Continuous Improvement and Validation with Customer Touchpoint Model in Software Development

Tanja Sauvola, Markus Kelanti, Jarkko Hyysalo, Pasi Kuvaja and Kari Liukkunen
M3S Research Unit, Faculty of Information Technology and Electrical Engineering,
University of Oulu
PO BOX 4500, FI-90014 University of Oulu, Finland
e-mail: {tanja.sauvola, markus.kelanti, jarkko.hyysalo, pasi.kuvaja, kari.liukkunen}@oulu.fi

Abstract—**Experimental-driven software development approach has gained momentum as a way to incrementally build and validate customer value. In-depth understanding of customer needs and reasons behind constantly changing requirements are essential for building successful software products. However, identifying, validating and reacting to these changes is often difficult and requires short iteration cycles and feedback from customers. This paper reports a 12-month case study conducted in an agile software team following a practical customer touchpoint (CTP) model for continuous improvement and validation. The objective of the study was to implement CTP into software team practices in order to determine what kind of effect it has on the development of a web application. The contribution of this paper is twofold. First, an in-depth case study is presented that identifies the practices a CTP model should adopt when implemented in the software development process. The CTP model is then extended based on the identified recommendations. Second, the benefits and challenges of the extended CTP in software development are presented. The main benefits relate to learning, decision-making, innovation, co-creation and communication. The model had a positive impact on the software development process, but some challenges, such as stakeholder availability and customer value measurement, were identified.**

Keywords—lean UX; service design; customer involvement; software development; continuous improvement.

I. INTRODUCTION

Customer value determines the success of a product or a service in the marketplace, and software has become essential to value creation and delivery [1]. Software development teams face high pressure to develop innovative products and services at increasing speeds in a dynamic and continuously changing business environment. To this aim, development teams seek to become more data-driven, which requires using customer feedback and product data to support learning and decision-making during the development process and throughout a product's lifecycle. This presents the challenge in software development to adopt iterative and agile practices for the continuous deployment of new features and enhancements to provide customers with added value [1]-[3]. Customer value and the ability to experiment with business ideas have been considered in Agile methods [4] and the Lean Startup [5] philosophy, both very popular in the software industry. Recently, concepts like continuous experimentation—where software development teams constantly experiment with product value—have been introduced in the literature as well [3][6][7]. The continuous experiment approach involves customers and end-users of the service in the decision-making process, providing feedback to developers by interacting with experimental materials like early prototypes of the features under development.

User experience (UX) has also become an increasingly important determinant of the success or failure of software systems. Approaches such as Design Sprint, introduced by Google Ventures [8], and Lean UX [9] using Lean Startup principles, now appear. These experiment-driven approaches interrelate with business strategy decisions and tend to focus on customer centricity. Forward-thinking companies see the benefits and importance of UX design in their product and service development activities. Even with these methodologies, software teams still encounter challenges when involving customers and integrating customer feedback into short development cycles [2][6].

In this paper, the customer touchpoint (CTP) model introduced by Sauvola et al. [2] is examined as a way for software development teams to become more customer centric and data driven. The study focuses on finding answers to the following research questions:

RQ1: What practices should the CTP model adopt when implemented in the software development process?

RQ2: What are the benefits and challenges of CTP in software development?

To address the research questions, a case study in the software development context has been performed. The CTP model is validated and extended by proposing practices for the continuous validation of customer value in short development cycles to increase customer understanding. The benefits and challenges of the extended CTP model are then identified.

The paper is organized as follows. Section 2 studies the related work. Section 3 presents research approach and describes the case project. Section 4 presents the results and analyses. Section 5 discusses the results. Section 6 concludes the work with a summary and topics for further research.

II. RELATED WORK

A. *Value creation through continuous experimentation*

In software development, short product development and deployment cycles and shorter feedback loops are enabled by practices such as agile and lean development [4][5], continuous deployment [10] and DevOps [11]. An organisation's ability to deploy new functional and non-functional features continuously enables faster responses to customer needs [10][12]. However, these practices provide only a little guidance on how to constantly validate product assumptions while experiment-driven approaches focus more on value delivery with validated learning.

