Chapter 5
The Technology History of Virtual Product Creation

Executive Summary

This chapter deals with the following topics:

- description of the evolution from shop floor to modern Virtual Product Creation and beyond, focusing on three different application fields: geometric modeling, verification and validation and product data management
- understanding of the interrelation between working technologies and traditions, knowledge about products and processes and collaboration aspects (local, regional, global) on the one hand, and the fast IT evolution on the other hand.

Quick Reader Orientation and Motivation

The intention of this chapter is:

- To gain a first insight into how technologies have emerged over time and in which sequence.
- To provide the necessary background to assess the origin and the maturity of virtual product creation tools and methods.
- To introduce the fundamental concepts of virtual geometric modeling.
- To give an overview over different computer simulation technologies for verification and validation activities.
- To present the core concepts and functionalities of Product Data Management (PDM) systems.

5.1 The History of Computer Aided Design (CAD) Systems and Geometric Modeling

Before geometries were modeled in Computer-Aided Design (CAD) software tools, geometries were drawn manually on paper. Since the tradition of manual drawings

R. Stark, *Virtual Product Creation in Industry*,
https://doi.org/10.1007/978-3-662-64301-3_5

dates back several hundreds of years, it has evolved and improved over time. In order to understand why such optimized practices have been replaced by computer-aided tools, one has to understand the advantages of using CAD over the traditional methods.

The first wave of CAD systems improved what had already been done before: the 2-dimensional drawing of geometries. But products have 3-dimensional shapes and thus multiple drawings of one product have to be done, that show the product from different perspectives and in different cross-sections. The high amount of drawings that have to be done results in a strong need for drawing creation productivity. If only a few minutes can be saved in creating one 2-dimensional drawing, then many hours of work could be saved.

Drawing geometries in a software provided several advantages over manual drawings. Digital drawings allow for more efficient ways for editing, storing, copying and distributing. The possibility to edit a digital drawing allows for removing mistakes or adjusting details without having to redo the entire drawing. Since design models need to be provided to multiple stakeholders (i.e. other engineers who need to align their design, and manufacturing experts) they must be copied and distributed which both is much easier with data rather than with paper. Furthermore, the storage of design models can be realized in many different ways. For example, they could be organized by their affiliation to different components, different engineering teams or different areas of manufacturing. While one single digital model may be referenced from different 'views' (i.e. data models that represent different structures for organizing engineering items), a drawing would have to be copied several times, if it was to be stored in different structures. Hence, the management of design models is more efficient for CAD models. All these advantages increase overall productivity and were the main reasons for the first development of CAD systems.

The first (2D) CAD system, named Sketchpad, was introduced in 1962 by Sutherland, a researcher of the MIT. All following systems were developed by significantly sized manufacturers (>10.000 employees) for use within their own company. Together the aerospace and the automotive industry were pioneering the field with notable systems such as DAC at General Motors in 1964 [1], CADD at McDonnel-Douglas in 1966 [2], PDGS at Ford in 1967 [3] or CADAM at Lockheed in 1967 [4].

After this first wave of CAD systems a new feature was introduced into CAD systems: the possibility to generate and modify 3-dimensional geometries. The advantage over traditional drawing approaches was evident: if 3-dimensional shapes are directly modeled in 3-dimensional space, they only need to be modeled once. This is also why design models were often modeled using clay, wood or other materials. Unfortunately, such approaches cannot provide the exactness that a drawing can provide, and drawings were still required. By modeling 3-dimensional shapes in a CAD tool, they need to be modeled only once and the resulting model is precise enough. Thus, the time for creating multiple drawings could be saved. Furthermore, new technologies allowed for precisely defining complex surfaces. With 2-dimensional drawings such precision could theoretically only be realized by creating an infinite number of fine-granular cross-sections of a 3-dimensional shape

or—depending on the topology of the geometry shape—by a few representative cross-sections in case of simple prismatic or rotational parts.

The first generation of 3D-CAD systems were mostly developed at research institutes, like in Europe most notably BUILD (University of Cambridge, 1978), PROREN (Ruhr-Universität Bochum, 1978) and Compac (Technische Universität Berlin) [5]. These systems built on research results that laid the foundations for modern 3-dimensional modeling: the mathematical concepts of non-uniform rational B-splines (NURBS), boundary representations (B-Rep), constructive solid geometry (CSG) models and wire frame models and specific 3D modelling languages such as the Part and Assembly Description Language (PADL) by Voelcker. It is important to understand that the first 3-dimensional modeling system did not support graphical modeling but instead modeling had to be done by writing mathematical formulas and code scripts.

Shortly after the first 3D-CAD systems were presented by academia, OEM would introduce these new technologies into their existing CAD solutions, subsequently replacing 2D modeling by 3D modeling. Examples comprise GEOMAP at Toyota [6, 7], PDGS at Ford or CADD at McDonnel-Douglas. While some of these solutions where custom-developed by external software companies, several OEM-internal developers would start their own businesses, leading to a wave of CAD vendors introducing ready-to-use CAD offerings for small and medium sized companies, too. During that period many of the CAD systems that still exist nowadays were born: PE CAD from HP (in 1980), UniSolids from Unigraphics (1981), CATIA from Dassault (in 1982), SDRC from I-DEAS (in 1982), InterAct (1983) and IGDS (1984) from Intergraph and Euclid from Matra (1985).

