Article:
Ghobadi, S and Mathiassen, L (2020) A Generational Perspective on the Software Workforce: Precocious Users of Social Networking in Software Development. Journal of Management Information Systems, 37 (1). pp. 96-128. ISSN 0742-1222

https://doi.org/10.1080/07421222.2019.1705508

A Generational Perspective on the Software Workforce: Precocious Users of Social Networking in Software Development

Shahla Ghobadi

Leeds University Business School, The University of Leeds

Lars Mathiassen

J. Mack Robinson College of Business Georgia State University

Abstract

Software is the lifeblood of technological advancement, and it progresses not only through emerging technologies, but also through the contributions of new generations of developers who have distinct technology-related experiences. We describe our qualitative investigation into how developers, who began regularly using social networking technology at an early age (referred to as *precocious users*), demonstrate distinct expectations about the goals of software development. We advance a theoretical perspective that explains how the increasingly socially infused nature of networking applications shapes generations of individuals—some of whom will go on to become creative developers in the software industry. Our perspective suggests software organizations can leverage developers who have been precocious users of more recent social networking technologies to reinforce intuitive usage, promote social impact, and re-energize experimentation and contribution to the software community. Our results also offer a comprehensive set of development goals that focus attention towards contemporary expectations about challenging usability and contribution to software ecosystems. We conclude by discussing how our methodological steps, data collection, and data analysis procedures empower future research to explore generational shifts in the career perceptions and competencies of the digital workforce.

Keywords. Software development, social networking, generation, digital workforce, digital native, human resource, career perceptions, comparative causal mapping, qualitative study

Ghobadi, S., Mathiassen, L. A Generational Perspective on the Software Workforce: Precocious Users of Social Networking in Software, Journal of Management Information Systems (forthcoming, accepted: 10.14. 2019).

Introduction

In 2010, two Irish brothers—Patrick and John Collison, born in 1988 and 1990, respectively—founded a software company. Their goal was to develop an online application that allows businesses to bypass the bank bureaucracy and accept payments instantly from customers across the globe. Following on the analogies of Facebook *posts*, Instagram *photos*, and Snapchat *videos*, the software they developed, called *Stripe*, makes it easy for anyone to share money anywhere and on any device. The Collison brothers' software ideas exploded into a multi-billion-dollar behemoth, placing them solidly among the generations of tech-savvy entrepreneurs who began regularly using social networking technology at an early age.

Social networking technology refers to Internet-based applications that build on the ideological and technological foundations of Web 2.0, enabling people to create and exchange user-generated content across a wide range of stationary and mobile devices [38]. Today, social networking applications such as Twitter, Facebook, Instagram, and Snapchat are ubiquitous and have become essential communication mediums that penetrate various spheres of our lives. Beyond being merely social playgrounds for users to express their thoughts, views, and feelings, social networking platforms offer spaces and mechanisms to orchestrate communication activities in increasingly technology-embedded social contexts [4]. Because precocious users[1] have ordered their social activities computationally from an early age, they may hold distinct expectations about the features and functionality of software, which increasingly borrows features from social networking technologies [17]. Moreover, precocious users who grow up to become software developers will likely build software through processes that revolve around creating, sharing, and blending information across diverse spheres of specialization [14, 24, 57]. Accordingly, they will likely

1 The *Oxford English Dictionary* defines "precocious" as an adjective, describing individuals who developed certain abilities or inclinations at an earlier age than is usual or expected. Researchers in learning and educational psychology often use the term to refer to young individuals who, at a significantly early age, demonstrate capabilities such as creativity, mathematical skills, and grammar proficiency [58]. Here, we use the term "precocious users of social networking technology" to refer to those individuals who began regularly using social networking applications at a notably early age.

demonstrate distinct expectations not only about new software products, but also about how we should conduct knowledge-intensive development processes.

Our understanding of what these distinct expectations might be is, however, limited; while the software industry increasingly recognizes that new generations of developers can accelerate the pace of innovation in software development, we don't yet know how that development will change as a result of their particular experiences and expectations. We can trace the theoretical flux that has discouraged advances in this area to various literature streams that are fragmented by debates about whether early life experiences with technology actually create generational differences, as well as about how we should study and operationalize emerging generations [10]. For example, while some argue that new generations of employees with early age experiences with modern technologies constitute a *digital workforce* that has distinct expectations and approaches [16, 70], others challenge the underlying idea that technology-related experiences give rise to generational shifts [10, 47].

Nonetheless, it can be useful to understand how software developers who were precocious users of social networking technology perceive product- and process-related goals in software development. Such an understanding can expand our knowledge of how new developers may contribute to innovation in the software industry [13, 31, 63]. This updated knowledge can, in turn, provide a fertile ground for continued theorization on emerging generations of developers. Since the Web's emergence in the 1990s, an influx of innovations has evolved social networking from a simplex, one-directional medium of communication to a full-duplex format that supports virtual communities, interest groups, and mobile devices. Developments in this area will continue to unfold as new generations of developers emerge, bringing with them new experiences as social networking users. A 2015 college graduate, for example, maybe have grown up with early social networking applications such as Friendster and MySpace, while a 2025 graduate will have likely had early experiences with Facebook, Instagram, and other more advanced applications. Efforts to understand such generational differences can contribute to our understanding of how new generations of employees approach work and how organizations can renew and revitalize their work practices by

leveraging this emerging workforce [16].

Here, we contribute to current understanding of these dynamics by asking a key question: *How might developers who were precocious users of social networking technology change the goals of software development?* We address this question through a systematic qualitative investigation that explores how the expectations of these developers (here, referred to as *Group 1*) differ from those of developers with less precocious social networking experiences (*Group 2*). In this investigation, we use comparative causal mapping (CCM), a variant of cognitive mapping [25, 42], to analyze the generational changes in developer expectations across two software companies. Our analysis (1) suggests both similarities and differences in how different developer groups express expectations about software products and processes, and (2) reveals software development goals and the relationships between various goals. By combining the empirical findings with extant literature, we advance theoretical propositions on the nature of generational shifts in product and process expectations, as well as on how software organizations can leverage these shifts to continuously expand and renew their software and their development approach. These propositions offer a generational edge to the co-evolutionary perspective on information systems development [55]. They also contribute to existing discussions on software development goals and expectations [67] and on how we should study and compare generations [10, 47].

Theoretical Background

Generations, Experiences, and Technology

Generation is an ancient yet vital concept that has been the subject of widespread media and academic study for decades [35]. It expresses both the passage of time and the boundary between "kinds of people" living in "kinds of time". A *generation* is also defined as a group of people who were born during a particular time period and experienced significant life events at critical developmental stages [72]. Research on generational issues traces back to the work of Karl Mannheim [46], who emphasized the importance of studying generations as a guide to understanding the structure of social and intellectual movements. In organizational

contexts, Joshi et al. (2011) suggest two notions of generations. The first is the *kith (*or *cohort*) notion, which explains how generational imprints emerge in a chronological sequence. The second is the *kin* (or *genealogy*) notion, which explains how intergenerational interactions transfer generational imprints on a continuum ranging from resistive to transmitive [35]. *Resistive interactions* correspond to the conflict-based view of generations, which is particularly emphasized in political sociology. In this view, generations assign and secure access to income and occupational prestige; the conflict occurs in the form of age-based stereotypes and pejorative perceptions of other generations' roles. In resistive interactions, preceding and succeeding generations engage in categorization-based responses to members of other generations. The result is an exacerbation of social divisions, which hinders communication and intensifies competition and conflict by accumulating organizational resources to each other's detriment. In *transmitive interactions,* which is a pronounced view in family sociology research, successive generations are bound in the reproduction of social life. In these transmitive interactions, generations engage in synergistic knowledge sharing and creative problem solving, demonstrating behaviors such as reciprocity, nurturance bonding, altruism, and beneficence.

A growing stream of generations research focuses on early experiences of "using new forms of technology" as a potentially important form of life experience [52]. Despite various efforts over the years, however, empirical research in this area is beset by debates over whether generational differences exist at all and, if they do, how emerging generations should be studied and operationalized (Appendix 1 summarizes this research). While some studies report some form of generational differences among individuals (e.g., in relation to comfort in using unfamiliar technology or in perceptions about technology addiction) [1, 52, 53], others report mixed results [3, 27, 28, 34, 47, 68] or methodological limitations in studying generations [40]. The latter studies suggest a sharp shift away from focusing on age-based differences to instead focus on embracing concepts such as digital literacy and computer engagement. These studies challenge prior research that generalizes generations into broad age groups—such as millennials or digital natives [5, 11, 35, 36]—and resonate with

generational scholars who emphasize that pure age-based examinations can lead to a proliferation of stereotypes that may not apply when other attributes are taken into account [34-36]. Because diversity based on demographic features does not necessarily correlate with diversity based on experience-based cognitive features [37, 49], it is important to go beyond cohort-based notions to explore how "experience-oriented" generations emerge in organizations and evolve to influence a wide range of outcomes [35, 36]. Despite research advancements in this area, however, important gaps in the literature remain.

Specifically, the existing literature suggests that early immersion into new forms of technology may shape individual beliefs and expectations about technology use, especially in education contexts. However, current discussions fall short in theorizing the manifestations and implications of generational issues as individuals enter work settings and contemporary organizations [16]. Moreover, the literature overlooks the original notion that *early experience* in using technologies such as the Internet can have profound implications and give rise to new forms of technology-driven generations of individuals [47, 70]. This insight relies on the notion that "repeated experience"—especially if deep-rooted in our early years—can create permanent "neural pathways" that either strengthen or weaken certain mental models in our brains. This is particularly true for technologies that are ubiquitous in various spheres of our lives; chief among the examples here is social networking, which has penetrated and transformed our everyday lives and interactions through continuously evolving applications. Such arguments resonate closely with cognitive development research, which has found that when learning certain things, a mere few years more of exposure makes a difference. For example, linguistic researchers have long asserted that the cortical centers, which are important to accent acquisition, lose plasticity around the onset of adolescence (at approximately age twelve) and that learning a new language after that age typically prohibits a person from acquiring the accent appropriate to it [23, 43]. Clearly, studying people's pronunciation of a new language in relation to age and their exposure to it is far more straightforward than studying intangible cognitive expectations such as development goals and expectations in relation to exposure to modern technologies such as social networking.

Nonetheless, as a systematic first step to addressing the literature gaps, our study explores the expectations of software industry developers who have precocious social networking experiences.

Expectations in Software Development

Software development requires innovative thinking to inspire and implement new product ideas and processes. Prior research suggests that *human agency* is key to how software is developed, and that developers' expectations shape how new software is designed and produced [18]. As such, the literature highlights two sets of software development expectations [60, 67]: (1) product-related expectations, and (2) process-related expectations.

Product-related expectations focus on how software should work, look, and provide value to a broad range of groups and individuals; these expectations include user satisfaction, software maintainability, software popularity, user impact, and social impact. Expectations for *user satisfaction* focus on the ability of the software's features and functions to both satisfy users and their needs [7, 15, 33] and ensure that users perceive that those needs and requirements are satisfied [19, 32, 64]. User satisfaction may also include users' reactions to the software's technological and informational capabilities, and how well the software addresses their psychological and behavioral needs [7, 64, 66]. *Software maintainability* expectations center on the software's ability to be flexible, adaptable, and sustainable in the face of changing business needs [8, 54]. *Software popularity* expectations typically exist in commercial product development contexts, which aim to ensure that the software can attract and retain loyal end users over the long term [56]. Research also identifies expectations in terms of *user impact*—that is, the software should enhance user welfare, productivity, and decision-making ability and quality [59]. Finally, research has begun to emphasize the importance of software products' *social impact*, or higher-level net benefits [15], including positive outcomes for society at large.

Process-related expectations refer to how software development should unfold in order to create value for a broad range of individuals and groups; these expectations focus on development productivity, team morale, and development dynamism. *Development*

productivity expectations focus on the need to balance the desires for high-quality software with a productive development process [66]. Productivity can improve by assessing software functionality and focusing on development scheduling and costs [66]. Further, research shows that when productivity is improved, developers write more lines of code with fewer customer-reported defects [45] and thus meet expectations for high product quality. *Team morale* research suggests that when software development focuses on improving team interactions, it can boost morale and provide value to the team by (1) providing a strong foundation for future development projects [67] and (2) creating an intrinsically motivating work environment that enhances the developers' capabilities to create new products . The latter argument is especially relevant because software developers tend to be very critical of the projects they work on [61]; they typically consider a project successful if they produce quality software and have an intrinsic sense of personal achievement. Similarly, *development dynamism*—that is, software development that embraces flexibility and experimentation with existing technology, such as third-party libraries and plugins [50, 69]—can enhance the software process and developer satisfaction. The expectations related to such dynamism, however, must include an appreciation of experimentation's risks and learning curve, which may require writing larger amounts of code to deliver ideal functionality [69].

