Holger Schlingloff, Fraunhofer FOKUS

1

CrESt Use Cases

In this chapter, we present three use cases that are used throughout this book to demonstrate the various systems engineering methods presented: vehicle platooning, adaptable and flexible factories, and autonomous transport robots. The use cases are chosen from real-life industrial tasks and exhibit all software engineering challenges that are specific to the development of collaborative embedded systems.

W. Böhm et al. (eds.), *Model-Based Engineering of Collaborative Embedded Systems*,
https://doi.org/10.1007/978-3-030-62136-0_1

1.1 Introduction

To derive and present the systems engineering methods described in this volume, three different industrial use cases are used throughout the book. These are vehicle platooning, adaptable and flexible factories, and autonomous transport robots. In the following, we describe each use case up to a level of detail that shows clearly how the respective process building blocks contribute to the overall development of the use case. For each use case, we first give some remarks on the historical evolution of the domain, then describe requirements and application scenarios for the use case, and finally describe the main challenges for development to be addressed in the rest of the book.

1.2 Vehicle Platooning

In the "Vehicle Platooning" use case, we consider a group of vehicles that share the goal of traveling together at high speed for some distance. With the vehicles driving in a low-distance formation, the overall air resistance is decreased and fuel consumption is significantly reduced. Furthermore, more vehicles fit onto the street and traffic may be more efficient. However, in order to avoid crashing into one another, the vehicles have to communicate constantly. Scenarios within this use case are as follows: forming and dissolving a platoon, as well as single vehicles joining and leaving a platoon.

Cruise control (CC) in cars has been known since the 1950s. Up to now, such systems have been and still are limited to isolated control decisions executed individually based on local sensor data. In the future, vehicle-to-vehicle and vehicle-to-roadside communication technology will enable the cruise control systems to consider a vast range of additional context information (e.g., general traffic conditions, dangerous situations ahead, etc.). This will enable the cruise control system to establish effective collaboration between vehicles. This kind of collaborative cruise control will be the central component of upcoming fully autonomous vehicles.

Adaptive cruise control (ACC) is a step towards such a collaborative cruise control. It is an enhancement of conventional cruise control systems that allows the vehicle equipped with ACC to follow a vehicle in front with a pre-selected time gap by controlling the engine, power train, and/or service brakes. This means that the

ACC is a system that requests the onboard computers to control the vehicle's acceleration and deceleration. The most common ACC systems generally use automotive radar systems, placed at the front of the car, and/or a camera placed on the interior rear mirror. The radar is used to identify obstacles and predict their speed by sending and receiving radio waves. Camera-only ACC systems are currently being researched but are not yet state of the art. The ACC increases and reduces the car speed and automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead. The system may not react to parked, stopped, or slow-moving vehicles; it alerts the driver of an imminent crash and may apply limited braking but the main responsibility for steering the car lies with the driver.

