements are typically written down in natural language without following any particular formalism and they come from brainstorming sessions among different stakeholders with different background like project managers, engineers, mechanics, software developers, etc. The consequence of this approach introduces a number of issues including incompleteness, inconsistency and ambiguity. Here, we typically catch the audience interest by claiming that Control Theory solves such issues since it encourages to use mathematics as a descriptive language for formulating control objectives. Hence, we show how to use quantifiers, relational operators, etc. to refine user-requirements and convert them into sound control objectives. In this way possible problems during the requirements definition phase easily pop-up and can be immediately corrected. Surprisingly, in the Author's experience, the formalization of a sound control problem is one of the most time-consuming phase in embedded software development.

Next, we observed that there is little or no work devoted to a careful study of the "environment" (process) behavior in V-model based Software Engineering, but almost all the focus is on the "system" (controller) development. Our aim is to induce a mindset shift such that the trainees understand the importance of analyzing the process behavior before initiating any controller design. In a V-model perspective this means to do the opposite of common practice. That is, we suggest to start from the right branch of the V-model (test) by running some open loop experiments and then go back to the left branch (design) to perform the controller design.

Finally, we invite the trainees to identify the four horsemen, namely the the states x, the input u, the disturbances d and the output y of the process along with the sets X, U, D and Y where they evolve and to use the outcome of this exercise to refine and adjust the control goals. We provide some simplified definitions to help the trainees with this task. For instance, we define the state x as the quantities that characterize the process behavior and that are of interest for the control objectives, the input u as the quantities that alter the state and that we can manipulate, the disturbance d as quantities that alter the state but we cannot manipulate (at most we can measure them) and the output as the part of the state that we can directly measure with some sensor or, more in general, a function of the state (and possibly of the input) that we want to control. Although these definitions may appear naive and incomplete, we observed that they are good enough for introducing these concepts.

We conclude this module by challenging the trainees to formalize control objectives and to identify the four horsemen on a number of prepared control problems that are typically connected to their business areas.

5.2 Signal processing

In the "What is Control Theory" definition given in the previous Section it appears the available information ingredient. We then provide some examples of bad decision making due to bad information through exploitation of everyday life examples, and we finally invite the trainees to come up with their own examples. At this point we provide highlights the about following topics

(1) Information provided as signals and signals representations: time and frequency domain,
(2) Acquisition of signals from the environment: sampling,
(3) Manipulation of signals: filters,
(4) Implementation of filters: discretization,
(5) Filters stability.

Information provided as signals and signals representations: time and frequency domain We show how the available information in embedded systems is essentially represented by signals. With the help of some computer animation we show how a signal can be decomposed into a finite or infinite sum or sine and cosine signals. The take away message is that a signal has two faces, one in time-domain and one in frequency-domain. Trainees understand that the Fourier transform is a tool used for switching from one representation into another.

Acquisition of signals from the environment: sampling By using Software Engineering wordings, we point out that the information from "the environment to the system" does not flow continuously but it is sampled. Although this is typically well-known, software developers tend to sample all the sensors and to schedule all the control functions running on the same platform at the same sampling rate regardless of the process time constants. As an example of drawbacks due to naïve sampling, we introduce the aliasing phenomenon with the support of some video. Then, we introduce the Nyquist criteria as a solution for the aliasing phenomenon.

Manipulation of signals: filters We start by showing how to clean a noisy sensor through a filter, thus obtaining better available information. Then, we generalize to the concept of filtering. We define filters as devices that transform one signal into another. Next, we propose the experiment of stimulating a filter with a sine wave signal at different frequencies and we invite the audience to annotate both gain and phase shift of the filter output signal in a table. We point out that this table can be viewed as a sort of data-sheet that tell us how the output signal of the filter in response to an input sine wave will be amplified (or attenuated) and phase-shifted at each frequency. The audience typically understands that the graphical representation of such a table are the famous Bode diagrams. The hook with mathematics is fairly easy: a filter can be nicely described by a complex number where the amplitude and the phase are parameterized by the frequency ω. At this point, the introduction of transfer functions becomes fairly easy.

For the sake of completeness, we also introduce filters in time-domain. This is done because sometimes we want to change the time behavior of a signal directly in the time-domain. We show examples of actuators that are sensitive to fast changes in their actuation requests. In these cases, we show how to smooth out the input signal to the actuators through smoothing filter.

Implementation of filters: discretization When looking at the software developed by the trainees, we usually identify a certain confusion in understanding continuous and discrete-time representations. We found easier to address the discretization problem starting from the time-domain. The take away message of this sub-module is that a filter representation can be continuous or discrete, and it can be both in time or frequency domain. We further explain that it is possible to move from one representation to another sometimes in an exact manner and some other times in an approximate manner by further explaining some classic discretization methods.

