CONSIDERING USER'S IMPACT IN VALIDATION ACTIVITIES – AN APPROACH FOR THE DETERMINATION OF REQUIREMENTS

Pinner, Tobias; Jost, Franz; Schmid, Daniel; Albers, Albert
Karlsruhe Institute of Technology (KIT), Germany

Abstract

Validation activities use virtual and physical validation models within a validation setup. In practice, a human user influences the product's functions and the overall user experience. Certain validation setups consider this user-evoked influence. Therefore, a suitable user model is required, as well as an interface system that supplies the model based user input to the systems under validation.
This paper presents an approach to support the designer in the definition of requirements for interface systems that transform virtual user models into physical actions. A system's analysis approach supports the designer in identifying the relevant user input to a technical system on the test bench. A descriptive model for interface systems supports a common understanding of these systems and helps to raise their modularity and adaptability.
The manual gear-shifting process is the application example within this paper. A generic gear-shifting model is presented. This model is part of a shifting robot that transforms a virtual model input into the physical shifting action.

Keywords: Design methodology, Requirements, Product modelling, Mechatronics, Validation

Contact:
Prof. Dr.-Ing. Albert Albers
Karlsruhe Institute of Technology (KIT)
IPEK Institute of Product Engineering
Germany
albert.albers@kit.edu

1 INTRODUCTION

Validation activities are essential in product engineering processes as they serve as a means of ensuring the achievement of objectives (Albers, 2010). These activities use virtual and physical validation models within a specific validation setup. From the *Hardware in the Loop* approaches (cp. Bringmann and Krämer, 2008), the data transmission between simulated environment and physical prototypes are well-known. For further developed validation approaches (like *X-in-the-Loop*, see below), the simulated system flows are no longer just virtual information flows but physical energy flows, like the application of torque to a gearbox. *Interface systems* transform virtual model output from the virtual into the physical domain. The transformation of virtual human user behaviour into physical action is a specific application for certain interface systems. Unlike other systems, the system *User* comprises manifold, variable, probabilistic output that complicates the selection of relevant parameters for a virtual model. Thus, the determination of the relevance of user outputs is one important step for the definition of virtual user models. In addition, the specific interface systems that connect the virtual user models to the physical domain are essential hardware elements in such a validation setup. Their hardware requirements depend much on the specific virtual or physical models.
This paper presents an approach to support the designer in the definition of requirements for interface systems that transform virtual user input into physical actions within validation setups, using the manual gearshift process as an application example.

2 STATE OF THE ART

The following section presents the state of the art in the context of drivetrain validation, with a focus on the involved virtual and physical models. Furthermore, a modelling technique for system analysis is presented.

2.1 X-in-the-Loop Framework for Drivetrain Validation

In the context of product engineering, validation activities can be regarded as an interaction between parts of a system under development and of the system's environment. Coming from the area of embedded systems, typical validation approaches are *Model in the Loop MiL, Software in the Loop SiL, Processor in the Loop PiL* or *Hardware in the Loop HiL* (cp. Bringmann and Krämer, 2008). The *Loop* is established for the simulation of the embedded system's environment. Within these environments, the mentioned validation approaches integrate parts from the system under development (SUD) at different levels of maturity, following a model based development approach. MiL integrates a software model into an environmental simulation; after the real code implementation, a SiL simulation evaluates the code. This code can be deployed on a microprocessor for a PiL simulation; after integration on the target hardware, a HiL simulation can be conducted (for early discussion on HiL environments for powertrain development refer to Powell et al., 1998; Raman et al., 1999).

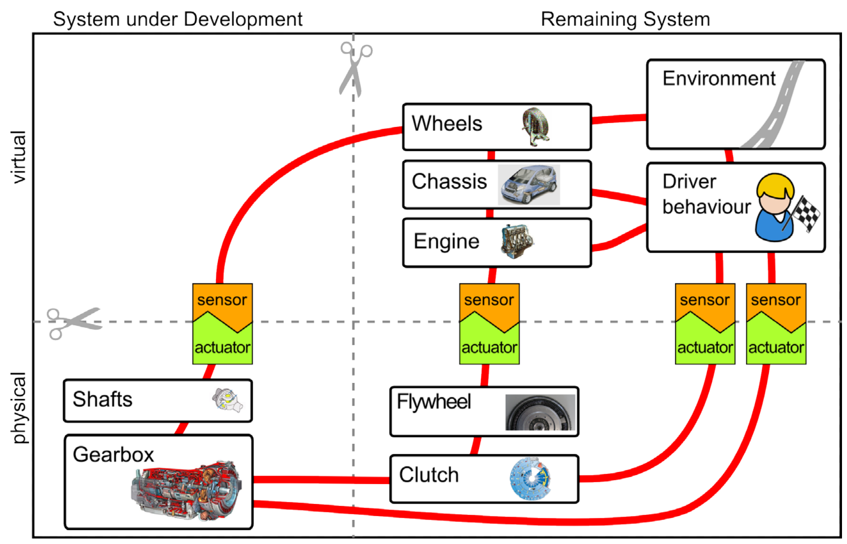

Figure 1. XiL arrangement for gearbox validation (cp. Albers and Pinner, 2013)

