SYSTEM BEHAVIOUR PREDICTION BY MODELLING AND SIMULATION

Modern product design requires a structured systems engineering approach, such as that characterised by the V-model. Different phases represent different levels of abstraction and detail, and modelling is essential for moving from one phase to the next. Iterations between phases are also necessary in order to arrive at a feasible and cost-effective design. What is an effective way to apply modelling during product design? To what extent will different models be applied during the various phases of product development? The first part of this article elaborates on the use of modelling during the various phases. In order to illustrate this, the content and scope for each phase are explained. It is emphasised that the extent of the requirements engineering phase is often underestimated because it comprises not only the specifications, but also the clarification of the design scope and interfaces. In the second part, the practical application of using such a model is illustrated by the implementation of the design for a linear motion system.

SVEN HOL

Product design

Modern product design requires the application of systems engineering, which is characterised by a high complexity and multidisciplinary interactions. The traditional scope of engineering comprises concept generation, design, development, production, and operation of physical systems. In that sense, systems engineering is part of this scope. But where, however, does systems engineering go beyond this?

First of all, systems engineering focuses on analysing and eliciting customer needs and required functionality in the early stage of the development cycle, then documenting the requirements and finally proceeding with design synthesis and system verification, all while considering the complete problem, the system lifecycle. This includes understanding and aligning to all the involved stake-holders. As well as customer involvement, the input from marketing teams, customer support engineers and product lifecycle analysts should be part of the focus of systems engineering [1].

Secondly, system development often requires input from multiple technical disciplines, such as mechanical engineering, electrical engineering, applied physics, control engineering and software engineering. By providing a holistic view of the development effort, systems engineering helps to unify all of the technical contributions and balances the trade-offs between cost, schedule and performance.

Finally, a systems engineering approach enables the design of systems with a high degree of complexity. Complexity here can be understood as technically challenging requirements, complex or poorly predictable environmental interactions, or complex interaction between various system components. An effective way to approach such systems is to apply a proper system decomposition, where the entire system is subdivided into smaller functional building blocks, or sub-systems, as in a jigsaw puzzle.

Structured approach

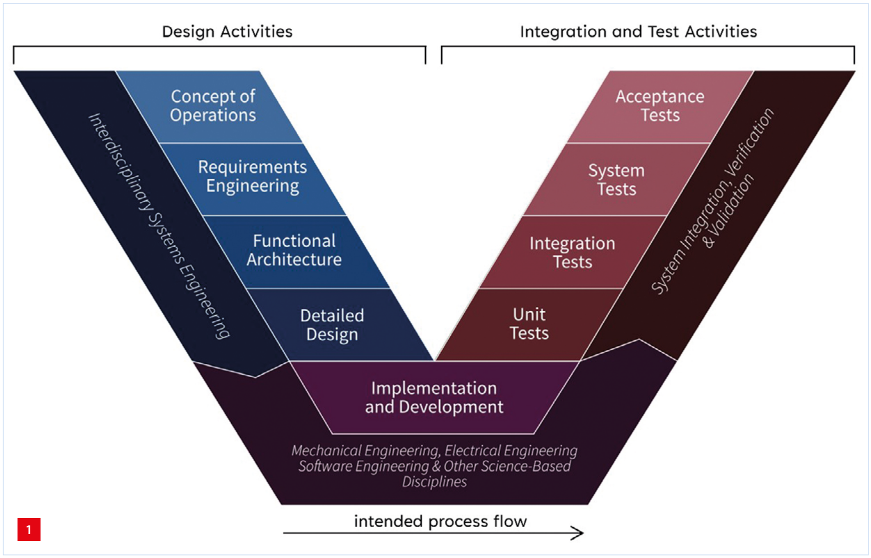
The V-model, which is used as a process guide in systems engineering, is shown in Figure 1 [2]. Multiple versions of the V-model exist: the version presented in Figure 1 will be used in this article as it defines clear and non-overlapping phases suitable for a multidisciplinary approach. The intended process flow is indicated on the horizontal axis, starting with the *Concept of Operations*, in which the high-level definition of the product is determined. Answers need to be found for questions such as: what is the product used for; how is the product used; what possible interaction exists between this and other products; why is the customer requesting this product; and will the customer's real need be satisfied with this product?

The second step, *Requirements Engineering*, consists of the definition of specifications [3]. In general, it starts on a high

AUTHOR'S NOTE

Sven Hol holds a Ph.D. degree in mechatronic engineering (2004) from Eindhoven University of Technology (TU/e). He obtained his M.Sc. degree in mechanical engineering (1996) at the University of Twente and he finished his master of technological design at TU/e (1999).
He has worked in various research and development functions at ASML and as an assistant professor at TU/e, and is currently senior system engineer at Demcon, responsible for product development in various technical domains.
He is also a lecturer at Mikrocentrum for the Applied Mechatronics training course.
During his career, he has obtained more than 70 patents in the field of mechanical and electro-mechanical applications.

sven.hol@demcon.com



Design Activities
Integration and Test Activities
Concept of Operations
Requirements Engineering
Functional Architecture
Detailed Design
Implementation and Development
Unit Tests
Integration Tests
System Tests
Acceptance Tests
Interdisciplinary Systems Engineering
System Integration, Verification & Validation
Mechanical Engineering, Electrical Engineering Software Engineering & Other Science-Based Disciplines
intended process flow

1 V-model as process guide in systems engineering. [2]

and functional level, specifying the scope (what is and what is not part of the system's function) and the action(s) the product should perform by addressing parameters such as: the product should be able to perform actions within a limited amount of time; it should be accurate; users should be able to operate it via a predefined user interface; and it should cost less than 10,000 euros.

