






Establishing a User-Centered Design Process for Human-Machine Interfaces: Threats to Success

Mario Winterer¹(✉) , Christian Salomon¹ , Georg Buchgeher¹ ,
Martin Zehethofer², and Alexandra Derntl²

¹ Software Competence Center Hagenberg GmbH, Hagenberg, Austria
mario.winterer@gmail.com

² ENGEL Austria GmbH, Schwertberg, Austria

Abstract. While user-centered design (UCD) processes have widely been established in domains like end-user electronics and business-to-consumer products, such processes still lack widespread adaptation for the development of industrial human-machine interfaces (HMIs). Over a period of more than two years, we have worked as part of a development team at a company from the manufacturing domain in a pilot project to introduce a UCD process. During this period, we have - via participant observation - collected a set of observed practices and behaviors that violate well-known UCD principles. Furthermore, we derived some root causes of these violations. Our insights are that introducing a UCD processes cannot be performed isolated for a single development team but impacts the entire organization including management and requires trust as well as changes with regard to mindset, methods, technologies, and team organization.

Keywords: User-centered design · Design process · Industry 4.0

1 Introduction

User-centered design (UCD) processes are well established in development of end-consumer electronics and web-based business-to-consumer products, as a good user experience (UX) is considered as a key success factor in these domains. However, in industrial companies, most human-machine interfaces (HMIs) are still developed traditionally in a feature-oriented manner. The design of HMIs in the mechanical engineering domain, which are used to inspect and modify process parameters or to manipulate automated processes, is typically heavily influenced by the logical structure of the control system, more precisely, the information model of the programmable logic control (PLC), without taking human factors into account.

The resulting HMIs focus on data such as functional blocks and their parameters, rather than on workflows or tasks that need to be performed by their operators. This, combined with the increasing complexity of modern industrial

machines, leads to cumbersome HMIs that do not match the high expectations raised by modern user interfaces of business-to-consumer products like smartphones. Today, companies see user experience as differentiating factor to get competitive advantage over competitors [23]. The need for user participation to build flexible, nevertheless understandable and fault-tolerant HMIs is also motivated by the industry 4.0 initiative [6, 24].

In this case study we report on our experiences of introducing a UCD process at *ENGEL Austria GmbH*, a company from the manufacturing domain. The company is manufacturer of injection molding machines and is currently in the process of developing a new generation of its software stack. As part of this software stack a new version of a *Sequence Editor* application for the programming of robot arms is being developed. This project was selected as a pilot project for introducing UCD at *ENGEL*. As part of an industrial research cooperation, the authors of this paper have worked over two years as project members in this pilot project to supervise the introduction of the UCD process. As a result, we have obtained deep insights in the processes and social fabric of the company which is advantageous over other inquiry approaches as for this research we considered it important to be able to look behind the facade of the organization. Based on our experiences we have collected a set of practices and behaviors we encountered during the introduction of UCD, that violate the core UCD principles.

The remainder of this paper is organized as follows: In Sect. 2 we describe the industrial context of this work. Section 3 presents a brief overview of UCD including central principles. In Sect. 4 we present malpractices we have found. Section 5 discusses related work. Section 6 concludes this paper with a summary of our main findings.

2 Industrial Context

ENGEL is a large manufacturer of injection molding machines. Such machines are used across many industry domains like consumer electronics, automotive, avionics, food industry for producing different kinds of plastic parts like enclosures of cell phones and laptops, toys, car parts, bottles, tooth brushes, etc. By using different molds and adaptable machine parameters, a single machine is able to process varying types of material and hence produce many different products. Nevertheless, certain domains require very specific adaptations of machines. Providing almost any requested customization is one of the key success factors of the company.