Based on these approaches, the X-in-the-Loop-Framework was developed (XiL-Framework, Albers et al., 2013). It is an approach to support validation activities in the area of drivetrain development, without focusing on embedded systems but considering concerns of mechanics and mechatronics. Mechanical systems, single software systems or even the environmental or user systems can be the SUD (Albers et al., 2008b). The XiL deals with models of the SUD, its environment and the driver (Remaining System); like in MiL, SiL, PiL or HiL the SUD models can be virtual or physical. However, even the SUD's environment and driver can both comprise virtual or physical models. Due to their origin from embedded systems, MiL, SiL, PiL and HiL had no need for physical models in their environment, whereas the drivetrain development does. Model Based Testing (MBT) methods that aim to facilitate and objectify test derivation, conduction and analysis (compare Pretschner, 2005; Bringmann and Krämer, 2008) remain compatible with the XiL.
One suitable descriptive model to depict a XiL validation setup is presented by Geier et al. (2012). It consists of a canvas showing one area for the SUD and the Remaining System (RS) each. Additionally, the areas are subdivided into virtual and physical domains. Figure 1 shows a model of a validation setup used on a powertrain-in-the-loop test bench. There, the gearbox and the shafts is the physical system under development. It is connected to the clutch and the flywheel, which are physical elements of the RS. The further drivetrain elements that are connected to the gearbox exist in the virtual domain.
Requirements on MiL, SiL, PiL and HiL setups arise predominantly on the virtual domain, concerning real time model execution or model maturity, and on interfacing between SUD and virtual environment, concerning data exchange and coupling (cp. Monti et al., 2005; Demers et al., 2007). XiL-framework related requirements include as well the physical domain with regard to the physical models of the SUD's environment. On the physical domain, the simulated system flows are no longer virtual (information flows like a calculated torque within a Simulink® model) but physical (energy flows like a mechanical torque on the shafts or the movement of the shifting lever). The transfer from virtual to physical domains is realised by *interface systems* that pass the virtual-physical boarder. Such a transition from information to energy flows is a novelty of the XiL-approach compared with the HiL-setups presented above, as for HiL the transition between the virtual remaining system and SUD affects only the SUD's data interface. XiL interface systems combine typically actuators that can manipulate the systems on the physical domain of the validation experiment with sensors that integrate physical feedback.
In the area of combustion engine development, the XiL-approach (cp. Engine-in-the-Loop) is followed as well (Bier et al., 2012; Shidore et al., 2011).

2.2 Considering User Behaviour

The *driver* is a central system in the X-in-the-Loop framework, representing one part of the SUD's environment. Driver behaviour that crosses the border from virtual to physical domains of the XiL setup needs an interface system as described in section 2.1. Even the use of physical driver behaviour within a validation setup usually crosses the virtual-physical borders: As the system under development is integrated on a test bench without a suitable human machine interface, a ride simulator can supply this interface to a real user. The acquired driver actions are then digitized and, hence, exist in the virtual domain before being transferred onto the physical domain.
For HiL, the consideration of driver behaviour affects the software-based implementation (e.g. for considering driving patterns or driver's behaviour in Hung et al., 2010). For XiL, the consideration of user behaviour can affect a software-based implementation of a user model and as well the transformation from virtual to physical flows. This transformation usually needs special hardware, like a clutch actuator or a gear shift robot. The specific design of such a system is driven by requirements that arise from the validation objectives, the SUD and environmental hardware and the virtual user model itself. The identification of user behaviour related requirements on an interface system and the user model can be supported by a purposeful analysis of the SUD.

2.3 System Modelling using the Contact-and-Channel Approach

One way to analyse physical models is the Contact and Channel Approach (C&C^2-A). This approach relates a system's embodiment to its technical functions (Albers et al., 2008a). This is one way to describe a network *(Wirk-Net)* of interfaces and physical structures that performs a function (Albers and Wintergerst, 2014). For modelling such a Wirk-Net, three model elements are used: The *Working*

Surface Pairs (WSP) represent interfaces between physical structures. *Channel and Support Structure* (CSS) are such physical structures that conduct system flows. *Connectors* (C) connect to the Working Surface on the system border and represent the effect on and from the system's environment for the functions under consideration (Albers and Wintergerst, 2014).