During the *Requirements Engineering* phase, all relevant aspects of the entire product lifecycle will be clarified. More specifically, this will include issues such as: it must not be harmful to the environment; commissioning must not exceed one week; it should operate for at least two years without maintenance; system downtime must not exceed five hours; service action should only take three hours maximum; and decommissioning will occur according to a predefined plan.

When a system functional decomposition has taken place, the functional requirements of the sub-components will be derived hierarchically from the system level. In essence, what is known as the budget breakdown, which specifies the contribution of performance-limiting factors among the sub-systems, starts from here.

Another aspect of the *Requirements Engineering* phase is the definition of the system's interfaces by identifying its boundaries. How does the system interact with its environment, e.g. user interface, control signals to other systems, facilities (supplies of fluids, gases and energy)? In a similar way, the interfaces between sub-systems are consolidated.

During the next phase, *Functional Architecture*, the functional requirements are translated into concept designs. In this creative process, tangible design proposals are developed. Often the proposals comprise rudimentary sketches or geometry models, which still lack a high degree of detail. During this phase, design conflicts between the sub-modules become clearer and might require a readjustment of the budget breakdown as stated in the *Requirements Engineering* phase. Detailed knowledge of the relevant physics is vital for feasibility calculations: will the sub-systems be able to perform their intended functions properly? How large and heavy will they become? Will they survive?

In the *Detailed Design* step, the concept designs are worked out in detail so that production can take place. This step includes detailed drawings, CAD models, material choices, mechanical and electrical interfaces, software routines, work instructions, and user and service manuals. During the detailing phase, it might happen that an iteration back to the *Functional Architecture* needs to take place; for example, when implementation in a detailed design turns out to be too cost-intensive or even impossible.

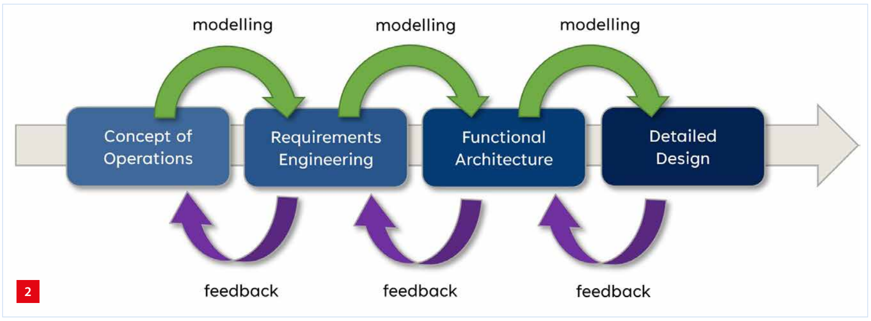
After *Implementation and Development*, where the actual production takes place, the *Integration, Verification and Validation* phases start, where the system is composed step by step by integrating and testing the ever-growing components and sub-systems leading towards the entire product.

Iteration

The left part of the V-model as illustrated in Figure 1 is addressed here as a design phase. As was demonstrated in the previous section, the design progression occurs in a downwards direction in that model. This progression is also depicted in Figure 2, illustrating the possible iteration loops.

Generally speaking, when progress in a phase is impossible, either by failing technical feasibility, exploding costs, safety issues or environmental effects, the iteration by feedback is required. When, for example, an intended solution in the *Detailed Design* phase becomes too expensive, we need to reconsider the different concepts previously found in the *Functional Architecture* phase.

If it happens that no other feasible concepts can be found, we need to reconsider the requirements as consolidated in the *Requirements Engineering* phase. Can we achieve



2 Iteration loops in the systems engineering process.

the customer's need with a different set of requirements? For example, imagine a case where a high production rate must be achieved. Unfortunately, several concept iterations did not lead to a feasible design. The question for the customer is then whether the specified production rate could still be achieved if other systems, currently out of our scope, were accelerated? In this case, the requirement for the production rate for our system could be relaxed, opening up possibilities for feasible design concepts.

Finally, if the requirements cannot be stretched or redistributed among several systems, then the customer should reconsider the concept of operation. Will different technology be able to overcome the limitations of the previously selected technology? For example, if the energy demand for a new isotope irradiation process becomes too high, it might be beneficial to stick to the conventional process or come up with another potential process with which isotopes could be irradiated.

The feedback loops mentioned above describe the cases for iteration. If the feasibility or design steps in a phase fail, a solution can be found in a previous phase. However, the intended forward-flow direction is accomplished by modelling. The modelling approach for each phase is different, as the levels of application are different. More specifically, to move from the *Requirements Engineering* to the *Functional Architecture* phase requires a different type of modelling than when moving from the *Functional Architecture* to the *Detailed Design* phase, as will be illustrated below.