Filters stability We show some unstable filter response and then we challenge the trainees by asking if there is anybody who knows why that happens. Some with some memories of some University Control Theory course typically answer correctly. However, we won't provide any stability notion here. We only claim that to avoid output blow-up in front of a bounded input it is necessary that the filter exhibit some stability property. We show how to check if a filter output will blow-up or not both by looking at the poles of its transfer function. Mathematics is hooked through the Routh and Yuri methods.

This sub-module concludes the part devoted to the acquisition of the available information in the embedded systems world. Next, we address the problem of modeling.

5.3 System modeling

We start this module by recalling the experiment of stimulating a filter with sine wave signals at different frequencies and to annotate the output signal gain and phase-shift in a table. Then, we extend this idea to physical processes by asking to the audience who prevents us to e.g. feed a pump with a sinusoidal current signal at different frequencies and to annotate the pump out fluid mass-flow gain and phase-shift at each frequency of the input signal? Or, who prevents us to do the same for a valve by consider a PWM voltage signal as input and the valve position as output? The take away message is that we can perform such an experiment on a large number of physical processes and therefore many physical processes can be represented in terms of transfer functions.

In general, we define dynamical systems as systems with memory where the current state depends on what has been done in the past. By following the same approach as in Section 4, we further detail such a definition by introducing three ingredients: initial state, action taken and time and we hook them to mathematics by introducing the explicit form $x(t) = \varphi(x(t_0), u(t), t \quad t_0)$. As usual, we provide a number of practical examples where we connect various physical process behaviors to the three ingredients provided above and then to the explicit representation of dynamical systems. Another hook to mathematics here is interesting: we use ordinary differential equations (ODEs) of the form $\dot{x} = f(x, u)$ by stating that their solutions are indeed of the form $x(t) = \varphi(x(t_0), u(t), t \quad t_0)$.

We finally introduce the task of System Identification, namely the art of building control-oriented models from input-output data obtained from experiments. We further explain the meaning of white, grey and black-box modeling and when to use the one or the others. The practical problem used as lever for justifying the necessity of Design of Experiments resides in the high cost of laboratory experiments.

5.4 Controller Synthesis, calibration and test

In this Section we describe how we introduce the concept of controller synthesis, which represents the software that the trainees are supposed to develop in their daily job. We further introduce testing methods such as Model-in-the-loop (MIL), Hardware-in-the-loop (HIL), Software-in-the-loop (SIL) and field tests. Given that such activities represent almost the totality of their daily work, the trainees interest is typically very high.

Software developers have a strong tendency of jumping straight into coding without having enough insights of the process behavior or of the control objectives. Then, depending on the test results, the software is patched by following a trial-and-error approach. Such a patchwork is iterated until the closed-loop system behaves in an acceptable manner, but it also typically leads to an uncontrolled and disorganized software growth. Furthermore, the final software poorly reads and it is difficult to maintain and scale. Among others, typical control software developed by employees with little knowledge of Control Theory includes a plethora of if-then-else conditions, switches, state-machines and map-based controllers. It sometimes includes some PID controllers but very rarely it includes controllers in some state-space form or transfer functions. We address the following topics:

(1) Classic closed-loop specifications in Control Theory,
(2) Open-loop, feed-forward and feedback control and common techniques in the frequency domain,
(3) PID control,
(4) Control design in the time-domain,
(5) Controller implementation, calibration and test.

Classic closed-loop specifications in Control Theory The first task is to translate the formulated control objectives into classic closed-loop specifications. We introduce transient specifications such as rising-time, overshoot and settling time as well as steady state specification such as steady-state error and disturbance rejections. We further show typical specifications in frequency domain like bandwidth, gain margin, cross-over frequency, etc. how a change of a specification in one domain

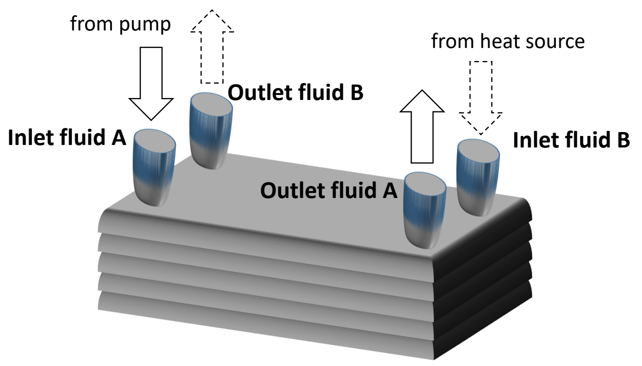

Fig. 2. Heat exchanger example: fluid A is pumped into the heat exchanger and it is warmed through an heat exchange process with fluid B which comes from an external heat of source. The control goal is to actuate the pump in such a way the temperature of the fluid A at the outlet of the exchanger is constant regardless of the fluid B state. Possible four horsemen are: $x = y =$ outlet temperature of fluid A, $u =$ inlet massflow of fluid A, $d =$ inlet temperature and massflow of fluid B.

is reflected by a change of a specification in the other domain and vice-versa.