3 MOTIVATION AND FOCUS

As stated above, a XiL remaining system may integrate human models that represent certain properties and behaviours of the user. Some of these model related user flows might be transferred from the virtual to the physical domain, like a model-based shifting movement that affects the physical shifting lever of a gearbox. For these tasks, specific interface systems are needed. Methodological support exists for requirements determination and system implementation concerning the interface between MiL, SiL, PiL and HiL SUD's and their referring environments (Bringmann and Krämer, 2008; Demers et al., 2007). Contrary to this, the requirement determination for XiL specific interface systems that take account of the XiL features with regard to the virtual and/or physical user models is not yet supported methodologically and comprehensively.

This paper presents an excerpt of a methodological approach to support requirements determination for such XiL interface systems. Section 4 describes a method to identify possible user related system flows that can get to the SUD (4.1), based on a system analysis. Furthermore, a new descriptive model for XiL interface systems is presented (4.2). Along with this, generic requirements for XiL interface systems are derived. Section 5 presents the application of the methods from section 4 to the practical example (development of a gear shift robot).

4 METHODOLOGY AND DEVELOPED APPROACH

4.1 Establishing of the relevant user-system-interaction

A central question for selecting the right user influence in a validation setup is the relation between the possible user output in its entirety and the resulting input flows to the system under development (SUD). The following method supports the identification of system flows that get to the SUD. It bases on the C&C^2-approach and conducts an effects analysis along the transmission path between user input and SUD. User and SUD represent the connectors to this path. The (desired or undesired) function *flow transmission* within the path can be analysed by subdividing it into a set of CSS and WSP (see Figure 3 for an application example):

1. Definition of working surface pairs (WSP), channel and support structures (CSS) and connectors for modelling the transmission path. The human user is represented by one system connector comprising a comprehensive set of human output flows (e.g. force, movement, touching, gesture or airflow). The SUD is one possible second connector, respectively a concrete sub-system of the SUD, according to the necessary granularity.
2. The transmission properties of WSP and CSS are analysed and put into a supporting software tool. Questions guide the operator during this process (e.g. *"is the working surface bendable?", "are the working surfaces electrically isolating?").*
3. The relevant human output is selected. A predefined (but adjustable) dependency matrix ensures the consideration of hidden influences that arise from a selected output flow. E.g., a force related input arises from the exertion of movement, as no movement exists without force application.
4. Based on the defined properties of the transmission path (within the tool) from step 2, it is checked automatically whether the user emitted system flows are transmitted between the system elements or not. The ceasing system flows are presented at the position of the system path that causes this ceasing.
5. Although a system flow stops at a certain positon, it can transform into (or evoke) another flow. Therefore, several questions support the developer in identifying this transformation (e.g. *"does the system flow cause a system deformation?").* A possible transformation could be rising temperature and force because of a blocked airflow with pressurisation.
6. All user output flows are listed that can get to the SUD. Every one of these flows (its intensity or dynamics) may be influenced by other human outputs. For example, sweat influences the force flow's dynamics, as slippery fingers affect the force transmission from the user's hand to the transmission path.

7. As an option, the system's connector (SUD in Figure 3) can be shifted along the transmission path prior to the evaluation in step 8. This helps to reuse an already modelled transmission path for varying validation objectives and various SUDs. With this, the user's effect on sub-systems (and not only one single SUD) becomes visible, which enables a flexible adaption of the effects analysis in retrospect.
8. For evaluation and as a summary, the ceased or transformed user system flows are presented and assigned to the corresponding WSP and CSS. This raises the method's transparency and enables the evaluation and retracing.

As a software tool, an interactive spreadsheet supports the application of this method, based on the spreadsheet application Microsoft Excel. The method helps to analyse a technical system concerning its sensitivity related to user influence. This knowledge is a base for developing specific user models that integrate into interface systems (cp. Figure 2) and user related requirements for validation purposes.

4.2 Development of Requirements for Virtual-Physical-Interface-Systems (VPIS)

The following approach focuses on systems that transform a virtual user model input into a physical output. For these virtual-physical-interface-systems (VPIS), requirements arise from various sources. Some requirements do not depend on the specific validation task and are generally valid for every VPIS. In contrast, some requirements interact deeply with the user model that is one source of specific and variable requirements.

4.2.1 A Descriptive Model for Interface Systems

In order to clarify the various requirements on VPIS and to raise their transparency, the *VPIS model* is introduced. This is a descriptive, conceptual model of the relevant VPIS structure that is affected by transforming the model-based, virtual user behaviour into physical action. In this way, the VPIS model is an abstract representation of a XiL interface system that realises the transformation between virtual and physical XiL domains.