The literature presents several methods for establishing continuous experimentation with customers. Bosch et al. [13], for example, suggests exposing products to customers in two- to four-week experimentation iterations to solicit feedback. Studies of software engineering also propose ideas for continuous experimentation. Fagerholm et al. [3] suggest their RIGHT model for continuous experimentation, in which the key element is the start-up company's ability to release prototypes with suitable instrumentation. In this model, business strategies form the basis of experiments, and results guide future development activities. Holmström Olsson et al. [6] present the concept of an innovation experiment system (IES) in which software development teams constantly develop hypotheses and test them with customers. The IES approach also recommends that software teams continuously deploy individual features rather than plan larger product releases. This enables short feedback loops and facilitates data-driven decisions, reducing the risk of failing to build customer value by continuously identifying, prioritising and validating product assumptions in all software development phases. Holmström Olsson et al. also coined the hypothesis experiment data-driven development (HYPEX) model, which introduces a set of practices to integrate feature experimentation into the software development process by combining feature experiments and customer feedback. The model aims to improve the correlation between customer needs and research and development efforts [14]. More recently, Holmström Olsson et al. also presented a quantitative/qualitative customer-driven development (QCD) model. This model presents available customer feedback techniques and aims to help companies become more data-driven by combining qualitative feedback with quantitative customer data [15]. Such methods are supported by Kohavi et al. [16], who report that companies use experiments to guide product development and accelerate innovation. Companies adapting the experimentation approach are typically developing internet related product and services, such as Amazon, eBay, Facebook, Google, LinkedIn, Microsoft and Netflix [16]. While experimentations are recognized as beneficial approach to software development, barriers to related resources, organisational culture and data knowledge persist [17].

B. *Service design in software development*

Service design (SD) is a methodological approach that can be utilized during software development to involve customers and collect feedback. It is a holistic, multidisciplinary approach that aids innovation and improves existing products to make them more useful and desirable to customers [18]. SD provides an outside-in-development approach, where products and services are developed holistically from customers' and end-users' points of view, and applies design thinking and methodologies in product and service development. Recently, some process models and working practices, such as Lean UX [9] and Design Sprint [8], have been introduced under SD and UX design titles with the aim of synthesising design thinking, agile software development and lean start-up philosophies. Lean principles and Lean Startup apply to Lean UX in three ways: 1) removing waste by only creating the design artifact that enables the software development team to move their learning forward; 2) embracing cross-functional collaboration and bringing all relevant stakeholders into the design process; and 3) adopting the experimentation mindset. Lean UX is the evolution of product design, aiming at breaking down the barriers between software development teams, designers and users [9]. Similarly, Google Ventures introduced Design Sprint to integrate product discovery, product validation and delivery activities in a five-day process [8].

Experiments with customers require the ability to elicit customer feedback. For example, visualisation techniques nurture the co-design process and elicit feedback without relying on existing technical infrastructure or up-front development efforts. Different visualisation techniques are used, e.g., in agile development to design a product or a service, support the development process, enhance communication and track the process [19]. The SD approach emphasises the co-design process, whereby stakeholders are involved in concrete, productive design tasks such as workshop sessions. These sessions typically include collaborative prototyping and other means of expressing the information needed in the design and development process [20].

III. RESEARCH APPROACH

The software development process and related activities are presented in this paper through a case study that follows the guidelines set by Runeson and Höst [21]. A qualitative case study method was chosen to gain a practical view on the topic. In principle, the study is an in-depth, single-case study, where the involved participants represent different public and private organisations with fundamentally different roles, such as user, data provider, company representative and ecosystem leader. According to Yin [22], the single-case design is appropriate for testing a specific theory, which in this case is the CTP model. The study is also confirmatory in nature, as it aims to evaluate the robustness of a theory.

A. Case description

Our case study was conducted in the Fenix project (www.fenix.vip), where the software team develops a cloud-based service for real-time business case management for the Allied ICT Finland (AIF) (www.alliedict.fi) In the AIF program, a need was identified as currently information about companies, research institutes, business opportunities and product offerings are scattered into various databases and services. Thus, there is a need to aggregate this information under one digital service-point. Fenix was therefore designed to provide one place for finding and meeting new business opportunities, solution providers, experts, start-ups, business incubators and research institutes. The aim is that the tool would allow users to browse information about potential business partners and opportunities across multiple third-party systems from a single point. For example, the utilization of Business Finland (www.businessfinland.fi) database to follow cases and offers from municipalities and governments as starting cases.

The specific challenge of the Fenix system was to gain the commitment of user stakeholders, namely companies, universities and other organisations, which AIF and Business Finland aimed to support. The idea was that Fenix would promote Finnish industry by increasing overall trade and exports through the improved exposition of business opportunities, expertise and new company and research ecosystem growth. The challenge was that there were no clear requirements to fulfil due to the changing group of stakeholders and the need to develop an efficient mechanism to expose and promote business cases and ecosystem formation. Further more, development funding depended on the success of committing enough stakeholders to create ecosystems in Fenix.

The development team consisted of nine people: a designer/product owner, seven developers and a scrum master in total. During development, one researcher acted as product owner and UX designer and another adopted the role of software architect. Experience levels ranged from novice to expert with several years of industrial experience. The software team followed scrum methodology with iterative two-week development cycles. Customer collaboration was guided by the CTP approach during the design and implementation of the system (see [2] for details).