System behaviour by modelling

The four phases in the engineering process as illustrated by Figure 2 are characterised by different levels of abstraction and detail. For example, a manufacturer of integrated circuits is considering establishing a production facility for integrated circuits for the electronics industry. In the *Concepts of Operations* phase, this customer will identify the required steps in the production process: wafer processing and oxidation, photolithography, etching, deposition and ion implementation, metal wiring, and packaging [4].

They will start with a marketing prediction of the required production rates by using estimates and marketing models for the demand for integrated circuits in the various sectors of the industry they intend to supply. They will set up models of the product flow with estimates on achievable production rates (bottom up) and specify minimum required production rates that guarantee a certain turnover that would cover the investment costs with sufficient margin (top down). The definition of the subsequent production steps, the testing and the logistics steps, describes their *Concept of Operations*: how will the integrated circuits be made, tested and distributed?

Once having identified the required production, testing and logistics steps, the requirements can be formulated. The overall specified product (i.e. size, accuracy, costs) and production demands (i.e. throughput, costs) should be translated into requirements for the intended sub-systems of the production process (oxidation, lithography, deposition station, wiring, packaging). Answers have to be found for questions such as to what extent will these sub-systems contribute to the product and production requirements? Throughput models and supplier information on sub-system capabilities will serve as input for sub-budgets on the main requirements, which will include such questions as what should each sub-system cost, and how quickly and accurately should it operate? These types of predictions and models will be used to finalise the *Requirements Engineering* phase.

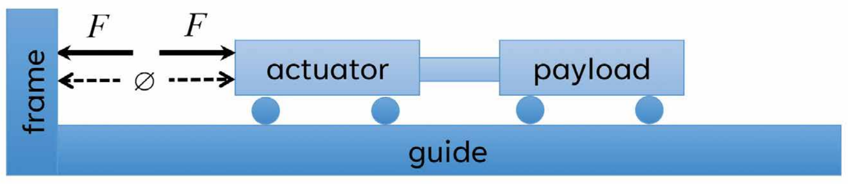
In the *Functional Architecture* phase, the different options for each sub-system will be enumerated. Possible production machines will be ranked on performance and costs. Interfaces between these production machines will be clarified: how will these machines be coupled mechanically; how will they be controlled; what handshakes and safety aspects are necessary to guarantee proper and safe operation? More detailed logistic models will predict the nominal throughput and the production loss should the sub-systems go down.

Finally, during the *Detailed Design* phase, models will be made of the fab building layout, including detailed information on which machine will be placed where, along with a fab facility plan, i.e. cables, air, vacuum and chemical supplies. The phase will also include detailed design on software implementation: what input and output has to be provided? What communication protocols will be used? What does the software user interface look like?

The following sections will illustrate the application of modelling by using a practical project implementation.

Applied model-based systems engineering

The goal of the project described here was to design a linear motion system that will move a payload in a repeating alternating movement (forwards and backwards) and accurately position the payload. The requirements for this project, as listed in Table 1, were consolidated in the

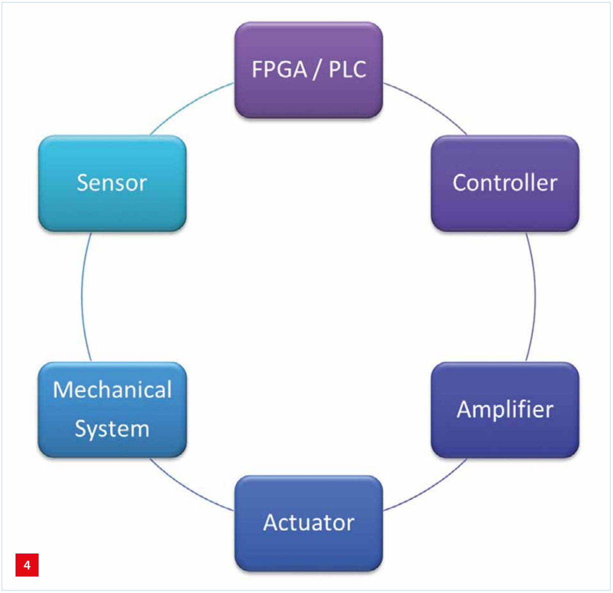


3

Proposed concept for the linear motion system.

Table 1
Requirements for the linear motion system.

Property	Value	Unit	Remark
Payload mass	0.3	kg	
Stroke	100	mm	Repeating sequence as shown in Figure 5.
Move time	175	ms	Allowed time to move payload.
Settling time	30	ms	Total time to reduce position error to < 20 µm (after moving).
Standstill time	325	ms	Time during which the payload should be positioned steadily.
Maximum error during move	200	µm	
Maximum error after settling	20	µm	
Maximum coil temperature	50	°C	
Maximum hardware costs	1,000	Euros	Including PLC, construction, actuator, amplifier.