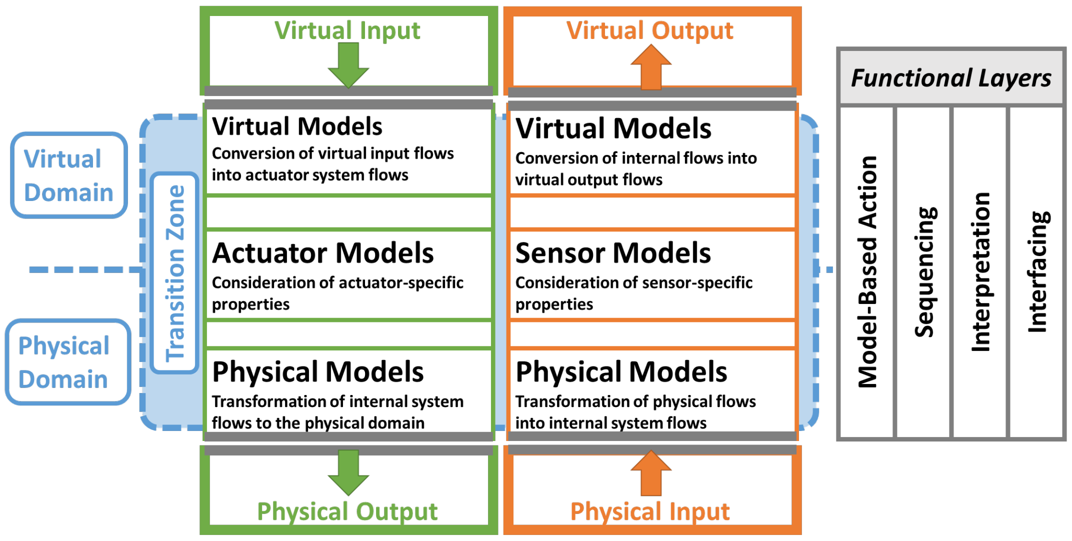
Virtual Input
Virtual Output
Functional Layers
Virtual Domain
Transition Zone
Physical Domain
Virtual Models
Conversion of virtual input flows into actuator system flows
Actuator Models
Consideration of actuator-specific properties
Physical Models
Transformation of internal system flows to the physical domain
Virtual Models
Conversion of internal flows into virtual output flows
Sensor Models
Consideration of sensor-specific properties
Physical Models
Transformation of physical flows into internal system flows
Model-Based Action
Sequencing
Interpretation
Interfacing
Physical Output
Physical Input

Figure 2. VPIS model with two transmission paths: Virtual input to physical output (left) and physical input to virtual output (middle). Four abstraction layers of VPIS functions (right)

Figure 2 shows the border between physical and virtual domains as seen in Figure 1. A VPIS is shown in the transition zone within the border, supplying connections to both the virtual and the physical parts of a XiL setup. There are two main paths for virtual or physical system flows: The conversion of virtual input to physical output (left side) and the conversion of physical input to virtual output (middle). For both paths, three relevant models are presented that conduct the flow conversion.

For the first path (virtual to physical): The *virtual models* serve for a model-based conversion of the virtual input to VPIS internal system flows. Such a model can be used to subdivide a complex user task (like "shift gear with characteristic 1") into application oriented parts (like "apply a force of 15N within shifting phase 2"). Such models can also exist unattached from a VPIS, whereas the models

shown here belong to the VPIS' design space. The *actuator models* take the physical structure of a VPIS into account. They incorporate the system kinematics, gear ratios or coordinate transformations. Furthermore, they calculate flows needed from the specific hardware, like current or voltage. The *physical models* are the hardware that transforms these flows into physical action. They are the physical representation of the actuator models, like a motor with a gearbox or a robotic manipulator.
For the second path (physical to virtual): The *physical models* are the hardware (e.g. sensors with specific connection) that transforms physical flows, like movement or force, into internal system flows, like voltage or current. The *sensor models* are used to convert these flows, considering the sensor's properties, their transfer behaviour or system kinematics. The *virtual models* interpret several values (like "force" or "travel") to a resulting virtual output (like "gearshift completed").
Additionally, the VPIS model comprises four abstraction layers of functions (see Figure 2, right) that interact with their neighbouring layers (a well-known approach based on abstraction layers is the OSI-Model, see (ISO/IEC 7498-1, 1994)). This is another view on a VPIS, with a more hardware-oriented focus. Four layers of the VPIS are suggested as follows:

1. **Interfacing layer:** This layer contains the real connections to the VIPS surrounding. For the virtual inputs, this may be a data connection plug that connects to an analogue or digital input signal. For the physical input, this could be a gripper or a mechanical coupling that connects to the physical systems of the validation setup.
2. **Interpretation layer:** This layer implements the conversion between the raw information input and the processed information that is prepared for further processing within the model-based actuation layer. For the virtual input flows, this is e.g. a protocol interpretation and a conversion into the needed physical quantity. For the physical inputs, this is the transformation of physical (e.g. force, speed, temperature) input into measures and their transformation into computable (i.e. digital) values.
3. **Sequencing layer:** This layer arranges and regulates the overall virtual-physical transformation process. It creates a chronological (but not functional) connection between virtual and physical in- and outputs.
4. **Model-Based Actuation layer:** This layer contains the model-based, comprehensive, functional link between virtual and physical in- and outputs. It contains e.g. mathematical models that map the physical output to the virtual input and vice versa.