B. Data collection and analysis

Empirical data was collected between January and December 2017 from observations and field notes, 13 interview recordings and 20 workshops. Semi-structured interviews with open-ended questions [23] were used to collect data from six customer interviews and seven development team interviews. Customer participants were selected to best represent the companies, research institutions and local public actors.

The interviewed customers represented the AIF, Business Finland, trade associations and local companies. The themes discussed in the interviews were customer collaboration practices, agile ways of working and the benefits and challenges involved in the current way of working. The interviews were conducted face-to-face and lasted approximately 60 minutes. All interviews were recorded and transcribed for analysis in QSR NVivo tool (www.qsrinternational.com). The obtained material was analysed in continuous collaboration by three researchers, following the recommendations for thematic analysis in software engineering by Cruzes and Dybå [24]. The interview data was first coded based on the interview topics and then analysed and coded according to emerging themes. The themes were first examined independently, then cross-analysed.

The workshops, besides being an integral part of the development cycles, occurred bi-weekly after every sprint. Workshops lasted approximately one hour and included software product demonstrations and collaborative prototyping sessions. Customers and other stakeholders were invited to see the latest version of the prototype software and to try, test and give feedback. Workshop participant groups consisted of the whole development team and between two and 15 customers. Before these workshops, a new version of the software in question was deployed. During the bi-weekly workshops, the product owner presented the changes and new features introduced in that particular software version and collected feedback and new development ideas. New features ideas were experimented with using UI mock-ups and interactive prototypes during the workshop sessions. The developers summarised the key points of each workshop, and the researchers observed and made notes about stakeholder reactions and documented relevant comments. Following the workshops, the development team analysed the feedback, prioritised next tasks and planned the next iteration.

C. Validity and reliability of the study

The design of the present study was carefully planned to consider validity concerns. We discuss four threats to validity: internal and external validity, construct validity, and reliability [22].

Internal validity regarding cause–effect relations was addressed via multiple sources of evidence, and with iterative research gradually building the final outcome. Evaluation of utility, quality and efficacy was done extensively with the help of industrial experts and real users. Immediate feedback was gathered, and the use of prototypes was observed. Based on the rich feedback and analysis, further development and corrective actions were carried out.

External validity concerning the generalisability of results was addressed by involving several industrial experts to provide their views. In addition, the involvement of several organisations of different types and domains increased the external validity and generalisability of the results. However, further study will be needed to generalise the results fully.

Construct validity was mitigated by several activities. First, an interview guide was developed and piloted. A pilot interview was used to refine and clarify the interview questions. Second, interview candidates were vetted for suitability to the study, and interview themes were introduced with background information.

The reliability of the data and derived results was ensured by applying a peer-reviewed research protocol. Threats to reliability were mitigated, particularly in the analysis phase, by involving all three researchers. The data itself was recorded using Jira (atlassian.com/software/jira) and Confluence tools (atlassian.com/software/confluence), which allowed other researchers and experts to review and correct the data.

Some danger of positive bias exists in the study's results and activities due the way the case study was implemented. The constant communication among researchers, the development team and customer representatives, as well as the inclusion of researchers on the development team, increased the likelihood of producing only positive results. This danger was mitigated by having one researcher to evaluate the plan, results and actions without participating in the development team activities. Further, clear roles and rigorous research methods helped the researchers to maintain an objective perspective throughout the study.

IV. RESULTS AND ANALYSIS

This section presents the extended CTP model (Figure 1.) with identified practices for continuous improvement and customer validation adapted from SD and Lean UX approaches. The practices are defined based on the empirical findings of the data collection and analysis as well as the authors' previous experiences when working with software development teams. The benefits and challenges of applying the extended CTP model in agile software development are then discussed.

A. *Recommended practices for the CTP model*

Continuous improvement of the service unfolds true qualitative and quantitative data collected from customers and products in the field. Various experimentation techniques and combinations can be applied throughout the development process. The role of the CTP model in the present study was to guide the development team to experiment and collaborate with several internal and external stakeholders during the development process and collect feedback.

The purpose of extending the CTP model was to allow the continuous identification, prioritisation and validation of product assumptions in short development cycles with the help of customers, using identified practices presented in touchpoints T_1-T_4 in the model. These practices should be applied as continuous activities to improve service and validate product hypotheses. The findings of this study indicate that to fully utilise the practices, the development team must have an agile and lean mindset, be self-organising and share responsibility. Also, the team must be cross-functional, with one designer onboard. In-depth analysis of the quantitative data requires data analytics and data knowledge expertise, as analysing and utilising data is much more complicated than collecting it.

Discovery: The first customer touchpoint T_1, is used to discover and compile customer needs and translate them into product requirements and product hypotheses, which can be used in experiments. Typically, new ideas and requirements are collected from various sources, including market and competitor intelligence, feedback from customers and product usage data (after the first release of the product has been made).