Summary of Insights and Research Focus

The reviewed literature provides insights into technology-driven generations of individuals and emphasizes the need to study the manifestations and implications of those generations as part of the emerging digital workforce. We build on this literature and draw upon the ubiquitous nature of social networking technology to highlight its potential role in creating new generations of software developers who have distinct software-related expectations. In the following section, we adopt an exploratory approach to understand developers' expectations based on the established conceptualizations of both software product expectations (including user satisfaction, user impact, software maintainability, and software popularity) and software process expectations (development productivity, team morale, and development dynamism) [60, 67].

Research Method

To address the research question, we undertake an interpretation-centric qualitative investigation—a common approach for examining and revealing phenomena that require in-depth analysis [65, 71]. Our approach assesses how the expectations of software developers who were precocious users of social networking technology (Group 1) differ from those with less precocious social networking experiences (Group 2).

Sample and Sites

We sampled developers (our unit of analysis) from software settings, which helped us build a rich understanding of development expectations and generational differences. The software settings we sampled from were two software organizations in the Asia Pacific; we refer to them here by the pseudonyms *iSolution* and *iSirva*. Several features of the two organizations make them suitable for sampling individual developers. First, they gave us access to a mix of developers working in different software contexts. iSolution has approximately 60 employees and develops software for capturing, storing, sharing, and analyzing scientific data. iSirva, which has approximately 100 employees, builds products for research and development in financial companies. Second, the two companies gave us access to developers who had varying experiences with social networking technology. So, while both companies had seasoned developers, they also strategically chose to recruit new software graduates as a major component of their development teams. Both CEOs have long-term professional engagements with universities, and they regularly present their software to the universities' schools of computer science and software engineering for feedback; they also recruit graduates from these schools to join their development teams. Third, the lead researcher on our project has a professional relationship with development managers at both iSolution and iSirva. This allowed us to recruit additional people to clarify the interviewees' statements and to explore the knowledge management system that records aspects of the development processes. These additional sources of information were instrumental in helping us develop a more in-depth, informed understanding of generational characteristics and implications.

Data Collection

Following an emergent strategy, our first step in data collection consisted of fieldwork to identify developers who had grown up using social networking. We asked the development managers to provide an initial list of both new and experienced developers working on their latest software product releases. We then contacted the listed people (by phone or face-to-face) to ask about their development background and prior exposure to the Internet and social networking, as well as their openness to participating in the study. Prior research refers to *demand characteristics* [56]—that is, when subjects interpret a study's intention and unconsciously change their behavior accordingly—as a major methodological challenge in studies that ask individuals direct questions about variations in human beliefs and expectations in organizational contexts. In this study, we therefore designed and asked our interview questions using neutral language, without any explicated intention to understand generational characteristics. This methodological choice enhanced the solidity and robustness of our empirical findings and helped us to avoid the limitations associated with demand characteristics.

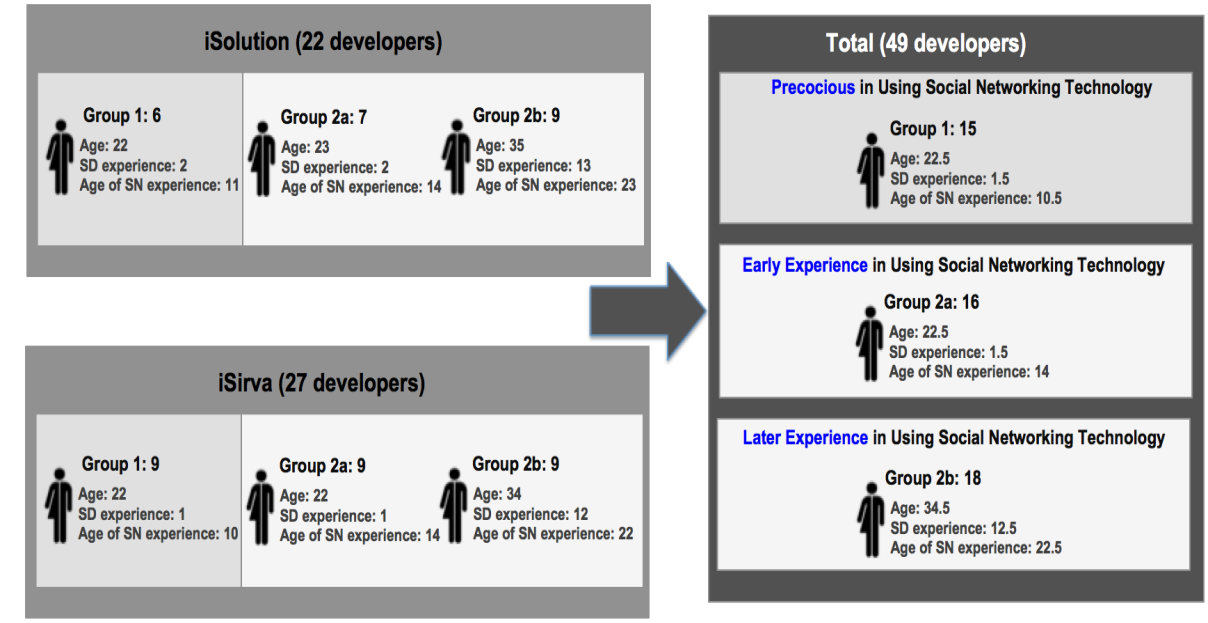
When conducting the interviews, the lead author asked participants the questions listed in Step 1 of the interview guide (Appendix 2). Their responses suggested that 27% of the developers (15 out of 55) had been precocious in adopting social networking applications such as Orkut, MySpace, and Friendster, and that they typically did so before age 11. Given their age at the time of technology adoption, the observation of precociousness resonates with research studies on cognitive learning and personality development [48], which suggest the milestone of age 12 as the onset of adolescence. We categorized these precocious developers as *Group 1*. The remaining 73% (40 out of 55) of developers had adopted social networking technology later in life (Group 2). Mindful that the two groups likely overlapped on other influential factors—in particular, age and development experience—and that such overlap

may inhibit the comparison's robustness, we looked more carefully into the age and development experiences of both groups of developers.

Our results indicated that Group 1 developers were very similar in terms of age (on average, they were 22.5 years old) and corporate development experiences (on average 1.5 years). Group 2 developers, however, had distinct differences. Specifically, 55% (22 out of 40) were roughly the same age and had the same development experience as the Group 1 developers. The Group 2 developers also had some early experiences with social networking, but they were not as precocious as the previous group in adopting social networking applications (they typically did so after the age of 14). Further, 45% (18 out of 40) of the Group 2 developers were considerably older (34.5 years on average), had more professional development experience (12.5 years on average), and had adopted social networking technology later in life (typically after the age of 20). This observation meant that a simple comparison of expectations across Group 1 and Group 2 might lead to unreliable results because some potential similarities and differences might be due to age or development experiences. To be more concise in the comparisons, we therefore broke Group 2 into two groups: we put younger, less-experienced developers in Group 2a, and older, more experienced ones in Group 2b. So, for example, an in-depth analysis of expectations may indicate that Group 1 developers have specific differences with Group 2b developers, and those differences might be due to age and development experience. If, however, the same differences hold true in comparison with Group 2a developers, it is more likely that they point to a generational characteristic.

We also removed six developers from Group 2a who were on the borderline of being precocious users of social networking applications (that is, they adopted applications when they were 12 or 13 years old). This helped us ensure that Group 1 and Group 2a developers

did not have a very close age overlap. Figure 1 illustrates the remaining 49 developers categorized into the three groups: Group 1, Group 2a, and Group 2b. Appendix 3 offers relevant details about each interviewee. In Step 2 of data collection, the lead researcher prepared for the interviews by spending a few days in each organization to get a sense of the environment, the common development practices and norms, [71] and each interviewee's work background. The researcher then conducted a total of 56 interviews, each lasting an average of 60 minutes. These interviews (Step 2, Appendix 2) sought to identify how the different groups describe their expectations for software development. Immediately after each interview, the researcher wrote a reflective memo, and crosschecked facts and impressions within 24 hours. Later in the study, we asked interviewees to review the findings to resolve ambiguities and validate the credibility and trustworthiness of the interpretations [65].



Age and experience refer to average numbers in years.

Figure 1. Sample Data

Data Analysis

We followed an exploratory and inductive approach to analyze developers' expectations, moving from empirical data to theoretical insights [20, 71]. Specifically, we used an *interpretation-centric* approach [65, 71] and various data sources—including interviews,

archives, industry trends, and existing research—which let us triangulate, challenge, and enhance our interpretations. The empirical inquiry's comparative nature also required that we use a systematic approach to analyze the data and pinpoint similarities and differences in perceptions across the developer groups. Hence, we chose comparative casual mapping (CCM) [25, 42], a variant of cognitive mapping, because it offers detailed guidance for constructing causal maps and for using measures that help systematize comparative investigations. For example, CCM guided us to use several measures to compare how developer groups reveal different patterns of development expectations, as well as how each group emphasizes different expectations and relations among those expectations. In keeping with the CCM methodology, we analyzed and made sense of the qualitative data in five steps: (1) create and use standard vocabularies, (2) process data for the causal maps, (3) construct the causal maps, (4) analyze the causal maps, and (5) develop the theory. We now describe these steps in more detail. Table 1 and Appendix 4 (Tables A-B) summarize the steps and provide supporting information.

Table 1. CCM Methodological Steps

CCM Step	Activities	Deliverables	Related documentation
1. Create and use standard vocabularies	1. Conduct an initial coding and categorization of the empirical data. This process is highly exploratory and necessitates returning to the literature to make sense of the findings. We managed the coding process using Nvivo 11.0 to store interview transcripts, field notes, and documents. This step's deliverables form the basis of the theory development (Step 5).	• A list of preliminary codes • Codes categorized into aggregate dimensions, concepts, and constructs	• Table 2
2. Process data for causal maps	1. Re-code the empirical data with a focus on extracting causal statements that describe relationships between different concepts relevant to the research question. 2. Pay attention to the directionality of each linkage and record them based on the expressed language. 3. Use different coders to code the data to help minimize coding biases and strengthen our interpretations.	• A table that demonstrates causal links between concepts, supported by sample quotes	• Table A (Appendix 4)
3. Construct causal maps	1. Break the identified causal links from Step 2 for each group and record each set of causal links in one file. 2. For each file, visualize the identified dimensions, concepts, and links. This process involves circling the revealed concepts and using arrows to show linkages between them. 3. Enhance readability of the maps by distinguishing between direct and indirect linkages related to the research question. 4. Calculate numerical insights (CCM measures) that facilitate a systematic comparison of the maps, then mark the results of the calculations on each map.	• Causal maps • A table that presents the results of the CCM measures for each concept and linkage	• Figures 2–4 • Table B (Appendix 4)
4. Analyze causal maps	1. Conduct iterative comparisons among the maps including their concepts, linkages, and CCM measure calculations. 2. Explore the empirical data to generate new, comparative insights by paying attention to and comparing the	• Tables that summarize and present the results of comparisons across causal maps	• Tables 3–5

	constructs that underlie the revealed concepts.		
5. Develop the Theory	1. Consider additional sets of data collection to triangulate and complement comparative insights. 2. Go back and forth between the findings, extant literature, and contemporary trends to challenge the identified insights. This process involves looking for both confirming and disconfirming evidence to challenge interpretations and reflect upon them to enhance the credibility of the theoretical account. 3. Develop and model theoretical propositions.	• Theoretical insights	• Propositions 1–4 • Figure 5

Step 1. Create and use standard vocabularies. In this step, the lead researcher read all interview transcripts and conducted initial coding of the data. The following interview quote, for example, was coded as expressing the importance of creating software that is flexible and scalable to changes in requirements:

> *"Scalability should be considered from the early stages of design and development ... because it is essential to develop software that can operate efficiently over a wide range of configurations, like handling large computing jobs or many users."*

The researcher grouped together words that were frequently mentioned in the interview transcripts to generate a list of preliminary codes. Some observations required a return to the software development literature to make sense of the findings, and this in turn prompted a more intensive reading of research on product development. For example, the interviewees used various expressions when referring to software maintainability, including *high-quality code*, *sustainable products*, and *scalable to new changes*. The researcher labeled each code—in this case, *software maintainability*—to summarize the meaning of words or phrases used by interviewees. The investigation was highly exploratory, but it was also a mindful process that looked for unifying concepts recommended by prior research. The process in Step 1 led us to identify 2 aggregate dimensions, 9 expectation concepts, and 23 lower-order constructs; continued reading of the data did not yield substantially new ideas. Table 2 summarizes these initial findings and offers examples of each expectation (which we describe in the findings section) to support the analysis. The aggregate dimensions and concepts we identified in this first step eventually formed the basis for our findings and theory development.