In 2016, the company started the development of its next generation of software for injection molding machines. This project encompasses the development of a new HMI (framework and applications), and a new middleware tier based on the OPC Unified Architecture (OPC UA) specification [19] for a unified communication with PLC control systems and auxiliary devices (e.g. robot arms and conveyor belts) of different vendors. As a consequence, large parts of the software have to be re-engineered and migrated to new technologies and frameworks. This undertaking affects many different stakeholders across several organizational units in the company. The core parts of the HMI are developed by four

different agile teams in a Scrum process. Once, the HMI framework is released, several customization teams, which adapt the machine to the customer's needs, will also make use of it. In addition, many other teams are working on software products that are not directly related to the HMI, but may have influence on the overall user experience (e.g. the customer portal).

One application of the HMI is a *Sequence Editor* for the programming of industrial robots and the manipulation of machine workflows. The *Sequence Editor* supports visual motion-level programming and is used by a wide range of technicians from well-trained maintenance engineers to novice factory attendants. The company selected the *Sequence Editor* as a suitable application for piloting a UCD process.

3 User-Centered Design Principles

UCD processes focus on putting users into the center of product design and development [18]. Existing approaches aim to integrate users in the development process because user involvement is a critical factor for system acceptance and success [1, 3, 10]. No matter which concrete methods [4] are applied by a particular approach, they all share the following common principles:

Integrated and Comprehensive Solution. In order to provide a consistent user experience, surrounding services and products must be developed together with core functions. Therefore, the development teams should cooperate closely with surrounding departments like marketing, training, and customer service [9].

Focus on Users and Tasks. For a good and minimal system design it is necessary to understand which people are using the system and what goals they are trying to achieve [8].

Active User Participation. End-users and domain experts, which in the manufacturing domain are often end-users as well, should participate through all process stages [12] beginning with early analysis. Identifying and selecting representative users is an ongoing process [13] that is crucial for project success.

Continuous Evaluation and Iteration. Development is iterative and based on prototypes even in very early states of the project. These prototypes are incrementally evaluated by either experts or (again) in collaboration with potential users [17]. Insights of these evaluations are used to build enhanced prototypes in subsequent iterations.

Interdisciplinary Teams. As a consequence of the preceding principles, a team must allocate a broad range of skills and knowledge to satisfy the UCD process needs. Usually, (software) engineers alone cannot cover this range but must be assisted sporadically by members of other departments and supported consistently by UX designers and usability engineers [5]. It is highly recommended to integrate these experts into the development teams.

One approach that transfers these principles to agile development environments is *Lean UX* as proposed by Gothelf and Seiden [7]. Lean UX focuses on vertical prototypes and minimum viable products (MVP) to gain rapid feedback and test the relevance and usability of implemented concepts (there are a lot of different definitions for MVP [16], we agreed on the definition given by Ries [21]).

4 Experiences When Introducing UCD

In this Section, we report on our experiences of introducing a UCD process for the development of the *Sequence Editor* in the UCD pilot project after about two years. For this purpose, we identified malpractices that symptomatically violate the UCD principles and methods we presented in Sect. 3. Figure 1 gives an overview of all findings. The figure lists all observed symptoms on the left side and categorize them by the principles (see Sect. 3) violated. Outgoing arrows mean that the source item is caused by the target item. So each of the symptoms can ultimately be tracked down to at least one root cause that originates in the behavior of the project team or their surrounding (processes, supervisors, etc.). As Fig. 1 shows we have identified four major root causes:

Inappropriate Development Organization, Tools, and Mindset. User interface development based on UCD requires appropriate mindsets. As industrial companies don't see themselves as software developers, they are much more traditional concerning methods, organization, tools, and mindset than modern software development organizations. These outdated attitudes may have severe impact on UCD based software product development.

UCD Intrinsic Issues. The user-centered design process is not perfect and has also some drawbacks [2]. Issues that are related to these drawbacks are summarized by this root cause.