It was also during that time, the early 1980s, that the computer hardware market was shaken by the introduction of RISC processors and the first workstation computers, most notably UNIX workstations. While CAD systems usually ran on computer hardware that was built for the single purpose of running CAD systems, the new workstation concept allowed for different usage scenarios. In the mid-1980s, graphical processing power also allowed for more advanced graphical editing for the first time. The company PTC profited from that development first by introducing their CAD system Pro/Engineer in 1987 that revolutionized the way 3D modeling was done in a graphical user interface.

This second wave laid the fundamentals for today's CAD-systems: the approach of solid modeling by which shape generation is done by sequentially adding basic shapes such as cuboids, spheres or cylinders (often represented as B-Reps) to a 3D model and combining them (using CSG models) in order to build more complex shapes. It was only in 1996 when a new approach to modeling was presented by Lüddemann that suggested to virtually imitate the process of clay modeling [8]. Nevertheless, solids modeling established itself as the most widely adopted approach and it can be found in every current CAD system.

While the basic modeling kernel remained stable for many years, CAD systems provided new functionality in other ways. In 1976 Grayer introduced an approach that allows for automatically generating machine control code for a milling machine directly from a CAD model. The concept of programming the machine's routines,

using so-called numerical control (NC) code, had already been introduced in 1952 by Parson [9]. This new approach allowed for automating the task of manually writing such NC code. Other approaches followed thereafter and nowadays the generation of NC code is supported by almost all CAD systems for many types of machines (plate work, grinders, etc.).

In 1987 Pratt and Wilson presented the concept of features [10] and parametric models that drastically increased engineering productivity. The basic idea of features and parametric models is to allow engineers to model design intent like 'hole' or 'thickness' explicitly with semantics instead of doing so indirectly via geometry only. The concept of a 'hole' can be selected from a set of reusable 'features' and the engineer only needs to place the 'hole' in the given coordinate system and specify its main parameters (such as diameter and depth). This is especially useful for standardized shapes such as screw threads and it saves time and ensures correctness by automating modeling tasks. In 1992 Schulte and Stark suggested using features as "higher level primitives" in order to transfer manufacturing relevant information about geometries from a CAD system to a CAPP System [11]. In 1994 Rieger developed one of the first feature modeling editors [12]. In 1998 Dassault Systems introduced its new CAD System Catia v5 that implemented the idea of features allowing the user to specify parameterized templates for parts [13, 14].

While CAD systems were thus providing ever more useful functionality many product models were still available as drawings and a very pragmatic question arose: how to convert these drawings into 3D models? In 1981 Jansen developed an approach for automatically converting technical drawings into 3D models [15–17]. Later, in 1995 and in 1997 Liu and Luth improved this approach by also generating more complex splines and semantic information in the resulting 3D models [18].

Until today, the CAD system market has been heavily consolidated and only few of the former system vendors have survived. The main competitors in the 2010s were Dassault's CATIA and Solid Works, Siemens NX (former Unigraphics) and Solid Edge, PTC's Creo (former Pro/Engineer) and Autodesk's AutoCAD (2D) and Inventor (3D).

While early CAD system offerings subsequently introduced substantially new modeling approaches, the focus of CAD vendors today is on iteratively improving productivity. Approaches like "shape morphing" in CATIA v5 [19] allow users to easily reshape freeform surfaces based on fixed feature points. Approaches like "direct modeling" from Spaceclaim (2007) or "synchronous technology" from Siemens NX (2007) aim at allowing the user to resize and reshape geometries more easily by intuitively pulling or pushing them with simple mouse-movements instead of typing parameter values into the forms, and by partially recognizing dependencies between parameters automatically. In the 2010th first digital platform (*Software as a Service, SaaS*) based CAD modeling environments were founded, such as on shape in 2012, which was acquired by PTC Inc. in 2019.

5.2 Digital Product Validation and Verification

While the previous section described the historical development of technologies for modeling geometries of a product, this section presents the development of validation and verification technologies.

5.2.1 Introduction into Validation and Verification (V&V)

Verification is the process of confirming that a technical system or a digital model (of a technical system) complies with all its specifications ("Did we build it right?"). Validation, on the other hand, is the process of confirming that a technical system complies with the customers' and all relevant stakeholders' expectations ("Did we built the right thing?") [20–25].

It is important to understand the difference between both. If a specification fully reflected all customers' and all stakeholders' expectations, then the process of verification would, at the same time, validate a technical system. However, this is usually never the case. Therefore, both processes must be performed along the product development process.

