2nd Conference on Production Systems and Logistics

Decisions And Characteristics During The Development Process Of A Software Demonstrator For Data Analysis In Production Logistics

Janine Tatjana Maier[1], Simon Berger[1], Tammo Heuer[2], Peter Nyhuis[2], Matthias Schmidt[1]

[1]*Institute of Product and Process Innovation, Leuphana University of Lüneburg, Universitätsallee 1, 21335 Lüneburg, Germany*
[2]*Institute of Production Systems and Logistics, Leibniz Universität Hannover, An der Universität 2, 30823 Garbsen, Germany*

Abstract

Software demonstrators enable the transfer of theoretical concepts into industrial practice and therefore are used in various research areas. Although the stages of the development of a software demonstrator are already described in literature, it lacks an examination of fundamental decisions and characteristics throughout the development process. Characteristics, such as transparency, hold across all research areas and industries. A specification and extension depends on the individual use case. This paper describes the necessary decisions regarding the framework of a software demonstrator for data analysis in the field of production logistics and the characteristics applicable before and during the actual implementation of the software demonstrator. Key decisions are for example the design of the software architecture or the choice of the programming language. The characteristics equal the requirements to a certain extent and relate directly to the users and to the application itself. Discussions with companies from a wide range of industries revealed underlying conflicts and diverse priorities in terms of the aspects of functionality. Especially the size of the company seems to play an important role regarding the expectations of a software demonstrator. Therefore, the integration of potential future users into the development process of a software demonstrator presents a major advantage. It reduces the number of cycling stages at the end of the development process and increases the willingness to use the software demonstrator after its completion. As the objectives behind software demonstrators can highly differ in the field of production logistics, concepts of demonstrators and software engineering and the related objectives were examined. A demonstrator for determining the order processing strategy serves as an example of how these different approaches interrelate.

Keywords

Software demonstrator; Decision making; Application-related characteristics; User-related characteristics; Order processing strategy

1. Introduction

Demonstrators point out the usefulness of concepts or new technologies. Functioning as testing and experience-generating tools, they help to remove the barriers between theory-based research and actual application in day-to-day industrial practice. Highlighting the potential of applications for a variety of industries, demonstrators often appear in showrooms or at industry fairs. In this case, the appearance as well as the transportability of the demonstrators play a major role. Increasing digitization leads to changes in the nature of demonstrators. There is a noticeable growth in all-software demonstrators and a trend towards combining traditional physical demonstrators with software demonstrators.

Software demonstrators offer the opportunity to automate the calculations of complex models and thus allow access to scientific solutions even for people unfamiliar with the subject. A popular example is a quality control system supported by machine vision [1]. Researchers discuss the use of intelligent systems for planning and controlling production processes for decades. For example, already in 1992 Yang et al. analyzed the similarity of components for the automated generation of process plans [2]. However, the transfer of similar procedures into industrial practice has just started to pick up speed in the last few years. Reasons for this are the increasing availability of data and the rising acceptance of these methods by companies. Nevertheless, a widespread use is still not visible. While companies appreciate the potential of intelligent systems for decision making in production planning and control [3], they also recognize the dependency of the quality of the decision on the quality and integrity of the used data [4]. The analysis of huge amounts of data presents a challenge, especially for small and medium-sized companies. In this area, software demonstrators can provide assistance to a certain extent.

In the field of production logistics various software demonstrators have been developed so far. They range from presenting cause-effect relationships [5] to tools providing assistance to apply machine learning methods [6]. Regardless of the different interpretations and applications of software demonstrators in this field, they are all built upon data analysis. Recent studies indicate that there is still a shortage regarding skilled worker in this field [7]. Companies have the opportunity to compensate this through hiring external experts at high cost. However, the development of software demonstrators for research purposes cannot be outsourced. Instead, this requires a holistic approach including knowledge of theoretical models and scientific research methods. For an adequate support, an examination of the fundamental decisions and characteristics throughout the development process of a software demonstrator is missing. A description of these decisions and characteristics can help to improve the development process and allows focusing on the specifications and extensions for the individual use case.