Domain Specific Difficulties. HMI development in the industrial domain is very special due to its tight coupling to the machinery hardware and its special usage environment. Although hard- and software must work together perfectly, the software development process differs significantly from the hardware development process. Apart from that, there is the very long product life-cycle, which can last 20 years or longer. Within this time, the company must provide support and maintenance of both, software and hardware. As many machines are not connected to the internet, updating the software system requires maintenance personnel to be on-site. So for cost reasons, updates should not be done too frequently. Last, but not least, industrial companies want to keep their production knowledge secret. Due to this and because the companies are spread worldwide, it is not too easy to perform UCD related tasks like observations or interviews with end-users.

Too Less UCD Experience. 'Exercise makes perfect' is also true for introducing a new process. The team members as well as all other people concerned have to learn new ways of doing things and - even more important - accepting

that things are different now. Lack of experience is especially noticeable when something goes wrong. But even when everything runs fine, people tend to revert to old habits.

The following sections (Sects. 4.1, 4.2, 4.3, 4.4 and 4.5) systematically describe all found problems and their symptoms grouped by UCD principle. Even more, where appropriate, mitigation strategies to overcome the corresponding problem are given. These strategies arise mainly from personal experiences of the authors mixed with tried and tested statements of literature. It is important to note that currently not all of these optimal situations are already established in the pilot project, hence their effectiveness is not proven yet.

4.1 Integrated and Comprehensive Solution

Feature-Driven Vs. User-Driven Development. While the pilot project follows a user-centric approach from start, all other teams continued to work feature oriented. This situation is a continuous source of conflict in a multi-team project. In a feature-driven development process, the overall model and the feature list are specified first; then the features are implemented step-wise. This is inconsistent with the user-centered design, where new features are defined and refined gradually based on user research.

Symptom: The framework team is busy implementing components like UI controls or input dialogues without any user need. Special framework features defined by the pilot project team are postponed as they do not match the predefined feature list of the framework team. As a consequence, the pilot project team must either implement the features by themselves, or wait until the framework team is able to deliver the requested feature. As the latter is irreconcilable with the UCD process (which demands early user-testing of implemented features), the pilot project team has to do much more work than planned.

Mitigation: All teams of the multi-team project follow the UCD approach. User research is done in tight cooperation. New features can be defined on demand.

Departmental Thinking. Traditionally, there is no communication channel between departments like marketing and the development teams. As a consequence, business goals do not necessarily align with product requirements nor do they drive innovation.

Symptom: There is no general design system that covers all different communication channels between company and customer: print media, the company web page, the web based customer portal, the product, and auxiliary apps. All these parts are developed independently and tell their own story to the user. As the marketing department is not interested in HMI development, and the product manager is not informed about marketing activities, the business goals of marketing and product development do not match.

Another example is the missing link between development department and customer training. While the customer training team usually has deep knowledge

