When Prototyping Meets Storytelling

Practices and Malpractices in Innovating Software Firms

Raffaele F. Ciriello[1,2], Alexander Richter[1,2]
[1]Department of Business IT
IT University of Copenhagen
Copenhagen, Denmark

Gerhard Schwabe[2]
[2]Department of Informatics
University of Zurich
Zurich, Switzerland
{ciriello,arichter,schwabe}@ifi.uzh.ch

***Abstract*—**Storytelling is an important but often underestimated practice in software engineering. Whereas existing research widely regards storytelling as creating a common understanding between developers and users, we argue that storytelling and prototyping are intertwined practices for innovators to persuade decision makers. Based on a two-year qualitative case study in two innovating software firms, we identify and dialectically examine practices of storytelling and prototyping. Our study implies that storytelling and prototyping should be integrated together into software engineering methods.

Keywords: Prototyping, Storytelling, Innovation, Practices, Malpractices, Case Study, Software Engineering

I. Introduction

In software engineering, prototyping and storytelling are widely regarded as distinct but related approaches to support requirements elicitation and validation [1]–[5], idea experimentation and exploration [6]–[10], facilitating communication [5], [10], and decision making [10]–[12]. However, existing research regards storytelling merely as creating a common understanding between developers and users in the sense of use cases and usage scenarios. Here we report on an in-depth qualitative case study of software prototyping in an innovation context within two Swiss software companies. Our data shows that innovators combine prototyping and storytelling to persuade decision makers and transfer implicit knowledge.

Prototyping is a complex, multifaceted activity whose outcome depends on many factors, such as form and function of the prototype [13], how it is used [14], by whom it is used [15], and various context factors such as project setup, development approach, organizational environment, and infrastructure [9]. We studied and observed prototyping and storytelling practices for over two years in depth via a case study involving extensive interviewing, observation, and collection of documentary material. We observed that a key problem in innovating software firms is communicating ideas purposefully to different audiences. This requires skillful use of communication tools like prototypes to persuade and collaborate with relevant stakeholders (cf. [16]). Thus, we address the question: *How do people communicate innovative ideas with software prototypes?*

Our contribution informs software engineering scholars and practitioners about the importance of prototyping and storytelling in organizations. We identify a set of practices that help to better understand the role of prototyping and storytelling for communicating ideas, persuading decision makers, and transferring implicit knowledge. An important practical implication is that storytelling and prototyping are deeply intertwined and should thus be integrated together into software engineering methods like agile software development. When combined and integrated into agile software development, prototyping and storytelling can increase customer involvement and satisfaction through early, continuous, and frequent delivery of working software (i.e. prototypes and the stories inextricably bound to them), facilitate close, co-located, and periodical cooperation between business people and developers (effectively by means of stories), and improve product simplicity.

II. Related Work

A. A Practice Perspective on Innovating Software Firms

Developing innovative software systems is risky and failure-prone, but essential for software firms to thrive and survive in a dynamic and globalized market [17]. Prototypes can help to innovate with reduced cost and risk, as they support the early clarification of relevant problems and serve as a basis of discussion and further development [5]. However, we know yet relatively little about the many roles of prototyping as a practice to support the communication about innovative ideas in organizations. Here, an *idea* is defined as an underspecified, abstract conception of an envisaged product in someone's mental model, i.e. an intangible and volatile image in the mind of a person [18, p. 303f]. Ideas often originate from problem-solving engagements [19, p. 25ff]. Only when somebody communicates an idea, it meets the realm of reality and becomes a germ cell of innovation [20]. Hence, we adopt a practice perspective to highlight how people use prototypes to communicate ideas, allowing us to better understand the role of prototypes in developing innovative software. When referring to *practices* in this paper, we refer to materially-mediated sequences of human activity centrally organized around shared understandings [21].

B. Software Prototyping

A *software prototype* (in the following just *prototype*) can be defined as a model of an envisioned software system that provides a basis for discussion, clarification, decision-making, experimentation, and learning between different stakeholders [5]. Prototypes represent ideas and simultaneously highlight and exclude aspects that are deemed critical or unimportant, respectively [14], [22]. *Prototyping* can be understood broadly as a practice to develop, demonstrate, evaluate, modify, and experiment with prototypes [10], [23]. Prototypes differ in *fidelity*, i.e. the prototype's closeness to the 'original', most commonly understood as the degree to which the prototype's visual refinement accurately represents the appearance and interaction of the envisioned system, not the accuracy of code or other attributes invisible to the user [24]. Low-fidelity prototypes (e.g. sketches, wireframes, or paper prototypes) are useful when a team tries to identify requirements, whereas high-fidelity prototypes are useful to create living specifications [24]. Prototypes can be represented as breadboards, presentation prototypes, functional prototypes, or pilot systems [9], [10]. *Breadboards* are used as proof-of-concept to investigate technical aspects in the back-end system [10] (e.g. system architecture, algorithms, data processing) and are not normally evaluated by end users [9]. *Presentation prototypes* provide a concrete preview of an abstract idea [10] by illustrating how the envisioned system may solve given requirements and focus mostly on the user interface [9]. *Functional prototypes* implement only the critical features with which the user needs to work in essence [9], [10]. And *pilot systems* are working systems that can be practically applied but still need technical finalization to count as a full system [9], [10].

Prototypes are useful for requirements elicitation and validation [1]–[5], idea experimentation and exploration [6]–[10], facilitating communication [5], [8], [10], [23], and decision making [10]–[12].

Requirements Elicitation and Validation: Because it is often not until the finished system is in use that users are able to explicitly formulate their system requirements [2], [5], prototypes can reduce time, risk, and cost of software development projects while improving quality and usability through iterative cycles of requirements elicitation and validation in early phases [1], [3], [4], [9]. Prototyping allows users to see and interact with a prototype, and to provide more constructive and detailed feedback [5], [12], [25].