To incorporate different views on the term software demonstrator and to identify the criteria and influences a literature research based on the established scientific databases Scopus, Web of Science and Google Scholar [8] was carried out. Besides standard literature on software engineering [9, 10], different concepts of demonstrators and the related objective, such as proof of concept, were examined. To evaluate the results and derive conclusions, interviews with companies from a variety of industries were conducted. This included small companies with around ten employees to large multinational companies. Apart from manufacturing companies, logistics service providers, consulting firms and software developers were considered. In the following, a demonstrator for determining the order processing is described to provide an example for the application of the presented general framework. Figure 1 shows the simplified architecture of the software demonstrator for determining the order processing strategy. As an abstract description of the overall system the architecture contains the general functionalities, the modules as well as the methods and patterns. Thus, it acts as a bridge between the requirements and the actual programming [11].

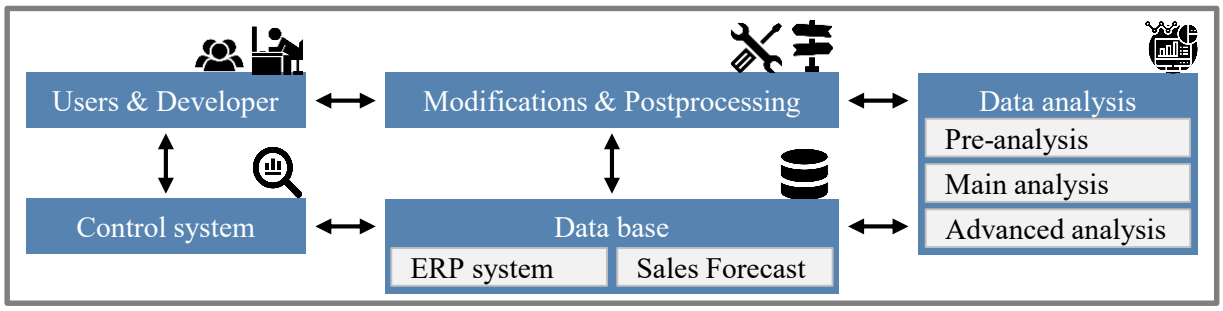


Figure 1: Simplified architecture of a software demonstrator using the determination of the order processing strategy as an example.

The subsequent section describes the development process of a software demonstrator and its challenges. This provides the framework for the analysis of the decisions and characteristics throughout the development process in section three. Furthermore, the characteristics are linked to the application and the users. Lastly, the conclusions of the paper are summarized and future research possibilities are outlined.

2. Development Process of a Software Demonstrator

The principles of a software development process have been examined widely in the literature. Already in 1979, DeMarco described the five phases of software development [12]. Figure 2 shows these phases and highlights the challenges of the application to the development process of a software demonstrator for data analysis in the field of productions logistics. The challenges are derived from literature research, interviews with companies and experience from previous research projects.

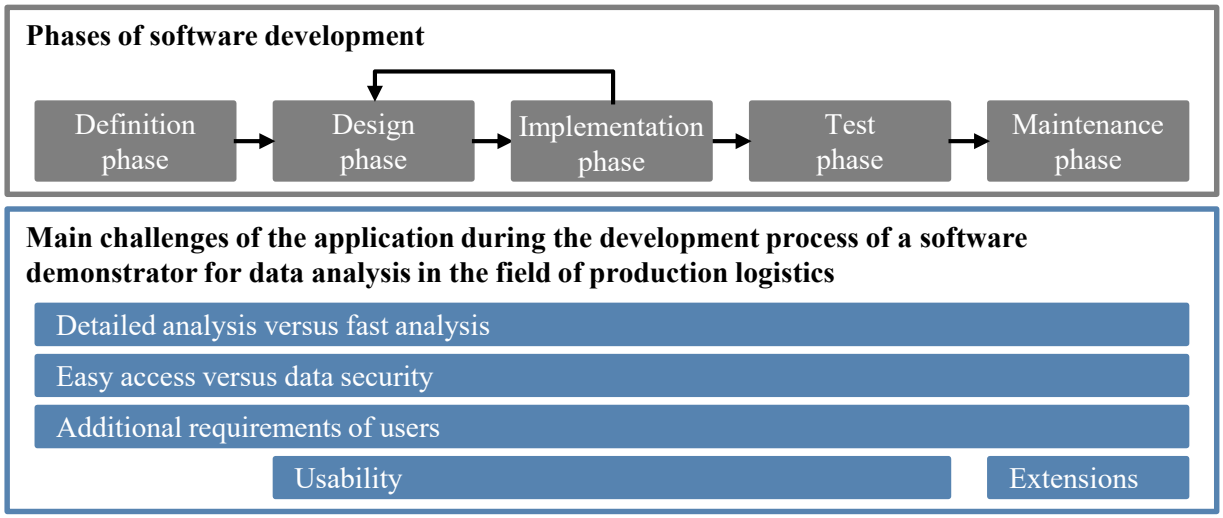


Figure 2: Main challenges for the application of the five phases of software development after DeMarco [12] during the development process of a software demonstrator