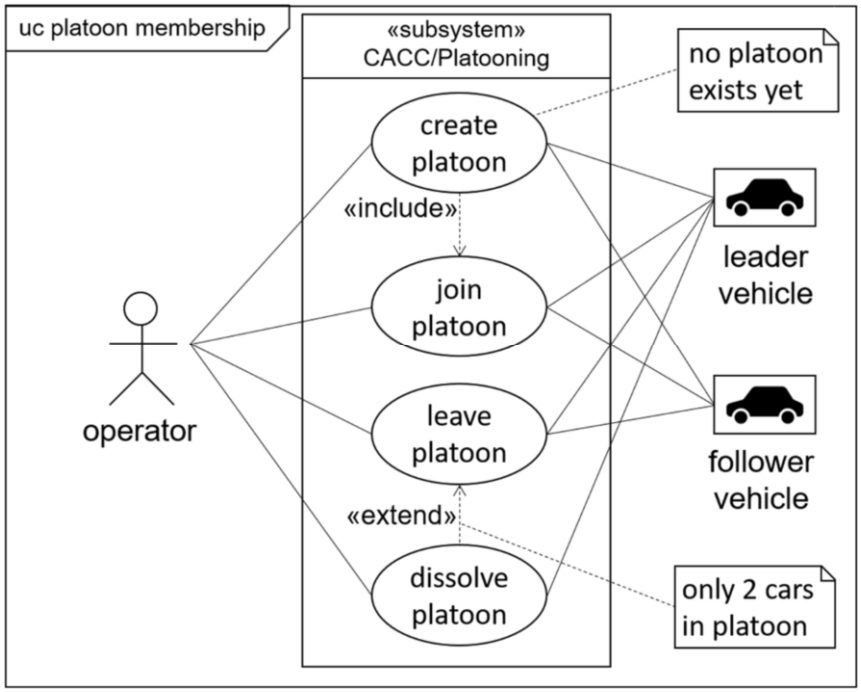

Fig. 1-1: *SysML use case diagram "Platoon Membership"*

Collaborative adaptive cruise control (CACC) takes the ACC technology to the next level, enabling vehicles to adjust their speed to the preceding vehicle in their lane with direct car-to-car communication. In the following, we use "CACC" to denote the cyber-physical system of communicating controllers in collaborating vehicles (that is, the collaborative system group (CSG)) and "CACC ECU" to denote the electronic control unit(s) in an individual vehicle (that is, the collaborative embedded system (CES)). Compared to

classical ACC, a CACC can respond faster to speed changes by preceding vehicles and even vehicles beyond the line of sight. These advancements improve the stability of the traffic flow, increase driver confidence, and allow distances to be minimized for vehicle-following. Ultimately, this results in better use of a highway's capacity and greater fuel efficiency. To increase efficiency by leveraging the collaborative aspect, the CACC may be observing several of the following common goals and targets:

- Same destination (at least partially)
- Support when driving on an unknown road/to an unknown destination
- Desired and steady cruising speed
- Reduced time and fuel consumption

Figure 1-1 shows the main SysML use case diagram[1] for platooning. Most of the collaborative aspects of the CACC functionality occur when a platoon is formed. Before any automated vehicle control can start, the vehicles have to notice each other and agree on a common driving strategy. During this phase, several aspects have to be considered:

- The vehicles must be in a close range so that a platoon can be formed physically. Therefore, the CACC must be aware of the physical location, speed, and direction of each vehicle. As a minimum, the vehicles must be aware of other CACC-capable vehicles and cars in their immediate vicinity.
- The vehicles must have a common driving direction. In the simplest case, the CACC would know the complete routes that all participating vehicles are about to travel. However, due to privacy concerns, this may not be the case; only partial information may be available from some vehicles.
- The vehicles should have a common or at least similar driving characteristic or goal. A truck platoon that wants to drive as economically and safely as possible might not be acceptable for a driver of a powerful car who wants to travel as fast as possible. Other drivers might not be willing to accept a very close distance to the surrounding vehicles, which is necessary to maximize the fuel savings. Such driving characteristics have to be negotiated between the participants.

[1] The term "SysML use case" should not be confused with the three use cases for collaborative embedded systems presented in this chapter. A SysML use case describes a dedicated functionality for a certain actor.

- The vehicles must agree on their roles in a platoon. A lead vehicle (LV) has to be selected; all other platoon members will be assigned the role of a follower vehicle (FV). Either role might not be acceptable for some drivers. During the negotiations, a car can be a potential lead vehicle (PLV) or a potential follower vehicle (PFV).

A typical scenario for this use case is as follows. A vehicle drives on the highway and wants to create a platoon. The CACC ECU of this vehicle generates a platoon proposal and continuously broadcasts it to other vehicles that might join the platoon. Another vehicle's CACC ECU receives the proposal and accepts it. After the acceptance, both vehicles start a "platoon verification" routine, which includes a platoon role allocation (PLV and PFV). During the verification, no other vehicle can connect to the platoon. The PFV joins the PLV longitudinally at the rear. The speed of both vehicles is synchronized to establish a pairing. When the verification is closed and the platoon is created, PLV becomes LV and PFV becomes FV_1.

In the meantime, the platoon proposal remains active. Invitations for other cars to join the platoon are continuously broadcast. If a PFV_2 receives this request and accepts the proposal, the existing platoon will be extended by another FV. In the simplest scenario, PFV_2 must join at the rear of the platoon — in other words, behind FV_1. More complex scenarios would allow a vehicle to also join somewhere in the middle of an existing platoon. Assuming that the communication is organized as a peer-to-peer network, PFV_2 can pair with FV_1 or LV, depending on the platoon network topology. Once the pairing is finished, the platoon join is closed; PFV_2 becomes FV_2 and the platoon regulation takes control.