Idea Experimentation and Exploration: Prototyping helps to explore and experiment with ideas in early innovation project phases [7], especially in the software industry where new product developments are characterized by high uncertainties and risk [8], [17]. By prototyping their way to a solution, teams can learn-by-doing what works and what not [8], [9]. As prototypes embody implicit and explicit design hypotheses that can be tested [26], the creation of knowledge about problems and potential solutions is often seen as their raison d'être [10], [25]. The outcome of such a process is often a prototype contaning just the essential features for validated learning, also called a Minimum Viable Product (MVP) [27].

Facilitating Communication: Prototypes can support and enrich coordination, communication, interaction, and motivation among various stakeholders [8], [9], [23]. Prototyping requires mutual coordination between developers and users throughout the entire development process [28], with each group continuously acquiring knowledge about the work practices of the other [5] and about the role the new system plays in the user's life [9], [24]. Prototypes provide a concrete basis for communication between users, developers, and decision makers. They support discussions of particular problems, clarification of particular questions, or preparation of particular decisions [5]. When used in social interaction, prototyping can spark creativity [6] and is helpful for externalizing and representing ideas [9], [13]. In addition, they play an influential role in generating and motivating teams that are bound together by the common purpose of fulfilling the prototype [8].

Decision Making: Prototypes can be used to sell ideas [29], prevent misunderstandings [9], [12], assess risks [11], and gain insights about feasibility, desirability, and viability [9], [10]. Thus, they influence decision making in ways not possible for written reports [9], [12]. Awareness of the needs of the different stakeholders and their views on a prototype is crucially important for decision making. Different audiences have different roles in the joint activity of prototyping, and thus bring different perspectives and interpretations to a prototype [15].

C. Storytelling

Storytelling can be simply understood as the communication of ideas, beliefs, and experiences via stories [30], [31]. A *story* can be defined as an account of actions that is formulated from authentic events, either real or imagined [32], [33]. Stories comprise characters, whereby there is usually one character (the "hero") the listeners identify with. Their structure consists of a beginning, middle, and an end, that is held together by a plot [34]. The sequence of actions can unfold in time or thematically [35].

In recent years, storytelling has gained importance as a technique for contextualizing information in business and technical domains, like knowledge management [30], [35], [36] or software development [32], [33], [37], [38]. In an organizational environment, a story can be understood as a detailed narrative of past actions and interactions of employees and managers that are communicated informally within or across organizations [36]. Stories are an inherently appealing form of communication that outperforms other formats in terms of memorability, learnability, persuasiveness, and ability to bind different communities together [36]. Because they are more vivid, engaging,

entertaining, and easily related to personal experience than specifications, rules, or guidelines, stories are more likely to be internalized, acted upon, and guide behavior [36]. In addition, stories encode rich contextual details, which makes them suitable carriers of implicit knowledge [36]. For instance, video stories facilitate seeing and hearing the characters and the environment in which they are situated, and thus add further cues and detail [37].

In software development, stories can be superior to more abstract use cases or usage scenarios because they are more vivid and thus effective for getting the attention of people [33]. Listening to people's stories helps to understand their needs, and crafting them together helps to shape the vision of a desired system [33]. Several attenuated forms of stories have been introduced into software development practice to explore and define a system's requirements. For instance, use cases are sequences of actions that describe how a generic actor or user interacts with a system, usually as a list of short, written steps [36]. And usage scenarios describe real-world situations of how people interact with the system, usually in the form of short prose involving various personas, steps, events, and/or actions that occur during the interaction [37]. Usage scenarios can be problem-oriented descriptions of the current state of affairs, activity-oriented descriptions of broadly defined actions the user performs with the system, or interaction-oriented description of detailed actions [39].

A good story complements a prototype in ways that cannot be achieved with traditional use cases and usage scenarios. Use cases and usage scenarios offer a brief statement of the requirements, usually comprising technical details of work packages or features to be implemented. Their purpose is essentially to map user requirements to system requirements, and to facilitate communication between users and developers [37]. In turn, stories provide a detailed, narrative illustration of the situation in a real context, thus enabling a high-resolution understanding of the involved people, along with their reasoning, interests, desires, needs, and environmental context [32], [33], [38]. Their purpose is in essence to provide rich context and map envisioned ideas to enacted ideas. Stories facilitate interaction between innovative employees and decision makers, such as managers in their formal role or even users or peers in their informal role when they support decisions with feedback [36]. Stories help designers to make detailed descriptions of human living people during the design process and can be used to give people an orientation [38]. The promise of introducing storytelling in the system development process is to use the potential of flexible interpretations [38]. A story that is interesting, appealing, and well-positioned in relationship to the real-world experience of decision makers can be a tremendously helpful means to persuade people and communicate an idea effectively [33]. It can help designers, users, customers, and developers to understand how the idea will change the existing situation in that it facilitates communication and shared understanding between stakeholders [37].

In sum, there is a research gap concerning the roles of prototyping and storytelling in relation to the communication of innovative ideas in organizations. Existing literature sees storytelling merely as creating a common understanding between developers and users, in the sense of use cases and usage scenarios. Thus, the role of storytelling in relation to prototyping remains unclear and underestimated, particularly when communicating ideas to persuade decision makers and transfer implicit knowledge. Thus, we address the guiding research question: *How do people communicate innovative ideas with software prototypes?*

III. Research Method

This research is situated within a larger study of innovation practices in the software industry. Over the course of more than two years (02/2013-12/2015), we obtained and sustained in-depth field access to two Swiss software companies, where we engaged in substantive interviewing, participant observation, and collection of prototypes and related documentary material. Our study is exploratory in nature. Following the principle of concatenation, we used the emerging findings of an initial study on idea communication to feed the next sub study [40]. More specifically, we realized that prototypes played a vital role in processes of generation, communication, negotiation, and development of ideas. This led us to analyze the prototyping practices in detail. We embarked on an iterative journey of data collection and analysis until we identified the key theme, namely the role of storytelling in software prototyping.

A. Research Relationship with the Case Companies

We selected two different kinds of companies, namely one product company and one project engineering company, to study them in depth and compare prototyping practices across different organizations. Both companies are characterized by a large percentage of employees who graduated from one of the leading computer science departments in the world. The company names are anonymous at their wish.