The first step is the exact and complete definition of the methodology to be developed. Already during the definition of the functionalities developers face various challenges. An overall objective of software demonstrators is a broad applicability. This leads to a conflict between a detailed analysis and a fast analysis. In addition, easy access as well as high data security standards are required. This is especially important as industrial data are often confidential. These conflicts remain relevant during all phases of the software development process. When transferring theory-based research findings into a software demonstrator, the existing theoretical validated framework can be used as reference during the definition and design phase. As the users of a theoretical model and a software demonstrator can differ additional requirements needs to be taken into account.

Actively involving the interest groups during all phases helps to prevent a deviation between expectations from the actual functionality of the software demonstrator, and thus minimizes the number of cycling stages resulting from non-occurrence. As an attempt to reduce the complexity, researchers further differentiated the definition phase. Balzert provides a good example by distinguishing between requirements in terms of functionality, performance, quality and framework conditions [13]. Models from the area of software development such as the waterfall model [14] or the spiral model [15] are early examples of the importance of a detailed analysis of the realization of the development processes. Such differentiations and descriptions are beneficial. Nevertheless, the identification of the future users is essential as differing needs can have a high impact on the software demonstrator [16].

In this context, it is important to understand the existing types of interest groups and evaluate the benefits of extending the system. Existing software demonstrators, such as the FRDISI [17] or the 4SECURail demonstrator [18], point out diverse interest groups for their respective areas and mainly base the general architecture of their demonstrators on these. This supports the assumption that a key factor behind the generation of the so-called usability and the associated user acceptance is the involvement of future users in the entire development process. The success of the incorporation of the results provided by the software demonstrator in a company highly depends on the acceptance of the employees regarding new technologies or processes. A human-centered design process helps to incorporate the user in an appropriate way. Besides other characteristics, a human-centered design process understands and actively involves the users and according to ISO 13407 [19]. Combining approaches of human-human and human-computer interaction research creates the basis for identifying the actual underlying criteria of such a process [20]. Trending topics such as artificial intelligence push this idea further [21]. Depending on the individual use case, additional approaches from fields like decision support systems [22] or simulation [23] can complement these criteria.

The design of user interfaces directly relates to usability. In 1993, Nielsen defined practices for usability engineering [24]. For novel technologies and concepts, trustworthiness is one of the most critical aspects. Mathis et al. show how personalized interfaces create trustworthiness, persuasion and even affection towards automated transport vehicles [25]. Similarly, a study on mobile services adoption points out the importance of trust and the user's desire to be in control [26]. With increasing digitalization and the growing interest in topics like artificial intelligence, the requirements of users broaden and result in design principles for specific topics, such as Industry 4.0 components [27]. The rising level of expectations has a strong impact on the data requirements. Data should not only be available, but also transparent and able to process in real-time [28]. The usability of data also demands easy data inputs and modifications of the data base [26]. For example, well-known technologies such as radio frequency identification provide opportunities for real-time data processing and therefore enable dynamic decision making [29].

Regardless of the degree of involvement of future users, an evaluation of the usability before, during and after the implementation of a system is mandatory. Already in 1997, Lin et al. proposed an index to measure the usability of software interfaces [30]. They take a wide range of requirements into account such as the fit to the user’s expectations, the consistency, the flexibility, the learnability and the provided guidance. A variety of methods exists to conduct evaluations throughout the development process. As a way to eliminate obvious usability problems prior to an actual test, it is possible to carry out a cognitive walkthrough [31]. In terms of an actual test, Holzinger defines thinking out loud as one of the most valuable methods to identify shortcomings regarding usability [32]. Scholtz provides a good overview of evaluation methods [33].

