People-driven Process-enabled Software Development: A 21st Century Imperative

Azad M. Madni, USC and Intelligent Systems Technology

Abstract. In the 21st century, software will continue to "grow" in a sociotechnical ecosystem comprising customers, end users, developers, maintainers, testers, and other stakeholders. Their continued participation is crucial to software acceptance both in the DOD and the commercial sector. In the recent past, software has been a process-driven product. However, with increasing software complexity, it is becoming apparent that the people aspect of software deserves greater attention and emphasis. The people aspect comprises people decisions, personnel skillset, training, motivation, creativity, and talent. This paper explores the shift from process-driven to people-driven, process-enabled software development. The key enablers to accomplish this shift are also discussed. This paper concludes with a reminder that while people are becoming increasingly important in software development, process will continue to be a key enabler.

Introduction

Software has been a process-driven product for the last few decades. This view has inadvertently de-emphasized the importance of people in the software lifecycle [1]. The reality today is that:

People with appropriate training perform software-related activities, often subject to governing standards and legacy constraints within development environments to achieve desired outcomes.

Today with ever-increasing software sophistication, human ingenuity is being challenged like never before. No longer does it suffice to just follow a disciplined development process because people are becoming increasingly crucial in performing trade-off analysis and in creating a satisfying user experience [1-2]. In addition, people are key to ensuring that software performance, quality attributes, schedule and cost objectives are being met. Exclusive focus on software process can potentially stifle human creativity and inhibit human contributions throughout the software lifecycle. Furthermore, as software continues to grow in complexity and humans continue to become more and more an integral part of software-based systems, predictable software behavior is becoming crucial to software system safety [3].

Process-Driven	People-Driven, Process Enabled
Process flows	Technical stories
Process enforcement	Process guidance
Process prescription	Software patterns
Process integration	People collaboration
Process recipe	Human creativity/innovation
Disciplinary focus	Transdisciplinary perspective
Process knowledge	Human imagination
Process discipline	People passion

Table 1: From Process-Driven to People-Driven Process-enabled Development

Today the proportion of software in systems continues to increase dramatically. This recognition has led to the creation of the term "software-intensive systems." And people contribute in a variety of ways to software-intensive systems. For example, humans create new paradigms, explore the software design tradespace, discover patterns and trends, provide decision rationale, attempt to explain anomalous behavior, and assure smooth integration of people and software. Yet, the importance of people in the software lifecycle continues to be underemphasized. This is surprising in that software is largely a people creation that is maintained, supported, and adapted by people. People are also responsible for software quality, and yet scant attention is devoted to the talent, training, creativity and motivation of people responsible for assuring software quality [4]. Clearly, process will always play an important role, but more as an enabler than a driver. This paper argues that to achieve dramatic advances in software quality, the people dimension needs to become a central focus with process as an enabler. After all, software innovation is primarily the result of human creativity, passion and motivation. While process will continue to play an important role in the software life cycle and provide context for collaboration, the process perspective will be a necessary and valuable adjunct to the people perspective as software continues to increase in complexity [5]. People-driven software spans the 5 P's: people, purpose, passion, patterns, perspectives, and processes. Table 1 presents the key elements underlying the shift in mindset from process-driven to people-driven, process-enabled software development.

There are several compelling reasons to make people the primary focus in software development today (Table 1). First, software is a creation of people, and quite frequently for the use of people. Exclusive focus on process can stifle creativity, and compromise user acceptance. Second, safety is becoming an increasingly important consideration in software-intensive systems. Safety subsumes predictable software behavior in the face of disruptive events [3]. It is important to note that processes do not automatically address safety concerns. It is people who introduce safety concerns in the software life cycle. Third, with the need for adaptive processes (e.g., agile), and the need for adaptable systems (to survive and operate in changing operational environments), the shift toward people-driven development is becoming inevitable [6-8]. Finally, with the advent of multi-domain software that cuts across multiple domains (e.g., electrical, optical, mechanical) and multiple disciplines (e.g., physics, social sciences, cognitive science), software complexity has increased dramatically. Collectively, these trends speak to the need for people-driven, process-enabled software development and use (Figure 1).

Figure 2 presents a notional graph illustrating the approximate relationships between process importance and software complexity, and between people importance and software complexity. As shown in this figure, as software complexity increases, software development becomes less and less process-driven, and more and more people-driven, albeit process-enabled. A key implication of this trend is that if the developing organization expects software to grow in scale and complexity, the organization is better off adopting people-driven, process-enabled software development practices [1,3,4,9].