Table 2. Theoretical Dimensions, Concepts, Constructs

Dimension / Concepts		Constructs	Sample Quotes
Product Expectations	**User satisfaction** refers to software that makes users feel that the software and the ecosystem around it meet their needs, requirements, and expectations.	1. User satisfaction with software features and functionalities 2. User satisfaction with their perception of the software's usefulness 3. User satisfaction with the ecosystem around the software	"You can get all development aspects right, but if users aren't feeling the software is useful to them, the process is a waste. When users are convinced, they can make the whole business successful. This means fewer complaints, increased conversions, less frustration, and fewer sleepless nights for developers." —Aaron (Group 2a) "With any software, there should come a healthy and growing platform that connects users to internal developers, community of experts, and external developers. People are fond of user communities where they can share their experience of using the software, make direct contact with developers, and suggest changes in software." —Bobby (Group 2b)
	User impact refers to improvements that the software can make to users' lives, such as improving users' health, well-being, and productivity; transforming established ways that users interact with existing applications; and creating new career paths for users.	1. User productivity 2. User well-being 3. User career path 4. Challenging established norms	"The final software should help alleviate the pain points of users in some way. If we can make something that people can use to help them with one of their tasks, we have created something great, maybe by coming up with new product ideas, adding new features to a product, creating new tools to make work more efficient or easier to test." —Kate (Group 1). "Software should challenge conventional expectation and make using the software as natural and straightforward as possible. As software developers, we can constantly look for ways that shorten the learning curve for users." —Anne (Group 1). "The communities surrounding the software can provide opportunities for users with some programming or consulting expertise to make changes in their career path. We've seen many examples of people who have been inspired to create companies that build apps that extend the original software." —Brian (Group 2b).
	Software popularity refers to developing products that are used by as many users as possible and to establishing long-term popularity among a selected group of users.	1. Wide-spread use 2. Long-term use	"More users mean more revenue for developers—but perhaps more importantly, popularity of the software sends a powerful signal to the market that we have developed a great software." —Zack (Group 1). "Size of the user base isn't the whole point here. I'd rather have 10 happy users than 1,000 users, 990 of which are unhappy. More users demand more time to make sure the product can handle a large number of users. And if we focus only on acquiring more users, we are more likely to be missing out the chance of continuing to retain the existing users." —Aiden (Group 2b).
	Software maintainability refers to software with well-structured, elegant code, with features and functionalities that are flexible and adaptable to change.	1. Code quality 2. Scalable software 3. Business maintainability	"Scalability should be taken into account from the early stages of design and development ... because it is essential to develop software that can operate efficiently over a wide range of configurations, like handling large computing jobs or many users." —Zack (Group 1). "Building strong community around the software is vital for long-term survival. When days come that we can no longer maintain the software, a supportive community can organize itself and make new contributions possible." —Jackie (Group 2b).
	Social impact refers to software that relates to social causes to help address societal needs, enhance social capital, and contribute to social ecosystems that are dynamic and growing.	1. Societal needs 2. Dynamic ecosystems	"It is good to write software that contributes to the betterment of society ... The world is increasingly running on software products, and in many cases, those products have social and ethical implications. Software developers can engage in initiatives like hackathons to align the software we develop to social issues that need to be addressed." —Hedi (Group 1). "I deeply think software is one of the most useful inventions of humankind ... And I think it can influence the world in a positive manner. That's unfortunately not the world we live in. I think we can do so much more if we hold ourselves to higher standards." —Mina (Group

			2b). "Software industry should be mindful of the impact that it can make by designing and nurturing powerful ecosystems. This is because those ecosystems can grow into a vibrant ecosystem that connects millions of developers and users, creates jobs, and enhances social capital." —Tom (Group2b).
Process Expectations	**Development productivity** refers to development processes that continuously stretch to reduce release time, costs, and waste in software development.	1. Time reduction 2. Cost reduction **3.** Wasted-effort reduction	"Looking for cost reduction possibilities is necessary because cost and risk go hand-in-hand. The less costly it is to release the software the more viable it becomes to release more often, and consequently, the less risky development would become as developers are more likely to run into bugs and deployment problems." —Jason (Group 1). "Waste reduction should be a core part of development process to make it possible to spend more time doing what provides value to users ... Because when developers focus on building unnecessary features, they can miss important development opportunities. The development team should always look for 20% of the code that can make the overall performance of the program." —Andrew (Group 2b).
	Team satisfaction refers to development conditions that feature enjoyable and rewarding interactions between team members as well as reflect a sense of development achievement across team members.	1. Team morale 2. Team learning	"Development works should be designed to regularly take a look at team interactions to take the pulse of the team's morale. That way, development culture can be tracked and improved alongside the rest of the work … Developing software is not a short-term activity. It happens over multiple releases and enhancements over time. Team members must take pride in building a product together over this long-term." —Vincent Group 2b).
	Development contributions refer to methodological advancements that developers offer to the software community. They can include code contributions, technical advice, or even new models that can be used by other development teams.	1. Code contributions 2. Contributing new models for organizing development works	"It would be optimal to contribute to the knowledge in the software community, either by contributing some of our code to open source projects, distributing any useful toolkit we invented during the process, or even sharing some technical advice with similar projects." —Randy (Group 1). "Look at the example of the [iSXy development team]; while working hard to scale their development works without getting bogged down by dependencies and coordination, they created a new agile culture that is now being used by many other development teams. This is not quite common in development, but it can lead to contributions that will become a de facto standard." —Nik (Group 1).
	Development dynamism refers to the continuous practice of experimentation with existing technology and innovation in developing new tools and techniques.	1. Continuous experimentation in development 2. Continuous innovation in the use of development technologies	"Last year, Randy and I started a weekly lunch at work where developers can hang out and share the coolest tools they have recently seen or used ... By making these lunch breaks fun, we've got team members interested in taking the time to experiment with new [development] technology. Even if we decide against it, we've learned a lot and have now knowledge in that area." —Bob (Group 1). "Software is a competitive and fast-paced market ... A software developer's nemesis is getting behind industry trends. I strongly believe that each project should challenge us to learn new things and stay on top of the profession." —Luc (Group 2a).

Step 2. Process data for the causal maps. With a better understanding of developers' expectations, the lead researcher, along with a research assistant with software development expertise, coded all the interview transcripts separately to verify the reliability

of the coding process. They resolved cases of disagreement through discussion to minimize researcher biases. The coding process consisted of two steps: (1) extracting statements that described interviewees' expectations for achieving effective software development (*concepts*); and (2) noting statements that articulate a relationship between different expectation concepts (*linkages*; we recorded the directionality of each linkage based on the interviewee's specific language). Table A (Appendix 4) explains the coding process and offers examples of how we coded the statements to extract expectation concepts and their relations.

Step 3. Construct the causal maps. We aggregated the coded data for each of the three groups of developers into one file. We then created a causal map for each group, circling the expectation concepts revealed in the interviews and using arrows to represent the relations among those concepts (see Figures 2–4). To enhance the maps' readability, we used solid arrows to distinguish between direct relations connecting expectations and effective software development, and dotted arrows to indicate indirect relations connecting expectation concepts. We also delved into the coded data to create numerical insights to help us systematically compare the maps and reveal generational insights about precocious users of social networking technology. This data dive involved measuring density, centrality, and reachability—measures that are specific to CCM methodology (Table B, Appendix 4). Each map shows the results of our calculations.

Step 4. Analyze the causal maps. We compared the three groups using the causal maps, which included the revealed expectation concepts, the relations among them, and each map's density, centrality, and reachability numbers. This analysis generated comparative insights at a high conceptual level. For each map, we observed and compared: (1) the number of concepts, (2) the map density, (3) the concepts with the highest centrality measure, and (4) the concepts with the highest reachability to effective software development (Table 5). We complemented this analysis by delving into each of the interviewees' statements to consider the constructs underlying the concepts. This process led us to create three tables (Tables 3–5) that illuminate different aspects of the comparisons

across the three groups. As Table 5 shows, the comparison across the causal maps revealed four expectations that were unique to Group 1: user impact, social impact, development dynamism, and development contributions.

Step 5. Develop the theory. During theory development, we used additional data sets to challenge and enhance our interpretations of the four unique Group 1 characteristics identified in Step 4. First, we relied on complementary data—including archives recommended by the interviewees and additional discussions with some of the key informants—to help us re-examine the implications of some of the statements that had led to our interpretations of Group 1's distinct expectations. For example, developers who had grown up using social networking made claims about *continuous invention* in the use of development technologies. We then looked at available meeting notes for evidence (if any) that those developers had played roles in experimentation processes. We also further interviewed development managers to inquire about experimentation processes in their specific contexts and to track the source of significant changes they had experienced. Second, we used existing theories and evolving industry trends to challenge our interpretations' generalizability and enhance the credibility of our theoretical account [65, 71]. We did so by looking for both confirming and disconfirming evidence. Step 4, for example, suggested that Group 1 developers expect software to challenge established norms and transform how users interact with technology (see *user impact* in Table 5). We found confirming evidence of this when examining how the evolutionary path of social networking applications—first UseNet and Six Degrees, followed by MySpace/Friendster/Facebook/Twitter, and then Snapchat/Instagram—mirrors the impact that evolving generations of developers have had in building social networking applications that challenge established norms and better fit into real-life experiences. Challenging our interpretations from Step 4 by triangulating them with additional insights from Step 5 led us to develop Proposition 1 in our theory (described in the theory development section).

Similarly, Step 4 suggested developers in Group 1 and Group 2b—but not those in Group 2a—raised meaningful concerns about software's role in helping address societal needs and enhance social capital (see *social impact* in Table 5). Still, we considered that today's developers in Group 2a could also be advocates of social impact in software development. We included such disconfirming insights into our theoretical interpretations (1) by relying on Social Impact Theory to argue that the extent of social impact is influenced by the temporal and spatial proximity between the relevant parties [41], and (2) by suggesting that such expectations are more likely to be emphasized in generations that have experienced higher levels of social interactions. These detailed analyses informed and strengthened our theorizing and led us to develop four propositions that explain how the seemingly small areas of difference we identified can accumulate and become the source of important changes in software products and processes. Finally, to benefit from peer debriefing, we discussed the emerging theory with colleagues who were not involved in the study.

Findings

The data on the 49 software developers suggested a mixture of precocious users (Group 1, 30%), early age users (Group 2a, 32%), and later age users (Group 2b, 38%) of social networking technology. Developers in both Group 1 and Group 2b had generally begun using the new technology in almost the same year that it was released (Appendix 3). At the time of that release, however, Group 1 developers were very young (<11 years old), whereas developers in Group 2b were significantly older (<20 years old). Like Group 1 developers, those in Group 2a were also young at the time of the new technology's release, but they were not as agile in adopting that technology into their daily life. Hence, they had adopted social networking technology a few years after its release.[2]

[2] Investigating the underlying reasons for a group of individuals being precocious in new technology use is beyond the focus of this paper. Nevertheless, our informants consistently described contextual reasons that encouraged or required them to use the new technology—most notably, having had a family member either working in computer science or living overseas.

The causal maps illustrate how developers across the three groups described their expectations of software development (Figures 2–4). As the maps indicate, the groups refer to similar types of product and process expectations, yet our in-depth analysis shows that their emphasis on expectation concepts and their underlying constructs varies considerably. We now offer a detailed explanation of how developers across the three groups articulated their concerns about (1) product expectations, (2) process expectations, and (3) the relations between them. We then summarize the findings, report comparisons between the three maps, and unearth the implications of these findings for generational changes in software developers' expectations.