3. Decisions and Characteristics

Throughout the development process of a software demonstrator, multiple decisions are required. The decisions are mainly driven by the defined objectives, but also directly interrelate with the users and the application itself. In the following, possible influences and criteria are outlined. Figure 3 visualizes the key decisions and characteristics during the development process of a software demonstrator for data analysis in the field of production logistics. It shows the underlying conflicts and highlights the importance of a continuous analysis, validation, and verification. The characteristics partly equal the requirements. Characteristics can influence the development process, but their existence is not necessary in every case. Requirements are considered mandatory for the development process of a software demonstrator. For example, the project lead-time can be seen as an externally given specification and have a strong impact on the entire developing process. Therefore, it classifies as a characteristic and as a requirement. The pre-existing infrastructure influences the development process of a software demonstrator, but the existence of an instructor prior to the development process is not necessary. Thus, it is only a characteristic.

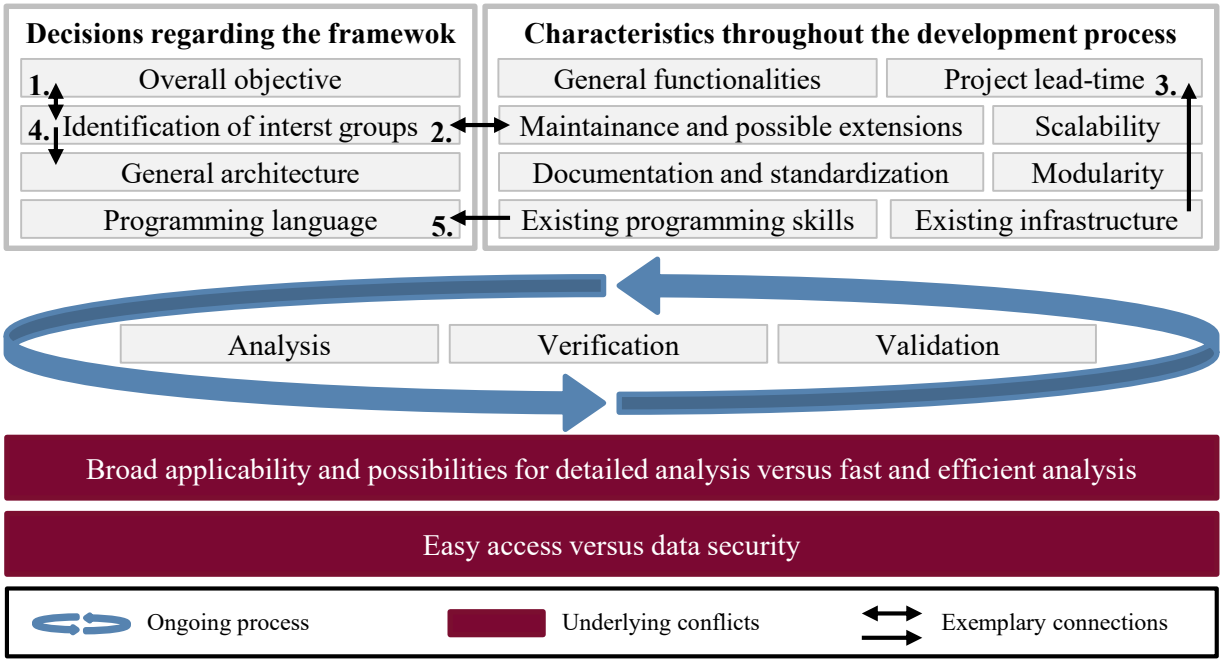


Figure 3: Decisions and characteristics, ongoing processes and underlying conflicts during the development process of a software demonstrator