There are many more aspects and parameters that have to be considered or negotiated during the build-up phase of a platoon. As the vehicle platooning use case is considered in various chapters throughout the book, we do not go into detail here. Moreover, there are operations that may be reasonable but are not considered in this book, such as changing the order or the leader of a platoon, fusing two platoons into one, or splitting one platoon into two. A collaborative platoon management system has to be flexible enough to cope with such diverse information.

The CACC use case exhibits many challenges for advanced software engineering, described as typical for the development of CESs in Chapter 3: the complex functionality is realized mainly by software, there is a high degree of networking of heterogeneous components, and the system must act reliably and autonomously. Furthermore, the development must take into account common and

conflicting goals of the CESs. The challenges addressed in this book can be summarized as follows:

- Conception, implementation, and validation of a CACC that realizes the function of driving in a platoon
- Assessment of the quality of the platoon regulation concept, especially with respect to safety and reliability
- Platoon communication concept and its quality, especially the security
- Heterogeneity of CESs built by different vendors (and the resulting challenges for information exchange between these systems, including standardization)
- Means to cope with uncertainties caused by imprecise and possibly differing context perceptions of vehicles

Further challenges, which are not addressed here, include:

- Reliability of artificial intelligence techniques used for context perception and the related uncertainty
- Elicitation of requirements for engineering methods and tools for generalized collaborative car-to-car and car-to-X functionalities.

1.3 Adaptable and Flexible Factory

The use case “Adaptable and Flexible Factory” deals with production modules that collaborate to build products on demand. Each module consists of one or more production machines and offers one or more production functions (e.g., cutting, assembly, inspection, or forwarding of a workpiece). These functions can be combined in different ways, and even dynamically recombined according to changing customer needs. The common goal is to optimize the use of production resources and machines for different usage scenarios.

According to the VDI 5201 standard, flexibility and adaptability are concepts that describe *“the ability of manufacturing companies to change in response to changing general conditions. […] Adaptability refers to the ability to change involving structural changes to the system, while flexibility refers to the ability to change without structural changes.”* Present day industrial production facilities mostly consist of specialized production machines that are connected in a fixed way via stationary transport devices such as belt or chain conveyors. The need for adaptable and flexible factories is driven by several demands:

- Individualization and customization of products
- Variability of products in globalized markets

- New or changed customer requirements
- Shorter product life cycles
- Changing markets and varying sales figures

Clearly, these demands cannot be met with traditional production systems. Adaptable and flexible factories are at the center of the fourth industrial revolution, comparable to the transition from individual manual production methods to mass production by machines in the 19th century. The ultimate vision of Industry 4.0 is to allow fully automatic production of individualized goods, reducing changeover times to zero. In order to realize this vision, several fundamental properties of a production system are required. The production process must be modular and arranged in several stages. Each production module must have a clearly defined set of capabilities and must be decoupled from other modules. Finally, the mapping of the process to modules and the topological layout of the process in the factory must be flexible. As most modern production facilities satisfy these requirements to some degree, the major obstacle to adaptable and flexible factories lies in the complexity of the corresponding systems engineering process.

Within this use case, we assume a factory is composed of multiple independent units called *production modules*. A production module can be thought of as a specific machine or device, or a tightly coupled group of machines. This covers both process industries and discrete manufacturing, where production modules are sometimes called *production cells*. Modules may be aggregated into different *production lines* that are substructures of a *production facility*. A *factory* may host several such facilities.

In our terminology, a production module or cell is a CES. The CSG is formed (statically or dynamically) according to a specific production job: it consists of all modules in the factory which take part in this particular production process. For a specific product component (e.g., a motor), this can be the corresponding production line. For a complete product (e.g., a car), the CSG consist of all modules in the corresponding production facility.

A production module is characterized by its ability to interact with the environment, which also includes communication with other production modules, humans (e.g., operators or maintenance engineers), and other entities within a factory (e.g., control systems or manufacturing execution systems). Collaboration arises from this possibility of interaction: several modules can form a production chain for a certain type of product. General functions of a module are:

- Processing
- Assembly
- Quality control — for example, visual inspection
- Transportation
- Storage of products

Flexible production modules are capable of performing different functions in the production chain. One example is a robot arm that can change the tool fitted (e.g., a welding gun) for another one (e.g., a digital camera). Adaptable production facilities are capable of changing the way the different modules are interconnected. An example is a mobile robot that can work in different production lines. This example shows that in an adaptable production facility, membership of a CES in a CSG can change dynamically.