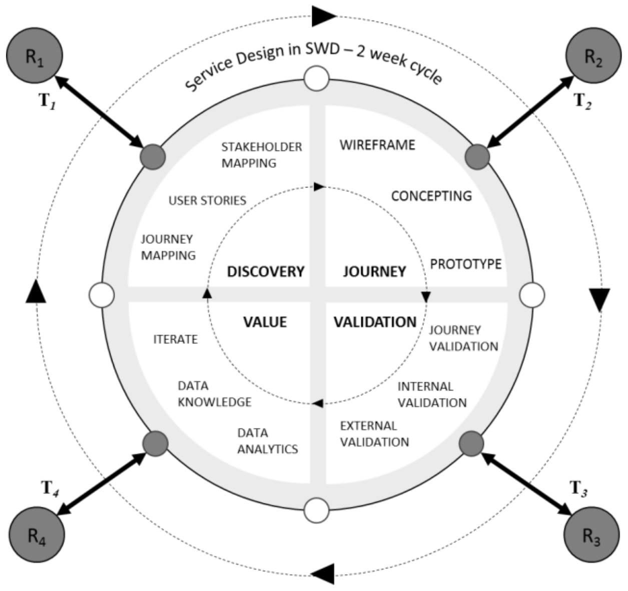

Figure 1. Lean UX practices in extended CTP model. R refers to the main activities: R_1 Collection; R_2 Prioritization; R_3 R&D Verification and R_4 Deployment. T indicate where companies and customers exchange information: T_1 Release learnings or new customer requirements, T_2 Release trade-offs and cost/benefit analysis, T_3 Release features and delivery commitments and T_4 Release configurations and real usage vs. planned usage of the product/feature. R_1–R_4 are also sources for requirements for new product ideas, including market and competitor intelligence, product usage data, customer feedback data collected, etc.

Techniques for collecting qualitative data can vary, from surveys and interviews to observations. Product assumptions are then turned into hypotheses for validation. The practices identified and utilised by the development team in this study included customer journey and stakeholder mapping, user analysis and creation of user stories. Journey mapping [9] provides a structured visualisation of a specific element, be it a single feature or an overview of the entire customer experience. Stakeholder mapping [18] on the other hand governs the overall complex situation and the expectations from various stakeholder groups. Benchmarking through online research and market data collection can also be used as an input to these practices.

Key learning: Balance discovery activities with development efforts by testing only high-risk hypotheses and conducting smaller amounts of research more often.

Journey: The second customer touchpoint T_2, is used to visualise an organisation's service vision and the customer journey. Such visualisation, considered as MVPs in this study, lends insight to improve the understanding of information. In the early phases of product development, many uncertainties related to customer expectations arise. Concept work is required when envisioning the idea for a service, or even at the feature level when designing prototypes and mock-ups. Visualisation techniques like wireframes, UI mock-ups, interactive prototypes and evolutionary prototypes guide development teams to prioritise development tasks continuously through experimentation with product concepts and design artefacts. In terms of prototyping, interactive prototypes are simulations that typically illustrate few predefined scenarios. They look and function like end product but do not handle real data input, processing or output. This technique proposes a quick and cost-effective way to concretise and test new ideas before any development work is done and help software teams to avoid developing unnecessary features. Evolutionary prototypes, build based on lessons learned from the interactive prototypes, however handle real data and form the basis for the actual software evolving after every iteration. Results of the present study indicate that design tasks should be treated as equal to development tasks and executed in the same or parallel cycles during software development. Digital product prototyping tools, such as Invision (invisionapp.com), UXPin (uxpin.com), and Sketch (sketchapp.com) were determined to be useful in this phase because they allowed for more effective collaboration and experimentation with product assumptions.

Key learning: Visualisation techniques create a common understanding by which to resolve communication-related issues. However, adapting these techniques require the role of UX designer in the development team.

Validation: Most of the validation in touchpoint T_3 occurs through testing activities. Testing activities are an opportunity to validate mismatches between customer needs and product offerings. As experienced in our previous research [2], this opportunity is often overlooked. Adopting the journey validation practice, trough different visualisation techniques, helps a development team to identify their needed steps, possible pain points and gaps. Internal validation then covers practices like daily scrum meetings, wikis and other communication channels to provide quick and frequent feedback from executives and team members. During daily scrum meetings in the present study, it was observed that the work that had been visualised on Kanban boards was often the work that got done. This demonstrates the importance of making product discovery and UX design tasks visible in a backlog. In this case, the UX tasks were visualised, executed and tracked in two-week sprint cycles alongside development tasks. External validation was done by exposing experiment materials to customers in two-week cycles to elicit and collect feedback. The experiment materials included interactive prototypes, mock-ups and evolutionary prototypes. Feedback was collected from customer workshops after every cycle, and utilizing usability testing with direct feedback mechanism in the working software integrated into the product development practices. It was concluded that it is important to involve the entire development team in external validation activities like workshops, as doing so helps the team avoid misunderstandings and the incorrect implementation of features.