In the following, the individual aspects are described in more detail and examples of connections based on a software demonstrator for determining the order processing strategy are given. It should be noted, that all decisions and characteristics are directly or indirectly connected. Defining the overall objective of a software demonstrator represents the beginning of the development process. The development requires a considerable amount of resources such as time, effort and money, and is a never-ending process in case the software demonstrator changes along with evolving companies over the years. To limit the effort the project lead-time and the general functionalities should be specified prior to any decisions. In contrast to commercial software development projects, there is no contract detailing these aspects. Nevertheless, building a software demonstrator based on an existing theoretical model has a major advantage. The concept has usually already been validated, for example by simulation studies. Thus, the risk of a project failure is comparatively low. The overall objective is directly related to the general functionalities and the project lead-time. The definition of the overall objective leads to the obvious interest group, the users. Identifying additional interest groups and incorporating their requirements can lead to adjustments of the objective definition (figure 1, arrow 1). The objectives of all interest groups should withstand in the real world. In the case of programming software, a wide range of possible requirements exists and thus additional support is required to merge the requirements together. Nuseibeh and Easterbrook suggest a roadmap to specify the requirements of software products [34]. Moreover, further classifications of the requirements such as a distinction between functional and non-functional can be beneficial. Aurum and Wohlin provide a list with examples for different requirement classifications [35]. Risk management approaches suggest to confront the interest groups in case of non-occurrence and to rank all requirements. Disguising non-occurrence will simply postpone the problem and even intensify it [36]. Therefore, the first and probably most continuously present decision is the selection of the interest groups. This calls for an additional distinction between the users and the generally possible interest groups. For the majority of cases the interest groups equal the users. Nevertheless, people do not need to be direct users to influence the development process. This includes people related to the data collection and preprocessing as well as people whose work is influenced by the results of the demonstrator. Neglecting to align the demonstrator with data acquisition and preprocessing results in the need of additional data standardization, cleansing and format transformation. Although this is possible without programming skills, for example through software such as KNIME [37], these additional process steps affect the usability of the demonstrator negatively.

Apart from that, it is important to know about the data already collected but so far not used and the cost-benefit ratio between the gathering of additional data and the quality of the results from the demonstrator. Involving the responsible parties eases the estimation of the necessary time and effort due to existing experience and even helps to reduce barriers regarding possible future extensions of the data collection. The interpretation of data requires a considerable amount of effort as well. Therefore, the results from the demonstrator need to be easy to evaluate and appropriate actions should be derivable without a high amount of effort and the consultation of experts [22]. A way to achieve this is the identification of tools, which comely exist in companies and possibly will be used for postprocessing the data of the demonstrator. This also raises the question who will maintain and potentially upgrade the system from a long-term perspective. Unless these tasks belong to the developer himself, more people need to be actively included in the development process (figure 1, arrow 2). Especially important in this context are a high level of detail in the documentation and the use of standards. On top of simplifying maintenance and extensions, standards provide the foundation for scalability [38]. Individual use cases imply different levels of detail and complexity. The main challenge is to enable a broad applicability and possibilities for detailed analysis, but at the same time provide a fast and efficient analysis [22].

The demonstrator should be applicable to both small and very large companies [23]. The number of people potentially connected to the demonstrator as well as the expectations tend to increase along with the company size. Small companies tend to focus on aspects such as reusability due to the usually more limited resources available for development. Using pre-defined elements can help to achieve a higher level of detail while meeting the defined project lead-time (figure 1, arrow 3). In addition, not needing to start modeling from scratch reduces the occurrence of errors [22]. On the other hand, fitting a new system to an existing infrastructure can result in a less efficient analysis. Aiming for a reusable and scalable system generates interest for more companies and therefore is a common objective for demonstrator developments [38]. Adaptability to different companies includes different industries as well as a variety of products and processes. Thus, modularity is one of the key aspects to achieve a scalable and reusable system [23]. In addition, a modular architecture is beneficial in case users want or are expected to make individual adjustments to the demonstrator. The challenge is to balance a universally applicable solution, a high level of detail and easy to follow calculation. Unless the demonstrator is considered as a sort of black box, a common ground regarding terms and data flow structure is essential [39]. After determining the interest groups, the general architecture is designed with regards to the identified requirements (figure 1, arrow 4).

The next step is the selection of the programming language. Consideration the programming skills of the developer, the users and if possible the long-term system manager helps to reduce the project lead-time as well as the maintenance and modification effort. On the downside, it heavily restricts the programming language (figure 1, arrow 5). As a result, a programming language or evaluation method that is actually more suitable for the application may be rejected due to a lack of experience. For example, R is a common software for statistical data analysis, but requires a considerable amount of knowledge [40]. In addition, the question of how to balance easy access and data security arises. Web-based solutions ease the linkage to other systems and enable a wide user base. Nevertheless, in the case of processing confidential data, companies commonly favor non web-based solutions. Data and users are also key aspects in the verification and validation. Only a high level of data quality and the correct usage lead to the desired result. A verification is especially important in the case of modular systems as it ensures the correctness of both the individual modules and the overall system. Just like models, demonstrators represent abstractions of real-world problems or systems. They support decision making by helping to estimate the outcome prior to an actual implementing [41]. Therefore, the validation by a real industrial application is mandatory to control the level of abstraction [22].