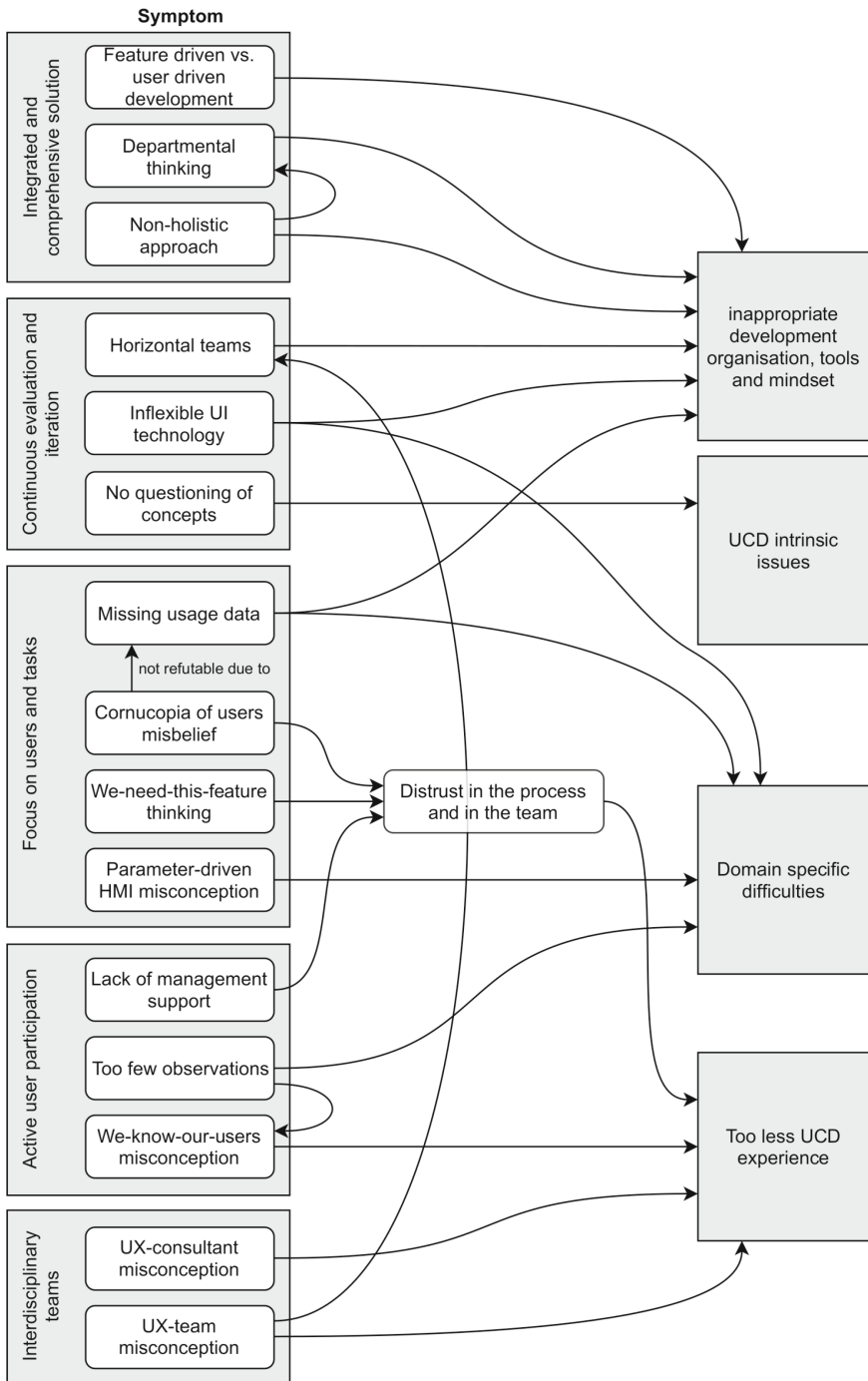


Fig. 1. Identified issues grouped by UCD principles and their causes.

about the needs and sufferings of many customers, they are not really integrated in the development process. Therefore, valuable information that is actually already within the company remains unused.

Mitigation: Development is driven by business goals. There is a clear vision for the next generation HMI which is defined interdisciplinary by UX experts, business executives, marketing experts, technicians and more. The vision is not necessarily restricted to virtual user interfaces. Every development iteration cycle generates value for the user and hence for the company.

Non-holistic Approach. Although the HMI is part of an integrated industrial environment, the HMI development is restricted to the graphical display only. This is disadvantageous in situations, where the user research findings demand a holistic approach which touches both, display and machine hardware as well. This issue is related to *Departmental thinking*, which is one of the root causes of this misconception.

Symptom: The team is presented with a *fait accompli*. Important decisions which have deep impact on user experience, are already made and cannot be (easily) changed. These may be size, orientation and position of the display panels, specification of the visualization hardware, or form and position of hardware keys. Adding additional hardware, like sensors or input devices are out of the question.

Mitigation: Due to a holistic approach, UCD means rethinking the entire machine and its environment from the point of view of user interaction. This provides an integrated solution that works best for the user.