Key learning: Customer needs and requirements change constantly. Backlog prioritisation should be done with high flexibility in short cycles against validated learnings.

Value: The ability to combine qualitative and quantitative customer feedback data is important, as often they complement each other. For example, by monitoring feature usage, quantitative data can be used to validate qualitative data. Analytic tools and collected product operation and performance data allows development teams to make data-driven decisions to improve the service in customer touchpoint Value T_4 continuously. In the present case, project log files and product usage data such as clicks, page views and number of visits were recorded and analysed. Especially in the field of web development, collecting and analysing product usage data has been eased by general tools such as Google Analytics. Target areas for quantitative data collection can range from system performance and UX improvements to business-level decision-making. In the present study, corrective actions based on quantitative data were done for example by improving the system level performance as well as placement and visibility of UI elements and workflows.

Techniques such as A/B testing can also serve as a potential approach to collect product usage data. In the present case study, A/B testing was not conducted because developing different variations of the same feature required vast resources and a large user base. In addition, a more detailed analysis of the log data would require specific skills and resources, such as adding a data scientist to the team.

Key learning: Quantitative data reveals tacit and complex knowledge of product usage. Detailed analyses of data and systematic, controlled experiments require the active presence of a data scientist.

B. *Benefits of the extended CTP model in software development*

The use of the extended CTP model with the identified practices as part of the two-week software development cycle was determined to be beneficial from the customer and development team points of view, but some challenges were identified. The benefits (Table I) and challenges (Table II) based on the research data are discussed next.

The most significant observed benefit of using the extended CTP model in software development was that it assisted both business and technical decision-making by promoting a better understanding of information (i.e., tacit knowledge is transformed into tangible knowledge). For the development team, direct interaction with customers enabled them to understand the reasons behind customer requirements and validate which features brought real value. Shared understanding also facilitated faster decision-making, which enabled the team to prioritise tasks 'on the fly' and estimate the validity of tasks in the product backlog.

The findings from this case study demonstrate that at their best, these practices increased customer involvement and nurtured innovation. Involving users in the early phases of development made customers more active and experienced with the product. This increased the motivation of development team and especially users, as they experienced the impact of their involvement during the process.

In terms of decision-making, the present results indicate that visualisation is one of the most important ways to concretise and test new ideas, as information is made visible to both customers and the development team through design artefacts.

TABLE I. IDENTIFIED BENEFITS

Benefit	Description
Learning and decision-making	Accelerate the decision-making process and concretise functions before actual development work.
Innovation and value co-creation	Accelerate co-creation and innovation between development team and users. Receive the first user feedback in the product/service idea phase.
Motivation	Interaction and co-creation between development team and users motivated both the development team and the users.
Communication improvement	Visualisation and prototyping methods improved communication and helped avoid misunderstandings between different stakeholders (e.g., management, development team, customers and end-users).
Transparency	Increase transparency between customers, users and the development team. Reveals grassroots knowledge exploitable in development.
Direct feedback	Presents an opportunity to receive instant and direct feedback from end-users in short cycles.
Integrated UX work	Design activities take place in the same cycle with development activities.
Service vision in communicable form	Developers have a clear, precise schematic by which to see the intended service from the customer perspective and can choose precise specifications.
Prioritisation	Prioritisation 'on the fly' allows the development team to capture changing priorities in short cycles and react flexibly and accurately.
Holistic approach	Holistic approach to software development from customers' and end-users' point of view.

The interviews stressed the need to produce proper visualisations, such as interactive prototypes and UI mock-ups, to concretise the service vision and improve communication to avoid misunderstandings. According to the interviewees, UX work was often perceived as a separate function, one asynchronous with research and development activities. It was also stated in the development team interviews that sometimes UX work was overlooked, as things like code quality, software scalability and security issues were deemed more important. *"Usability and user interface design… perhaps we cannot appreciate these or we consider more important that the software itself works or the code is good quality, the software is scalable and safe and so one, which is of course important… but from my perspective, I see that it is old-fashioned engineering thinking. …today, in my mind it (usability) is the biggest competitive advantage of software, (senior software developer)."*

Visualisation techniques used in the workshop sessions also bridged the communication gap between those with business mindsets and those more technically oriented. It was important that the whole development team participated in the workshop sessions to get direct and instant feedback from customers. This helped the development team to understand the reasons behind customer needs, the kind of behaviours expected from the service and what created value for customer. During the sessions, the development team was able to identify design improvement ideas for the future and problem areas in the existing design. This had a clear impact on project work such as planning, scheduling and prioritising tasks, as the team was able to capture and react to changing priorities in short cycles. The practices presented in the extended CTP model facilitated better alignment and transparency of different functions, from UX design, business development and product management to short cycles with research and development activities.