In case of the software demonstrator for determining the order processing strategy the central challenge was to connect the various requirements resulting from the fact that the order processing strategy interacts with upstream strategic aspects and at the same time with downstream production planning and control tasks. As

a result, not only different companies and their diverse production processes but also numerous divisions within this companies had to be involved in the development process. One way of achieving this was to set up a working group for the companies to exchange views and experience amongst each other. In addition, several companies were visited to generate a common understanding of their production processes and the associated procedures.

4. Conclusions and Outlook

Transferring theory-based research findings into the day-to-day industrial practice is challenging. Software demonstrators can help to speed up this process by linking theory-based research to the real world problems of companies. Nevertheless, they are only temporary solutions. Companies might use a software demonstrator as a starting point for their own customized developments. Researchers can upgrade the software demonstrator as they gain new insights or connect it to other demonstrators to take a step towards a smart factory. Existing literature often neglects the decisions and characteristics during the development of a software demonstrator or focuses on a very specific case of implementation. Interviewing companies from a variety of industries revealed a rising level of expectations for software demonstrators in the field of productions logistics aligned with the company size. Large companies tend to have more people involved in the development process and more pre-existing infrastructure. In this case, the decision, which requirements should be prioritized, and which requirements should not be taken into account, is essential. Employees in small and medium-sized companies usually have less time available to take an active part in the development process. This shifts the focus to deciding when users are actually needed and to what extent testing with fictitious data or freely available data from other areas is sufficient. Incorporating different sized companies from various industries in the development process ensures a broad applicability. Thus, a software demonstrator should be designed to fit shifting objectives. Even though the decisions and characteristics are influenced by the company size, it does not change their strong interrelations and the underlying conflicts. Only a continuous analysis, verification and validation alongside a clear position between the conflicting objectives can lead to the development of a suitable software demonstrator. As the development process of a software demonstrator differs from a classical software engineering process, further research is required. Analyzing the reusability of pre-defined software elements could help to standardize the development process. For example, examining the wide range of existing projects containing software demonstrators could help to highlight similarities and at the same time specify the decisions and characteristics tendencies for different research areas.

Acknowledgements

The research project was carried out in the framework of the industrial collective research programme (IGF no. 20906 N). It was supported by the Federal Ministry for Economic Affairs and Energy (BMWi) through the AiF (German Federation of Industrial Research Associations eV) and the BVL (Bundesvereinigung Logistik eV) based on a decision taken by the German Bundestag.

References

[1] Louw, L., Droomer M., 2019. Development of a low cost machine vision based quality control system for a learning factory. Procedia Manufacturing 31, 264-269. https://doi.org/10.1016/j.promfg.2019.03.042

[2] Yang, H., Lu, W.F., Lin, A. C., 1992 Intelligent Process Planning Using a Machine Learning Approach. IFAC Proceedings Volumes 25 (28), 147-151. https://doi.org/10.1016/S1474-6670(17)49482-1

[3] Schmidt, M., Maier, J.T., Grothkopp, M., 2020. Eine bibliometrische Analyse - Produktionsplanung und -steuerung und maschinelles Lernen (english title: PPC and machine learning - A bibliometric analysis). Wt werkstattstechnik online 110 (4), 220-225.

[4] Schuh, G., Reuter, C., Prote, J.P., Brambring, F., Ays, J., 2017. Increasing data integrity for improving decision making in production planning and control. CIRP Annals 66 (1), 425-428. https://doi.org/10.1016/j.cirp.2017.04.003

[5] Härtel, L., Nyhuis, P., 2018. Systematic Data Analysis in Production Controlling Systems to Increase Logistics Performance, Schmitt R., Schuh G. (eds) Advances in Production Research. Proceedings of the 8th Congress of the German Academic Association for Production Technology (WGP), Aachen, November 19-20, 2018. https://doi.org/10.1007/978-3-030-03451-1_1

[6] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., Google Brain, 2016. TensorFlow: A System for Large-Scale Machine Learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265-283

[7] Rammer, C., Bertschek, I., Schuck, B., Demary, V., Goecke, H., 2020. Einsatz von Künstlicher Intelligenz in der Deutschen Wirtschaft: Stand der KI-Nutzung im Jahr 2019. ZEW-Gutachten und Forschungsberichte.