In our use case, we consider a CSG for the production of quadrocopters. Each product consists essentially of components from five different classes:

- Mechanical sub-components
- Onboard electronic components
- Motors for the rotors
- Batteries
- Remote control units

Each of these components is available in several different variants, hence there are a large number of different products that can be built. The production process consists of several steps, which are performed either in sequence, in parallel, or independently of each other. Typical production steps are:

- Pre-assembly of rotor arms and rotor
- Pre-assembly of the body, including mounting of onboard electronics and battery
- Attachment of four arms and rotors to the body
- Final assembly of the full product

For each individual production step, activities such as turning, sticking, molding, drilling, screwing, etc. are necessary. The order of assembly of the different parts, and a production system which can realize this production task are shown in Figures 1-2 and 1-3.

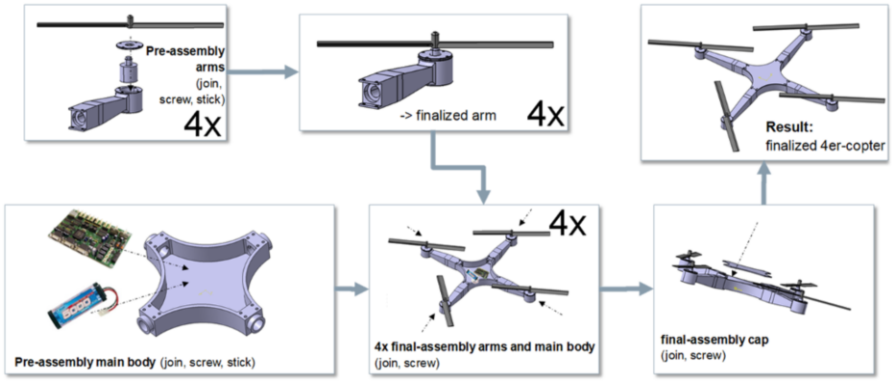

Fig. 1-2: *Process sequences "Quadrocopter" – order of assembly*

The production facility (i.e., the CSG) is structured into two main lines and several sidelines. Each line contains several production modules (i.e., the CESs). Each module is capable of performing different processing tasks (joining, sticking, gluing, soldering, etc.), allowing a flexible production of parts for different quadrocopters within one line. Moreover, the connection between sidelines and main lines can be adapted dynamically according to changing demands. Given a certain sequence of quadrocopters to be produced, the modules collaborate to accomplish this job as quickly as possible and with the most effective use of resources. Usually, this collaboration is orchestrated by a central manufacturing execution system (MES). The MES assigns each specific step of the production process to an individual production module and adapts the flow between the production lines accordingly. However, such a centralized control component is not really necessary; it would be feasible to imagine the production modules distributing the workload among themselves.

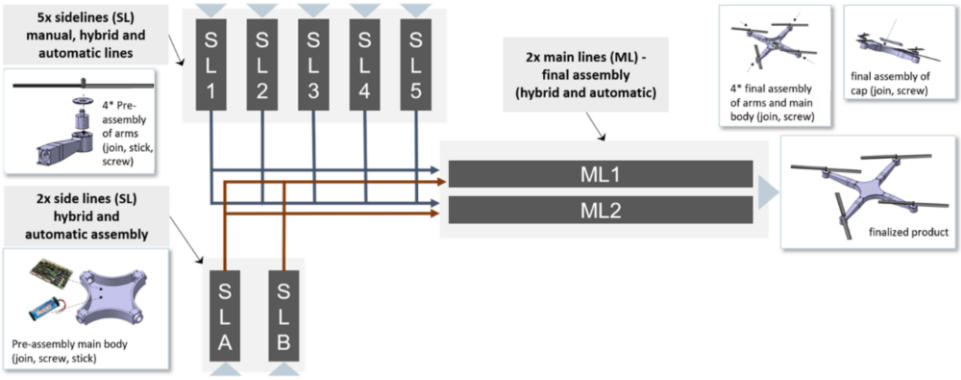

Fig. 1-3*: Example production system for the assembly of a quadrocopter*

The diagram in Figure 1-3 is an abstract model of the production facility. Given appropriate models of the production modules and their interconnections in the production facility, plus a description of the necessary production steps for each product and the estimated demand for each product, the best possible system configuration can be determined via simulation. In particular, simulation can be used to show the manufacturability of certain products or sequences of products, to determine the best timing of the modules and lines, to avoid bottlenecks and optimize the layout and output of the facility, and to calculate the cost per unit and management costs. Chapter 12 shows how to create adequate models for this use case.