Banking and IT Solutions (BITS): For more than 20 years, the traditional business model of this company with around 1400 employees has been the development, distribution, and operation of its proprietary core banking system. After the executive board became increasingly concerned that the lifecycle of this product might have peaked, BITS took various extensive measures to develop new products and services in the areas of mobile banking, outsourcing, financial services, and consulting. Our style of involvement with BITS was that of a closely involved researcher having in-depth access to data, issues, and people, who viewed the researcher as one of ‘them’, trying to make a valid contribution to the field site [41].

Custom Software Engineering (CustomSoft): For almost 20 years, the core business of this company with around 350 employees has been the development of and consultancy for custom business software in segments including transport, health, space agencies, public administration, banks, and insurances. CustomSoft recently initiated efforts to rethink its business model from a project engineering to a product company to reduce the financial risk stemming from the company's high dependence on client orders. The style of involvement with CustomSoft was that of an outside observer who was not seen as having a direct personal stake in various interpretations at outcomes, with personnel frankly expressing their views [41].

B. Data Collection and Analysis

The first author conducted 95 semi-structured interviews ranging from 19 to 104 minutes (average 60 minutes) with experts involved in recent innovation projects at BITS and CustomSoft. By interviewing a wide range of participants with differing roles and from different units we were able to seek out and document multiple interpretations of the actions under study [42, p. 77]. The author used a semi-structured interview guide to ensure topical focus and consistency while also allowing respondents to freely express their own views. We recorded and transcribed all but two interviews to capture a full description of what was said and facilitate later in-depth analysis. Through these interviews, it was possible for us to step back and access the interpretations of the fellow participants in more detail [41]. We wrote up detailed interview notes within a day.

Following the idea of triangulation [43, p. 291], we relied on multiple sources of evidence, compiling multiple interpretations obtained from interviews, observations, field notes, and documentary material into a coherent picture [42]. For instance, we collected and analyzed 17 prototypes and related documents that participants sent us. In addition, the author conducted a series of participant observations at formal gatherings (meetings, workshops, presentations and fairs) and informal gatherings (lunches, impromptu meetings) in the context of the innovation projects, spending in total 211 full days at the research sites between 2013 and 2015. Where possible, photographs and field reports complemented the observations.

We carried out the data analysis collaboratively relying mostly on interview transcripts, collected documentary material, and field reports. The interviews were recorded, transcribed, and processed using MAXQDA, where two researchers developed a codebook to facilitate joint analysis and increase confidence in the findings [44]. Two additional researchers carried out coding checks to ensure intercoder reliability and develop a shared conception of reflection [44], through which we identified key themes.

C. Structured Literature Analysis

Parallel to the case data collection cycles, we conducted a structured literature analysis in which we followed the well-established framework by Vom Brocke et al. [45]. Hence, we conducted the five generic steps: *1) definition of review scope 2) conceptualization of topic 3) literature scope 4) literature analysis and synthesis 5) research agenda*. Steps 1-2 followed from the field study in which we identified research topics and the scope, namely the role of storytelling in software prototyping. In step 3 we searched on ACM digital library, AIS electronic library, and Google Scholar for the keywords "software, prototype, prototyping, storytelling, narrative, scenario, development, engineering", selected 68 sources from reading the titles, abstracts, and introductions. We then proceeded with the snowball technique, selecting further texts from the references cited in the sources and synthesized them into the literature review in section II (step 4). We finally framed the research agenda (step 5) by moving back and forth between data and literature, interrogating field material to check whether the data supported emerging claims, and whether literature helped us to make sense of the empirics [46].

IV. Results

This section provides detailed insights into the observed storytelling and prototyping practices in our case study at the two Swiss software companies BITS and CustomSoft. As shown in table 1, we group the practices into four categories: *Choosing the Script, Determining the Level of Detail, Engaging with the Audience, and Spreading the Message*. The categories represent essential aspects of prototyping and storytelling that emerged from our iterative analysis and interpretation of the data. Each category comprises a dyadic pair of practice and malpractice, which we examine dialectically in the following.

TABLE I - Overview of Practices and Malpractices

Category	**Practice**	**Malpractice**
Choosing the Script	Holding an "I Have a Dream" Speech	Telling Fairy Tales
Determining the Level of Detail	Presenting an "Elevator Pitch"	Using a Sledgehammer to Crack a Nut
Engaging with the Audience	Crafting the Story Together	1) Take It or Leave It 2) Premature Closure
Spreading the Message	Coupling Prototype and Narrative	Running from Pillar to Post

A. Choosing the Script

The script or plot is the structure that holds the story together and defines the sequence of actions that unfolds. This is a decisive factor of a story's success or failure and the narratives around a prototype should be chosen accordingly to accurately reflect real world situations of actual users.

1) Holding an "I Have a Dream" Speech

A good practice is to hold an "*I Have a Dream"* speech, i.e. using prototypes to show how one can bridge the difference between the current status quo ("what is") and the

future desirable state ("what could be"), along with a call for action to implement the idea. For this purpose, the storyteller needs to pick up the listeners in the world they live in and take them on a promising journey to a bright future. One observed challenge in this context is that not all potential users are trained to think in the same abstract categories like software professionals are. Here prototypes can be helpful as visual aids to provide living examples and speak the listeners' language. For instance, the mobile banking suite project team at BITS used a photo story to show how the prototype could make payment, wealth management, and real estate purchases easier on mobile devices. After inviting senior managers of various Swiss banks to learn about the product idea, several of them showed interest in the future product. The program manager remembers: "*The customers were excited. They understood: These are people with ideas. [...] Of course, we raised high expectations, for which to be fulfilled one has to wait several years, but the message to the market was important: We want to go in that direction". (I25)* An innovation partnership with several banks came about and, as a next step, the project team developed a functional prototype. Then, they sent the CEOs of the banks an iPad with the prototype app installed as a Christmas present. This was reportedly an effective instrument to help the listeners understand where the journey would lead to and in the end, the project was successfully developed and implemented. One software engineer remembers: *"[the prototype] was insofar helpful that they could see 'ah, that's how it could look like.' [...] The sole looking and touching already helped to explain what we wanted to show.*" (I1).