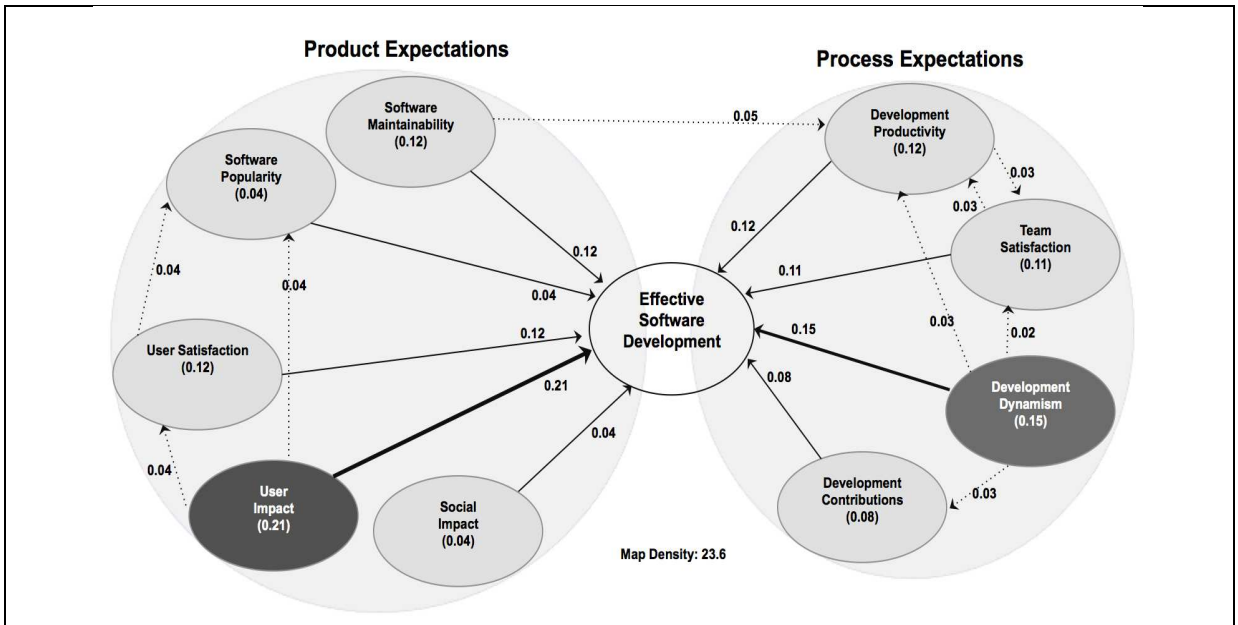


Figure 2. Group 1 Map

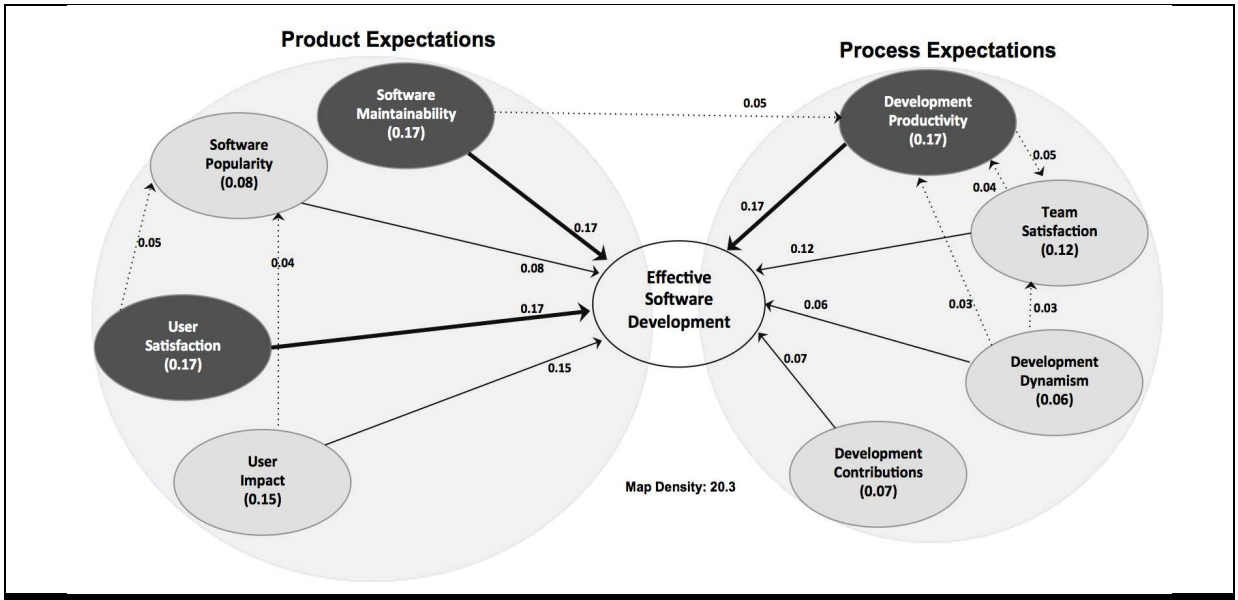


Figure 3. Group 2a Map

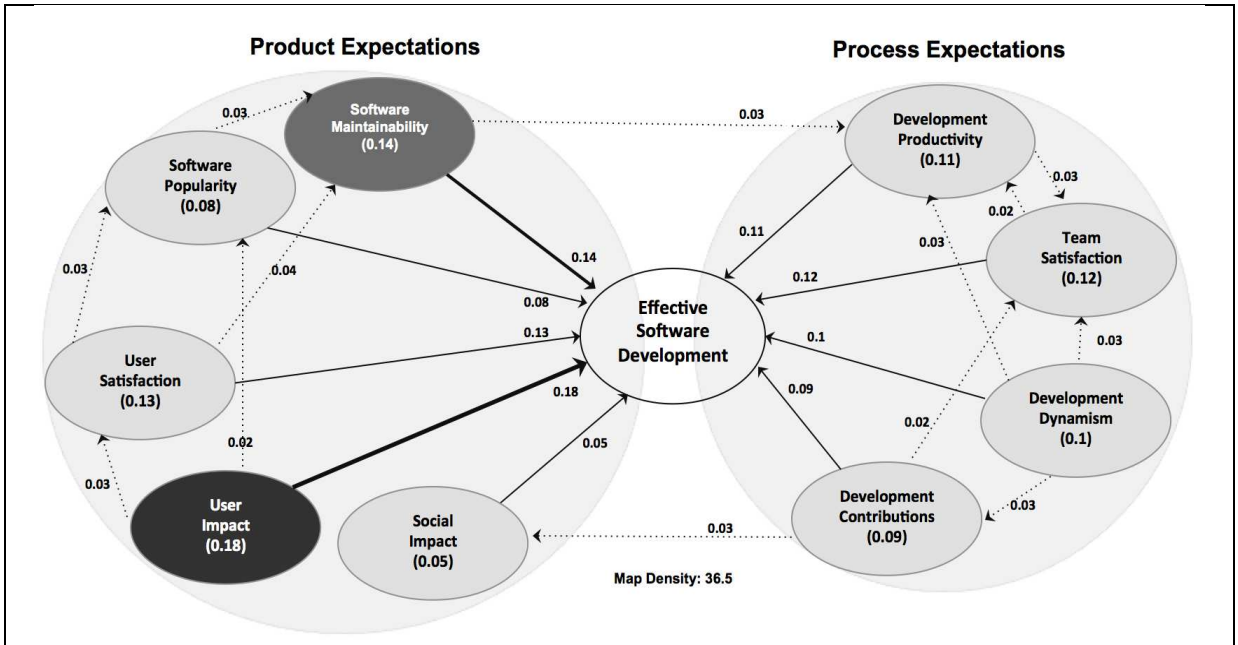


Figure 4. Group 2b Map

Product Expectations

User satisfaction. Developers across all groups emphasized the importance of user satisfaction in software development. According to these developers, software should meet user needs, requirements, and expectations, and users should feel the software's usefulness in their activities. However, Group 2b—experienced developers—was the only group that

emphasized the role of software ecosystems in shaping user satisfaction. These developers argued that user communities and software ecosystems help users communicate with each other and with the development team in helpful and satisfying ways.

User impact. Developers frequently mentioned user impact, and it was a central expectation in the cognitive maps of Group 1 and Group 2b. Developers across the three groups referred to the importance of creating impactful software that helps users become more productive, fulfilled, and successful in their activities. Experienced developers in Group 2b also touched on another expectation about user impact: to develop software that enables communities that inspire users and offer opportunities to enhance users' career paths. Finally, Group 1 drew attention to software's vital role in challenging existing applications and offering solutions that improve how users interact with technology. This finding is important; none of the developers in Group 2a or Group 2b raised this expectation.

Software popularity. All developers consistently emphasized that development activities should focus on software that can attract and support as many users as possible. Although experienced developers (Group 2b) agreed with this view, they emphasized that the software's long-term popularity among a selected group of users is more important than merely attracting a lot of users.

Software maintainability. All developers consistently mentioned the importance of software maintainability. This suggests the expectation that software should have well-structured, elegant code and include features and functionalities that are easy to maintain, adapt, and scale to different platforms and for future purposes. For example, developers argued that poor code is difficult to change and that when changes are made, bugs are more likely to be introduced. Hence, software should be adaptable across a wide range of configurations. Although all groups highlighted the expectation of software maintainability, it was particularly central in the cognitive maps of Group 2a and Group 2b. Developers in Group 2b also raised the expectation that software be maintainable from a business perspective and that rich ecosystems have the potential to maintain resources needed for the software's ongoing development.

Social impact. Only Group 1 and Group 2b developers mentioned the importance of software's social impact. Both groups linked software to social causes aimed at addressing societal needs and enhancing social capital. Group 1's developers had deep-rooted experience in networked social settings and raised meaningful expectations that the software they develop should cross commercial development boundaries and help society. Group 2a's developers, despite being similar to Group 1 in both age and background experience, did not share this emphasis. Like Group 1, however, the experienced developers in Group 2b highlighted the impact of their software at a global level and shared the expectation that products should make meaningful and useful contributions to society. These experienced developers further expected that software should build dynamic communities of interest with tangible benefits for society.

Process Expectations

Development productivity. All developers consistently discussed their expectation about ongoing improvements in the productivity of development processes. Interestingly, developers did not point to traditional norms of productivity in terms of meeting schedules or budget requirements; instead, they said that development teams should actively look for innovative ways to get software changes to users faster and reduce release costs. They argued that better productivity is not only crucial to development teams, but also gives users confidence that developers are making adequate progress. Experienced developers in Group 2b also raised an additional expectation here: that software processes should aim for less waste in both effort and resources, and that development teams should continuously search for the sources of waste and how to address them.

Team satisfaction. All developers suggested that software development should offer healthy team interactions and a sense of personal achievement. They also raised the expectation that development should unfold as a fun daily activity and create teamwork conditions that energize team members and inspire them to learn, make smarter choices, and produce superior products.

Development contributions. All developers expected software development to make

methodological contributions to the software community. They noted that such contributions might come in the form of code or technical advice. Group 1 developers also raised another expectation: that reflective software development can and should allow developers to innovate new models for organizing development work, and that such models can turn into applicable standards that can be used by other development teams.

Development dynamism. All developers—but especially those in Group 1—emphasized dynamism, suggesting that development embrace continuous reflection, experimentation, and invention in using development technologies to help developers stay on top of their professional careers. They said development processes should be taken as an opportunity to learn and experiment with new development tools and technologies such as plugins, frameworks, and pieces of code. Group 1 developers also emphasized the importance of experimenting with meaningful actions to invent new development tools. Further interviews with development managers confirmed their desires. For example, one senior product manager explained how a group of these developers recently challenged the team to explore a new open source tool in their development process:

> "*These folks have been very active in examining new tools as soon as they discover them to see if they can find a use case. For example, it was Nik that suggestion that we recently try a useful tool to quickly find obsolete API usage in our Xcode projects.*"

Relationships among Expectations

The interviewees reinforced their arguments about software product and process expectations by articulating relationships among the concepts. The causal maps collectively illustrate these relations with linkages (1) among product concepts, (2) among process concepts, and (3) between product and process concepts.

Linkages among product concepts suggests that achieving certain product-related qualities enhances the chances of achieving other product-related qualities. For example, Vincent (Group 2b) explained that software popularity improves software maintainability because a large ecosystem of users is more likely to help maintain the software's growth. Similarly, Manuel (Group 2b) described how software impact increases user satisfaction with the software, which in turn enhances the software's popularity:

"It wouldn't be sufficient to have users satisfied with the final product. The software should delight users, improve their well-being, and make some noticeable changes to their daily life [user impact]. *This way they will be happier with the software* [user satisfaction], *and they are more likely to spread the word to their network* [software popularity]."