4.2 Focus on Users and Tasks

Missing Usage Data. Due to missing usage data, the pilot project team has no idea about how the thousands of users interact with the HMI of the machines in-use. Knowledge about usage can make time-consuming observations and discussion obsolete. The reasons for the lack of data are manifold. Most industrial machines in-use are either not connected to the internet at all, or are not accessible from outside due to security reasons. So usage data has to be collected manually. Apart from that, many of the machines in-use are rather old and outdated from a technical point of view and provide too little data storage to collect user interaction data and its usage context (e.g. machine state) over time.

Symptom: Although the stakeholders pretend to know the users (see 4.3), they are not able to answer questions like ‘Which UI parts are used most?’, ‘How many minutes/hours per day do users use the HMI?’, ‘Which navigation paths are used most?’, ‘Do the users use swipe gestures or previous/next buttons for navigating between views?’, or ‘What are the top ten operation errors?’. Based on such information, the development team could focus on UIs that are really relevant to the user instead of laboriously gathering such information through user research.

Mitigation: Usage data is collected automatically and periodically uploaded to a centralized cloud storage so that it can be used for detailed usage analysis. The results are an important basis for further development.

Cornucopia of Users Misbelief. The process of defining personas based on observations is regularly distrusted by stakeholders. They believe that the company has so many end-users, and all of them work differently, so it is impossible to unify their personalities in just a few personas. As a non-domain expert, these believes are hard to assess or even declare invalid, especially when there is no usage data to verify this (see *Missing usage data*).

Symptom: Experts that act as stakeholders of the project often point out the great functionality of the existing product by telling stories about a special user or use case, which, at first glance, seem to render the prospected solution impractical or incomplete.

Mitigation: Although special users and use cases are real and respected by the HMI team, they do not drive HMI development. The stakeholders have trust in the team and the process and know, that the result of a design iteration does not support all possible use cases. There are enough domain experts that defend the design iteration result against disbelievers.

We-Need-This-Feature Thinking. Stakeholders tend to use the old system as requirement reference. They demand features from this system to be transferred to the new system without taking user needs into account. As a result, they question feature-incomplete iteration results. Similar to *Cornucopia of users misbelief*, the main causes are distrust in the process and in the team, but the symptoms are different.

Symptom: Again - similar to *Cornucopia of users misbelief*, experts act as stakeholder. But instead of telling a story about individuals, they pretend a certain feature is crucial to most of the users. For non-domain experts, it is very hard or even impossible to refute this claim, hence these features are often re-implemented without any confirmation by user research or testing.

Mitigation: Stakeholders focus on the iterative progress of the team, even if they know that the current product still misses features that seem to be important at first glance. This requires a certain level of trust in the team and in the UCD process.

Parameter-Driven HMI Misconception. The PLC of *ENGEL* defines about 16.000 parameters that may be relevant to the HMI. Due to multiple product lines and individual customization, the parameters actually viewed in the HMI vary heavily. The easiest way to support this flexibility is to just visualize the logical structure of the control system, ignoring any user tasks or workflows.

Symptom: Instead of focusing on tasks, the UI focuses on parameters. Most of the views are just parameter lists without additional information. The grouping and ordering of these parameters are defined by the PLC and customer

customization developers without assistance by UI/UX experts. Concepts like wizards or 'intelligent' workflow assistants are missing.

Mitigation: The HMI is two-layered. The parameter layer provides a flexible mechanism for both, the developer and the end user to define easily, which parameters should be displayed on which page and in which order. This layer is sufficient to control and operate the machine. Apart from that there is the workflow layer, that provides explicitly developed user interfaces that support important workflows and tasks. These UIs can be introduced step-by-step each improving the overall user experience.

4.3 Active User Participation

We-Know-Our-Users Misconception. As the company is unfamiliar with user-driven development, the project stakeholders are still tempted to ignore user research and demand features, they think are relevant instead. They argue this by mentioning their many years of experience. Although the company has sufficient knowledge about the customer's usage scenarios, it is almost exclusively in the minds of service technicians and customer advisers. The knowledge is not structured and therefore not directly usable.