4.2.2 Generic Requirements for VPIS

A specific VPIS embeds into a validation setup with other VPIS, virtual and physical models and the interface to a superordinate controlling system. From these general boundary conditions of VPIS use and application, five general requirements arise for the components of the abstraction layers and the models of both paths from Figure 2:

- *Polymorphism:* Different validation questions and different levels of abstraction concerning the validation models lead to specific and varying characteristics of VPIS. To operate under these different boundary conditions, a VPIS must implement multiple alternative operation strategies that cope with the variance in supplied input and expected output. This addresses the adaption to a variety of test cases, to different validation models, to different validation setups with different SUDs and to different users. This affects mainly the interfacing layer and parts of the virtual models.
- *Modularity:* In order to react to substantial changes within the validation environment (e.g. entire new products or models), the layout of VPIS must be modular. Thus, existing (sub)-systems of the VPIS can be exchanged separately. This affects signal connections and protocols, interfaces to the physical domain or the central conversion model between physical and virtual domains. Each functional layer acts as an independent subsystem of the whole VPIS and, thus, separate layers can be exchanged without the need for adaption in other layers.
- *Connectivity:* The VPIS is no standalone system but it integrates into a surrounding validation environment. Therefore, the VPIS must provide the right level of connectivity to both the models and other VPIS and the overall framework. This affects plugs and interfaces, as well as protocols, processes and models. The connectivity between VPIS is realised between similar layers of different VPIS. Interface components ensure the physical connectivity between different VPIS and between a VPIS and the validation environment. These elements have to be physically compatible. The further layers enable the connectivity on higher levels of abstraction, like

protocols or measurement units on the interpretation layer, or chronologically balanced connectivity on the sequencing layer and model compatibility on the model layers.

- *Interoperability:* The interoperability requirement is based on the request for connectivity. Multiple VPIS can interact with one another and cooperatively realise a virtual physical conversion with a mutual user model in the background. This requirement necessitates a common understanding of user behaviour between the involved VPIS and a suitable coordination of their processes.
- *Transparency:* To consider all of the requirements mentioned before, there is a crucial need for transparency and openness within the VPIS. Without the knowledge about the applied models or the meaning of parameterisation options, there is no chance to polymorphism or interoperability. Without the knowledge about internal or external interfaces, the demand for modularity and connectivity cannot be fulfilled.

5 APPLICATION FOR THE DESIGN OF A GEAR SHIFT ROBOT

The manual gear change in a car is an interactive process between user and gearbox. The realisation of this process on the test bench requires an adequate user model and a suitable VPIS that transform this model based, virtual user input into physical output. This section presents an analysis of the driver-gearbox-interaction in order to identify relevant system flows for the definition of virtual models (5.1) and a developed virtual gear shifting model (5.2). Furthermore, requirements (5.3) and realization (5.4) of the gear shift robot are presented.

5.1 Establishing of relevant driver-gearbox-interaction

With the method for the identification of relevant user input (cp. 4.1) the possible human influence on the gearbox has been studied.

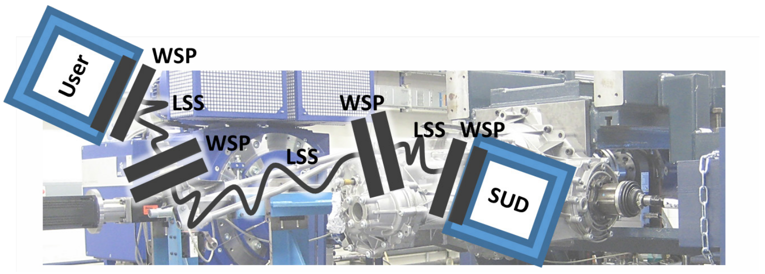

Figure 3. User and SUD as connectors to the shifting transmission path, with C&C-A

The system connector is set at the shifting fork inside the gearbox. Table 1 represents an excerpt of the possible human output flows. Every starred (*) system flow reaches the subsystem under consideration; the others end or transform into another flow.