2) Telling Fairy Tales

A malpractice for choosing the script is to tell *Fairy Tales*. While prototypes can be compelling backbones of a story, one should pay close attention to how close the told story actually reflects a real-world situation in the listener's experience. For instance, the mobile banking suite project team once conducted a workshop with a private bank where they used the prototype to walk the customer through a seemingly illustrative user scenario. In this case the scenario was how a bank customer uses the app to finance the purchase of a new house. But then, the client advisors of the private bank protested that this is not even a real use case because their wealthy customers do usually not lend money to buy a house. They are rather more interested in optimizing cash flows in their portfolio. One project member remembers: "*Having the abstraction ability to see that instead of buying a house you could do the same thing with cash flows was already somehow difficult. So, the closer you are to the real life situation of the client advisors, the better." (I1)*. This implies that the storyteller should carefully choose the content of the story presented as context of the prototype. Our data revealed several cases of "dummy data" (e.g. "lorem ipsum" texts) leading to confusion, because the story was harder to interpret. While listeners may sometimes tolerate discussing over a prototype that looks sketchy and unfinished, they rarely tolerate any deviation from real life situations in a story. Hence, prototyping may involve stubs, sketches, and mockups to a certain extent, yet the story that is told behind must not be fictional but as close to reality as possible. In one workshop, a team of developers at BITS learned the hard way that using dummy data can have serious drawbacks: "*There were 30 people in the room [...] and everybody could see how it would look like in the end. That was good. At the same time, it was bad that we had dummy data in it, [because the sponsor] took it too seriously. [...] For us, it was only dummy data, arbitrary strings, could also have been the names of Beatles songs. But [the sponsor] insisted on it until we explained that it was only a string, not relevant for what we wanted to show. And in the end we created a set of test data with which the developers could then work with for long. That was the positive aspect of all that." (I28).* These drawbacks can be even more severe when the users confuse the prototype with the finished system and underestimate the necessary effort to finalize the development. If the storytellers do not make transparent what is 'behind the scenes' of the prototype, they do not only risk getting less constructive feedback, but they also risk raising false expectations among listeners and ultimately disappointing them: "*I made some HiFi clickable prototypes. [...] The executive board saw that and immediately raced around and said: 'In one week we go online with that!' [I thought:] 'Ah! Panic! There was nothing programmed, no HTML-code whatsoever, the whole implementation in the CMS, nothing was there.' And then, well, we could not stop it anymore. We had to take night shifts to bring the whole thing online just in time somehow."* (I82). In short, innovators should be careful that a) the context is as close to reality as possible and b) the prototype is not misunderstood or misused as the final system.

B. Determining the Level of Detail

The level of detail of the story being told is a critical issue. Including many details (e.g. actors, steps in the sequence of action, high fidelity of the prototype) may be tempting but may bore or confuse the target audience (especially managers and decision makers who are busy and want to focus on the essential idea aspects).

1) Presenting an "Elevator Pitch"

One observed way to choose the level of detail effectively is to present an "*Elevator Pitch"*, i.e. the storytellers include only the central details that are necessary to understand the key idea in as little time as an elevator ride. One CustomSoft software engineer told us of a small prototype app for a table soccer game. He regularly discussed it with his colleagues during lunch to decide what should be the next features to bring a benefit and what should be excluded. He reflects on this experience as follows: *"The ideal artifact is a prototype, as lean and light as possible, such that I can gather quick feedback, but also as fat as necessary, such that you can see the idea." (*I85*)*

2) Using a Sledgehammer to Crack a Nut

A malpractice to determine the level of detail is *Using a Sledgehammer to Crack a Nut*, i.e. over-engineering irrelevant aspects and neglecting relevant ones. Especially technically versed developers are at risk of building something just for pleasure. Instead of developing a

minimum viable story that contains just the necessary core aspects that are required to show how the idea creates value for the audience, people tend to put unnecessarily high efforts in technical gold-plating of solutions to ideas that are not yet well thought through: "*We are so used to building huge applications for lots of money. But actually, it takes much less to show an idea."* (I85). Although some study participants are aware of potential advantages of storytelling over gold-plating, they consider the main barrier to be a lack of writing skills and relatively high effort of writing a story clearly and concisely. For instance, one lead developer states that a good story is far more laborious than a prototype "*because you have to write it well." (*I11). Hence, not everyone agrees that the benefit of storytelling is worth the effort. One lead developer even states that it would show him that the employee already spent too much time on the idea if a well elaborated story would be the first thing the employee came up with. Here, it may help if storytellers are explicit about the purpose of storytelling, because managers usually wish for conciseness when ideas are communicated to them. As one interviewed manager argues, it is crucial that the storyteller is able to explain the main benefit clearly and concisely in few words: "*Many ideas are not presented concisely enough. People talk a lot but sometimes it needs just one precise sentence."* (I31).

C. Engaging with the Audience

Storytelling in the context of software prototyping should not be confused with a frontal presentation. Instead, we observed that it is a crucial factor how the storytellers engage and interact with the audience.