Linkages among process concepts suggests that achieving certain process-related qualities enhances the chances of achieving other process-related qualities. For example, Kate (Group 1) explained that experimentation and innovation in the use of development technologies (*development dynamism*) enhance not only development productivity and team satisfaction, but also the chances of making development contributions to the software community (*development contributions*):

> *"The web has come a long way, but there's still so much more growing to do. The Internet of Things and cloud components have enormous potential to revolutionize how we connect and deliver information. By encouraging developers to experiment with and use the new competencies, software organizations can release true open source projects and even make code contributions to nonprofit software foundations."*

Developers also consistently emphasized *relations between product and process expectations*. For example, they argued that improved levels of *software maintainability* reduce time, cost, and waste in software releases, hence increasing *development productivity*. Furthermore, experienced developers noted that code contributions during software development (*development contributions*) help enrich the software's *social impact*; Nik (Group 1) explained this as follows:

> *"I think software developers have all the tools they need to build software with social impact. Right now, there are several platforms like GitHub to contribute code* [development contributions] *that can lead to software aimed at social impact."*

Comparisons across Groups

An initial comparison of the causal maps indicates that the three groups are generally similar in their expressed expectations. Apart from the concept of social impact—which Group 2a developers did not raise—all groups referred to all nine product and process expectations. Despite these similarities, however, a closer look into the details of their statements suggests that the groups differ in how they explain both the concepts and the relations among them. Regarding the concepts, as Table 3 shows, the three groups shed light on different aspects of expectation (that is, the underlying constructs). The experienced developers in Group 2b provided the most comprehensive view of development expectations, referring to 20 of the

23 underlying constructs, while Group 1 referred to 17 and Group 2a to 13. As the most experienced group, Group 2b developers are the only ones who emphasized the business ecosystems surrounding software products, suggesting that software should create communities of interest that (1) users are satisfied with (row 3); (2) contribute to the software's business growth (row 12); (3) enhance collective capital (row 14); and (4) contribute to the user's career path (row 6). Moreover, developers in Group 2b provided a more long-term perspective on software development by emphasizing the importance of nourishing long-term users (row 9) and reducing waste in long-term development efforts (row 17).

Table 3. Articulation of Expectations Across Groups

Dimension/Concepts/Constructs			Group		
			1	2	3
Product Expectations	User satisfaction	1. Satisfaction with software features and functionalities	*	*	*
		2. Satisfaction with their perception of the software's usefulness	*	*	*
		3. Satisfaction with the ecosystem around the software			*
	User impact	4. User productivity	*	*	*
		5. User well-being	*	*	*
		6. User career path			*
		7. Challenging established norms	*		
	Software popularity	8. Wide-spread use	*	*	*
		9. Long-term use			*
	Software maintainability	10. Code quality	*	*	*
		11. Scalable software	*	*	*
		12. Business maintainability			*
	Social impact	13. Societal needs	*		*
		14. Dynamic ecosystems			*
Process Expectations	Development productivity	15. Time reduction	*	*	*
		16. Cost reduction	*	*	*
		17. Wasted-effort reduction			*
	Team satisfaction	18. Team morale	*	*	*
		19. Team learning	*	*	*
	Development contributions	20. Code contributions	*	*	*
		21. Contributing new models for organizing development works	*		
	Development dynamism	22. Continuous experimentation in development	*	*	*
		23. Continuous innovation in the use of development tools	*		

However, Table 3 also indicates that Group 1 developers drew attention to specific expectations about *user impact*, *development contributions,* and *development dynamism* (rows 7, 21, and 23). They did so by explaining the importance of (1) challenging established norms and transforming how users interact with technology, (2) continuous experimentation and innovation in the use of new development technologies, and (3) contributing both new and standard models for organizing development work. The significance of this finding for Group 1—the precocious users of social networking—is

reinforced by comparing them to the perceptions of developers in Groups 2a and 2b, none of whom raised these expectations. Table 3 also suggests that Group 1's developers emphasized the importance of social impact, as did Group 2b's experienced developers, who are influenced by their more advanced age and development experience. This expectation, however, was not emphasized in the cognitive mindsets of Group 2a's developers. Specifically, none of that group's developers referred to social impact as an expectation. Instead, they frequently highlighted classic expectations including user satisfaction, development productivity, and software maintainability.

Table 4 summarizes the relations among concepts by developer groups. Group 1 and Group 2a referred to 9 and 7 linkages between expectation concepts, respectively, whereas developers in Group 2b referred to 13 linkages. The critical difference here is that Group 2b is the only group that (1) articulated how popularity and user satisfaction of software ecosystems improve software growth and maintainability (items 2, 5), and (2) shed light on the impact of development contributions on increasing team satisfaction and enhancing the social impact of software (items 11, 13). Still, as we noted, Group 1 developers highlighted *user impact* by challenging established norms as well as *development contributions* to the software community. Hence, unlike developers in Group 2a, precocious users of social networking emphasized two linkages that suggest both that (1) software that improves how users interact with technology increases user satisfaction (row 3), and that (2) continuous experimentation and invention in development technology use increase the chance of making development contributions (row 10). Based on these insights, Table 5 summarizes the key findings across the three developer groups.

Table 4. Relations

		Linkages	Group 1	Group 2a	Group 2b
Product-Product Linkages					
1	**Product-Product**	User satisfaction→Software popularity	X	X	X
2		User satisfaction→Software maintainability			X
3		User impact→User satisfaction	X		X
4		User impact→Software popularity	X	X	X
5		Software popularity→Software maintainability			X
Process-Process Linkages			**3**	**2**	**5**
6	**Process-Process**	Development productivity→Team satisfaction	X	X	X
7		Team satisfaction→Development productivity	X	X	X
8		Development dynamism→Development productivity	X	X	X
9		Development dynamism→Team satisfaction	X	X	X

10		Development dynamism→Development contributions	X		X
11		Development contributions→Team satisfaction			X
Process-Product/Product-Process Linkages			**5**	**4**	**6**
12	**Product-Process**	Software popularity→Development productivity	X	X	X
13		Development contributions→Social impact			X
Total			**9**	**7**	**13**

Table 5. Comparisons

Group 1	Group 2a	Group 2b
(1) 9 out of 9 expectation concepts (2) 17 out of 23 expectation constructs (3) 9 linkages and density 23.6 (4) The only group that emphasized the role of software in transforming how users interact with new technology (***user impact, product expectations***). (5) The only group that reported taking major steps to experiment and innovate while using new development technologies (***development dynamism, process expectations***). (6) The only group that characterized development contributions by explaining that software development can expand standard models for organizing development works (***development contributions, process expectations***). (7) As with experienced developers in Group 2b, this group raised meaningful concerns about social impact regarding how software helps address societal needs and enhance social capital (***social impact, product expectations***). (8) User impact (0.22) is the most central and reachable concept to software development.	(1) 8 out of 9 expectation concepts (2) 13 out of 23 expectation constructs (3) 7 linkages and density 20.1 (4) User satisfaction (0.17), Development productivity (0.17), Software maintainability (0.17) are the most central and reachable concepts.	(1) 9 out of 9 expectation concepts (2) 20 out of 23 expectation constructs (3) 13 linkages and density 36.5 (4) The only group that emphasized software ecosystems, including user satisfaction with the software ecosystem (user satisfaction), and software ecosystems that enhance user career path (user impact), create lively communities that enhance social capital (social impact), and help maintain business growth (software maintainability). (5) The only group that emphasized the importance of nourishing long-term users (software popularity) and reducing waste in development efforts (development productivity). (6) User impact (0.18) is the most central and reachable concept.

Theoretical Propositions

The empirical analyses suggest that developers who have been precocious in using social networking technology have many similarities and particular differences in how they express expectations about software products and processes as compared to other developers. In a few years, many software graduates will likely have been precocious users of popular social networking applications such as Facebook, YouTube, and Instagram. However, with significant developments in social networking technology—including virtual reality (VR) and artificial intelligence (AI)—new social networking applications will continue to emerge. Accordingly, the software workforce will continue to experience an inflow of developers who, as precocious users of newer social networking technologies, can help renew software products and processes. This evolving nature of the development workforce motivates the following theorization.

Shifts in Product Expectations and Renewal of Software Products

Developers in Group 1 had specific expectations about *user impact* that emphasized the importance of challenging established norms and transforming how users interact with technology. These developers also linked higher levels of user impact with improved user satisfaction and enhanced software popularity. Indeed, when asked about key innovations in their product portfolio, iSolution's engineering manager highlighted how these developers changed product expectations:

"Having [Group 1 developers] *has brought new expectations about what the final software should look like. As such, we have started to develop more creative and intuitive solutions … I found the newer generation is grown up having easy access to abundant information. Back in our time, it was more about knowing information … these developers come to work expecting everything works instantly, to be at their fingertips."*

These findings are better understood in the light of development trends in social networking technology. From MySpace and Friendster to Facebook and Twitter to Instagram and Snapchat, social networking has advanced from a technology that replicates the real world in digital space to one that intuitively embraces the nature of social interactions. As an example, by building UseNet in 1979, developers gave users the ability to communicate through online newsletters. In 1997, the Six Degrees developers let users—for the first time—create a profile and become friends with other users, opening a path to current forms of social networking. Developers of similar applications such as Facebook and Snapchat built on this concept, working to creatively connect users in more personal, intuitive ways that better approximate real-world interactions. Facebook, for example, developed its revolutionary timeline feature in 2011, allowing it to leverage existing content to motivate user engagement and further contribution. Similarly, in 2012, Snapchat shifted social networking's primary focus from "creating the real world to recreate it online" to "creating digitally disposable content that mirrors the fleeting nature of real-life interactions."

We thus expect that individuals who grow up with emerging social networking applications, such as those utilizing VR, will be more likely to expect software applications to fit seamlessly into their real-life experiences. If those individuals choose to become developers, they will likely go beyond satisfying personal and work-related needs and

requirements of users to develop software that supports intuitive usage and challenges the established ways that users interact with software applications (*shifts in expectations*). Because beliefs and attitudes toward a particular behavior can lead to considerable variance in the actual behavior [2], developers who have been precocious users of newer forms of social networking technology will be more likely to critically assess software usability and create software solutions that support intuitive usage. These solutions have the potential to improve user satisfaction with the software and expand the ecosystem of the software's potential users and external developers (*renewal*). These arguments motivate the first proposition:

> ***Proposition 1.*** *Developers who have been precocious users of newer forms of social networking applications will be more likely to expect a critical assessment of software usability and the development of solutions that support intuitive usage, which in turn will increase user satisfaction with the software and its popularity in the market.*

Our results demonstrated that Group 1 and Group 2b developers, in contrast to those in Group 2a, emphasized a desire to develop software that helps address societal needs, enhance social capital, and contribute to evolving ecosystems. A Group 1 developer eloquently expressed this insight as follows:

> *"Coding to benefit the society is appealing to me. I used to volunteer to code for putting open government data to good use. Here, I'm working on a pet project to align some of our products with social causes ... The CEO* [previously a senior developer] encourages *participation in* [a competition for hacking government systems] *to invent new data management apps. Last year, I got two awards. He liked the prototype, so we've got some funding this year to turn it into a real solution."*

Social Impact Theory (SIT) [41] contends that the extent of social impact is influenced by the temporal and spatial proximity of the relevant parties [51]. Accordingly, for social influence to be manifest, a large number of individuals must be present at the time and close to each other in social space. This insight relates to the finding that experienced developers have likely developed an awareness of the *social impact* of their software by the virtue of having worked in various positions and contributed to and collaborated with other developers on different projects over time. On a similar note, the fundamental driver behind

social networking technology is to create value from social activity by attracting users with shared interests. Social networking applications continuously add new features to encourage human contact and expand the social network, letting users employ tags and other mechanisms to organize content and help others find useful information. These features empower individuals to become immediate information sources and to share their opinions, experiences, and perspectives with others. More recent forms of social networking applications such as Instagram and Snapchat increase these interactions by offering features that motivate more frequent and intense user engagements. This argument reconfirms the finding that if individuals grow up using emerging social networking applications, they have experienced increased opportunities to cross time and space boundaries and developed an early taste of the social impact of daily life interactions. If those individuals become software developers, they will likely bring expectations about the societal impacts of the software they develop to the industry (*shifts in expectations*). We thus argue that a collaboration between developers who were precocious users of newer social networking technologies and experienced developers is likely to reinforce trends emphasizing social impact in software development (*renewal*). This argument constitutes the second proposition:

> ***Proposition 2.*** *Developers who were precocious users of newer forms of social networking applications are more likely to emphasize social impact during development of software and to further reinforce trends to do so when collaborating with experienced developers with similar expectations.*

Shifts in Process Expectations and Renewal of Software Processes

Developers in Group 1 conveyed specific expectations about *development dynamism* by highlighting the importance of continuous experimentation and innovation when using new development technologies. A Group 1 developer summarized this finding well:

"Last year, Randy and I started a weekly lunch at work where developers can hang out and share the coolest tools they have recently seen or used ... By making these lunch breaks fun, we've got team members interested in taking the time to experiment with new [development] *technology. Even if we decide against adopting it, we've learned a lot and have knowledge in that area."*

Prior research on generations and technology suggest that, while people who began using a major technological innovation at an early age may not differ from preceding

generations in how tech-savvy they are in using that technology, they will likely demonstrate behavioral distinctions in how they use it [26]. Specifically, newer generations are more likely to prefer experimentation and content creation to simply consuming existing content when using a technology they grew up with. These behavioral distinctions stem from the deep-rooted history of regular use from their early years, and they will likely increase because more recent forms of social networking technology put a stronger emphasis on content creation and interaction among users. Such an insight has especially important implications for the software industry because developing software requires creating, sharing, and blending information over time and across diverse spheres of specialization [24]. Contemporary development technologies also increasingly borrow features from social networking applications to enable collaboration and teamwork across distributed groups of developers. For example, developers have traditionally used face-to-face or email notifications for user-requirement updates. This trend has begun to change as social software technology relocates requests and notifications into a live, dynamic chat room. A chat room is an inherently social space that lets developers interact in real time, overcoming the constraints of asynchronous email to enhance their learning and decision making. These arguments concur with our empirical finding that developers who have grown up using recent forms of social networking applications are more likely to experiment with and innovate development technologies that share features with social networking applications (*shifts in expectations*).