Table 1. Relevant human output flows (excerpt)

No.	Human Flows	Source	Further influenced by (excerpt)
*1	Exertion of force	Direct	Sweat, fluids, age, health, gender
*2	Guiding	Direct	Force, sweat
3	Gripping	Direct	Sweat
4	Swiping	Direct	
5	Touching	Direct	
*6	Temperature flow	Indirect	Force, sweat
*7	Conductivity	Indirect	Sweat, fluids
*8	Movement	Transformed	

The column 'source' shows whether the flow was selected directly for analysis (i.e. from the user), indirectly (i.e. from a dependency table) or if it has been transformed (within the system, from a transformation table). On the synchronisation ring, the exertion of force (#1), the guiding of the lever (#2) and movement (#8) are relevant. Although the human flows #6 and #7 may arrive the synchronization ring, a further physical analysis shows their negligibility by considering the distance to the ring and the used material.

As a result, the development of a human gearshift model should concentrate on the exertion of force, the lever's guiding and its movement, and dynamic variation of these inputs.

5.2 Development of a generic shifting model as a virtual model for VPIS

For the generic gearshift model (i.e. virtual model from Figure 2) the mechanical (i.e. force and travel related) user influence on the gearbox according to the conclusion from the method application in section 5.1. is focussed. To create a generic user model, the shifting process is subdivided into five sequences, corresponding to theoretical findings and prominent gradients within the shifting force and lever position distribution. These sequences are analysed concerning user related mechanical influence on the shifting lever and the findings are transformed into model description and behaviour.

Figure 4 (left) depicts the force based generic user model derived from a literature review and from own measurements with a modified shifting lever that records shifting forces and positions in a car. The model defines parameters that are used to modify the model behaviour in order to realize the polymorphism requirement presented in section 4.2. With the model, the required physical and virtual input flows are determined, which have to be provided along the virtual and physical system flows:

- *Phase I:* The shifting lever moves from its starting position to the beginning of synchronisation. Therefore, it must be grabbed from its detached position that allows the lever to vibrate freely during its engaged status. The end of phase I is defined with a threshold value for the force *F_thresh*. Further parameters are the maximum speed *v_max* and a target force *F_target*. These values are variable within the model (polymorphism) and need recording, transforming and embedding into the computation process along the physical domain of all VPIS layers.
- *Phase II:* After finishing phase I, the target force *F_target* is set to a desired value that represents the maximum force for the synchronisation process. To ensure a smooth transition between phases I and II, the parameter *F'_max* limits the time derivative of the actual force. The adaptability of *F'_max* support the polymorphic approach of VPIS by supplying a model parameter that represents different types of force application during the synchronization process (i.e. different kinds of drivers).
- *Phase III:* After reaching target force of phase II, the actual force keeps a constant value. After the full engagement of the synchronization partners, the difference between actual force and target force grows as the actual force diminishes. Exceeding a threshold value *ΔF_max* for this difference finishes phase III.
- *Phase IV:* After the engagement, the shifting lever moves into its final position. Therefore, a new target force *F_target* is defined. Exceeding a new threshold value *F_thresh* identifies that the lever reaches its target position.
- *Phase V:* This phase considers different approaches of finishing the shifting process. The lever can be pushed slightly beyond the end position, keeping the target force of phase IV. Furthermore, the lever can be released using a zero force control (*F_target=0*), both with or without pushing the lever beyond the end position.

The gearshift model presented above concerns the manually driven shifting process without taking further elements (like clutch or throttle control) into consideration. Hence, the affected VPIS has no need for interoperability with other VPIS but with the test bench (providing process or model information) and the physical gearbox system.

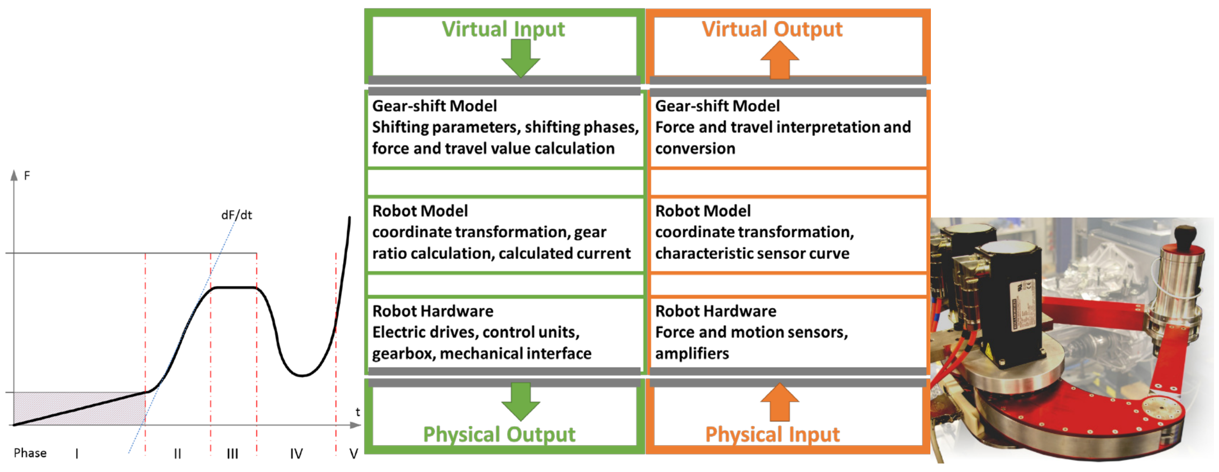
F
dF/dt
t
Phase I II III IV V
Virtual Input
Gear-shift Model
Shifting parameters, shifting phases, force and travel value calculation
Robot Model
coordinate transformation, gear ratio calculation, calculated current
Robot Hardware
Electric drives, control units, gearbox, mechanical interface
Physical Output
Virtual Output
Gear-shift Model
Force and travel interpretation and conversion
Robot Model
coordinate transformation, characteristic sensor curve
Robot Hardware
Force and motion sensors, amplifiers
Physical Input