Open-loop, feed-forward and feedback control The most common approach used by the trainees is to attempt to compensate the process behavior through open-loop control. This is typically done through maps, switches, if-then-else logic, etc. rather than considering some transfer function of the process. Here we re-present the same idea but in a Control theoretical setting where we cancel the process dynamics through exploitation of the the process transfer function. As usual, we show this control technique on some simple real-world example, possibly in the audience application domain, and then we generalize it saying that with this method the control input u is typically of the form $u = k(r)$, where r is the desired set-point.

Along the same line, we introduce only feed-forward control method in a colloquial way by claiming that the controller merely tries to counterbalance the effect of the disturbance on the process output by changing its input. With the help of some real-world example we formalize this method a bit more appropriately by claiming that the control input u is typically of the form $u = k(d)$. Based on some practical examples, we ask some target questions to the trainees with the aim of making the drawbacks this control strategy to emerge. In this way, the audience self-convince on the limitations of only feed-forward control.

While feed-forward control schemes attempt at anticipating the effects of a disturbance on the process output, feedback control schemes react only after something on the process out has been detected and it is therefore of the form $u = k(x)$. Hence, we refer to feed-forward as proactive type of control and feedback as reactive type of control, further discussing pros and cons of each approach. We finally point out that it is obviously possible to use a combination of the two approaches and therefore the control signal becomes of the form $u = k(x, d)$.

PID control Given that the majority of the trainees often deal with control problems that can be nicely solved with a simple PID controller, we pay particular attention to this submodule. We introduce the "king of controllers" by claiming that it merely does two things but it does them very well: firstly it makes the process output to tracks constant set-points; secondly it rejects constant disturbances from the process output. Through a number of real-world examples, the trainees easily realize how many engineering problems can be solved by the solely utilization of a simple PI controller. We further highlight that such a controller poorly loads the CPU and it has a very small memory footprint and therefore is very appealing for embedded software products.

Control in time-domain For the sake of completeness, we warn the audience that it is possible to design control strategies also in the time-domain. The first explained method is again by plant inversion. We use an explicit representation of a discrete-time process of the form $x(t) = \varphi(x(t), u(t), t \quad t_0)$. The explanation is easy: given the current state $x(t)$ and a desired value of $x^*(t+T)$ we compute $u(t)$ that steers $x(t)$ into $x^*(t+T)$. This is possible because we have the model φ. Then, we only introduce a couple of well-known control techniques in time-domain: the LQ control and the MPC. We solely explain the working principle of both by stating that the goal is to find a sequence of control actions $u(t), u(t+T), \ldots, u(t+(N \quad 1)T)$ that generates a state sequence $x(t+T), x(t+2T), \ldots, x(t+NT)$ such that a quadratic costs of input sequence the generated trajectories is minimized. We show the physical meaning of a quadratic cost. We finally remark that although the LQ problem is solved in closed-form, the MPC problem generally requires some iterative methods for finding the optimal solution. We conclude this section by stressing that LQ and MPC are not the only possible choices for feedback control design in time-domain, but there are many more.

Controller implementation, calibration and test Once a controller is designed, we need to implement, calibrate and test it.

The implementation depends on the specific software development platform used by a specific Company. Such platforms include programming languages, available hardware, software tools, etc. We remark that Control Theory generally addresses only the algorithmic part independently of the underlying platform. Hence, details like programming language, processor used, etc. are often left out the scope of Control Theory. This remark is important due to the confusion around the word "control engineer" in Industry as it may include personnel with little or no background on Control Theory.

Regarding the calibration and test, we divide testing activities in different layers that we denote as Model-in-the-loop (MIL), Software-in-the-loop (SIL), Hardware-in-the-loop (HIL) and field test. The purpose of MIL is to test a model of the controller closed in loop with a model of the process in a desktop environment. The SIL is the same as MIL but the model of the controller is replaced with the compiled code of the whole software, including other functions running on the same target platform. In this way it is possible to check the control strategy robustness with respect to aspects that have been left out during the design phase like signals quantization, sampling period, time-delays, co-existence with other software functions, etc. Eventual coding bugs like for example signals data type mismatch can be also identified and fixed during this phase.