Verification happens at several stages in the product development process. It usually beginsg when the first digital models (e.g. geometries, simulation models, etc.) have been created. While the digital models only constitute parts of the whole technical system and while they are not physically built yet, they can be compared with specifications. When all partial digital models have been created, they should ideally be integrated (e.g. as virtual assemblies or co-simulation models) and then again be compared to the specification. Finally, when the real physical system has actually been built, it should again be compared to its specification. Verification can be performed by engineers completely and does not require the involvement of the customer or other stakeholders.

Validation can only be truly performed when a prototype of the technical system exists or when it has been finally physically built. Before this exists, validation can only be performed against a set of assumed performances of a product without sufficient confidence that this can actually be achieved (as it is, for example, the case in the quality function deployment approach). Virtual prototypes allow for partially validating a product before it is physically built. Nevertheless, the final physically built system that exists must be validated again. Validation must always involve customers and/or other stakeholders.

The specifications relevant for verification and building prototypes usually consist of requirements and digital models that describe a technical system's behavior. Requirements are first specified at product/system level and are subsequently broken down into detailed specifications for its subcomponents and parts. The discipline of requirements management provides methodologies for collecting and detailing requirements but is not focused upon in this section. It is thus assumed that detailed

requirements and related digital models already exist and need to be checked against each other. Requirements that are verified mainly comprise the following aspects:

- spatial constraints (boundaries) for geometries,
- kinematic behavior of parts,
- physical behavior of parts and forces,
- behavior of interrelated processes, and
- user experience.

Boundaries for geometries can refer to a static geometry or to the space that a geometry occupies considering its possible movement or positioning. The latter is tightly related to the kinematic behavior of parts that analyses how a set of parts that share geometric interfaces or simply a common space can be moved or positioned. A kinematic analysis thus provides the foundation for the verification of spatial constraints that consider kinematics.

The physical behavior of a part focuses on the interrelation between geometries (with specific material characteristics) and physical forces and movements that are applied to it. Typical examples comprise material deformation under different pressures applied, vibration of bodies, or movement of air or water on surfaces.

The behavior of processes is relevant when many different physical or digital processes are dependent on each other. For example, the technical execution of the physical function "braking a vehicle" involves many different system interactions such as the physical behavior between the ground and the tire and between the brake disc and the brake pad, the behavior of sensors that measure the forces that work on the brake and the behavior of the software that reacts to the sensors signals and that may control the brake pad in return. A separate analysis of all these system interactions without taking into considering the cross-effects may result in unforeseen behavior of the complete technical system. Therefore, this aspect is very important with regard to validation.

Finally, user experience is an important factor in order to focus on how a human user (the customer) perceives a product when interacting with it. It mainly focuses on the effects of product characteristics that may directly affect human sensory perception. These may comprise noise, haptics like textures of surfaces, odor, visible shapes, colors and different aspects of dynamic interactions.

In order to verify the different aspects mentioned before, different computer-based simulation technologies have been developed and evolved over the last decades. They are presented in the following sections.

5.2.2 Evolution of V&V Technologies and Computer Aided Engineering (CAE)

The development of the first algorithms, languages and theoretical approaches to simulate physical aspects of a system or process flows dates back to the 1930s. During

that time Enrico Fermi used Monte Carlo algorithms to calculate the properties of neutrons and presented according to numerical methods for investigating statistical problems. In the 1940s Jon Von Neumann and Stanislaw Ulam presented the roulette wheel technique that was applied to the same problem. The simulation of physical phenomena was thus one of the first fields for applied simulation approaches. At this time though computer technology was not available and hence the presented approaches were not yet implemented as software.

In the 1950s discrete event computer simulation was introduced. The IBM 650 computer was used and the algorithms were implemented in assembler language (i.e. not a high-level programming language as commonly used nowadays). In this case, no physical phenomena were investigated but abstract process flows. At this point they were not yet applied to engineering use cases.

In the 1960s computer simulation gained strong momentum and many different formal simulation languages (for describing simulation models and setups) were presented. Carl Adam Petri presented petri-nets as an approach to model process flows. Geoffrey Gordon presented the General Purpose Systems Simulator (GPSS), also an approach for simulating process flows, and applied it to the problem of weather prediction. Harry Markowitz, Bernard Hausner, and Herbert Karr presented SIMSCRIPT, a language for modeling and simulating events and schedules, and used it to simulate inventory problems. Ole-Johan Dahl and Kristen Nygaard presented the programming language SIMULA that was also used for modeling object flows through processes [26]. SIMULA build the foundation for later programming languages such as Smalltalk and thus introduced basic concepts for the object-oriented programming paradigm that is one of the most commonly applied approaches in software development nowadays. Further simulation languages comprised SOL (A symbolic Language for General Purpose System Simulation) from Don Knuth and J. McNeley, the General Simulation Program (GSP) by Keith Douglas Tocker and CSL (control and simulation language) from John Buxton and John Laski.

With all these new simulation approaches and technologies openly available manufacturer's interest in simulation increased. Companies like Boeing, Martin Marietta, General Dynamics, Raytheon, or Southern Railway built simulation groups that investigated the applicability of these approaches to their engineering-specific problems. At the same time, computer manufacturers like IBM, Control Data, and UNIVAC focused on providing suitable hardware solutions allowing the industrial application of simulation languages. Computer performance was limited at this point of time though, thus limiting the complexity of simulation models that could be simulated.