1) Craft the Story Together

One good practice to engage with the audience is to *Craft the Story Together*, i.e. using prototypes to attract potential customers or users, to keep them interested by continuously showing progress, and to obtain input on the current state of the prototype. For instance, one successful innovation project at BITS started when a major Swiss bank posted a public tender for a fund management system. Here, a prototype played an important role to sell an idea and persuade a funding decision, but also to act as a common object of work to craft a story together. Making use of the fact that the bank was already a long-standing customer, a team at BITS created a working prototype in the form of a new module in the existing system that was already implemented at the bank. Being able to demonstrate with a living example how the solution would look like, and to show an early proof-of-concept with a working prototype, BITS had a major advantage over its competitors and convinced the sponsors at the bank to win the bid. The project leader reflects on this experience: "*If you want to convince a bank, you need a prototype. Slides are not enough [...] You should build a prototype as quickly as possible, such that you can talk to the customer soon. People need to see something. [...] Of course, a prototype is a higher investment, but it also has much higher persuasive power.* (I10)". As the innovation project continued, the project team used the prototype regularly to support continuous interactions in product review workshops with the customer, and to discuss possible design options internally. The project leader compares this experience with building a Lego house together: "*You add one or two bricks, remove some others, refine a whole chunk, then start with a new plate. [...] The whole idea is only a sketch until you build a prototype and validate it with someone who has the business knowledge. But unless you create something tangible, you never get to the next level.".* (I10) And when the first pilot product was rolled out to an initial set of test users, the prototype was useful to show the banking expert how the final product could be used. "*When you talk to a customer a prototype usually works best [because] it is always a challenge to understand what the customer actually wants. [...] It is easier if you just try to implement it briefly with a pilot-prototype and then say 'look, if you click here, than this happens, and in the end, it looks like that'.* (I23).

Similarly, we found various cases at CustomSoft where prototypes were used to craft stories together with various stakeholders. In general, CustomSoft engineers report that using a prototype is in most cases the best way to present an idea to both internal and external stakeholders, because the quality of the feedback and interaction is much higher: "*I call this 'sounding'. You go there, make some music and see if the music goes down well or not. So, in principle, I could just go there, do a PowerPoint presentation, and say 'look, I have a super cool idea, this is how I imagine it'. And then there are bullet points, right? And then they say 'Well, thanks, um, we will see. Bye.' Or I go there and say 'now look at this'. Then I hand them a tablet and say 'now you can click and imagine [it], right?' Then they say: 'Wow! This is exactly what I imagined!'.* (I79) In sum, crafting the story together refers to how prototypes can be used to make a compelling case for the development of an innovative idea first and then, once a social coalition has been built, to co-create the story centering around the prototype together.

2) Take It or Leave It

A malpractice to engage with the audience is *Take It or Leave It,* i.e. developing something in a quiet chamber and then pushing it involuntarily to the user, often done by developers thinking they know best how the solution should look like. Our interview partners understand this practice is not ideal, but they report that it happens all too often. Their offered explanation is that, besides having to explore new terrain and constantly look for new markets to enter with innovative products, a company like BITS also must exploit the existing business and make sure that the current product portfolio runs stable to satisfy existing customers. More 'boring' tasks like release planning, responding to customer issues, fixing bugs, meeting new regulatory requirements, or maintenance usually determine the day-to-day business at BITS. Larger parts of the organization are explicitly devoted to keep the existing business running and stable instead of innovating. In such an environment, there is a risk that innovative ideas are assigned lower priority or even drown in daily business because they are naturally riskier. Such a state can frustrate many employees in the long run, especially the more technically versed who often have a keen interest in new technologies and a natural inclination to creativity and

curiosity. Perhaps because of these hurdles, we have encountered the rather dubious practice of *take it or leave it* quite frequently at BITS. We often observed that people decide to take their own initiative and conduct a 'submarine attack'. That is, they develop something at their own discretion in a quiet chamber, push it into the release and wait to see how the customer likes it. The result can be anywhere between exciting and horrifying for the customer: "*People simply build something, let the customer work with it and see what happens. What is generally considered rapid prototyping is that you build a prototype and get feedback before it goes live, but that's not how [they] build prototypes. Yes, there are some workshops and then they build what they understood there. But often, you do not have time for a second or third iteration, and then you go live with what you have."* (I20) As a standard software provider, BITS always has the challenge that some features may provide value for some customers but can be counterproductive for others. When a submarine attack goes into the productive system, this can lead to undesired complexity or even violate regulatory requirements: "*under the cover of prototyping and pragmatism, people build and check in complete solutions without too many further considerations, and then the customers have to live with it. [...] Sometimes it works well, you're more efficient and all, but sometimes it creates incredibly high collateral damage [...] because it was implemented quick and dirty."* (I20)

3) Premature Closure

Another malpractice to engage with the audience is *Premature Closure*, i.e. jumping to conclusions too early about what is to be built. As we learned from our study, it requires some courage and energy to challenge important stakeholders, especially sponsors, and thereby risk that an innovation project might be cancelled or not initiated at all. Even when they find an idea attractive, stakeholders often do not have the patience (or willingness) to support prototyping and experimentation for long, because this naturally implies high levels of risk and uncertainty: "*In such a project setting, the customers' willingness to prototype surely conflicts their willingness to have some specification, which you can sign and say 'this is what we get'. Ideally, you want to start with a rough prototype and refine it with the customer. [But] the customer, who pays for it and also wants to implement it in a year, insists on signing a binding document first."* (I1). A typical challenge in this situation is to find the right level of fidelity, such that the envisaged solution is concrete enough to be both feasible and desirable for all stakeholders while also not to narrow down the solution path too early and thereby waste innovative potential. One project team at BITS had to learn the hard way that committing oneself too early may save the project, but the final product will eventually not be used: "*We worked on this for a year, created prototypes and all. Now we come up with the final product and the customers say it is not quite what they expected. [...] It was way too abstract for them. They cannot grasp what it means if you do not show them exactly how it looks like*." (I12). Inherent to the immaterial nature of software products is the danger of confusing a prototype with a final system. This increases the risk of premature closure even further, because interactions with the audience may focus too much on unimportant issues while overlooking important ones: "*The customer cannot differentiate what is easy or difficult to change. Very often, we get tangled up in extremely tedious discussions about things that are completely irrelevant. Things like, it does not matter if we build that in green or blue. Then again, some things are not even brought to the table, because the customer thinks 'that's easy, that's just some workflow parameterization'. But there we have to say 'sorry, what you want is impossible, the whole server-construct is missing there'.*" (I12). In addition, storytellers frequently overestimate their understanding of the listeners' real needs. When they lull themselves into a false sense of security about the requirements too early, the probability is high that the final product will not be satisfying. One experienced software engineer summarizes it as follows: "*At the moment, we spend a bit of time with the customer and the rest is development and testing. We should reverse that ratio and really challenge the customer. Certain things that are not worth building should simply not be built. [...] We should look much longer at the whole integration process on the customer side and build prototypes without code, just wireframes. And we should take these prototypes to the customers to challenge them until they are fed up and say 'this is what I really want' instead of only saying 'yes, yes' and then build something.*" (I9).