We can better understand the importance of such generational elements for improving software development outcomes by attending to the technological changes in software development and the driving role of experimentation. The Internet era, for example, made it possible for developers to build high-speed, low-cost business systems that go beyond the traditional physical and geographical boundaries of development. Web-based development brought different requirements and expectations, including increased project scale, scope, technical complexity, and team diversity [9]. Despite such differences, however, scholars have found that software developers face remarkably similar challenges over time, including

time pressures and the need for team diversity [39]. Given this, some have suggested that the challenge in developing new software is not to invent new, incremental approaches that reflect unique technological trends; rather, the challenge is to give developers the freedom to experiment with shifting technological regimes so they can better understand how to use new and different approaches in a planned, manageable, and consistent way [39]. Hence, the emphasis of new generations on *dynamism* in using development technologies can help software organizations to embrace—and learn how best to use—new technologies for developing software. Specifically, this dynamism can help create best practices and push innovations forward as developers discover new methods and metrics that increase development productivity [22, 62]. A dynamic software development context can subsequently enhance the team satisfaction of developers and increase the chances that they will contribute to the software (*renewal*). These arguments lead to the third proposition:

> ***Proposition 3.*** *Developers who were precocious users of newer forms of social networking applications are more likely to experiment with development technologies, which in turn will enhance development productivity, improve team satisfaction, and contribute new models and standards to the software community.*

Shifts in Process Expectations and Renewal of Software Products

We found that developers generally believed that software teams should make *development contributions*—such as through code and open source inputs—to the broader community. Prior research has also shown an increasing emphasis on software development communities [21]. However, we found that Group 1 developers sometimes went beyond this in noting that ambitious software development teams could package their development experiences as new standards and models for conducting and organizing development tasks. Indeed, when asked about engagement with the broader development community, one development manager at iSirva highlighted contributions by Group 1 members:

"*They are comfortable with open source ecosystems that are built on the premise of capitalizing on good ideas and contributions from the entire community. Some of our projects aim to develop data management software for different fields. They contributed creative ways for collecting external ideas and building online communities."*

This finding is consistent with our previous discussion on how social networking

applications have evolved to motivate users to contribute rather than simply consume information. Snapchat, for example, opens by launching users immediately into a camera, suggesting: "*Here's the main thing; figure out the rest and create new content*." In contrast, earlier social networking technologies such as Facebook and even Instagram open on a timeline of existing content of other users. Building on the increasing focus on content contributions, we expect that developers who were precocious users of newer social technologies will increasingly challenge norms about software and development processes, actively engaging in turning innovations into new process models and sharing them widely with the software community (*shifts in expectations*). With their emphasis on contributing new models and standards to the software community, these developers can, in turn, enhance the trend of social impact in software (*renewal*). We thus propose:

> ***Proposition 4.*** *Developers who were precocious users of newer forms of social networking applications will more likely contribute new models and standards to the software community, which in turn will reinforce the emphasis on the social impact of the software they develop.*

As Figure 5 summarizes, Propositions 1–4 suggest the ways in which socially infused networking technologies have shaped—and continue to shape—new generations of developers, who may emerge as creative forces that enhance software products and processes by perceiving and approaching their development differently from preceding generations of developers.

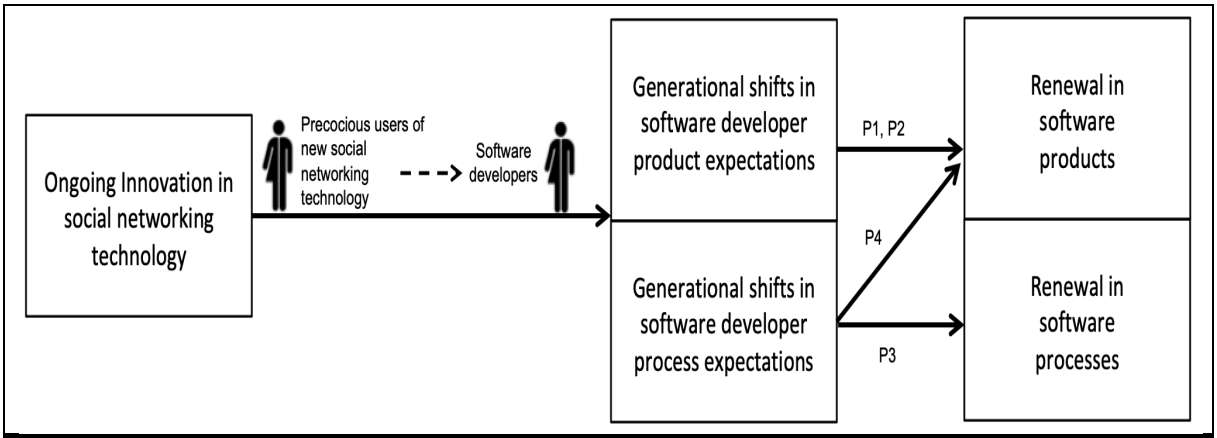


Figure 5. Gefnerational Perspective on the Software Workforce

Discussion and Future Research

The findings from this exploratory study have important implications for information systems research, generational studies, and the development literature. Here, we recapitulate the major insights into these implications to enhance understanding of (1) the relationship between innovations in social networking and new generations of software developers, (2) software expectations and the relations among them, and (3) technology-related experiences and new generations of employees. We also outline several lines of research that are ripe for further investigation and may advance our findings.

New Generations of Software Developers

Our theorization adds to and advances socio-technical perspectives [6, 29, 30] by suggesting that changes in software development are not driven purely by changes in technological advancements [44]. Rather, new generations of software developers, who enter the workplace with distinct expectations and contributions, can feed into how changes in technological innovations emerge and advance over time. These findings are an expression of the *co-evolutionary perspective* [55], which recognizes this interrelation between software developers and new technology. The co-evolutionary perspective suggests that the software industry continuously benefits from employees with new sets of knowledge, skills, and abilities that allow them to advance existing technology; however, thus far, it explains these changes in software development professionals in relation to external forces—such as the industry's drive to adopt new technology, innovation, and low-cost development. We expand the co-evolutionary perspective by contributing a generational edge that explains how software industry innovations can themselves influence the cognitive expectations of individual users, who then enter the industry as engines of change. By leveraging the diversity and inclusiveness of these emerging perspectives, software organizations—and the industry as a whole—can renew their development practices.

Our propositions suggest that software organizations can leverage developers who

have been precocious users of newer forms of social networking technology to serve several key purposes, including (1) creating ideas that challenge software usability and offer solutions that support intuitive usage; (2) reinforcing social impact through engagements with experienced developers; and (3) re-energizing innovation in development processes through experimentation, invention, and contribution to the software community.

These results contribute a new foundation for advancing the literature on the co-evolutionary relationship between the precocious use of technological innovations and information systems personnel. Future research can build on our results by incorporating generational concepts and cognitive models into investigations of how software industry changes happen over time. Such research could be an ongoing effort as advancements in technology continue to influence individuals' mental models and cognitive mindsets.

Software Development Expectations

We make two contributions to the software literature on development goals and expectations [60, 67].

First, as Table 2 summarizes, the cognitive maps of the three developer groups contribute a more comprehensive and contemporary set of development goals and expectations than currently exists in the literature. The conceptualization of product expectations reveals new constructs about increasing (1) user satisfaction in the software ecosystem, (2) software maintainability through ecosystem efforts, (3) user impact through career advancement opportunities, and (4) user impact through solutions that challenge usability and support intuitive usage. Similarly, the conceptualization of process expectations reveals new constructs about expanding (1) development contributions by introducing new ways of socializing and organizing work in software development, and (2) development dynamism by inventing and using new development tools and technologies. These additions mirror changes that have occurred in the software industry's values and norms [60, 67]. Future research can expand our findings by focusing on the ongoing changes in values, norms, and practices related to development goals in the software industry. Our results, for

example, highlighted the importance of code quality as part of the software maintainability concept. The interviewees—who develop software for the R&D and science sector—characterized code quality as elegant code that is both flexible and adaptable to required changes. Software development, however, increasingly confronts demands for device interconnectivity, integration, and platform compatibility. These demands give rise to specialized code review applications that expand the boundary of code quality to include the need to address security vulnerabilities and thereby avoid hostile attacks.

Second, our findings expand on software development literature that has drawn attention to product and process expectations about developing software, but has not yet systematically investigated how those expectations might contribute to better outcomes [12]. Aggregating the cognitive maps of developers with different perspectives lets us offer a new explanation of how product and process expectations interact with each other. For example, by building software that features maintainable code and community (product expectation), developers can be more productive (process expectation); by offering knowledge contributions to the software community (process expectation), developers can increase the software's maintainability (product expectation) and contribute to the software's long-term impact on social contexts (product expectation). Future research can build on these findings and illuminate further relations among different development goals, expectations, and outcomes.

Technology-Driven Generations

Prior research reports on the rise of emerging generations—such as digital natives—in education contexts [27] and as end users of ubiquitous information systems [70]. More recently, scholars have expanded this conversation into organizations, focusing on how the digital competencies of the workforce have changed and will continue to do so [16]. Indeed, researchers emphasize the importance of understanding how these new generations of employees, who have early age experiences using modern technologies, approach work, as well as how organizations might best utilize emerging generations in service of their goals [16].

Existing discussions in this area, however, are beset by debates over whether generational differences exist and how emerging generations should be studied and operationalized [10, 28]. Thus far, we have seen mixed results and discouraging research on technology-driven generations (e.g., digital natives), and part of the problem relates to methodological choices. Examples here include studying generations by virtue of birth year (e.g., everyone born after 1982) or fundamentally challenging the idea of generational shifts in favor of concepts such as digital literacy [3, 27, 28, 34, 47, 68] and computer engagement [40]. Further, understanding how technology-related experiences—such as being a precocious user of a specific technology—influence individuals' mental models is not as straightforward as understanding other similar but more tangible phenomena, such as native accent acquisition [23, 43].

Against the dual backdrops of debate on technology-driven generations [10, 27, 70] and literature on the digital workforce [16], our study offers a systematic approach to investigating experience-based similarities and differences across different groups of individuals. As we explain in detail in Table 1 and Appendix 4's method section, our approach includes rigorous methodological steps, data collection, and data analysis procedures that future research can rely upon to better understand generational shifts in the career perceptions and competencies of the workforce. We encourage future studies to use and build on our approach to delve into different cognitive aspects of technology-related experiences. For example, researchers can use our approach and recent recommendations from cognitive development research to conduct larger-scale quantitative examinations into generational issues in technology and management research. Also, we removed six developers who were on the edge of being precocious users of social networking technology to reduce the possibility of overlap with fully precocious users. While we expect such nearly precocious individuals to exhibit mixed results, it would be insightful to further investigate this issue and its implications for work-related settings. Finally, although our methodological approach helped overcome potential demand characteristics in the data collection process and continuously challenged our interpretations toward theory development, we suggest

researchers involve different coders throughout the analysis to further to enhance interpretations.