The 1970s continued where the 1960s ended, and further event- and process-centered simulation approaches and languages were presented at scientific conferences. Alan Pritsker presented multiple event simulation languages such as GASP IV, SLAM or SAINT [27]. Parkin and Coats presented a new algorithm for event-based discrete simulation [28].

But the 1970s also marked the advent of the first computer aided engineering (CAE) systems. This term summarizes software systems for finite-element analysis

(FEA), for computational fluid dynamics (CFD), and multibody dynamics (MBD). In the early 1970s the first three dimensional models for calculating fluid flows were introduced at Boeing [29]. In the late 1970s the FEA systems ANSYS and Abaqus were developed, providing means to companies to model, simulate and analyze material deformation or heat transfer problems. In 1977, Orlandea et al. [30] introduced the MBD system ADAMS (automatic dynamic analysis of mechanical systems) that allowed for calculating the kinematics of three-dimensional objects.

The introduction of CAE plays a major role from a product verification perspective. Process- and event-centered simulation approaches can be used to analyze abstract system behavior models, while CAE can be used to analyze geometry models. Hence, both approaches support different engineering activities at different stages of the product development process. Furthermore, the analysis of geometry models always matters when developing a (physical) technical system, while the analysis of abstract system behavior is only relevant for rather complex systems such as airplanes. Hence, CAE put simulation technologies on the map of many more companies, from tool machining companies to car manufacturers.

The 1980s mark an important change in the history of simulation technology. Computer hardware became significantly cheaper, thus also allowing smaller companies to profit from simulation software without having to commit to massive financial investments. With cheaper hardware, more powerful computers could be afforded and more complex simulations became possible. Furthermore, an increasing number of off-the-shelf software solutions was offered on the market, on one hand in the area of material requirements planning (MRP) for manufacturing and Computer Aided Process Planning (CAPP), and on the other hand for solving complex mathematical equations. While MRP and CAPP represent solutions focused on specialized engineering tasks, toolboxes such as Matlab, which was introduced in 1984, were generic solutions that could be applied for solving simulation tasks for different purposes. In addition to advanced math functionality, Matlab also provided a graphical user interface for modeling data flow and visualizing simulation results. Thanks to its large acceptance and deep market penetration it still is an important offering on today's market (marketed as SIMULINK since 1992).

MATLAB marks a cut in the way simulation software was used. While earlier, simulation models were programmed in a specific language, MATLAB allowed users to create simulation models graphically. Computer simulation thus became more accessible to a wider range of non-expert users. This trend continued in the 1990s and nowadays all important simulation software systems provide such graphical modeling interfaces.

Another noteworthy innovation that happened in the 1980s was the first introduction of a virtual reality setup with a head-mounted display (HMD) that included a motion tracking system (at the University of North Carolina). In 1989 VPL Research spawned the first commercial offer of such an HMD, called the "EyePhone".

In the 1990s simulation systems and computer hardware became increasingly powerful, yet no substantial theoretical innovations were introduced. Manufacturing planning was the most common application scenario for process-centered simulation approaches. The market of off-the-shelf software solutions for computer simulation

expanded and consolidated. Systems such as GPSS, EXTEND, MAST, Micro Saint were developed, replacing former solutions that required programming. It is important to understand though that even such graphical simulation modeling systems still require the programming of scripts to some extent. Since different systems employed different proprietary scripting languages, Hilding Elmqvist introduced Modelica in 1997. That is an object-oriented language for the modeling of technical systems providing a standardized format for reusing and exchanging dynamic system models. Modelica is still used today in many simulation software systems such as SimulationX or Dymola.

The 2000s marked the advent of hardware-in-the-loop (HIL) simulation where real, physical electronical/or mechanical components are connected to a simulation software (through sensors and actuators that are connected to computer interfaces). This allows for verifying the interaction of multiple electronical, mechanical and software components where a part of the components already exists physically and other parts are still under development. Such functionality is often provided by development tools for modeling and programming the data flow between electronical components, such as LabView (first introduced in 1983) or dSPACE (first introduced in 1988). HIL is widely used in the development of cars and trucks but also in the development of all other kinds of mechatronic systems.

Basic Explanation of simulation approaches and technologies

Simulation Technologies have evolved with one main goal in mind: minimizing the efforts of testing physical prototypes. Instead of building a costly physical prototype, simulation software allows for testing a virtual prototype instead. Since the second half of the 90s an overall Digital Mock-Up (DMU) can be created if all geometries are well structured in a product information database, and a broad range of different digital models exist to allow for specific virtual prototype simulations.

This approach also allows for testing a product (or one of its components) early in the design process, i.e. even before aspects such as manufacturing need to be considered. Hence, problems can be discovered earlier and the duration of development iterations can be shortened. Finally, manual testing tasks can be automated, further lowering testing costs.

As emphasized in the previous section there exist different simulation approaches, each one suited for different verification purposes.