By adopting visualisation techniques as suggested in the extended CTP model, the development team was able to obtain their first user feedback after very short iterations (a few days to a few weeks). The model also guided the entire development team to holistic thinking by following a customer centric design and development process throughout the project lifecycle.

C. Challenges of the extended CTP model in software development

Stakeholder availability and commitment represented some challenges for example, if few or no visible functions were delivered in a short iteration cycle, customer and stakeholder commitment to participate in the workshops could suffer. In addition, customers may feel disrupted from their own work by being involved in development activities too often. The development team reported the two-week cycle as optimal, but based on the interviews it could be shortened to one week. For customers, the two-week development cycle with a workshop at the end of each sprint was sometimes seen too often.

New requirements appear all the times and they may disrupt the old ones, often this is related to different user groups and customer roles with conflicting needs. This may also add complexity to defining feature maturity and identifying metrics for measuring customer value.

For a software team to adapt practices presented in the extended CTP model, certain key roles must be filled, which may cause some challenges. The current case study required expertise of back-end and front-end development, database development, server and tool management, a product owner to represent the customer and a UX/UI designer to conduct and facilitate design tasks. While the development team and effort were kept as minimal as possible, it was evident that UX design and different visualisation techniques required significant effort; the person in charge of the design had to discuss and make decisions on the design with different stakeholders who often held conflicting demands and priorities. In order to mitigate this problem, a stakeholder analysis method [25] was used to gain further insight on the conflicting issues and resolve them. The UX work alone was often full-time work for a single person, even in a lightweight web application project. Furthermore, detailed analysis of the log data would require specific skills and resources, such as adding a data scientist to the team and a large user base. In this study, analytic support was integrated to working software and some corrective actions were continuously taken based on quantitative data. However, small user base and lack of resources hindered more detailed analyses. It should be noted that in some industries, laws and regulations or other limitations could prevent quantitative feedback collection.

TABLE II. IDENTIFIED CHALLENGES

Challenge	Description
Stakeholder availability and commitment	Finding suitable time for all stakeholders to participate workshops. Balancing workshop frequency and getting customer commitment to participate and give feedback.
Customer role	Different user groups might have different needs, and those can change according to context.
Measurement of customer value	Identify metrics for how to measure customer value is challenging.
Resources	Successful implementation required having a cross-functional team with an active UX/UI designer and data scientist in the team.
Direct communication	Developers may not always be confident or comfortable with increased and direct customer interaction.
Visualised features might be understood as 'easy to do'	Workload estimation can be challenging from the customer perspective. E.g., features in an interactive prototype and real effort for actual implementation might come as a surprise.
Maturity and definition of done (DOD)	Changing priorities and stakeholders make it hard to analyse the maturity and DOD of features.
Quantitative metrics / analytics	Quantitative metrics and analyses become possible after implementing analytics support, once the software is in use and there is a sufficient stakeholder pool using it. Before that, obtaining quantitative data is nearly impossible.
Physical distances	Technical issues and lack of tools for sharing local demonstrations.

Development team interviews of the present study revealed that some developers were more confident when allowed to focus simply on building the software, preferring to keep their interactions with customers at a minimum or restricted to key individuals.

While a working prototype proved an efficient way to involve stakeholders in the development process, the downside was that customer expectations rose when they saw how 'easy' it was to produce something visible that matched their needs. This led to situations where customers expected that the feature's implementation would be easy as well. The challenge therefore is the increased risk of customers developing unrealistic expectations related to work estimations for the complete functional feature. During the study, this was addressed by maintaining a careful balance between keeping stakeholders satisfied even the workload would be bigger and prioritizing these features over those that are not visible but can be important for system stability, security and other issues that do not show in day-to-day work.

Physical distances reduced the development teams ability to interact with their customers face to face, as well as posed a challenge during the workshops and meetings due to technical issues, such as connection quality and faulty communication equipment. While these technologies enable participation regardless of physical location, communication tools do fail at times, or stakeholders are not used to hold online meetings. Mistakes like using a low volume of voice, moving away from the microphone or demonstrating locally something not visible to those participating remotely can slow down the meeting, make communication difficult and increase the risk of misunderstanding.

V. DISCUSSION

Software development is not an easy or straightforward task. It is even more challenging when requirements are constantly changing and there are many involved customer organisations with different demands. Previous research [2] shows that, delivering value and meeting customer needs in constantly changing markets are key success factors for any business. Multitude of existing methods and approaches for identification, prioritization and validation of customer needs are presented in literature, with more recent being continuous experimentation. From a process perspective, Schermann et al. [26] reports a lack of clear guidelines for conducting experiments. Our extended CTP model fills the gap and presents an approach for continuous improvement and customer validation.