Symptom: When presenting insights gained from user observations in the field, experienced employees, which are not part of the project team, claim that they already knew about that and this information could have easily been requested.

Mitigation: Although service technicians and customer advisers are important sources of information, the main user needs are based on user research in the fields.

Too Few User Research There are many reasons, why user research in the industrial domain is difficult. Obviously, there are safety and information security reasons. In addition, intrusive techniques like interviews keep workers away from their work, so not all companies are suitable for that. We also found that observing infrequent tasks requires good planning, so it is important to synchronize the schedule of the UX researchers with the work schedule of the participants. Hence, often the right user is not next door. Last, but not least, typical workflows often consist of many technical steps, which are less interesting to the UX expert. All in all, observing the entire workflow may take a few hours or even several days. All this causes high costs. As a result, the team tends to do less observations than necessary.

Symptom: The symptoms are obvious: for many scenarios, confirmation by observation is still pending; results from ideation workshops are not validated with end-users; colleagues are used as representative for real end-users.

Trade-off: Observations happen on a regular basis for important workflows. Missing user needs due to missing observations are mitigated by defining user need assumptions and trying to confirm or refute them early by user testing rapid prototypes. Participants are real end-users, but also service technicians, customer advisers, trainers, apprentices and other personnel of the company.

Lack of Management Support. Although the management supports the pilot project and the UCD process, it has too little knowledge about the philosophy of UCD. The consequences are lack of trust and demand for intervention.

Symptom: Time spent on user observations is criticized by supervisors (see *Too few user research*), especially if their main findings are already known by stakeholders (see *We-know-our-users misconception*).

4.4 Continuous Evaluation and Iteration

No Questioning of Concepts. Once, an early prototype has been tested and proven to work at a certain degree, it is never questioned any more. As a consequence, iterations just improve existing prototypes gradually and never raise radical changes. Although this issue is inherent to UCD methods in general, it is even worse in this industry. Due to high domain complexity, it is almost impossible to test all technical details of concepts, so there is always the risk of improving a prototype that is basically broken without knowing it. A similar issue has already been identified by [15] in 1997.

Symptom: Shortly after project start a central prototype was elaborated in detail to overcome some doubts about the user-centered approach. Even so user tests have shown that the prototype basically works for experienced users another promising concept has never been tested, because of the effort already spent.

Mitigation: Interaction concepts are tested at a very early stage. In this phase, there are often several concept proposals that can be tested against each other using A/B tests. This makes it possible to find concept errors early on and to optimally combine the best solutions. In addition, special domain expert reviews improve the prototype quality on a conceptual level.

Horizontal Teams. Currently, the multi-team project is set up with four horizontal teams. One team is responsible for the OPC UA based layer set up on top of machine and robot control, which is developed by a second team. Third, a team implements the HMI framework and the HMI base application accessing information of the OPC UA layer. Fourth, the pilot project team develops the *Sequence Editor* by means of the HMI framework and integrates it into the HMI base application. As a consequence, new interaction concepts designed by the pilot project team cannot be integrated into the system without support from the other teams. This causes latency which makes it hard to evaluate new UI concepts in time.

Symptom: A new concept that should facilitate trouble shooting in the *Sequence Editor* caused the robot control layer to provide novel data. This circumstance was not foreseen by the team implementing the *Sequence Editor* and so the group of persons participating in the technical coordination meetings on this issue has been successively increased, with a lead time of more than two Scrum sprints (3 weeks each) [22].

Mitigation: The teams are vertically organized, so they can work independently most of the time. Dependencies between teams arise only when both teams share the same user needs.

Inflexible UI Technology. Both, technology-in-use and system architecture did not support exchanging UI parts and modifying interaction concepts easily. Even more, due to the limited capabilities of the mobile touch device, interaction concepts are limited too.