D. Spreading the Message

Stories can be an effective way to build social coalitions. Thus, one important factor to consider when using prototypes for storytelling is also how to spread the message. There are a variety of ways how storytellers can spread the idea in a desired or undesired way.

1) Coupling Prototype and Narrative

A good practice to spread the message is *Coupling Prototype and Narrative*. A prototype can support and enrich the direct face-to-face communication between innovation teams and different stakeholders. By showing with a prototype how a finished system will look like in the future while telling a story, it is not only easier to explain complex issues, but it is also possible to create 'wow moments' to attract and persuade potential customers or users. One common challenge in this situation is that, after such an interaction, these people usually talk to other people about the state of the prototype and its underlying idea. One product owner at BITS calls this the 'indirect audience': "*The direct audience are those you present the idea to. But they take this to others and you must roughly know who these others are to know how to present your design."* (I6). While it can be quite desirable that an idea spreads, we observed that it is important to couple together prototype and narrative such that the idea maintains its integrity while it spreads. Otherwise negative effects can occur, such as misunderstandings or even political unrest when people

object or reject the idea before it is well thought out. Hence, the sender of an idea must have the indirect audience in mind and ensure that the idea is repeated consistently. An often-observed way to couple together prototype and narrative is to create artifacts that not only show the prototype but also how it could be used. Ideally, the artifact itself tells the story in a concise and simple manner, such that one does not have to tell the same story again and again in different contexts. Additionally, one should pick a conventional file format that runs on the common platforms without requiring much technical expertise from the viewers. A simple and frequently used artifact in this context is PowerPoint. Because the tool is easy to use and widespread, it often serves as a container to couple together prototypes and narratives: "*If I do not narrate the prototype, it does not mean anything to you. So [I usually] make a PowerPoint presentation with screens of the prototype and descriptions what it does, so I can send it by mail [and] do not have to narrate it in person." (I65).* However, PowerPoint presentations still have a high level of ambiguity [47]. A more sophisticated, though somewhat more laborious, way is to create video films. These can either involve real actors showing how to use the envisaged system, or simply tutorial-like screens of the prototype with audiovisual explanations: "*We tried to tell the story in individual conversations, but we found out that this dilutes the message over time. So we made films [that show] how we imagine the next stage. We put clickable prototypes into films such that the people, when they tell something internally, always tell the same story without having to make any 'soundtracks' for slides. We gave them readily assembled films with bubbles and all, which they could simply show on a [USB] stick and say 'this is how [it] looks today. This is how we think it looks tomorrow', such that the message stays stringently the same." (I75).*

2) Running from Pillar to Post

In turn, one less effective way to spread the message is *Running from Pillar to Post*, i.e. trying to please all stakeholders equally. Budgeting and initiation processes of an innovation project typically involve interactions with sponsors and upper managers, but the later development and integration requires appropriate support from both internal and external middle management and employees. However, it is not conducive for a project team to try satisfying all stakeholders equally. Instead, it is crucially important to be aware of how decision power is distributed among relevant stakeholders and give decision makers the necessary arguments at hand to convince others. For instance, the mobile banking project team told us an infamous anecdote of them trying to satisfy the conflicting demands of two different stakeholder groups, and then failing to meet both of them: "*The top management loved our design, but when we went to the middle management, they all said 'it has to look classic, we are doing boring e-banking here'. So we did that, went back again to the steering committee with the top managers, and they said 'how boring is that prototype?' [chuckles], and threw it out again*." (I17). This is especially the case when the listener is a sponsor who pays for the envisaged product, but not a user who works with the final product it in the end. "*We have this challenge that the product we build will not be used primarily by bankers, but by bank customers. And the banker in between is actually more of a problem than an aid. Because they have a different view than the customer on the banking business."* (I25). Or if the management overlooks that developers must support an idea, too: "*So I created paper prototypes and HTML UI prototypes, collected feedback that reinforced me to continue, [...] but at that point where it had to be implemented it was also a task for the developers. But the developers complained and did not want to do it, so suddenly there was no time and it was off the table."* (I11)

V. Discussion

In the previous section, we provided rich empirical insights into observed practices and malpractices of prototyping in our case study, focusing particularly on the role of storytelling. In sum, prototypes can be a compelling backbone of a story and help to make a persuasive case for a desirable future (i.e. holding *I Have a Dream* speeches*)*, but it is crucially important that the script is as close to an existing or envisioned real world situation of the listener as possible (i.e. avoiding *Fairy Tales)*. Storytellers should use prototypes to focus the story's level of detail on highlighting only the aspects relevant to the listeners while leaving out irrelevant ones (i.e. telling *Elevator Pitches*) instead of gold-plating and over-engineering technical aspects of the prototypes that are not conducive to illustrate the core features (i.e. *Using a Sledgehammer to Crack a Nut*). *Crafting the Story Together* by using prototypes to attract listeners and to obtain feedback continuously is an effective way to engage with the audience, as opposed to pushing the listener to use something that has been developed in a quiet chamber (i.e. forcing them to *Take It or Leave It*), or jumping to conclusions about what the listeners really want too early (i.e. *Premature Closure*). And *Coupling Prototype and Narrative* can be an effective means to spread the message convincingly and consistently among stakeholders, while sparing storytellers the efforts of *Running from Pillar to Post* and trying to please everybody.