Spatial constraints (boundaries) for geometries are usually verified directly in a CAD environment and do not require additional simulation software. Modern CAD environments meanwhile provide easy-to-use clash analysis functionality, which for a long time was a privilege of specialized DMU tools only. When an engineer places multiple CAD parts in one shared space, the CAD environment is able to analyze the resulting assembly and identify all spots where parts ‘collide’. If parts are moveable then their kinematics can be modeled in the CAD environment, too, and the clash analysis functionality will consider the whole space that each moveable object may occupy in any of its possible positions.

Usually though, CAD environments do not provide means for modeling physical behavior. While they can detect clashes of parts they cannot compute what exactly

happens if these parts interact with each other with specific forces applied to them. Analyzing the physical behavior of a product or its parts thus requires specialized CAE simulation software.

In order to analyze physical product behavior, continuous dynamic simulation approaches are applied. This is what FEA models are used for, which have been mentioned in the previous section. Examples for physical behavior comprise:

- the way a car body deforms when it crashes into another object,
- the turbulences resulting from a current of wind meeting an airplane's wings or
- the vibration required to make a building structure collapse.

In the continuous dynamic simulation, a geometry model is translated into a set of differential–algebraic equations modeling continuum mechanics. Since solids and fluids behave differently, different models are used for describing solid mechanics and fluid mechanics. The car body deformation and the collapsing building structure are both examples for solid mechanics. The air turbulences are an example for fluid mechanics.

The algebraic equations are then solved by mathematical algorithms and the results are reflected back into the geometry model. This allows for visualizing them in a geometrical representation. Often, physical behavior (such as the degree of deformation measured in millimeters or the range of movement during vibration) is also visualized in charts and diagrams.

While the continuous dynamic simulation focuses on geometry and physics it is often also desirable to analyze the behavior of disembodied things such as signals or data flow. This is especially interesting in electronical and mechatronics engineering where components usually do not interact through the application of physical forces but the sending and receiving of electronic signals. This is what the process-centered simulation approaches are used for, that have been presented in the previous section.

In process-centric simulation models, functional components of a system are modeled as a graph of nodes that are interconnected through edges that transfer quantifiable signals (e.g. in software such as Dymola, LabView or SimulationX). Each node may have multiple input and output edges and it processes inputs into outputs. For each edge, a direction and a signal type (e.g. a visual signal such as light at a specific luminosity or a data input stream of digits) is specified. Each node can be modeled as a mathematical function with the signals from the incoming edges signals as its parameters.

Therefore, each functional component's behavior can be modeled separately and finally all functional components can be simulated in their aggregated behavior. Usually, such simulations reveal where components may receive input signals that are not out of their accepted range of values (e.g. a light signal that is too dark or too bright and that can thus not be measured properly by a photometer) or where functional components fail (either because they generate wrong outputs or because they do not generate outputs at all, e.g. in case of unsolvable mathematical equations). Such simulations are also used to optimize the behavior of functional components (by fine-tuning the mathematical function that represents their behavior).

As mentioned in the previous section, process-centric simulation models can be developed before the functional components themselves are developed and they can provide useful insights on the constraints for interfaces between functional components. At a later stage of the product development process, when functional components have been designed, the virtual models of the functional components can be used as inputs for continuous dynamic simulation software. That other simulation software (with one specific virtual model of a functional component) can then be linked to the process-centric simulation model, replacing the mathematical function of one node (that was only based on an assumption earlier in the product development process). Hence, the initially assumed behavior of one functional component can be replaced by its actual behavior (assuming that the continuous dynamic simulation model is valid). Finally, all developed functional components can be "co-simulated" and their real interplay can be analyzed and validated.

While process-centric simulation models are used for modeling the behavior of a system with respect to the input and output signals that its different components receive and generate, such models often do not provide any insight on the temporal aspects of a system's behavior. Functional components of a system send signals from one to another but sometimes it is essential to know at which point in time these signals are sent and how long one component needs to wait for another to send a specific signal. This is a very similar problem to that in business process or project planning where one wants to minimize idle times in the process/project but also wants to ensure that single activities have enough buffer time in case of unforeseen events.

In such cases, state machines (or process models with underlying state machines) are used for modeling the system behavior. Similar to process-centric simulation models, functional components are modeled as a graph of nodes that are interconnected through edges. Each node is modeled as a set of attributes, such as duration or likelihood of failure. Often, minimum, maximum and average values can be specified for such attributes. In addition to nodes that represent functional components, there also exist nodes that guide the process flow (decision, parallelization or synchronization points). This allows for modeling parallelization and iteration.

5.3 Product Data Management (PDM)

When the first 2D CAD systems were introduced in the late 60s no file systems existed yet and data could not be transferred through a computer network. That means the created drawings could not be saved as a local file in a folder on a computer and they could not be sent to a server that provided storage functionality. Instead, they could either be plotted/printed or saved on a magnetic tape. The management of the created models thus involved manual tasks dealing with physical objects (i.e. plots or magnetic tapes) that had to be stored in some physical storage place. Since drawings were made manually for decades before the introduction of the first CAD systems, approaches for the storage, indexing and access existed already. Nevertheless, these

approaches relied on human users for indexing, searching and securing engineering results, they required physical storage space, and the distribution of engineering results required physical copying and distribution through classical postal or delivery services.