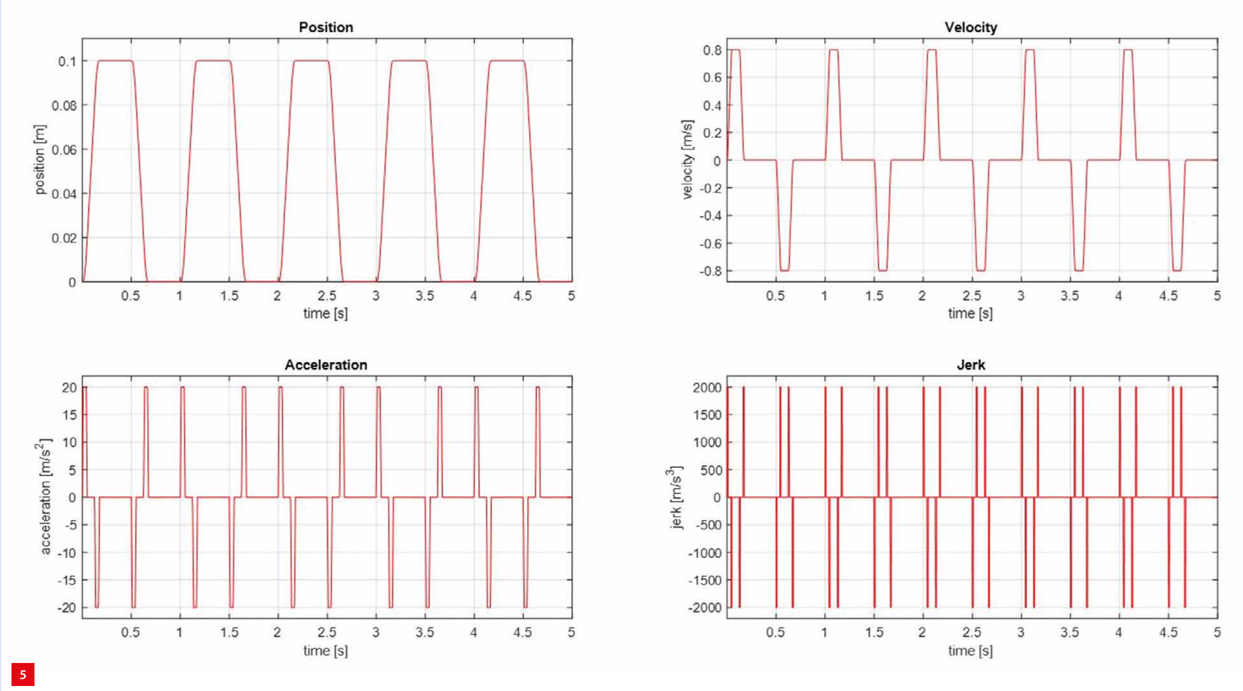
4

Hardware components for the concept.

Requirements Engineering phase. The modelling was necessary for illustrating the performance of (one of) the proposed concept(s), as shown in Figure 3, in order to be able to make the step from the *Functional Architecture* phase to the *Detailed Design* phase of the design process. In other words: is the proposed concept capable of meeting the requirements (Table 1)? In this concept, the payload is coupled mechanically to a linear actuator where its reaction force is exerted onto a frame.

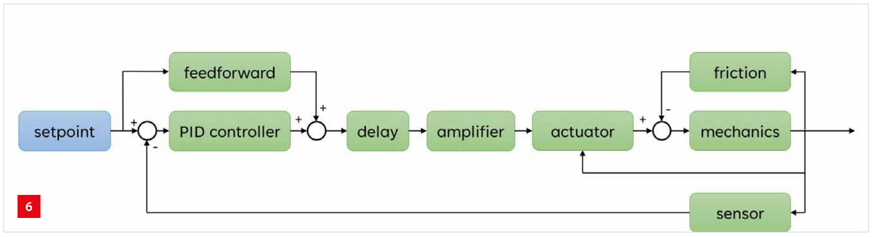
Both the actuator and the payload will be guided by a linear guide that is part of the frame. The intended components of the linear motion system are illustrated in Figure 4. The payload and actuator's moving mass are indicated by Mechanical System (mechanics). Their position is determined by a sensor and fed back to an FPGA (field programmable gate array) or PLC (programmable logic control), being the control hardware. On this control hardware, a controller algorithm provides a signal for the power amplifier that provides the electrical power for the actuator.

Figure 5 illustrates the required (third-order) repetitive setpoint. It shows position, velocity, acceleration and jerk (time derivative of the acceleration) as a function of time.



5

Setpoint requirements.



6 Block scheme representation of the linear motion system.

The stroke is 100 mm with a maximum velocity of 0.8 m/s, a maximum acceleration of 20 m/s^2 and a jerk of 2,000 m/s^3.

Figure 6 shows the block scheme for the linear motion system, which contains all the components as listed in Figure 4. This block scheme allows the motion performance to be studied. The delay block represents the delay in the system due to the finite sampling time of the FPGA/PLC. A (non-linear) block for friction was added to study the effect of friction. The actuator block contains a feedback of the position, as the effect of a non-linear actuator will be investigated. For all the components, a model representation will be derived. All model representations will be combined in the previously mentioned block scheme.