An important practical implication of our study is that prototypers need to see themselves as storytellers. The here identified set of practices can be used by software professionals as a guideline to enact storytelling and prototyping purposefully and simultaneously in practice. Our data shows that prototyping and storytelling are two complementary sides of the same coin. Both are essentially techniques for requirements elicitation and validation [1]–[5], idea experimentation and exploration [6]–[10], facilitating communication [5], [8], [10], [23], and decision making [10]–[12]. Prototyping can support and enrich the communication about innovative ideas and, if done properly, be a low-risk and cost-efficient approach to develop innovative software systems. However, a prototype alone does not elicit and validate requirements, explore and experiment with ideas, facilitate communication, or make decisions by itself. Just as a picture can be worth a thousand

words if we know what it shows, a prototype can be worth volumes of documents if, and only if, we know the story it is supposed to tell [12]. The prototype itself does not indicate what it does as it provides no explanations or judgements [12]. This knowledge cannot be fully explicated in the prototype alone, but rather resides implicitly in the minds of its developers, viewers, and users. When any of these stakeholders leaves the team or forgets lessons learned after the prototype is no longer used, part of the acquired knowledge that could have been useful in other contexts will be lost [12]. So far, research has mostly treated the construction, communication, and preservation of implicit knowledge as black box, overlooking the practices through which people enact these in social interactions with prototypes. In turn, storytelling has only ever been seen as a means to create a common understanding about as-is and to-be between storyteller and listener, in essence reducing stories to a bridge between developer and user in the sense of use cases and usage scenarios [37]. Thus, existing literature can only provide few answers in terms of theoretical concepts, empirical insights, or let alone practical guidelines [8], [26].

We offer a distinct perspective in which storytellers are innovators who need to convince decision makers by combining the expressiveness of an illustrative prototype with the persuasiveness of an appealing story. Here, decision makers are understood as managers and sponsors in their formal role, but also as users, business experts, technical experts, and other peers who are consulted during the decision-making process with the goal to not only understand each other mutually, but also to persuade others. Thus, a good storyteller should ensure that the story has an interesting, appealing, and authentic script, highlights only the relevant aspect with respect to the intended audiences, and provides listeners with opportunities to shape the idea while also staying clear and consistent as it spreads.

Prototypes and stories both embody many different requirements details explicitly and implicitly, some of which are hard to divide [36]. A prototype is the pivotal point of a story by means of which people can estimate potential impacts of the envisioned system. In turn, a story provides rich context and conveys a lot of implicit knowledge. The story's level of quality and elaboration decides over whether the prototype is persuasive or not. In stark contrast to traditional use cases and usage descriptions, storytelling provides richer context and centers on the actual users instead of some generic customer [37]. In addition, stories play an important role in communicating ideas within and across organizations. They help listeners to get a better understanding of the needs and desires of actual users and thus better support decision making. Storytelling with a prototype is an ideal situation to start a discussion focused on the problem and possible solutions.

Based on these insights, we suggest that storytelling and prototyping can and should be integrated into software engineering methods together. For instance, the here identified practices fit the principles of agile development, such as increased customer involvement and satisfaction through early, continuous, and frequent delivery of working software (i.e. prototypes and the stories inextricably bound to them), close and periodical cooperation between business people and developers (effectively by means of stories), co-located communication, and simplicity [48].

VI. Conclusions and Outlook

The present study explores the role of software prototypes in communicating innovative ideas in organizations. Our data shows that storytelling should not be reduced to the sole function of creating a common understanding between developers and users, but should also be seen as complementary practice, at eye level with prototyping, that can be an important means to support decision-making, transfer implicit knowledge, and facilitate communication, requirements elicitation and validation, as well as idea exploration and experimentation.

Our case study results are obviously bound to two organizations to limit complexity and explore practices in depth. Future work could examine whether the held observed prototyping practices and malpractices are specific to software firms, or can also be found in various industries where innovating involves software development. It could also be interesting to compare which of the here observed practices also hold in a physical prototyping environment.

Acknowledgements

We thank the employees and management of BITS and CustomSoft for their openness and support. We also thank our colleague Peter Heinrich for helpful suggestions.

References

[1] A. Hickey and D. Dean, “Prototyping for requirements elicitation and validation,” *AMCIS 1998 Proc.*, p. 268, 1998.

[2] D. L. Parnas and P. C. Clements, “A rational design process: How and why to fake it,” *IEEE Trans. Softw. Eng.*, no. 2, pp. 251–257, 1986.

[3] S. P. Overmyer, “Revolutionary vs. evolutionary rapid prototyping: balancing software productivity and HCI design concerns,” *Cent. Excell. Command Control Commun. Intell. C3I George Mason Univ.*, vol. 4400, 1991.

[4] F. Kordon and others, “An introduction to rapid system prototyping,” *IEEE Trans. Softw. Eng.*, vol. 28, no. 9, pp. 817–821, 2002.

[5] R. Budde and H. Zullighoven, “Prototyping revisited,” in *COMPEURO’90*, 1990, pp. 418–427.

[6] T. Kelley, “Prototyping is the shorthand of innovation,” *Des. Manag. J. Former Ser.*, vol. 12, no. 3, pp. 35–42, 2001.

[7] T. Carleton and W. Cockayne, “The power of prototypes in foresight engineering,” in *ICED 09*.

[8] B. Doll, *Prototyping zur Unterstützung sozialer Interaktionsprozesse*. Springer DE, 2009.