When the first wave of 3D CAD systems was introduced in the late 70s, the first file systems (e.g. System VFS, FAT) and the first data exchange protocols for computer networks (e.g. Ethernet, ARCNET) had already been presented by researchers, but were not yet widely adopted in business practice. Drawings were still managed physically.

It was only in the 1980s that the physical management of drawings (and all other kinds of documents) started being partially replaced by the digital management of files. More powerful file systems like the Berkeley Fast File System were introduced that allowed for storing drawings and documents directly on a computer. Also, the first relational database systems (e.g. POSTGRES, INGRES) were introduced. While they did not allow for managing complex data like documents or drawings, they could manage huge amounts of small data, and could thus be used to store and manage information about suppliers, orders, customers, etc. Data bases were used in software systems (introduced in the beginning of the 1980s) that allowed for managing metadata about paper-based documents digitally. That means that metadata (i.e. information about creation data, document type, author, version, etc.) was managed digitally while the corresponding documents were still stored physically. Later on these systems evolved into so-called Electronic Document Management (EDM) [31] systems and could then also manage documents digitally (in a file system), hence rendering physical storage obsolete. Examples for early EDM systems comprise SoftSolutions (1979), Saros Mezzanine (1986) and PC Docs (1989) [32–35]. Today such systems are called Document Management Systems (DMS).

In order to allow companies to "migrate" older documents, that only existed physically, into digital documents these systems also provided document imaging (i.e. scanning) functionality. Furthermore, text-analysis algorithms would allow for indexing text-based documents semi-automatically thus saving indexing efforts. And finally, documents could be searched using full-text search.

While the advances in file systems, network protocols and the introduction of EDM systems provided significant advantages for managing documents, they did only address "generic" data management challenges (e.g. indexing, searching, storing, etc.). Product development faced specific challenges though, that these systems did not address, mainly revision and configuration control and the management of the lifecycle of product data.

Version, revision and configuration control is an important field of activities in product development because one component can be used in multiple different product versions or configurations. Hence, one single version of a CAD model could be a part of different assemblies or a part in different bills of materials (BOM). While simple version control usually only allows for saving consecutive versions of one document (i.e. 1, 2, 3, etc.), in product development one document may exist in different versions in different "contexts" (e.g. assemblies, BOM, etc.). This complexity could not be handled with early EDM systems.

The other challenge with respect to product data is its life cycle. Different engineering artifacts (i.e. requirements, system models, CAD models, etc.) often go through different development and release stages (e.g. idea, concept, design, released design, etc.). Each of these stages may affect the access rights for the corresponding product data and the way it is stored and versioned. Early EDM systems did not allow for specifying such characteristics and all documents were simply treated the same way.

While the amount of product data increased steadily in the 1980s, the challenges mentioned before showed that there was a need for specific IT support for the management of product data. As a logical consequence, especially large manufacturers like Boeing or Ford with strong in-house research and development departments would develop their own company-specific PDM systems. For example, Ford's PDGS system would feature a component called Data Collector that provided PDM functionality connecting globally distributed development centers [3]. Smaller companies, on the other hand, were less affected by the challenge of overwhelming amounts of product data, but first and foremost they simply could not afford to develop their individual solutions.

The first PDM software that was sold on the market was SherpaWorks from Sherpa, that was released in 1984 [36]. In 1989 IBM introduced a PDM software called ProductManager [4, 37, 38]. But it was only in the 1990s that the market for PDM systems grew significantly. In the early 1990s Unigraphics and SDRC, two companies that already offered CAD solutions at that time, released respective PDM offerings (Unigraphics iMan in 1991 and SDRC Metaphase in 1992). In the late 1990s other CAD vendors followed their example, and in 1998 PTC released Windchill and Dassault Systemes released Enovia. BAAN introduced BAAN PDM in 1996 [39] and Eigner + Partner introduced CADIM/EDM in [40]. Hence, the "new" PDM market was (mainly) shared among CAD vendors, thus explaining the initial focus of most PDM systems on the management of CAD models. Many other types of product data, such as requirements, simulation models and results or factory layouts would still be managed outside of PDM systems. The CAD vendors, realizing this maladjustment, would thus redefine their image from CAD vendors to "Product Life-cycle management (PLM) solution providers" in the late 1990s, and enhance their products with corresponding, additional functionality.

It should be noted that the term PLM is not only limited to the management of product lifecycle information or data within a specialized IT system. PLM also comprises the management of information and information flow between processes at a more general level. Eigner and Stelzer even refer to PLM as a solution strategy [41]. A PLM system alone can thus not cover all aspects of PLM.

Today, PLM systems support the management of almost any kind of product data. Their typical components are:

- a central data vault where all product data is securely stored,
- a workflow engine for controlling product data centric processes such as release processes,
- user interface components for handling,

- bills of materials/product structures,
- product configurations,
- version management,
- (standard) parts management, and
- project management.