Conclusion

Our qualitative investigation into how precocious users of social networking technology demonstrate distinct expectations about the goals of software development (1) advances a generational perspective on the software workforce, and (2) offers a methodological approach for exploring the distinct characteristics and contributions of technology-driven generations and the changing workforce. Our generational perspective includes theoretical propositions on the reciprocal links in the evolution of social networking technology, new generations of software developers, and software renewal. Together with the empirical findings, the perspective reveals considerable opportunities for understanding the nature and drivers of technological innovations in today's increasingly dynamic and competitive software industry. As such, we hope that our approach for comparing different groups together with our empirical and theoretical contributions will inspire and guide future inquiry into generational dynamics to address some of the software industry's most critical and demanding challenges.

Acknowledgement

We are grateful to the editors and anonymous reviewers for constructive feedback during the review process. This research project also benefitted from valuable comments from Brian Pentland, Lewis Williams, Steven Johnson, Arun Rai, Jens Dibbern, Thomas Huber, and attendees of different research seminars. We thank the interviewees from the software organizations who shared their perspectives and insights. This study received funding from the British Academy Leverhulme Small Research Grant (project ECO2017- 88576-R).

References

1. Ahn, J. and Y. Jung. The common sense of dependence on smartphone: a comparison between digital natives and digital immigrants. *New Media & Society*, 18, 7 (2016), 1236-1256.
2. Ajzen, I. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50, 2 (1991), 179-211.
3. Akçayır, M., H. Dündar, and G. Akçayır. What makes you a digital native? Is it enough to be born after 1980? *Computers in Human Behavior*, 60, (2016), 435-440.
4. Alaimo, C. and J. Kallinikos. Computing the everyday: social media as data platforms. *The Information Society*, 33, 4 (2017), 175-191.
5. Alsop, R., *The Trophy Kids Grow Up: How the Millennial Generation is Shaking Up the Workplace*.San Francisco, CA: Jossey-Bass, Wiley, 2008.
6. Arthur, B., *The Nature of Technology: What It Is and How it Evolves*.New York: Free Press, 2009.
7. Bala, H., A.P. Massey, and M.M. Montoya. The effects of process orientations on collaboration technology use and outcomes in product development. *Journal of Management Information Systems*, 34, 2 (2017), 520-559.
8. Banker, R.D. and C.F. Kemerer. Performance evaluation metrics for information systems development: a principal-agent model. *Information Systems Research*, 3, 4 (1992), 379-400.
9. Baskerville, R. and J. Pries-Heje. Short cycle time systems development. *Information Systems Journal*, 14, 3 (2004), 237-264.
10. Bennett, S., K. Maton, and L. Kervin. The 'digital natives' debate: A critical review of the evidence. *British Journal of Educational Technology*, 39, 5 (2008), 775-786.
11. Biggs, S. Thinking about generations: conceptual positions and policy implications. *Journal of Social Issues*, 63, 4 (2007), 695-711.
12. Caldeira, M.M. and J.M. Ward. Understanding the successful adoption and use of IS/IT in SMEs: an explanation from Portuguese manufacturing industries. *Information Systems Journal*, 12, 2 (2002), 121-152.
13. Carlo, J.L., K. Lyytinen, and G.M. Rose. A knowledge-based model of radical innovation in small software firms. *MIS Quarterly*, 36, 3 (2012), 865-895.
14. Carstensen, P., C. Sorensen, and T. Tuikka. Let's talk about bugs. *Scandanavian Journal of Information Systems*, 7 (1995), 1.
15. Cedergren, S. and S. Larsson. Evaluating performance in the development of software-intensive products. *Information and Software Technology*, 56, 5 (2014), 516-526.
16. Colbert, A., N. Yee, and G. George. The digital workforce and the workplace of the future. *Academy of Management Journal*, 59, 3 (2016), 731-739.

17. Cramton, C.D. and S.S. Webber. Relationships among geographic dispersion, team processes, and effectiveness in software development work teams. *Journal of Business Research*, 58, 6 (2005), 758-765.
18. Davern, M., T. Shaft, and D. Te'eni. Cognition matters: enduring questions in cognitive IS research. *Journal of the Association for Information Systems*, 13, 4 (2012), 273-314.
19. Doll, W.J., W. Xia, and G. Torkzadeh. A confirmatory factor analysis of the end-user computing satisfaction instrument. *MIS Quarterly*, 1, 1 (1994), 453-461.
20. Eisenhardt, K.M. Building theories from case study research. *Academy of Management Review*, 14, 4 (1989), 532-550.
21. Fang, Y. and D. Neufeld. Understanding sustained participation in open source software projects. *Journal of Management Information Systems*, 25, 4 (2009), 9-50.
22. Fichman, R.G. and C.F. Kemerer. The assimilation of software process innovations: an organizational learning perspective. *Management Science*, 43, 10 (1997), 1345-1363.
23. Ganjoee, M.A. and M.H. Narafshan. A study on the effect of age on the representation and processing of second language grammar achievement. *Studies in English Language Teaching*, 4, 2 (2016), 223-240.
24. Ghobadi, S. What drives knowledge sharing in software team: a review and classification framework. *Information and Management*, 52, 1 (2015), 82-97.
25. Ghobadi, S. and L. Mathiassen. Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*, 26, 2 (2015), 95-125.
26. Greene, J.A., B.Y. Seung, and D.Z. Copeland. Measuring critical components of digital literacy and their relationships with learning. *Computers & Education*, 76, (2014), 55-69.
27. Guo, R.X., T. Dobson, and S. Petrina. Digital natives, digital immigrants: an analysis of age and ICT competency in teacher education. *Journal of educational computing research*, 38, 3 (2008), 235-254.
28. Helsper, E.J. and R. Eynon. Digital natives: where is the evidence?. *British educational research journal*, 36, 3 (2010), 503-520.
29. Henfridsson, O. and B. Bygstad. The generative mechanisms of digital infrastructure evolution. *MIS Quarterly*, 37, 3 (2013), 907-931.
30. Howcroft, D. and B. Light. The social shaping of packaged software selection. *Journal of the Association for Information Systems*, 11, 3 (2010), 122-148.
31. Iivari, N. Discursive construction of 'user innovations' in the open source software development context. *Information and Organization*, 20, 2 (2010), 111-132.
32. Ivari, J. An empirical test of the DeLone–McLean model of information system success. *The DATA BASE for Advances in Information Systems*, Spring, 1 (2005), 8-27.

33. Ives, B. and M.H. Olson. User involvement and MIS success: a review of research. *Management Science*, 30, 5 (1984), 586-603.
34. Jones, C., R. Ramanau, S. Cross, and G. Healing. Net generation or digital natives: is there a distinct new generation entering university? *Computers & education*, 54, 3 (2010), 722-732.
35. Joshi, A., J.C. Dencker, and G. Franz. Generations in organizations. *Research in Organizational Behavior*, 31, 1 (2011), 177-205.
36. Joshi, A., J.C. Dencker, G. Franz, and J.J. Martocchio. Unpacking generational identities in organizations. *Academy of Management Review*, 35, 3 (2010), 392-414.
37. Kang, H.-R., H.-D. Yang, and C. Rowley. Factors in team effectiveness: cognitive and demographic similarities of software development team members. *Human Relations*, 59, 12 (2006), 1681-1710.
38. Kaplan, A.M. and M. Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business horizons*, 53, 1 (2010), 59-68.
39. Kautz, K., S. Madsen, and J. Nørbjerg. Persistent problems and practices in information systems development. *Information Systems Journal*, 17, 3 (2007), 217-239.
40. Kesharwani, A. Do (how) digital natives adopt technology differently than digital immigrants? a longitudinal study. *Information & Management*, (2019), doi:10.1016/j.im.2019.103170.
41. Latane, B. The psychology of social impact. *American Psychologist* 36, 4 (1981), 343-56.
42. Laukkanen, M. Comparative cause mapping of organizational cognitions. *Organization Science*, 5, 3 (1994), 322-343.
43. Lenneberg, E.H. The biological foundations of language. *Hospital Practice*, 2, 12 (1967), 59-67.
44. Lyytinen, K. and G.M. Rose. The disruptive nature of information technology innovations: the case of internet computing in systems development organizations. *MIS Quarterly*, 27, 4 (2003), 557-596.
45. MacCormack, A., C.F. Kemerer, M. Cusumano, and B. Crandall. Trade-offs between productivity and quality in selecting software development practices. *IEEE Software*, 20, 5 (2003), 78-85.
46. Mannheim, K. The problem of generations. *Psychoanalytic review*, 57, 3 (1970), 378.
47. Margaryan, A., A. Littlejohn, and G. Vojt. Are digital natives a myth or reality? university students' use of digital technologies. *Computers & Education*, 56, 2 (2011), 429-440.
48. McCrae, R.R., P.T. Costa Jr, A. Terracciano, W.D. Parker, C.J. Mills, F. De Fruyt, and I. Mervielde. Personality trait development from age 12 to age 18: longitudinal, cross-sectional and cross-cultural analyses. *Journal of personality and social psychology*, 83, 6 (2002), 1456.

49. Miller, C.C., L.M. Burke, and W.H. Glick. Cognitive diversity among upper-echelon executives: Implications for strategic decision processes. *Strategic Management Journal*, 19, 1 (1998), 39-58.
50. Misra, S.C., V. Kumar, and U. Kumar. Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82, 11 (2009), 1869-1890.
51. Naylor, R.W., C.P. Lamberton, and P.M. West. Beyond the "like" button: the impact of mere virtual presence on brand evaluations and purchase intentions in social media settings. *Journal of Marketing*, 76, 6 (2012), 105-120.
52. Nelissen, S. and J. Van den Bulck. When digital natives instruct digital immigrants: active guidance of parental media use by children and conflict in the family. *Information, Communication & Society*, 21, 3 (2018), 375-387.
53. Ng, W. Can we teach digital natives digital literacy? *Computers & education*, 59, 3 (2012), 1065-1078.
54. Nidumolu, S. The effect of coordination and uncertainty on software project performance: residual performance risk as an intervening variable. *Information Systems Research*, 6, 3 (1995), 191-219.
55. Niederman, F., T.W. Ferratt, and E.M. Trauth. On the co-evolution of information technology and information systems personnel. *ACM SIGMIS Database*, 47, 1 (2016), 29-50.
56. Orne, M.T. On the social psychology of the psychological experiment. *American Psychologist*, 17, 11 (1962), 776-779.
57. Ozer, M. and D. Vogel. Contextualized relationship between knowledge sharing and performance in software development. *Journal of Management Information Systems*, 32, 2 (2015), 134-161.
58. Park, G., D. Lubinski, and C.P. Benbow. Contrasting intellectual patterns predict creativity in the arts and sciences: tracking intellectually precocious youth over 25 years. *Psychological Science*, 18, 11 (2007), 948-952.
59. Petter, S., W. DeLone, and E.R. McLean. Information systems success: the quest for the independent variables. *Journal of Management Information Systems*, 29, 4 (2013), 7-62.
60. Procaccino, J.D., J.M. Verner, M.E. Darter, and W.J. Amadio. Toward predicting software development success from the perspective of practitioners: an exploratory Bayesian model. *Journal of Information Technology*, 20, 3 (2005), 187-200.
61. Procaccino, J.D., J.M. Verner, K.M. Shelfer, and D. Gefen. What do software practitioners really think about project success: an exploratory study. *Journal of Systems and Software*, 78, 2 (2005), 194-203.

62. Quintas, P. A product-process model of innovation in software development. *Journal of Information Technology*, 9, 1 (1994), 3-17.
63. Quintas, P. Programmed innovation? trajectories of change in software development. *Information Technology & People*, 7, 1 (1994), 25-47.
64. Rai, A. and S.S.W. Lang. Assessing the validity of is success models: an empirical test and theoretical analysis. *Information System Research*, 13, 1 (2002), 50-69.
65. Sarker, S., X. Xiao, T. Beaulieu, and A.S. Lee. Learning from first-generation qualitative approaches in the IS discipline: an evolutionary view and some implications for authors and evaluators. *Journal of the Association for Information Systems*, 19, 8 (2018), 752-774.
66. Sawyer, S., P.J. Guinan, and J. Cooprider. Social interactions of information systems development teams: a performance perspective. *Information Systems Journal*, 20, 1 (2010), 81-107.
67. Shim, J.T., T.S. Sheu, H.G. Chen, J.J. Jiang, and G. Klein. Coproduction in successful software development projects. *Information and Software Technology*, 52, 1 (2010), 1062-1068.
68. Thompson, P. The digital natives as learners: Technology use patterns and approaches to learning. *Computers & Education*, 65, (2013), 12-33.
69. Tomaszewski, P. and L. Lundberg. Software development productivity on a new platform: an industrial case study. *Information and Software Technology*, 47, 4 (2005), 257-269.
70. Vodanovich, S., D. Sundaram, and M. Myers. Research commentary—digital natives and ubiquitous information systems. *Information Systems Research*, 21, 4 (2010), 711-723.
71. Walsham, G. Doing interpretive research. *European Journal of Information Systems*, 15, 1 (2005), 320-330.
72. Wey Smola, K. and C.D. Sutton. Generational differences: revisiting generational work values for the new millennium. *Journal of organizational behavior*, 23, 4 (2002), 363-382.