[8] Martín-Martín, A., Orduna-Malea, E., Thelwall, M., López-Cózar, 2018. Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. Journal of Informetrics 12 (4), 1160-1177. https://doi.org/10.1016/J.JOI.2018.09.00

[9] Bourque, P., Dupuis, R., Abran, A., Moore, J.W., Tripp, L., Wolff, S. 2002. Fundamental principles of software engineering – a journey. The Journal of Systems and Software 62 (1), 59–70

[10] Juristo, N., Moreno, A.M., 2013. Basics of software engineering experimentation. Springer Science & Business Media.

[11] Garlan, D., 2000. Software architecture: a roadmap, in: Proceedings of the Conference on the Future of Software Engineering, pp. 91-101.

[12] DeMarco, T., 1979. Concise Notes on Software Engineering, Yourdon Press, New York.

[13] Balzert, H., 2009. Lehrbuch der Softwareentwicklung. Basiskonzepte und Requirements Engineering, 3rd ed. Spektrum, Heidelberg.

[14] Adenowo, A.A., Adenowo, B.A., 2013. Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. International Journal of Scientific & Engineering Research 4 (7), 427-434.

[15] Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer 21 (5), 61-72. doi: 10.1109/2.59.

[16] Boehm, B., Bose, P., Horowitz, E., Lee, M.J., 1999. Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach. Proceedings - International Conference on Software Engineering. 10.1145/225014.225037.

[17] Coulibaly, M., Belkhala, S., Errami, A., Medromi, H., Saad, A., Rouissiya, M., Jaafari, A., 2018. Development of a demonstrator «smart-parking», in: 2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON), IEEE, pp. 172-176. doi: 10.1109/MELCON.2018.8379088.

[18] Basile, D., ter Beek, M.H., Fantechi, A., Ferrari, A., Gnesi, S., Masullo, L., Mazzanti, F., Piattiona, A., Trentini, D., 2020. Designing a demonstrator of formal methods for railways infrastructure managers, in: Tiziana, M., Bernhard, S. (Eds.), International Symposium on Leveraging Applications of Formal Methods, Springer, Cham, pp. 467-485.

[19] ISO 9241-210, 2019. Human-centred design processes for interactive systems. International standard, 2nd ed. International Standardization Organization

[20] den Os, E., Boves, L., 2003. Towards ambient intelligence: Multimodal computers that understand our intentions, in: Proceedings of eChallenges e-2003, pp. 22-24.

[21] Mou, Y., Xu, K., 2017. The media inequality: Comparing the initial human-human and human-AI social interactions. Computers in Human Behavior 72, 432-440. https://doi.org/10.1016/j.chb.2017.02.067

[22] Blume, S.A., 2020. Resource Efficiency in Manufacturing Value Chains, Springer, Cham.

[23] Randell, L.G., Bolmsjo, G.S., 2001. Database driven factory simulation: a proof-of-concept demonstrator, in: Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304), vol. 2, IEEE, pp. 977-983.

[24] Nielsen, J., 1993. Usability Engineering. Academic Press, Inc. ISBN 0-12-518405-0

[25] Mathis, L.A., Diederichs, F., Widlroither, H., Ruscio, D., Napoletano, L., Zofka, M. R., Viehl, A., Fröhlich, P., Friedrich, J., Lindström, A., Thorslund, B., Litke, A., Papadakis, N., Bakalos, N., Hernandez-Jayo, U., Espié, S., cavallo, V., Panou, M., Gaintnidou, E., Bekiaris, E., (2020). Creating informed public acceptance by a user-centered human-machine interface for all automated transport modes, in: Rethinking transport. 8th Transport Research Arena TRA 2020, April 27-30, 2020, Helsingfors, Finland.

[26] Kaasinen, E., 2005. User acceptance of mobile services: Value, ease of use, trust and ease of adoption.

[27] Hermann, M., Pentek, T., Otto, B., 2016. Design principles for industrie 4.0 scenarios, in: 2016 49th Hawaii international conference on system sciences (HICSS), IEEE, pp. 3928-3937.

[28] Mantravadi, S., Jansson, A. D., Møller, C., 2020. User-Friendly MES Interfaces: Recommendations for an AI-Based Chatbot Assistance in Industry 4.0 Shop Floors, in: Asian Conference on Intelligent Information and Database Systems, Springer, Cham, pp. 189-201.