Additional functionalities typically provided by PLM systems comprise [42] advanced search, file conversion, secure file transfer, task management and/or change notifications.

Together with a suitable PLM strategy, these functionalities aim to provide the following business benefits [43]:

- to save development time and cost through the reuse of parts, modules, platforms, etc.;
- to reduce the amount of engineering changes after the start of production through better support of V&V activities in the early PDP stages;
- to improve collaboration through well-defined processes and responsibilities;
- to confidently ensure the availability of relevant data;
- to increase the amount of time engineers spend on innovative and value-creating activities through reducing the efforts for laborious data management activities, and
- to provide continuous support of business processes through the reduction of information gaps between heterogeneous IT systems.

While initially PLM systems provided mostly data management functionality, current PDM systems support all kinds of processes, either through workflow functionality or through specialized, task-oriented plugins (e.g. requirements management views) for the graphical user interface. Nevertheless, they usually do not cover data and processes management from the entire product lifecycle, but only from the beginning of a products' life, the product development phase.

There exist multiple reasons for the PLM systems' focus on product development, the most important one being that later phases of the product lifecycle are often managed not by the same company that develops the product, but by external partners. Reaching an agreement on a common PLM approach in such an Extended Enterprise setting can be time-consuming and challenging [44]. Hence, traditionally, IT systems are used by one company only and companies do not interlink their IT systems or use shared IT systems. Instead, the different companies that are involved in the product lifecycle manage their own data and processes separately. As a result, there exists no single IT system that supports data management and process support for all phases of the product lifecycle, but a variety of specialized IT systems in each different phase of the life cycle.

Another practical reason stated by Grieves [45] is the fact the whole lifecycle of a product may last up to 100 years which is much more than the typical lifetime of an IT system. At the start of production of a product (often after multiple years of development) the initially introduced PLM system may already be out of date. If for that reason another IT system is introduced for managing information from the later

phases of the product lifecycle, then it makes no sense for PLM vendors to cover these phases in the first place.

Nevertheless, PLM system vendors are still aiming to provide solutions for later phases of the product life cycle, too. Currently, data from the use and the end-of-life phase of a product is often managed only in ERP (Enterprise Resource Planning) systems, if at all. Since ERP systems are focused on business-centric topics such as sales numbers, parts supply, logistics, etc. important information relevant for engineering is often not collected in them. Hence, there exists a demand for managing engineering-relevant information in these lifecycle phases that is likely to be addressed by PLM vendors in the near future. While there already exist partial solutions for supply-chain management and factory data management, the management of information about a product's usage, wearing, maintenance and disposal is still poorly covered.

Existing IT solution offers from competitors (for later product lifecycle phases), such as ERP, pose a practical challenge for this extension of PLM system's functionality though. PLM system vendors must penetrate new market areas facing stiff competition. It thus remains to be seen whether or not this will hold PLM systems back from actually covering the entire product lifecycle somewhere in the future. Please refer to Chap. 11 to gain more insight to PDM/BOM and PLM.

References

1. Krull F (1994) The origin of computer graphics within General Motors. IEEE Annals Hist Comput 16(3):40. https://doi.org/10.1109/MAHC.1994.298419
2. Rake W (1973) Computer-Aided Design and Drafting (CADD): an advanced designer's tool. SAE Technical Paper 730934
3. Whitney DE (1993) History of CAD/CAM at Ford
4. Weisberg DE (2008) The engineering design revolution. the people, companies and computer systems that changed forever the practice of engineering
5. Spur G, Krause FL, Harder JJ (1982) The Compaq solid modeler. In: Computers in mechanical engineering, New York, pp 44–53
6. Whitney DE (1991) Visits to Prof Kimura's CAD research Lab July 5 and July 19 for discussions about product realization, IMS, and Product Development Cycles
7. Kimura F (2009) Geomap-III: designing solids with free-form surfaces. Comput Graph 4(6):58–72
8. Krause JL (1997) Virtual clay modeling. In: Pratt MJ, Sriram RD, Wozny MJ (eds) Product modeling for computer integrated design and manufacture. Chapman & Hall, London, pp 162–175
9. Reintjes JF (1991) Numerical control. Making a new technology. Oxford series on advanced manufacturing, vol 9. Oxford University Press, New York
10. Shah JJ, Mäntylä M (1995) Parametric and feature-based CAD CAM. Concepts, techniques, and applications. a Wiley-Interscience publication. Wiley, New York
11. Bernardi A, Klauck C, Legleitner R, Schulte M, Stark R (1992) Feature based integration of CAD and CAPP. In: Krause F, Ruland D, Jansen H (eds) Informatik Aktuell. CAD'92 Neue Konzepte zur Realisierung anwendungsorientierter CAD-Systeme, Berlin, pp 295–311
12. Rieger E (1994) Semantikorientierte Features zur kontinuierlichen Unterstützung der Produktgestaltung. Produktionstechnik - Berlin, vol 158. Hanser, München, Berlin