Symptom: The UI framework in use does not or barely support multi-touch input. Implementing animations like fade-out of dialogs, transitions or rotations is hard and requires major code changes. Controls like text input fields, buttons or check boxes cannot be styled or skinned to be adapted to modern UI designs. Features like visualization of 3D models or embedding multimedia are missing or difficult to integrate.

Mitigation: Existing legacy components have been replaced and a more suitable UI framework has been introduced. Furthermore, a more capable mobile touch device has been prospected in favor of better user experience.

4.5 Interdisciplinary Teams

UX-Consultant Misconception. In the first months of the pilot project, the main UX work was done by external UX experts. As a result, the team had too little knowledge about UX related aspects to be able to develop the MVPs. Furthermore, external UX experts have too little domain know-how, which is necessary for a holistic understanding of scenarios.

Symptom: As the UX-consultants have only very few contact to the developer team, most of the user stories are already specified into detail when they are presented to the software developers. Although the stories might be perfect from a UX point of view, they are not technically validated, hence the developers might face several technical difficulties while implementing them. As they were not involved in the user research nor design process, they miss any reasoning and don't know if and how far they can deviate from the specification to circumvent these difficulties. Again, as the UX-consultants are separated from the development team, most of these problems are not discussed, thus the features are implemented exactly as specified - no regard to expenses. Even worse, experienced software developers often question the UX designs and concepts, which leads to disparaging opinions and disrespect toward the UX experts.

Mitigation: UX is an integrated part of the development process. The teams defines UX roles similar to the typical software development roles 'Software Architect', 'Tester' or 'DevOps Engineer'. All team members take part in UX-related tasks like user research or evaluation for the sake of knowledge transfer in both directions.

UX-Team Misconception. Separating the UX experts from the development team by building a UX team of its own was another misconception. This approach clearly conflicts with the vertical team thinking (see *Horizontal teams*). Although this keeps the UX know-how inside the company at least, it also keeps UX know-how away from the development teams.

Symptom: The symptoms are similar to *UX-consultant misconception*, although less severe, as at least there is UX know how in the company.

Mitigation: See *UX-consultant misconception*

5 Related Work

In the manufacturing industry the need for usability and user experience as explicit quality measures for user interfaces of cyber-physical systems (CPS) [24] is rather new. This need is based on changing requirements, a higher level of automation, and increasing complexity driven by the *Smart Factory* idea of the *Industry 4.0* initiative [14]. These requirements demand for appropriate and proper working UCD processes, as described by Pfeiffer et al. [20], but industry still lacks long-time experience on how to integrate these processes in their development practice.

Systematic reviews [4,11] have shown that most publications that discuss UCD processes in practice primarily discuss issues that emerge when introducing particular UCD methods (e.g. personas, user tests,...) in the context of agile processes. In [15] Lauesen investigates the introduction of UCD processes. We can confirm his findings, i.e., that early prototypes are only modified in details in later phases (see Sect. 4.4), and that there exists a friction between software developers and UX-experts (before UX-experts became part of the team). Compared to Lauesen, we have identified additional issues, which had negative impact on the project's pace.

6 Conclusion

Introducing UCD in the industrial domain represents a significant paradigm shift, since industrial HMIs are typically still developed in feature-oriented manner. UCD processes are based on a set of principles that must be followed in order to be successful. We have presented a set of issues that we have encountered when introducing UCD in a company from the manufacturing domain including symptoms and potential mitigation strategies. The root cause of most of the problems seems to be the lack of trust in the process on all organizational levels (line management, stakeholders, other teams, other departments), which itself originates from lack of knowledge about the UCD process.

Acknowledgement. The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Affairs, and the Province of Upper Austria in the frame of the COMET center SCCH.