[9] D. Bäumer, W. R. Bischofberger, H. Lichter, and H. Züllighoven, “User interface prototyping—concepts, tools, and experience,” in *ICSE 1996 Proc.*, pp. 532–541.
[10] H. Lichter, M. Schneider-Hufschmidt, and H. Züllighoven, “Prototyping in industrial software projects,” *ICSE 1993 Proc.*, pp. 221–229.
[11] J. E. Urban, “Software Prototyping and Requirements Engineering,” *Rome Lab.*, 1992.
[12] K. Schneider, “Prototypes as assets, not toys,” in *ICSE 1996 Proc.*, pp. 522–531.
[13] Y.-K. Lim, E. Stolterman, and J. Tenenberg, “The anatomy of prototypes,” *ACM Trans. Comput.-Hum. Interact. TOCHI*, vol. 15, no. 2, p. 7, 2008.
[14] S. Houde and C. Hill, “What do prototypes prototype,” *Handb. Hum.-Comput. Interact.*, vol. 2, pp. 367–381, 1997.
[15] N. Bryan-Kinns and F. Hamilton, “One for all and all for one?: case studies of using prototypes in commercial projects,” in *Proceedings of the second Nordic conference on Human-computer interaction*, 2002, pp. 91–100.
[16] R. F. Ciriello, F.-R. Aschoff, M. Dolata, and A. Richter, “Communicating Ideas Purposefully - Toward a Design Theory of Innovation Artifacts,” in *ECIS2014 Proc.*.
[17] K. M. Eisenhardt and B. N. Tabrizi, “Accelerating adaptive processes: Product innovation in the global computer industry,” *Adm. Sci. Q.*, pp. 84–110, 1995.
[18] E. Partridge, *Origins: A short etymological dictionary of modern English*. Routledge, 1991.
[19] K. C. Desouza, *Intrapreneurship: managing ideas within your organization*. Univ of Toronto Press, 2011.
[20] R. F. Ciriello and A. Richter, “Idea Hubs as Nexus of Collective Creativity in Digital Innovation,” in *ICIS2015 Proc., 2015*.
[21] T. R. Schatzki, K. Knorr-Cetina, and E. von Savigny, *The practice turn in contemporary theory*. Psychology Press, 2001.
[22] S. Holmlid and S. Evenson, “Prototyping and enacting services: Lessons learned from human-centered methods,” in *Proceedings from the 10th Quality in Services conference, QUIS*, 2007, vol. 10.
[23] C. Floyd, “A Systematic Look at Prototyping,” in *Approaches to Prototyping*, R. Budde, K. Kuhlenkamp, L. Mathiassen, and H. Züllighoven, Springer 1984, pp. 1–18.
[24] J. Rudd, K. Stern, and S. Isensee, “Low vs. High-fidelity Prototyping Debate,” *interactions*, vol. 3, no. 1, pp. 76–85, Jan. 1996.
[25] A. M. Davis, “Operational prototyping: A new development approach,” *IEEE Softw.*, 9, 5, pp. 70–78, 1992.
[26] T. Schlachtbauer, M. Schermann, and H. Krcmar, “Do Prototypes Hamper Innovative Behavior in Developing IT-based Services?,” in *ICIS2013 Proceedings*, 2013.
[27] S. Blank, “Why the lean start-up changes everything,” *Harv. Bus. Rev.*, vol. 91, no. 5, pp. 63–72, 2013.
[28] O. Gutierrez, “Prototyping techniques for different problem contexts,” in *ACM SIGCHI Bulletin*, 1989, vol. 20, pp. 259–264.
[29] C. A. Voss, “The role of users in the development of applications software,” *J. Prod. Innov. Manag.*,2, 2, pp. 113–121, 1985.
[30] G. Schreyögg, *Knowledge management and narratives: Organizational effectiveness through storytelling*. Erich Schmidt Verlag GmbH & Co KG, 2005.
[31] T. Oaks, *Storytelling: a natural mnemonic*. 1995.
[32] N. Gershon and W. Page, “What storytelling can do for information visualization,” *Commun. ACM*, vol. 44, no. 8, pp. 31–37, 2001.
[33] A. Uittenbogaard, “Storytelling for Software Professionals,” *IEEE Softw.*, vol. 30, no. 3, pp. 9–12, 2013.
[34] S. Hayne, *Using Storytelling to Enhance Information Systems Knowledge Transfer*. Las Vegas, 2009.
[35] E. Meyer, N. A. D. Connell, and J. H. Klein, “A narrative approach to knowledge exchange: an empirical investigation in two contrasting settings,” 2005.
[36] W. Swap, D. Leonard, and L. A. Mimi Shields, “Using mentoring and storytelling to transfer knowledge in the workplace,” *J. Manag. Inf. Syst.*, 18-1, pp. 95–114, 2001.
[37] E. Wende, G. King, and G. Schwabe, “Exploring Storytelling as a Knowledge Transfer Technique in Offshore Outsourcing,”, *ICIS 2014 Proceedings*.
[38] H. Clausen, “Designing Computer Systems from a Human Perspective: The Use of Narratives,” *Scand. J. Inf. Syst.*, vol. 6, no. 2, Jan. 1994.
[39] M. B. Rosson and J. J. M. Carroll, *Usability engineering: scenario-based development of human-computer interaction*. Morgan Kaufmann, 2002.
[40] R. A. Stebbins, *Exploratory Research in the Social Sciences*. SAGE, 2001.
[41] G. Walsham, “Doing interpretive research,” *Eur. J. Inf. Syst.*, vol. 15, no. 3, pp. 320–330, 2006.
[42] H. K. Klein and M. D. Myers, “A set of principles for conducting and evaluating interpretive field studies in information systems,” *MIS Q.*, pp. 67–93, 1999.
[43] D. Silverman, *Interpreting qualitative data: Methods for analyzing talk, text and interaction*. Sage, 2006.
[44] J. T. DeCuir-Gunby, P. L. Marshall, and A. W. McCulloch, “Developing and using a codebook for the analysis of interview data,” *Field Methods*, 23-2, pp. 136–155, 2011.
[45] J. vom Brocke, A. Simons, B. Niehaves, K. Reimer, R. Plattfaut, and A. Cleven, “Reconstructing the giant: On the importance of rigour in documenting the literature search process.,” *ECIS 2009 Proc.*, 2009.
[46] D. Yanow and P. Schwartz-Shea, *Interpretation and method: Empirical research methods and the interpretive turn*. ME Sharpe, 2013.
[47] R. F. Ciriello, A. Richter, and Schwabe, Gerhard, “PowerPoint Use and Misuse in Digital Innovation,” in *ECIS2015 Proceedings*.
[48] K. Beck *et al.*, “Manifesto for agile software development,” 2001.