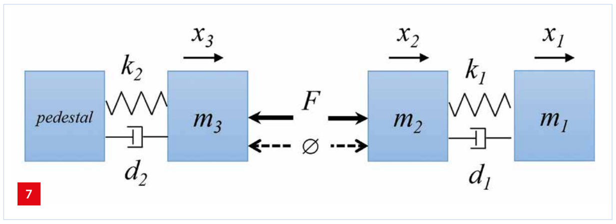
Mechanics

As a first step, a model representation of the mechanics was derived according to Figure 7. Mass m_1 represents the payload, coupled with a stiffness (k_1) and damping (d_1) to the actuator mass m_2 on which the actuator force F is exerted. The position sensing takes place between the actuator mass and the reaction frame mass, m_3. The reaction frame is coupled with a high stiffness (k_2) to the 'fixed' world (pedestal).

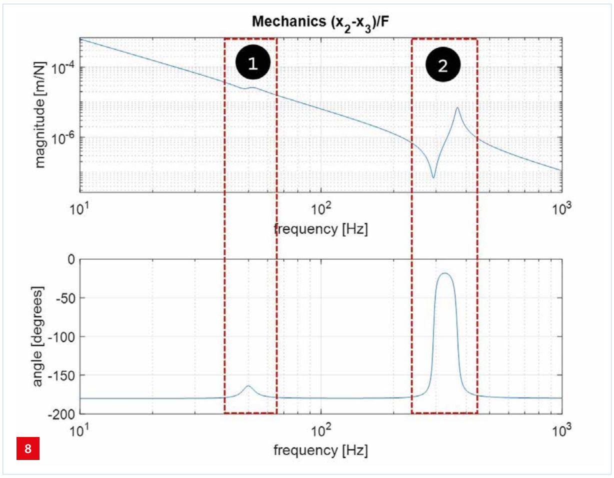
Figure 8 shows the behaviour of the mechanics in a Bode representation of the transfer function. At around 20 Hz the decoupling of mass m_3 is observed with a small peak in the magnitude (region 1). The decoupling of mass m_1 from m_2 occurs at 350 Hz (region 2). The frequencies at which these peaks occur depend on the masses and stiffnesses, and determine the achievable controller performance (which is out of scope for this article and thus not further discussed [5]).

Actuator

The intended linear actuator is a synchronous Lorentz actuator or a three-phase actuator. These actuators consist



7 Mass-spring-damper model.



8 Behaviour of the mechanics (excluding friction).

of three coils that are supplied with sinusoidal-shaped current, based on the relative position of the coils with respect to the magnets. The magnets are fixed on the stationary part and the coils are connected to the moving part. The generated force is substantially proportional to the current amplitude through the coils:

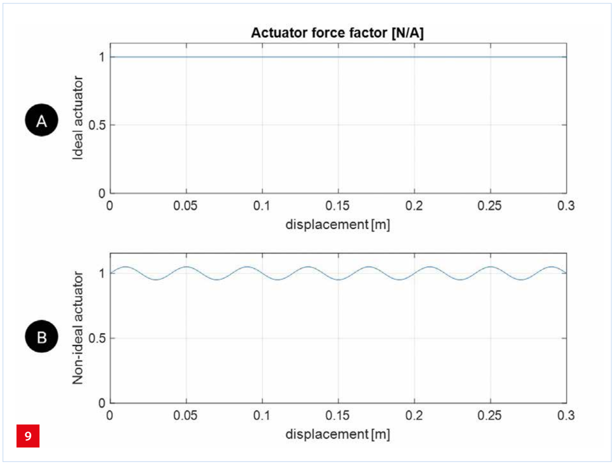
$$F = k \cdot i$$

Here, F is the force, k the actuator force factor, and i the amplitude of the current through the coils.

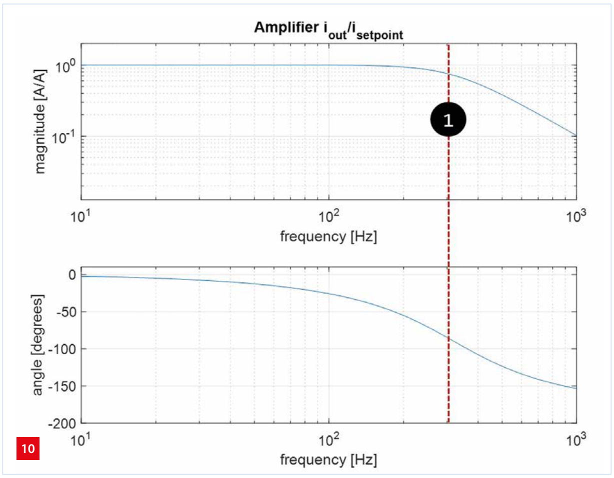
In practice there are force variations to be expected, for example due to:

- variations in magnet strength;
- variations in magnet dimensions and position;
- variation in coil dimensions;
- eddy currents, generated in electric conductive parts;
- heating of permanent magnets.

Figure 9 shows the force factor for an ideal actuator (upper picture, A) and an actuator where the force factor has a position-dependent ripple (lower picture, B). The actuator in the model of Figure 6 is modelled with this force ripple.



9 Force factor of an ideal (A) and non-ideal (B) actuator.



10

Behaviour of the current amplifier.