The HIL consists in downloading the actual software on the target electronic unit and to verify if there are additional issues connected to that specific platform like stack overflow or CPU overload. Finally, during the field tests, the final product is verified against the user-requirements in the real environment. The calibration during the testing phases consists in adjusting the controller parameters to have a desired level performance. We explain the classic trade-off between closed-loop performance and robustness and how to adjust it through calibration.

We stress that the closed-loop system performance shall be measured through both control and software metrics. The former are used to measure how well the user-requirements are satisfied and they typically change with the application domain. For example, in the case of combustion engine control, we may define fuel consumption, emissions, delivered power and noise as control performance metrics. The software performance metrics typically don't change with the application domain and they are used to measure the controller complexity in terms of CPU load, memory footprint, maintainability, portability, scalability, etc. The defined control and software metrics are essentially the functional and non-functional requirements in Software Engineering.

5.5 Observers

This part is not always necessary for the development of embedded software products but it represent a central topic in Control Theory and it is also compelling in Industry. One trivial reason for developing observers is that there is a strong push to use as less sensors as possible in embedded products. This is not only due to the cost of the sensor itself but also because of its administration cost, its influence to final product price, the probability of failure and its influence to the final product packaging, just for mentioning some. In addition, there are sometimes quantities that we just can't measure like for the example the enthalpy variation of a fluid in a heat exchanger but they may represent information needed to the controller and therefore we need to estimate them.

With this in mind, the introduction of the observers becomes fairly easy. We only explain what is the basic idea behind the observers and we only introduce the Luenberger observer and the Kalman filter.

5.6 Complementary concepts

The aim of this module is to provide the trainees with additional mathematical tools to better understand Control Theory books and literature. Such tools include signals and systems norms, coordinate transformations methods and so on. We further provide additional control notions such equilibrium points along with their stability properties in addition to provide reachability and observability notions.

6. RESULTS AND DISCUSSION

The proposed course has been evaluated through anonymous questionnaire. We experienced a fairly high level of satisfaction in all the editions. The main message from the collected answers is that exploitation of a large number of practical problems ultimately clarified why certain topics such as step-responses, transfer functions and Bode diagrams are taught in under-graduated courses. A genuine appreciation towards the creative usage of mathematics as a support tool for analyzing engineering problems has also been reported by a surprisingly high number of attenders.

Although a questionnaire provides some relevant information about an education program, it does not allow to determine its actual effectiveness. We should also attempt at measuring how trainees apply the learned topics for creating business value. For this purpose, we observed the deliveries quality change before and after the proposed course and we further observed the trainees approach when facing new engineering problems. Fortunately both have been improved: average software quality has been increased and the attitude when facing new engineering problems became more rigorous.

Nevertheless, the rate of trainees who become proficient heavily depended on how much they self-learn by practicing on the provided simulation models and exercises. In turn, the time spent on self-learning depends on the available budget allocated for educational activities. That is, if the budget is limited then self-learning practice during work hours is hard to justify and therefore trainees have a natural tendency to down-prioritize it. Since most industries struggle with their budget, we believe that the ideal case would be to improve under-graduate programs to take into account Industry needs and hopefully this work contributes to the cause. That is, we believe that a stronger involvement of Industry in defining under-graduate programs would be even more effective for facilitating the utilization of Control Theory in the embedded software development Industry.

ACKNOWLEDGEMENTS

The Author thanks Chiara Rinaldi at the University of Gothenburg and Damiano Varagnolo at NTNU for the inspiring discussions on pedagogical topics in Control Engineering. We also thank the former student and colleague Ricard Blanc at Volvo Group for his precious feedback on this work.

REFERENCES

Abramovitch, D.Y. (2019). Introducing feedback control to middle and high school stem students, part 1: Basic concepts. In 12th IFAC Symposium on Advances in Control Education.

Bennet, S. (1996). A brief history of automatic control. IEEE Control Systems Magazine.

Densten, I. (2001). Re-thinking burnout. Journal of Organizational Behavior.

Fowler, K.R. and Silver, C.L. (2015). Developing and Managing Embedded Systems and Products. Newnes.

Pines, A. (1993). Burnout: an existential perspective. Taylor and Francis: Washington DC.

Rossiter, J.A. et al. (2018). A survey of good practice in control education. European Journal of Engineering Education, 43(6), 801–823.

Rossiter, J. et al. (2014). Opportunities and good practice in control education: a survey. In 19th IFAC World Congress.

Samad, T. (2017). A survey on industry impact and challenges thereof. IEEE Control Systems Magazine.

Sinek, S. (2009). Start with why: how great leaders inspire everyone to take action. New York, N.Y.