13. Kent JR, Carlson WE, Parent RE (1992) Shape transformation for polyhedral objects. In: Proceedings of the 19th annual conference on computer graphics and interactive techniques. ACM, New York, NY
14. Brill M (2006) Parametrische Konstruktion mit CATIA V5. Methoden und Strategien für den Fahrzeugbau. Hanser, München [u.a.]
15. Krause F, Jansen H (1993) Luth N (1993) Neue Methoden der automatischen Zeichnungsinterpretation. Zeitschrift für wirtschaftliche Fertigung und Automatisierung 88(12):589–592
16. Spur G, Jansen H, Krause F (1986) Automatische Digitalisierung und Interpretation technischer Zeichnungen für CAD-Prozesse. ZWF CIM 81 (1986) 5:235–241
17. Spur G, Jansen H, Krause F (1986) Verarbeitungstechniken zur automatischen Zeichnungserfassung für CAD-Prozesse. ZWF CIM 81(5):235–241, 9:460–466
18. Luth N (1997) Eine Methode zur automatischen Strukturinterpretation in digitalisierten technischen Zeichnungen. Berichte aus dem Produktionstechnischen Zentrum Berlin. IPK, Berlin, Berlin
19. Ali A, Brebbia CA (2006) Digital architecture and construction. WIT transactions on the built environment, vol 90. WIT, Southampton
20. Aiaa guide for the verification and validation of computational fluid dynamics simulations. Aiaa, [S.l.]
21. Author Unknown (1979) Schlesinger terminology for model credibility. In: Simulation, vol 32, No 3, pp 103–104
22. American Society for Quality (ASQC) (1978) Quality systems terminology (A3-1978)
23. ASME (2006) Guide for verification and validation in computational solid mechanics
24. IEEE (1990) Standard glossary of software engineering terminology (Std 610.12-1990)
25. Kapurch SJ (2007) NASA systems engineering handbook. Retrieved from https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook.pdf. Accessed on 07 Nov 2017
26. Nance RE (2002) RGS Perspectives on the evolution of simulation. Oper Res 50(1):161–172
27. Rossetti MD, Hill RR, Johansson B, Dunkin A, Ingalls RG, Goldsman D, Nance RE, Wilson JR (2009) A brief history of simulation. In: Dunkin A (ed) Winter simulation conference. Winter simulation conference, s.l, pp 310–313
28. Nance RE (1993) A history of discrete event simulation programming languages. The second ACM SIGPLAN conference on History of programming languages. ACM, New York, NY, pp 149–175
29. Robert PE, Saaris GR (1972) Review and evaluation of a three-dimensional lifting potential flow computational method for arbitrary configurations. Boeing Co.
30. Orlandea N, Chace MA, Calahan DA (1977) A Sparsity-oriented approach to the dynamic analysis and design of mechanical systems. J Eng Industry
31. ISO Electronic document management -- Vocabulary -- Part 1: Electronic document imaging (12651-1:2012)
32. Author Unknown (1990) UTAH software firm forms new marketing company. Deseret News
33. Klaproth F (1998) Lossau N. The document management system Saros Mezzanine and the new product AGORA as key component in a digital library architecture at Göttingen University Library 1513:685–687
34. Delphi Consulting Group (1995) In its newest market research service, document management: the paperless revolution. Delphi Finds Three Vendors Dominate Market
35. Chen SCM (1994) The document masters: PC DOCS, Saros, and SoftSolutions. PC Magazine, pp 316–317
36. CADAZZ (2004) CAD software - history of CAD CAM. http://www.cadazz.com/cad-software-history.htm
37. IBM (1997) IBM ProductManager Version 3 Release 2
38. Sendler U (2009) Das PLM-Kompendium. Referenzbuch des Produkt-Lebenszyklus-Managements. Xpert.press, Springer, Berlin, Heidelberg
39. Baan Development B.V. (1998) BAAN IVc4. BaanPDM 5.1.3 DBA Guide. Retrieved online from: http://baansupport.com/docs/baan/U7175AUS.pdf

40. Schmidt S, Wierschin H (1998) Global Engineering auf der Basis weltweit verteilter Daten und Dokumente. In: Industrie-management, 10/1998
41. Eigner M, Stelzer R (2013) Product lifecycle management. Ein Leitfaden für Product Development und Life Cycle Management, 2nd edn. VDI. Springer, Dordrecht
42. Saaksvuori A, Immonen A (2008) Product lifecycle management, 3rd edn. Springer-Verlag, s.l
43. Feldhusen J, Gebhardt B (2008) Product Lifecycle Management für die Praxis. Springer, Berlin, Ein Leitfaden zur modularen Einführung, Umsetzung und Anwendung
44. Stark J (2009) Product lifecycle management. 21st Century paradigm for product realisation, 1st edn. Springer, London Limited, s.l
45. Grieves M (2006) Product lifecycle management. Driving the next generation of lean thinking; [how GE, P&G, Ford, Toyota, and other leading companies achieved dramatic increases in productivity and profit]. McGraw-Hill, New York