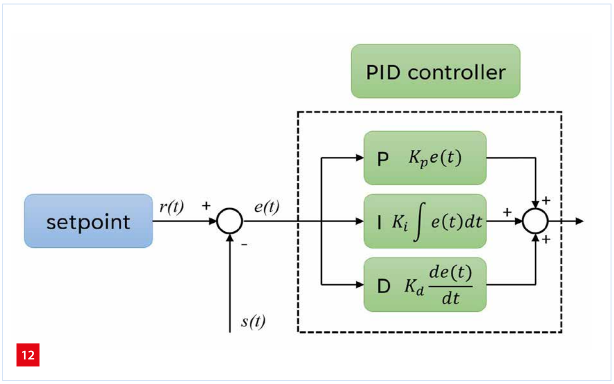
In this way, the effect of actuator force factor variations on the positioning performance is studied.

Amplifier

The intended amplifier is a low-cost current amplifier. The setpoint from the controller is sent to the amplifier, which provides the current to the actuator. In general, the amplifier will not be able to follow high-frequency input signals. For higher frequencies, the current levels will be lower than requested and there will be considerable phase shift. This is modelled as a second-order low-pass filter, according to the supplier. The transfer function, defined as the ratio between the current setpoint and the actually delivered current, is:

$$H_{\text{amplifier}} = \frac{i_{\text{output}}}{i_{\text{setpoint}}} = \frac{\omega_s^2}{s^2 + \sqrt{2} \cdot \omega_s s + \omega_s^2}$$

Here, ω_s [rad/s] corresponds to the bandwidth of this amplifier, which is about 300 Hz (2,000 rad/s). Above this frequency, the input signal is no longer followed properly, as is seen from the decreased gain and –90° phase shift as indicated by point 1 in Figure 10.



12

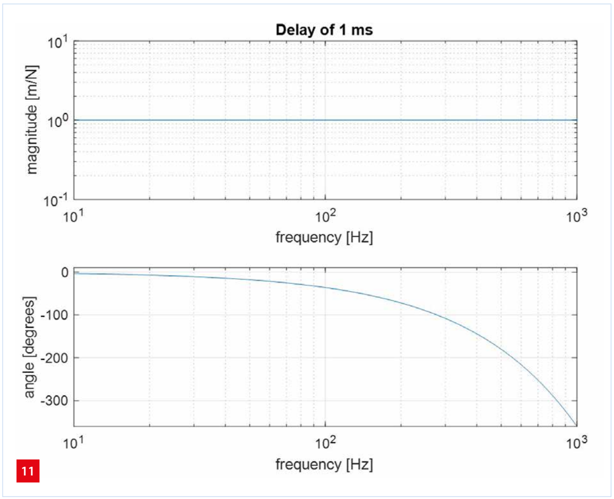
PID controller model.

Delay

Delay occurs in a digital system where sampling takes place. The sensor sends a signal to the FPGA/PLC that represents the position. As the FPGA needs time to firstly acquire this signal, then calculate the next setpoint signal and further derive the new setpoint for the amplifier (calculated by the controller algorithm) from these signals, the signal to the output of the FPGA will arrive with some time delay. A delay has no effect on the amplitude of the signal, but solely on the phase shift, as is observed in Figure 11, illustrating the model of the loop delay for a sample frequency of 1 kHz (the delay equals one sample, i.e. 1 ms).

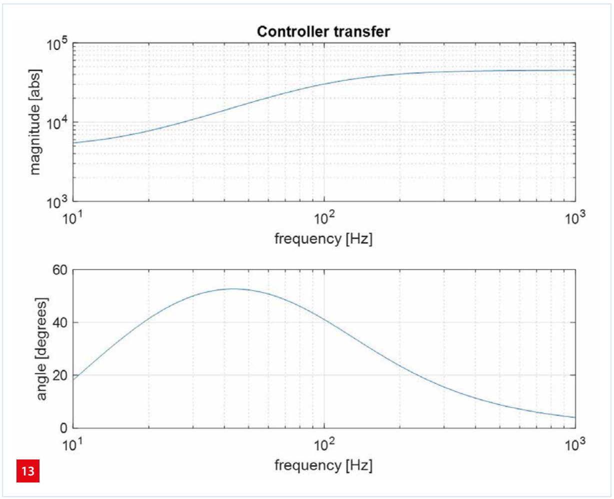
Controller

The controller is the algorithm that calculates the control output signal from the error signal, which is the setpoint value deducted from the sensor reading. In nearly all industrial applications, what is known as the PID controller is used. It consists of a proportional action (P), where the input is simply multiplied with a proportional gain K_p, an integrating action (I), where the input is integrated in time and multiplied by a factor K_i, and a differentiating action (D), with gain K_d. This type of controller is shown in Figure 12. Note: The D-action is often implemented as a tame D-action, meaning that high-frequency signals are no longer