Challenges for the design of adaptable and flexible factories, which are addressed in this book, are as follows:

- Definition of engineering methods and a corresponding process for the design of an adaptable and flexible factory
- Integration of qualities into the engineering methods and models — for example, safety, reliability, and security
- Creation of models for production modules and facilities
- Description of production processes and validation of orders
- Simulation and analysis methods for these models:
  - For proving properties of the CESs as well as the CSG
  - For managing variability in the CSG
  - For risk assessment and risk decomposition
- Engineering tools that support the adapted engineering methods
- Migration concept for converting a legacy production site into an adaptable and flexible factory step by step

1.4 Autonomous Transport Robots

Our third use case deals with autonomous transport robots, which are driverless vehicles for loading and unloading production modules in a factory. Since they are not stationary, autonomous transport robots can realize the material flow between flexible units in an adaptable production facility. In our terminology, each robot is a CES, and the fleet of robots is the CSG that provides the transport service to the production facility. We explore a decentralized control scenario, where each robot can decide which transport job to accept and accomplish. The common goal of the fleet is to keep production going — that is, no production module may ever stop due to lack of supply material or abundance of processed material.

In present-day factories, traditional transport systems such as conveyor belts or rollers are increasingly being replaced by automated guided vehicles (AGV). The task of these AGVs is to provide an automated flow of material between storage, machinery, workspaces, and shipping department — for example, to transport small load carriers, trays, barrels, and coils. Moreover, they can be used for the automated transport of components to quality control or refinishing operation spaces, and for the transport of tools and testing equipment to assembly lines or working spaces.

The advantages of AGVs in comparison to stationary conveyor systems are:

- Scalability: A fleet may grow as necessary with regard to the number of transportation tasks. If business demands grow, new vehicles can be added to the fleet easily.
- Changeability: The layout of a production process can be changed easily, as no stationary equipment has to be rebuilt.
- Fault tolerance: With stationary equipment, even a small failure of a single part often means that the whole process is halted. If one of several AGVs malfunctions, however, the others can simply take over its tasks.
- Reduced space: In general, vehicles use less space than conveyors; moreover, they can be stowed away if not in use. In fact, as modern transport robots use the same walkways as human factory workers, the additional space requirements are minimal.
- Easy deployment: Since there is no construction work necessary, AGVs can be deployed at a production site within a relatively short amount of time.

The first generation of AGVs, introduced in the 1950s, were capable of following a white line or other optical markers on the floor. They used to drive on circular one-way routes on dedicated lanes in the factory. Thus, there were only a few advantages compared to stationary conveyor systems. The second generation, which emerged around 1970, still had to use dedicated areas that humans were not allowed to enter but could localize themselves in these areas via photoelectric and inductive sensors. Thus, they could move more or less freely within a blocked segment of the traffic route, which allowed more flexibility. Laser scanners for distance measurement became available in the 1990s, with safety features available only from the 2000s. A rotating laser scanner for distance measurement can not only stop the AGV if a person approaches, it can also build a digital map of the factory environment and allow the AGV to move freely in the facility.

An autonomous transport robot is an AGV that can navigate autonomously. It does not require any kind of markings, reflectors, or track guidance. Using a pre-recorded map of the environment, it finds its path by itself, without the need for fixed routes on traffic ways. Localization is done via comparison of the data from the integrated laser scanner with an internal map of the factory. Routing is also autonomous: when a robot receives an order to transport a load from point A to point B, it uses the map to calculate an optimal path. In the case of there being an unexpected obstacle on this path—for example, a pallet that the vehicle cannot circumvent—the robot comes up with an alternative route. If no alternative exists, the robot reports to the central management software that the order cannot be executed.

In this use case, we consider a fleet of autonomous transport robots as a CSG. Currently, transport robot fleets are managed and controlled centrally. A fleet organization system AIC (AGV interface controller) is in contact with the customer's manufacturing execution system and translates material requisitions into transportation tasks for the fleet. Criteria for the AIC's choice can include the vehicle's distance to the pick-up-area, avoiding robots driving without a task, and the battery status of the robots. From the AIC, the robots receive simple instructions with a "pick up here, carry there" structure and then plan the route to get to their destination, with each robot taking little individual action and robots gathering information first and foremost from the central controlling system.