References

1. Abelein, U., Sharp, H., Paech, B.: Does involving users in software development really influence system success? *IEEE Softw.* **30**(6), 17–23 (2013)
2. Abras, C., Maloney-Krichmar, D., Preece, J., et al.: User-centered design. In: Bainbridge, W. (ed.) *Encyclopedia of Human-Computer Interaction*. Sage Publications, Thousand Oaks, 37(4), 445–456 (2004)
3. Bano, M., Zowghi, D.: A systematic review on the relationship between user involvement and system success. *Inf. Softw. Technol.* **58**, 148–169 (2015)
4. Da Silva, T.S., Martin, A., Maurer, F., Silveira, M.: User-centered design and agile methods: a systematic review. In: 2011 AGILE conference. pp. 77–86. IEEE (2011)
5. Göransson, B., Sandbäck, T.: Usability designers improve the user-centred design process. In: *Proceedings for INTERACT*, vol. 99, pp. 1–4 (1999)
6. Gorecky, D., Schmitt, M., Loskyll, M., Zühlke, D.: Human-machine-interaction in the industry 4.0 era. In: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 289–294. IEEE (2014)
7. Gothelf, J., Seiden, J.: *Lean UX: Applying Lean Principles to Improve User Experience*. O’Reilly Media, Inc., Sebastopol (2013)
8. Gould, J.D., Lewis, C.: Designing for usability: key principles and what designers think. *Commun. ACM* **28**(3), 300–311 (1985)
9. Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, Å.: Key principles for user-centred systems design. *Behav. Inf. Technol.* **22**(6), 397–409 (2003)
10. Harris, M.A., Weistroffer, H.R.: A new look at the relationship between user involvement in systems development and system success. *Commun. Assoc. Inf. Syst.* **24**(1), 42 (2009)
11. Jurca, G., Hellmann, T.D., Maurer, F.: Integrating agile and user-centered design: a systematic mapping and review of evaluation and validation studies of agile-ux. In: 2014 Agile Conference, pp. 24–32. IEEE (2014)
12. Kujala, S.: User involvement: a review of the benefits and challenges. *Behav. Inf. Technol.* **22**(1), 1–16 (2003)
13. Kujala, S., Kauppinen, M.: Identifying and selecting users for user-centered design. In: *Proceedings of the Third Nordic Conference on Human-Computer Interaction*, pp. 297–303. ACM (2004)
14. Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M.: Industry 4.0. *Bus. Inf. Syst. Eng.* **6**(4), 239–242 (2014)
15. Lauesen, S.: Usability engineering in industrial practice. In: Howard, S., Hammond, J., Lindgaard, G. (eds.) *Human-Computer Interaction INTERACT 1997. ITIFIP*, pp. 15–22. Springer, Boston, MA (1997). https://doi.org/10.1007/978-0-387-35175-9_4
16. Lenarduzzi, V., Taibi, D.: Mvp explained: a systematic mapping study on the definitions of minimal viable product. In: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 112–119. IEEE (2016)
17. Nielsen, J.: Usability inspection methods. In: *Conference Companion on Human Factors in Computing Systems*, pp. 413–414. ACM (1994)
18. Norman, D.A., Draper, S.W.: *User Centered System Design: New Perspectives on Human-Computer Interaction*. CRC Press, Boca Raton (1986)
19. OPC Foundation: IEC 62541: OPC Unified Architecture. Standard, International Electrotechnical Commission (2015–2016)

20. Pfeiffer, T., Hellmers, J., Schön, E.M., Thomaschewski, J.: Empowering user interfaces for industrie 4.0. *Proc. IEEE*. **104**(5), 986–996 (2016)
21. Ries, E.: *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Books (2011)
22. Schwaber, K., Beedle, M.: *Agile Software Development with Scrum*, vol. 1. Prentice Hall, Upper Saddle River (2002)
23. Vääätäjä, H., Seppänen, M., Paananen, A.: Creating value through user experience: a case study in the metals and engineering industry. *Int. J. Technol. Mark.* **9**(2), 163–186 (2014)
24. Wittenberg, C.: Human-CPS interaction-requirements and human-machine interaction methods for the industry 4.0. *IFAC-PapersOnLine* **49**(19), 420–425 (2016)