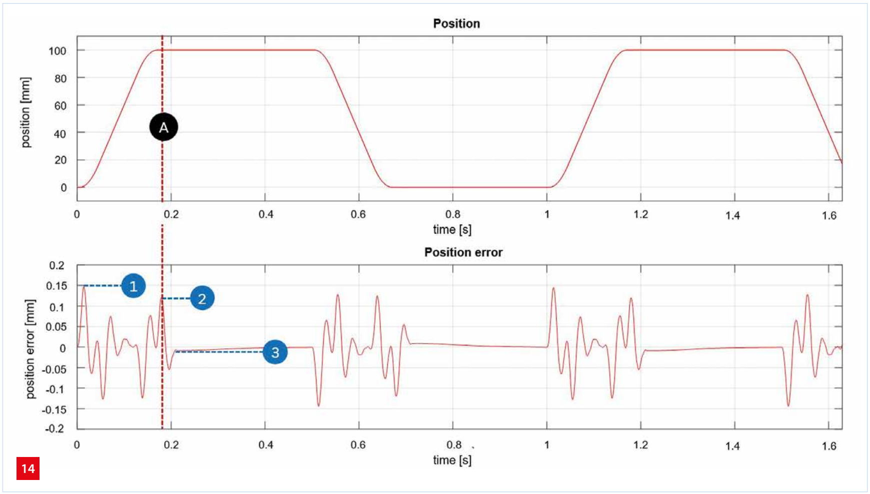
11

Behaviour of the delay.



13

Behaviour of the PID controller.



14

Motion performance prediction by the model.

differentiated. This is implemented in the controller, but the elaboration of this falls outside the scope of this article.

In control engineering terms, the transfer function for a PID controller is expressed as:

$$H_{\text{controller}} = K_\text{p} + \frac{K_\text{i}}{s} + K_\text{d} \cdot s$$

Here, the factors K_p, K_i and K_d are the proportional, integrating and differentiating gain, respectively. Figure 13 shows the behaviour of the PID controller (only for the frequency range from 10 Hz - 1 kHz, where the integrating action is not visible). Note: For reasons of simplicity, the D-action is expressed as a differentiating action rather than a tame-D action, as mentioned above.

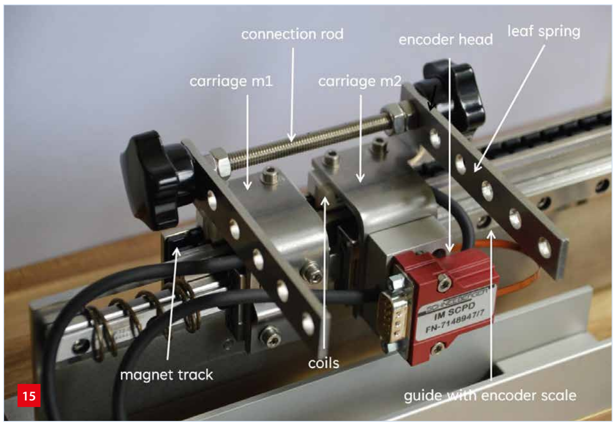
Simulation results

The above-mentioned component models are combined into one model describing the linear guide system. Friction and a non-ideal actuator behaviour were incorporated into this model. Figure 14 presents the motion performance prediction for this model, showing the actual position (upper graph) and the position deviation from the setpoint (lower graph). Time stamp A indicates the end of the movement: the point where the velocity setpoint has become zero again. Point 1 indicates the highest occurring position error during the movement, while point 2 shows the position error at the moment the velocity setpoint has become zero (stamp A). Finally, point 3 indicates the error at 30 ms after the velocity setpoint has become zero.

Comparison of model and hardware implementation

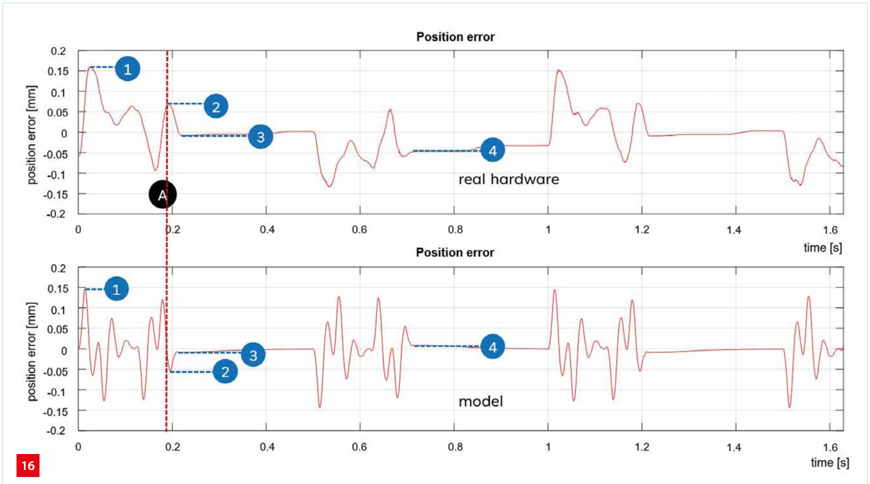
A key question is how accurate is the model? How much does the model deviate from a real hardware implementation? In order to verify its accuracy, the motion was performed on a real hardware set-up as illustrated in Figure 15. This set-up consists of two carriages, namely the payload (m_1 from Figure 7) and actuator mass (m_2). The carriages are connected by a connection rod, which was represented by the stiffness k_1 and damping d_1 in Figure 7. The linear actuator components, magnet track (connected to the stationary part) and the coils (connected to carriage m_2) are indicated as well.
Figure 16 shows the position error for the hardware and the model (as was shown in the lower graph of Figure 14). When comparing both graphs, it can be seen that both error signals are almost identical. The relevant performance parameters are listed in Table 2. There are also differences, however. Firstly, it is apparent that the model shows more higher frequency content in the position signal compared to the real hardware during the motion.