[29] Zhong, R.Y., Li, Z., Pang, L.Y., Pan, Y., Qu, T., Huang, G.Q., 2013. RFID-enabled real-time advanced planning and scheduling shell for production decision making. International Journal of Computer Integrated Manufacturing 26 (7), 649-662. https://doi.org/10.1080/0951192X.2012.749532

[30] Lin, H.X., Choong, Y., Salvendy, G., 1997. A proposed index of usability: A method for comparing the relative usability of different software systems. Behaviour & Information Technology 16 (4-5), 267-277. https://doi.org/10.1080/014492997119833

[31] Jeffries, R., Miller, J.R., Wharton, C., Uyeba, K.M., 1991. User interface evaluation in the real world: a comparison of four techniques, in: Procedings of ACM CHI'91 conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, pp. 119- 124.

[32] Holzinger, A., 2005. Usability engineering methods for software developers. Communication of the ACM 48 (1), 71-74. https://doi.org/10.1145/1039539.1039541

[33] Scholtz, J., 2004. Usability evaluation. National Institute of Standards and Technology

[34] Nusbeih, B., Easterbrook, S., 2000. Requirements Engineering: a Roadmap, in: Proceedings of the conference on The future of Software engineering - ICSE '00, vol. 21, no. 5, pp. 61-72. doi: 10.1145/336512.336523

[35] Aurum, A., Wohlin, C., 2005. Engineering and Managing Software Requirements, Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-28244-0

[36] DeMarco, T., Lister, T., 2003. Risk management during requirements. IEEE software 20(5), 99-101

[37] Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B. (2009). KNIME-the Konstanz information miner: version 2.0 and beyond. AcM SIGKDD explorations Newsletter 11(1), 26-31. https://doi.org/10.1145/1656274.1656280

[38] Alexandersson, J., Becker, T., Engel, R., Löckelt, M., Pecourt, E., Poller, P., Reithinger, N., 2004. Ends-based dialogue processing, in: Proceedings of the 2nd International Workshop on Scalable Natural Language Understanding (ScaNaLU 2004) at HLT-NAACL 2004, pp. 25-32.

[39] Maybury, M.T., Wahlster, W., 1988. Readings in Intelligent User Interfaces. Morgan Kaufmann, San Francisco.

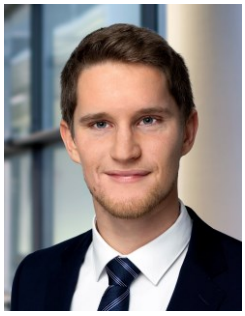
[40] Ihaka, R., Gentleman, R., 1996. R: a language for data analysis and graphics. Journal of computational and graphical statistics 5(3), 299-314.

[41] Law, A.M., Kelton, W.D., 1991. Simulation modeling and analysis, McGraw-Hill, New York.

Biography

Janine Tatjana Maier (*1994) studied industrial engineering at the Leibniz University Hannover. Since 2018, she works as a research associate in the field of production management at the Institute of Product and Process Innovation (PPI) at the Leuphana University of Lüneburg.

Simon Berger (*1992) studied industrial engineering at the Hochschule Ravensburg-Weingarten University of Applied Science and the Leuphana University of Lüneburg. Since 2021 he works as Production Process and Technology Engineer in the company Franatech GmbH.



Tammo Heuer (*1992) studied industrial engineering at the Leibniz University Hannover and has been working as a research associate at the Institute of Production Systems and Logistics (IFA) at the Leibniz University Hannover in the field of production management since 2018.

Peter Nyhuis (*1957) studied mechanical engineering at Leibniz University Hannover and subsequently worked as a research associate at the Institute of Production Systems and Logistics (IFA). After completing his doctorate in engineering, he received his habilitation before working as a manager in the field of supply chain management in the electronics and mechanical engineering industry. He is heading the IFA since 2003. In 2008 he became managing partner of the IPH - Institut für Integrierte Produktion Hannover gGmbH.

Matthias Schmidt (*1978) studied industrial engineering at the Leibniz University Hannover and subsequently worked as a research associate at the Institute of Production Systems and Logistics (IFA). After completing his doctorate in engineering, he became head of Research and Industry of the IFA and received his habilitation. Since 2018, he holds the chair of production management at the Institute for Product and Process Innovation (PPI) at the Leuphana University of Lüneburg. In addition, he became the head of the PPI in 2019.