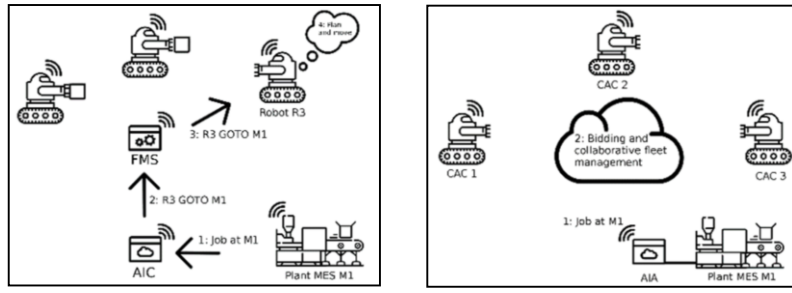

Fig. 1-4: *Central and decentralized fleet management*

Here, we are considering transport robots as individuals with goals, foresight, and an awareness of the other robots in the fleet. Individual robots are granted a higher level of autonomy, and the central AIC is no longer necessary. The task management system merely offers tasks that must be performed, and the robots distribute these amongst themselves according to individual capabilities (see Figure 1-4). This has several advantages. Among other improvements,

it increases the overall efficiency, making more sensible use of resources and moving in ways that ensure no robot becomes a hindrance for others.

The user story in Table 2-5 describes how autonomous cooperating robots can determine which one of them fulfils an order for transportation. If a new order is given and several robots are available to take it, there must be a decision about which one of these robots will actually perform the task. This can be accomplished via a "bidding" process in which each robot calculates its factors playing into this task — for example, how far away it currently is from the pick-up area or what its current battery charge status is. It then sends these combined factors to the other robots as a bid. Depending on which robots can offer the most practical circumstances, a distributed consensus protocol is used to decide which robot takes the order.

Table 2-5: *User story for distribution of transport jobs*

	Who?	**What?**	**When?**	**Why?**
1	Production Module	Broadcasts transportation need to robots	Every time a module has need of support or availability (may be in advance and/or may be with priority)	The production process of the module is not allowed to stop
2	Every Robot	Calculates a bid for this transport (may be based on individual cost and/or other criteria)	When a new transport need is notified	To get the information about which robot is the best fit for this transport
3	Every Robot	Determine winner by distributed leader election algorithm	After bidding	
4	Winning Robot	Adds the transport to its own transport queue	When a bid is won	That the transport need is satisfied

Further challenges in this use case are as follows:

- Cooperative path planning: Ideally, each robot should share the information about blocked paths with the other robots in the fleet. This information must be updated at frequent intervals. A more advanced option would allow path planning according to the traffic situation and the presumed paths of the other robots.
- Fault tolerance: The transportation system is not allowed to halt if some of the robots are offline (in a dead spot where there is no wireless reception) or cannot localize themselves because of massive differences between the observed and expected environment.
- Flexible fleet size: It should be possible to integrate a new robot into an existing and operating fleet without stopping production. After it has authorized itself, the new robot receives map and task information from the others and is able to collaborate in the fleet as a coequal partner.
- Distributed logging and monitoring: For a possible "transport as a service" operation mode, the fleet must remember all relevant transactions. The logging of this data must be safe and secure—for example, via a block chain mechanism.

Acknowledgement: The author wishes to acknowledge the contributions to the CrESt use case descriptions by the authors of the respective deliverables, in particular Oliver Kreuzmann, Stefan Penz, Jorge Castillo, Suryo Buono, Birthe Böhm, Roland Rosen, Jan Vollmar, Jan Christoph Wehrstedt, Wolfram Klein, Sebastian Schröck, Constantin Hildebrandt, Alexander Ludewig, Birte Caesar, Marian Vorderer, Michael Hassel, Sebastian Törsleff, Jan Winhuysen, Tobias Schüle, Michael Nieke, Jan Stefan Zernickel, Susanne Dannat, André Schmiljun, Henry Stubert, and Janina Samuel. Moreover, the author thanks the reviewers Torsten Bandyszak, Birthe Böhm, Birte Caesar, Alexander Hayward, Jörg Kirchhof, Vincent Malik, Nikolaus Regnat, Sebastian Schroeck, and Jan Christoph Wehrstedt for their valuable remarks.