Typically, existing methods and approaches for continuous experimentation (presented in Section II of the paper) are influenced by the 'build-measure-learn' loop and include the phases of planning, collecting customer feedback and analysing of data for learning and decision-making purposes. While some of the approaches focus on the technical implementation and instrumentation for collecting customer feedback data, the extended CTP model offers a more holistic outside-in approach and integrates UX work in the same development cycle. The model can be seen as a framework that guides development team to experiment and collaborate with internal and external stakeholders during the development process. Experimentation does not always require working software as experiments can be conducted with other kinds of experimental artefacts, such as visualizations or mock-ups. Multiple variations of the extended CTP model can be derived based on context and available resources.

The study found two main contributions from the extended CTP model. First, the extended CTP model enhances software development process with practices to improve feedback elicitation, continuous improvement and customer validation through its suggested practises in *1) Discovery, 2) Journey, Validation* and *4) Value* customer touchpoints. Second, the extended CTP model integrates collaborative product discovery with product delivery tasks in short cycles while recognizing the role and importance of UX work. Conducting smaller amounts of research but more often, involving the entire development team and testing the high-risk hypothesis with visualisation techniques are encouraged. In this way, the model offers a new holistic approach to continuous improvement and customer validation.

The research results highlight that teams adopting recommended practices need to have a cross-functional competences, innovative and experimental mindset in which experimentation is seen as a learning opportunity. This is in line with previous continuous experimentation research where organisational culture act as barrier to the incorporation of the practices [17]. In general, adopting these practices benefited both the development team and users and enabled the team to become more customer-driven while focusing on the tasks most relevant to the users. The main benefits relate to learning and decision-making, innovation, value co-creation and communication. We also identified number of challenges that may hinder customer involvement and adapting identified practises, such as stakeholder availability, commitment and measuring customer value. We acknowledge that adapting visualisation practices and quick prototyping requires a cross-functional team with UX design activities and tasks aligned together with the research and development tasks. In addition, experimentation with quantitative data requires a data scientist to be present in the team. Unlike large companies, such as Microsoft [27], this may not always be feasible due to limited resources. We also argue that analysis of quantitative data is not a sole responsibility of a data scientist, as it requires deep customer insight from qualitative data. It this sense, data analysis could fall under product management tasks as the shared responsibility of a product owner, UX designer and data scientist were also business aspects are taken into consideration. This reinforces the fact that there is not one right approach for software teams to become more customer centric. Rather, practices should be tailored to meet specific domain and contexts, distinct business goals and different organisational cultures.

VI. Conclusion and Future Work

In this paper, we presented a case study aiming to help software teams to continuously validate customer value in short development cycles to increase customer understanding trough practises presented in the extended CTP model. The current findings are based on empirical data gathered from a software development team developing a modern software-as-a-service platform called Fenix. The Fenix system was designed to help companies grow their revenue through digital partnering and ecosystem creation strategies.

This study shows that several methods and practices can be utilised during software development to capture and validate customer needs thus helping development teams to reduce the gap between user expectations and actual implementation. The extended CTP model presents the identified practises through customer touchpoints. By recognising the synergies between continuous customer collaboration, integration of design tasks and involving all the relevant stakeholders, along with the ability to combine qualitative and quantitative feedback, software teams can speed up their delivery process and become more data-driven in their learning and decision-making processes.

Future work on this topic should examine how the extended CTP model works when the web application in question is developed further and the user base is substantial enough to use controlled experiments. This would allow for interesting evaluations of the different practices used in the extended CTP model when used to develop already established software rather than restricting investigations into new software development.

Acknowledgments

This work was supported by the Six City Strategy (6Aika) and the European Regional Development Fund (ERDF), Business Finland, Allied ICT Finland and its ecosystems.

References

[1] A. Nguyen-Duc, X. Wang, and P. Abrahamsson, “What influences the speed of prototyping? An empirical investigation of twenty software startups”. In: Software Engineering and Extreme Programming, XP 2017. Lecture Notes in Business Information Processing, vol. 283, pp. 20–36. Springer, Cham, 2017.

[2] T. Sauvola et al., ”Towards customer-Centric software development, A multiple-Case study”. In: 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2015, pp. 9–17. IEEE, Funchal, Portugal, 2015.

[3] F. Fagerholm, A. S Guinea, H. Mäenpää, and J. Münch, “The RIGHT model for continuous experimentation”. J Syst Software vol. 123, pp. 292–305, 2017.