Appendix 1: Empirical Research on Generations and IT-related Behavior

Topic	Study	Summary of Findings
Suggesting differences across generations based on IT-related behavior	(Kesharwani 2019) [40]	Using a survey approach administrated in India, the author investigated the impact of user orientation toward technology and continued usage behavior. The author finds that (1) positive perceptions about technology-assisted learning would better explain the digital natives' continued usage than the digital immigrant's continued usage; (2) for digital immigrants, in addition to compatibility, the influence of significant others, such as instructors and friends, better explains continued usage behavior. However, for digital natives, system compatibility better explains their continued usage behavior.
	(Nelissen and Van den Bulck 2018) [52]	Using a survey approach administrated to examine child–parent digital media guidance in Belgium, the authors find that both children and parents reported that children guide their parents in how to use digital media, especially for newer media such as smartphones, tablets, and apps. They also find that families that had higher child–parent digital media guidance reported more conflicts about media.
	(Ahn and Jung 2016) [1]	Using a qualitative approach administrated through semi-structured interviews in South Korea, the authors find that (1) younger generations (born after the 1980s) think of dependence on smartphones as an inevitable phenomenon because the devices are the main means for communication and are very convenient for people, but that (2) older generations attribute causes of smartphone addiction to users' intrinsic characteristics.
	(Ng 2012) [53]	Using a survey approach administrated in Australia, the author shows that undergraduate students were generally able to use unfamiliar technologies easily in their learning to create useful artifacts. However, to use them for meaningful purposes, they first had to be provided the opportunity to do so.
Questioning the concept of generations based on IT-related behaviors	(Thompson 2013) [68]	Using a survey approach administrated among freshmen students in the U.S., the author explored the degree to which students identified with generational claims about their approaches to learning and with the productivity of these approaches. While the findings suggested some positive correlations between the use of digital technology and the characteristics ascribed in the popular press to digital native learners, it also finds negative correlations between some categories of technology usage and the productivity of student learning behaviors.
	(Margaryan et al. 2011) [47]	Using a mixed-method approach (survey and interviews) with undergraduate students in two U.K. universities, the authors find that engagement in technology does not depend on age because older populations can gain expertise in using a technology while younger generations may achieve only limited knowledge in its use.
	(Jones et al. 2010) [34]	Using a survey approach administrated in five U.K. universities, the authors suggest that while strong age-related variations exist in the use of new technology among the sample, it is not possible to describe first-year students born after 1983 as a single generation. The authors emphasize that those students are not homogenous in their use and appreciation of new technologies.
	(Helsper and Eynon 2010) [28]	Using a survey approach administrated in the U.K., the authors find that adults can become digital natives, especially in the area of learning, by acquiring skills and experience in interacting with information and communication technologies.
	(Guo et al. 2008) [27]	Using a survey approach administrated in the U.S. and Canada, the authors show a non-significant difference in IT competence among different age groups. The authors conclude that the digital divide between "native" and "immigrant" users may be misleading, and may distract researchers from carefully considering the diversity among IT users and the nuances of their IT competencies.

Appendix 2: Data Collection

Step 1. The lead author initiated the interviews by asking the following questions:

1. What is your age?
2. How long have you been working in the software development profession?
3. How long have you been working at [the company]?
4. When was the first time you started using the Internet? Any particular website?
5. When did you first begin using social networking? Any particular website?

Note: We gave the interviewees the following: (1) A brief explanation of what we mean by *social networking*—that is, Internet-based applications that enable users to create and exchange user-generated content. (2) Examples of major early and recent social networking applications, including SixDegrees, Classmates.com, LiveJournal, Friendster, LinkedIn, MySpace, Flickr, Orkut, MSN Spaces, Yahoo360, YouTube, Twitter, and Facebook (Boyd et al. 2007; Yu et al. 2010).

Note: The lead researcher conducted the first set of interviews by asking the questions listed below. The process of asking the questions, however, was not mechanical. Instead, the interviewer reflected on the responses and asked additional questions to encourage the interviewees to explain their early background of engagement with social networking sites. For example, if a developer said that s/he began using social networking technology at the age of 10, the interviewer inquired about the websites s/he had used, how s/he had initially learned about those websites, and why s/he had begun using those websites from an early age. Such follow-ups were instrumental in ensuring that our research builds on solid data to categorize developers.

Step 2. The lead researcher conducted the interviews by asking participants for their opinions on the following questions:

1. What are the priorities that warrant the most significant consideration during software development?
2. What do you think the goals of software development are?
3. What do you expect from successful software development?
4. Which development objectives are considered to be the most important?

Note: Once again, the interviewer reflected on both the interviewees' responses and their ability to elaborate on the subject. The interview process thus led the interviewer to ask probing questions to elicit further information about interviewee responses. For example, the researcher asked a developer who had emphasized social impact and development contributions to explain further by asking: "What do you mean by social impact?" and "Can you provide examples of contributions to the development community?"

Appendix 3: Empirical Data

Group		Name	Years of SD Experience	Birth		Social Networking Experience		
				Year	Age*	(First time, regular use)		
						Year	Age	Site
1	Group 1	Simone	1	1993	22	2003	10	MySpace (2003)
2		Alex	2	1993	22	2004	11	Friendster (2004)
3		Anne	3	1992	23	2003	11	Friendster (2004)
4		Jason	2	1993	22	2004	11	Orkut (2004)
5		Daniel	3	1993	22	2004	11	Friendster (2004)
6		Nilu	1	1992	23	2003	11	MySpace (2003)
7		Sam	2	1994	21	2004	10	Orkut (2004)
8		Hedi	1	1995	20	2004	9	MySpace (2003)
9		Bob	1	1994	21	2005	11	MySpace (2003)
10		Kate	1	1993	22	2003	10	Friendster (2004)
11		Arash	1	1993	22	2003	10	MySpace (2003)
12		Zack	1	1992	23	2003	11	Friendster (2004)
13		Randy	1	1994	21	2004	10	Friendster (2004)
14		Fayette	1	1993	22	2004	11	Friendster (2004)
15		Nik	1	1992	23	2003	11	MySpace (2003)
16	Group 2a	Veronica	1	1993	22	2007	14	Facebook (2007)
17		Jonny	2	1993	22	2007	14	Facebook (2007)
18		Nina	2	1992	23	2006	14	Friendster (2004)
19		Sean	3	1991	24	2006	15	Friendster (2004)
20		Sarah	3	1991	24	2005	14	MySpace (2003)
21		Luc	2	1991	24	2005	14	Orkut (2004)
22		Steve	1	1992	23	2006	14	Orkut (2004)
23		Jonathan	0	1993	22	2007	14	Facebook (2007)
23		Mark	1	1992	23	2006	14	MySpace (2003)
24		Robert	1	1992	23	2007	15	Facebook (2007)
25		Matthew	1	1994	21	2008	14	Facebook (2007)
26		Richard	1	1993	22	2007	14	Facebook (2007)
27		Scott	2	1991	24	2007	16	Facebook (2007)
28		Ann	1	1993	22	2007	14	Facebook (2007)
29		Nathan	0	1994	21	2008	14	Facebook (2007)
30		Jane	1	1993	22	2007	14	Facebook (2007)
32	Group 2b	Aaron	10	1981	34	2003	22	MySpace (2003)
33		Vincent	13	1980	35	2003	23	MySpace (2003)
34		Brian	13	1978	37	2003	25	Friendster (2004)
35		Jackie	14	1980	35	2002	22	Friendster (2004)
36		Mina	13	1983	32	2004	21	Orkut (2004)
37		Aiden	13	1978	37	2003	25	MySpace (2003)
38		Andrew	14	1979	36	2004	25	MySpace (2003)
39		Tom	14	1978	37	2003	25	MySpace (2003)
40		Bobby	10	1983	32	2004	21	Orkut (2004)
41		Shane	11	1982	33	2004	22	Orkut (2004)
42		Daren	14	1980	35	2003	23	Friendster (2004)
43		Meghan	14	1980	35	2003	23	Friendster (2004)
44		Liam	11	1981	34	2003	22	Friendster (2004)
45		Charlie	13	1981	34	2004	23	Orkut (2004)
46		Manuel	9	1983	32	2003	20	Friendster (2004)
47		Helen	11	1982	33	2003	21	MySpace (2003)
48		Jay	14	1979	36	2002	23	MySpace (2003)
49		Inez	12	1980	35	2003	23	Friendster (2004)

*The data about age was collected in 2015.

Appendix 4: Methodological Process

Table A. Processing Data for Causal Maps

Sample Quotes	Concepts	Link
"Users must be happy that the software does what they pay for [user satisfaction] because meeting the demands of the target audience is a big tick against development goals [effective software development]."	1. User satisfaction 2. Effective software development	User satisfaction→Effective software development
"The final software should help alleviate the pain points of users in some way [user impact]. If we can make something that people can use to help them with one of their tasks, we have created something great [effective software development]."	1. User impact 2. Effective software development	User impact→Effective software development
"Team members must take pride in building a product together over this long-term [team satisfaction]. That is the only way to build high-quality products that the team will be proud to stand behind [effective software development].	1. Team satisfaction 2. Effective software development	Team satisfaction→Effective software development
"By encouraging developers to experiment with and use the new competences [development dynamism], software organizations can release true open source projects and even make code contributions to nonprofit software foundations [development contributions]."	1. Development dynamism 2. Development contributions	Development dynamism→Development contributions

Table B. Constructing Causal Maps

Measure	Description	Examples
Density	*Density* indicates how interconnected the concepts of a causal map are; maps with high density indicate well-understood phenomenon, while those with low density indicate less understood phenomenon. We calculated the density of each map by dividing the number of links the interviewees expressed by the total number of the concepts in that map.	Measuring the density of the Group 1 map (Figure 2): The map has 10 concepts; our analysis identified 236 statements that articulate the interrelation between those concepts. Hence, we calculated the density of the Group 1 map as 236/10 = 23.6.
Centrality	*Centrality* measures show us how central a concept is to each map. Centrality numbers for each expectation concept are provided as numbers within each concept's circle. We calculate a concept's centrality by dividing the number of direct linkages involving that concept by the total number of linkages in the map.	Measuring the centrality of development dynamism in the Group 1 map: 36 out of 236 links expressed by the interviewees involve the concept of development dynamism (36/236 = 0.15). Measuring the centrality of user impact in the Group 1 map: 49 out of 236 links expressed by the interviewees involve the concept of user impact, making user impact the map's most central concept (49/236 = 0.21). Measuring the centrality of development dynamism in the Group 2b map (Figure 4): 66 out of 365 links expressed by the interviewees involve the user impact concept, making it the map's most central concept (66/365 = 0.18).
Reachability	*Reachability* measures help us understand the strength of the relationship between the concepts in the map. We calculate reachability as the strength of the direct and indirect linkages of one concept to another concept. The reachability number for each link is noted on that link in the map. We also use **bold** arrows in the maps to highlight the map's strongest linkages (highest reachability numbers); an example here is the link between user impact and effective software development in the Group 1 map (see Figure 2).	Measuring the reachability between development dynamism and effective software development in the Group 1 map: Analysis of the statements made by developers in Group 1 led us to identify 236 statements that articulate the interrelation between different concepts in the map. As shown, development dynamism can influence effective software development both directly and indirectly through development productivity, user satisfaction, and development contributions. We extracted 19 statements that articulate the importance of development dynamism for achieving effective software development. Similarly, 6, 5, and 6 statements articulated the impact of development productivity, team satisfaction, and development contributions, respectively, on effective software development. We thus calculated the reachability measure between development dynamism and effective software development as the sum of the number of the statements above (19 + 6 + 5 + 6 = 36) to the total number of all the expressed linkages (236). So, given that 36/236 = 0.15, we note "0.15" on the link between development dynamism and effective software development.