A possible explanation for this might be that in the model there is no damping modelled for the ball bearings and the cables (actuator and encoder cables). The encoder cable, not shown in Figure 15, has a considerable diameter and adds (position-dependent) stiffness and damping. That brings us to the second difference, as indicated by point 4 in Figure 16, representing the error after settling when moving backwards. This error is higher (about 50 μm) in the case of the hardware. Apparently, the cable stiffness and bearing damping need to be taken into account to arrive at a closer prediction.



15

Hardware implementation.

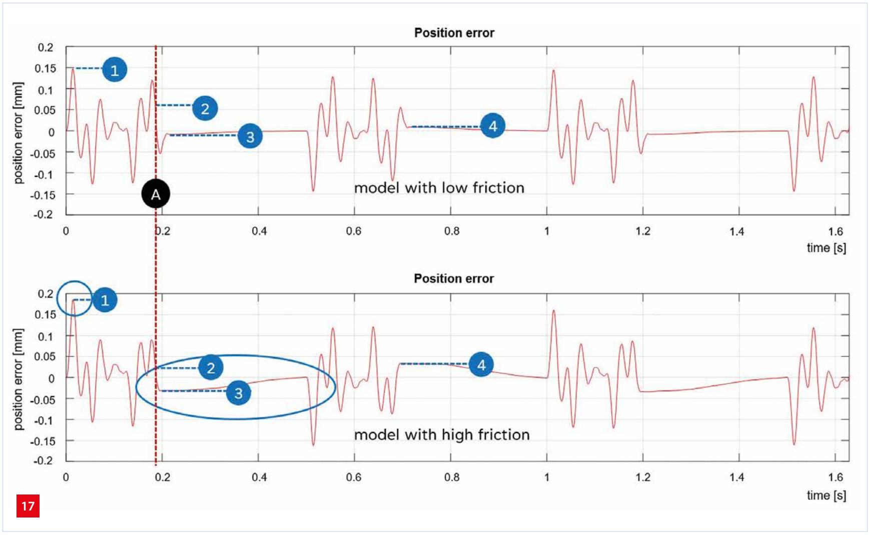


16

Error measurement at the real hardware (top) and error prediction by the model (bottom, from Figure 14).

Table 2
Motion performance according to model and real hardware.

Label	Property	Requirement	Model prediction	Hardware	Unit
1	Maximum error during move	200	150	155	µm
2	Error when velocity becomes zero	-	60	70	µm
3	Maximum error after settling	20	10	10	µm



17

Error prediction by model with low and high friction.

Motion performance effect study

The model was derived for predicting the motion performance. Such a model is usually also used to find the sensitivity for various parameters of influence. It is necessary, for example, to determine upfront to what extent the value of the stiffness (k_1) will impact the system behaviour or achievable controller bandwidth, or to what extent the bearing friction determines the motion performance. The following parameters (among others) affect the motion performance:

- payload and actuator mass;
- interconnecting stiffness and damping;
- bearing friction and damping;
- cable stiffness and damping;
- actuator force factor variations;
- amplifier cut-off frequency;
- amount of FPGA/PLC delay;
- control parameters (where achievable bandwidth is determined by the above-mentioned parameters);
- feed-forward implementation.

A designer needs to know the sensitivity of these parameters before starting the *Detailed Design* phase in order to ensure that proper design choices are made.
To illustrate this, the motion was performed for the case where the friction was increased from $F_{\text{w,dynamic}} = 0.15$ N to $F_{\text{w,dynamic}} = 0.5$ N (being the dynamic friction; the static friction was increased from $F_{\text{w,static}} = 0.3$ N to $F_{\text{w,static}} = 1.0$ N). The effect on the maximum occurring error (point 1) and the error 30 ms after the velocity becomes zero (points 3 and 4) is clearly visible in Figure 17.

Conclusions

Complex product development requires a systematic approach, as indicated by the V-model for systems engineering. Different phases are identified with differing levels of abstraction and detail. These phases will occur in a chronological order, where feedback iterations are necessary when progress fails. For progressing towards the next phase during product design (the left branch of the V-model), modelling is key to predicting intended functioning and behaviour. The level of detail in these models will increase in time along with the increased level of detail in the design.

Using a practical example, it was illustrated how a model can be derived to analyse a concept design of a linear motion system. This model was used to predict motion performance behaviour and to analyse the sensitivity of various parameters that determine its performance. The model results validated the feasibility and indicated the achievable performance of the studied concept, thus legitimating a move forwards to the next design phase.

REFERENCES
[1] G.M. Bonnema, K.T. Veenvliet, and J.F. Broenink, *Systems Design and Engineering*, CRC Press, London, 2016, ISBN 9781498751261.
[2] Sandia National laboratories.
[3] J. Dick, E. Hull, and K. Jackson, *Requirements engineering*, Springer, Cham (CH), ISBN 9783319610733.
[4] Electronics and you, *www.electronicsandyou.com*
[5] Mikrocentrum , Applied Mechatronics training course, *www.mikrocentrum.nl*