[4] K. Beck, J. Grenning, and R. C. Martin, Agile Manifesto. http://www.agilemanifesto.org/ [retrieved: August, 2018]

[5] E. Ries, The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. Penguin Group, London, 2011.

[6] H. Holmström Olsson, J. Bosch, and H. Alahyari, “Towards R&D as innovation experiment systems: A framework for moving beyond agile software development”. Proceedings of the IASTED, pp. 798–805, 2013.

[7] S. G. Yaman et al., “Transitioning towards continuous experimentation in a large software product and service

development organization: A case study". In: P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, and T. Mikkonen (eds). Product-focused Software Process Improvement, PROFES 2016. Lecture Notes in Computer Science, vol. 10027, pp. 344–359. Springer, Cham, 2016.

[8] J. Knapp, J. Zeratsky, and B. Kowitz, Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon and Schuster, New York, 2016.

[9] J. Gothelf and J. Seiden, Lean UX: Designing Great Products with Agile Teams. O'Reilly Media Inc., Sebastopol, 2016.

[10] J. Bosch, "Building products as innovation experiment systems". In: M. A. Cusumano, B. Iyer, N. Venkatraman (eds). Software Business, ICSOB 2012. Lecture Notes in Business Information Processing, vol 114, pp. 27–39. Springer, Berlin, Heidelberg, 2012.

[11] F. Elberzhager, T. Arif, M. Naab, I. Süß, and S. Koban, "From agile development to devops: Going towards faster releases at high quality: Experiences from an industrial context". International Conference on Software Quality, 2017, pp. 33–44. Springer, 2017.

[12] E. Anderson, S. Y. Lim, and N. Joglekar, "Are more frequent releases always better? Dynamics of pivoting, scaling, and the minimum viable product". Proceedings of the 50th Hawaii International Conference on System Sciences, 2017, pp. 5849–5858, 2017.

[13] J. Bosch, H. Holmström Olsson, J. Björk, and J. Ljungblad, "The early stage software startup development model: A framework for operationalizing lean principles in software startups". In: B. Fitzgerald, K. Conboy, K. Power, R. Valerdi, L. Morgan, K.-J Stol, (eds). LESS 2013. LNBIP, vol. 167, pp. 1–15, Springer, Heidelberg, 2013.

[14] H. Holmström Olsson and J. Bosch, "The HYPEX model: From opinions to data-driven software development". In: Bosch J. (ed). Continuous Software Engineering. Springer, Cham, 2014.

[15] H. Holmström Olsson and J. Bosch. "Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation". In: J. Fernandes, R. Machado, K. Wnuk (eds). Software Business, ICSOB 2015. Lecture Notes in Business Information Processing, vol. 210, pp 154-166, Springer, Cham, 2015.

[16] R. Kohavi, A. Deng, and B. Frasca. "Online controlled experiments at a large scale". Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 1168–1176, 2013.

[17] E. Lindgren and J. Münch, "Raising the odds of success: The current state of experimentation in product development". Inform Software Tech, vol. 77, pp. 80–91, 2016.

[18] M. Stickdorn and J. Schneider, This is Service Design Thinking (1st ed). 2012.

[19] J. Parades, C. Anslow, and F. Maurer, "Information visualization for agile software development teams". In: Second IEEE Working Conference on Software Visualization (VISSOFT), 2014, pp. 157–166, 2014.

[20] Y. Lee, "Design participation tactics: The challenges and new roles for designers in the co-design process". J Co-design, vol. 4(1), pp. 31–50, 2008.

[21] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering". Empirical Software Engineering, vol. 14(2), pp. 131–164, 2009.

[22] R. K. Yin, Case Study Research: Design and Methods (5th edn). Sage Publications, 2013.

[23] M. D. Myers and M. Newman, "The qualitative interview in IS research: Examining the craft". Inf Organ, vol. 17(1), pp. 2–26, 2007.

[24] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering". In: International Symposium on Empirical Software Engineering and Measurement (ESEM), 2011, pp. 275–284, 2011.

[25] M. Kelanti, J. Hyysalo, J. Lehto, S. Saukkonen, P. Kuvaja and M. Oivo "Soft Systems Stakeholder Analysis Methodology". Proceedings of the 10th International Conference on Software Engineering Advances (ICSEA 2015), pp.122–130, Barcelona, Spain, 2015.

[26] G. Schermann, J. Cito, P. Leitner, U.Zdun, and H.C. Gall, "We're doing it right live: A multi-method empirical study on continuous experimentation". Information and Software Technology, 99, pp.41-57, 2018

[27] M. Kim, T. Zimmermann, R.DeLine, and A. Begel. "Data Scientists in Software Teams: State of the Art and Challenges. IEEE Transactions on Software Engineering", 2017, pp. 1–17, http://dx.doi.org/10.1109/TSE.2017.2754374