Figure 4. Generic gear-shifting model (left), VPIS model (middle) and physical representation (right) of the shifting robot

5.3 Excerpt of requirements for a humanlike gear shift robot

The following requirements are directly related to the developed generic gearshift model. It is only the human model related subset of all requirements that apply to the development of a shifting robot as a VPIS. Further requirements arise directly from the validation environment, from the validation models available and from direct customer request.

General requirements that affect multiple VPIS model layers: The control loop, the physical value acquisition and conversion must run in a cycle frequency that suits the human motor control and sensing characteristics. This affects all related elements on every VPIS layer (mechanical interface, interpretation, sequencing, data interface, protocol).

Requirements affecting the model layer of the VPIS model: Human model's must provide the following input parameters: Target shifting force, force thresholds and maximum speeds for different phases, force gradient for phase II, desired behaviour for phase V. Furthermore, the test environment must provide the desired target gear for engagement.

Requirements affecting the sequencing layer of the VPIS model: Implementation of a force based motion controller, like an admittance control model, to enable the definition of target forces.

Requirements affecting the interpretation layer of the VPIS model: Acquisition and conversion of the following shifting lever's measures: Shifting force, lever speed, time derivative of the force as an input from the physical domain into the VPIS.

Requirements affecting the interfacing layer of the VPIS model: Supply of a physical connection to the actual shifting lever and not directly to the gearbox as the model focuses on the shifting lever as the interface to the physical domain.

5.4 Mechanical, Electrical and Software Realisation

The realisation of a gear shift robot shows the transfer of the general requirements from section 4.2 into practice.

Transparency reasons were the main cause for the decision to develop an all-new gear shift robot and not to buy a commercially available one. Only an in-house development ensures the full knowledge of the mechanical, electrical and software structure that is necessary to fully adapt or modify a VPIS to meet new research requirements.

The *connectivity* requirement affects every VPIS layer on the robot: The communication protocol between robot and test bench is CAN. Additionally, the control box connects to the network to enable remote maintenance. A common database file (CAN dbc) defines the contents of the transmitted data and integrates into the test bench software. The mechanical connectivity realises a shifting lever connector that realises position and force transmission.

This shifting lever connector consists of a bushing that connects to a ball, both of circular profile. The ball replaces the gearbox specific lever geometry with a universal adapter to support the system's *modularity*. As well as for modularity, the motion control loop is subdivided into two sub systems, the logic controller and the motion controller. The logic controller interprets the input tasks, calculates motion paths, and selects suitable motion controllers. The motion controller sets the robots

positions according to the provided motion paths, using suitable motion control loops (like PID control or admittance control). This subdivision supports the system's *modularity* as new logic controllers can easily be integrated without needing new motion controllers. In this way, the robot can be adapted to new shifting paths (e.g. automatic transmission) or new shifting behaviours.
Up to now, multiple logic and motion controllers have been implemented into the robot supporting the *polymorphic* approach. The standard logic controller implements an H-shaped shifting geometry; further variants differ in the selection of motion controllers.

6 SUMMARY AND OUTLOOK

The VPIS model is introduced as new way to coordinate requirements that are directly related with the model based transformation from the virtual into the physical domains. Furthermore, it allows a mapping between requirements and the corresponding system properties. The model's structure, the involved models, the division into functional layers and the knowledge about the layer's interdependencies are meant for supporting the understanding of requirements engineering for interface systems. The partitioning into functional VPIS layers simplifies the identification of relevant connections and reduces complexity in realisation by encapsulation and separation between the abstraction layers. The encapsulation of system layers eases the *polymorphic* design of VPIS. The clarification and knowledge of system flows and deliverables between each VPIS layer supports their *modularity*. Documenting and classification of VPIS elements to the VPIS model raise the *transparency* and support the exploitation of the model's advantages.
Furthermore, an approach for identifying possible user related system flows is presented that can be conducted prior or during the development of VPIS.
Up to now, the VPIS model has shown its applicability regarding the development of a gear shift robot. The model aims to support the definition of clear interfaces between the involved elements (mechanics, electronics and software) in order to raise modularity and adaptability. Future work will be carried out concerning the quantification of the approaches' practical use and their further evaluation.

REFERENCES

Albers, A., 2010. Five Hypotheses about Engineering Processes and their Consequences, in: Horváth, I., Mandorli, F., Rusák, Z. (Eds.), Proceedings of the Eighth International Symposium on Tools and Methods of Competitive Engineering - TMCE 2010. Delft University of Technology, Delft, pp. 343–356.

Albers, A., Alink, T., Matthiesen, S., Thau, S., 2008a. Support of System Analyses and Improvement in Industrial Design through the Contact & Channel Model, in: Marjanovic, D., Storga, M., Pavkovic, N., Bojcetic, N. (Eds.), Proceedings of the DESIGN 2008. The Design Society, Dubrovnik, Croatia, pp. 245–252.

Albers, A., Behrendt, M., Schroeter, J., Ott, S., Klingler, S., 2013. X-in-the-Loop: A Framework for Supporting Central Engineering Activities and Contracting Complexity in Product Engineering Processes, in: Proceedings of the International Conference on Engineering Design, ICED 2013.

Albers, A., Düser, T., Ott, S., 2008b. X-in-the-loop als integrierte Entwicklungsumgebung von komplexen Antriebsystemen, in: 8. Tagung Hardware-in-the-Loop-Simulation. Haus der Technik.

Albers, A., Pinner, T., 2013. Schaltroboter schaltet im automatisierten Prüfstandbetrieb wie ein Mensch. Newsl. Wiss. Ges. Für Produktentwicklung WiGeP 15–16.

Albers, A., Wintergerst, E., 2014. The Contact and Channel Approach (C&C2-A): Relating a System's Physical Structure to Its Functionality, in: Chakrabarti, A., Blessing, L.T.M. (Eds.), An Anthology of Theories and Models of Design. Springer London, London, pp. 151–171.

Bier, M., Buch, D., Kluin, M., Beidl, C., 2012. Entwicklung und Optimierung von Hybridantrieben am X-in-the-Loop-Motorenprüfstand. MTZ 03/12.

Bringmann, E., Krämer, A., 2008. Model-Based Testing of Automotive Systems, in: Proceedings of the International Conference on Software Testing, Verification, and Validation. Presented at the International Conference on Software Testing, Verification, and Validation, IEEE, Lillehammer, Norway, pp. 485–493.

Demers, S., Gopalakrishnan, P., Kant, L., 2007. A Generic Solution to Software-in-the-Loop. Presented at the Military Communications Conference, IEEE, Orlando, FL, USA, pp. 1–6.

Geier, M., Jäger, S., Stier, C., Albers, A., ASME Verification and Validation Symposium, 2012. Combined real and virtual domain product validation using top-down strategies.

Hung, Y.-H., Wu, C.-H., Lo, S.-M., Chen, B.-R., Wu, E.-I., Chen, P.-Y., 2010. Development of a hardware in-the-loop platform for plug-in hybrid electric vehicles. Presented at the International Symposium on Computer Communication Control and Automation (3CA), IEEE, Tainan, pp. 45–48.

ISO/IEC JTC 1 ISO/IEC Joint Technical Commitee for Information Technology, 1994. ISO/IEC 7498-1:1994 Information technology -- Open Systems Interconnection - Basic Reference Model: The Basic Model.

Monti, A., Figueroa, H., Lentijo, S., Wu, X., Dougal, R., 2005. Interface issues in hardware-in-the-loop simulation. Presented at the Electric Ship Technologies Symposium, IEEE, pp. 39–45.

Powell, B.K., Sureshbabu, N., Bailey, K.E., Dunn, M.T., 1998. Hardware-in-the-loop vehicle and powertrain analysis and control design issues, in: Proceedings of the American Control Conference 1998. Presented at the American Control Conference, IEEE, Philadelphia, PA, USA, pp. 483–492 vol.1.

Pretschner, A., 2005. Model-Based Testing in Practice, in: Fitzgerald, J., Hayes, I., Tarlecki, A. (Eds.), FM 2005: Formal Methods, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 537–541.

Raman, S., Sivashankar, N., Milam, W., Stuart, W., Nabi, S., 1999. Design and implementation of HIL simulators for powertrain control system software development, in: Proceedings of the American Control Conference 1999. Presented at the American Control Conference, IEEE, San Diego, CA, USA, pp. 709–713.

Shidore, N., Ickes, A., Wallner, T., Rousseau, A., Ehsani, M., 2011. Evaluation of ethanol blends for PHEVs using engine-in-the-loop. Presented at the Vehicle Power and Propulsion Conference, IEEE, Chicago, IL, USA, pp. 1–8.