=In the recent past, several developments have collectively pointed to a much needed shift from process-driven to people-driven software development. First and foremost, is the uncertainty about the operational environment, rate of maturation of promising technologies, and personnel turbulence resulting from retirements, layoffs and personnel moves. Second, software is becoming increasingly more complex because of ever-increasing scale, and ever-growing need for adaptability in light of the changing roles of humans in relation to software. These trends are being driven by the need for systems to be long-lived and capable of coping with unknown operational environments. Third, organizations are increasingly turning to adaptive processes such as agile development paradigm, which is increasingly being viewed as a source of competitive advantage when applied correctly. It requires an accomplished team of developers, effective leadership in pulling the team together, and a change in mindset associated with traditional process-driven development in which roles are important but individual people are viewed as interchangeable/substitutable parts, with people availability trumping people skillset [5,9,10].

Alistair Cockburn, in his book “Characterizing People as Non-linear, First Order Components in Software Development” argues that predictable processes require components with predictable behavior. And, people are anything but predictable. Treating humans as interchangeable components or replaceable parts in software development is a misjudgment. Human behavior tends to be variable and nonlinear. Humans exhibit an uncanny ability to succeed in novel ways, while also exhibiting a disconcerting capacity to fail in unimagined ways. It is the failure to account for these factors in software development that inevitably result in schedule and cost over-runs. In fact, it is fair to say that humans strongly figure in both project successes and failures [3].

Unfortunately, the mistaken belief that people are interchangeable resources is deeply ingrained in business thinking. It dates back to Frederick Taylor’s Scientific Management approach for performing repetitive tasks such as running a factory [11]. However, for highly creative work such as software development, this view is clearly inapplicable. And today, with the advent of smart manufacturing, manufacturing also no longer abides by this tenet. Another key tenet of Taylor’s theory is that the people doing the work are not best-suited to determining how best to do the work. While this tenet may hold, to a degree, on the factory floor, it is untrue of software development. In fact, people attracted to software engineering tend to be the best and the brightest, with the culture of youth pervading the field [3, 11].

So, what is it that people bring to software? People bring imagination, novel insights, storytelling ability, and an uncanny ability to discern and exploit patterns [2, 4]. These capabilities have the potential to transform software development in unprecedented ways to achieve dramatic improvement in software quality, responsiveness, cycle times, and life cycle costs. Some of the unique human capabilities that bear on software quality and costs are presented in Table 2.

A people-driven, process-enabled view of software goes well beyond the process perspective. It is sensitive to business concerns and constraints, implications of software-related decisions on short-term, mid-term, and long-term concerns of a program or business. It is cognizant of the available skillset in both management and development teams. It shows understanding of programmatic and technical trade-offs, and the importance of collaboration and full stakeholder participation in the software lifecycle. The latter is essential for reasoned compromise that addresses stakeholders’ concerns and resolves issues. It is also essential for stakeholder acceptance of collaboratively made decisions, and elimination of extraneous design iterations and rework [1].

The people-driven view of software is especially sensitive to the required skillset and available expertise when it comes to the selection of the software development process (e.g., spiral, waterfall, evolutionary prototyping, incremental commitment) [3]. With a people perspective, software development process selection is not based just on problem particulars (i.e., objectives, schedule, budget,

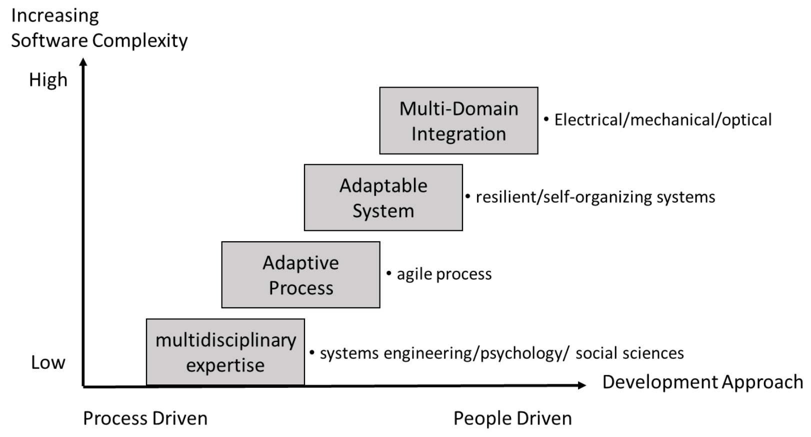

Increasing Software Complexity
High
Low
Multi-Domain Integration
• Electrical/mechanical/optical
Adaptable System
• resilient/self-organizing systems
Adaptive Process
• agile process
multidisciplinary expertise
• systems engineering/psychology/ social sciences
Development Approach
Process Driven
People Driven

Figure 1: Developments Contributing to Increasing Software Complexity

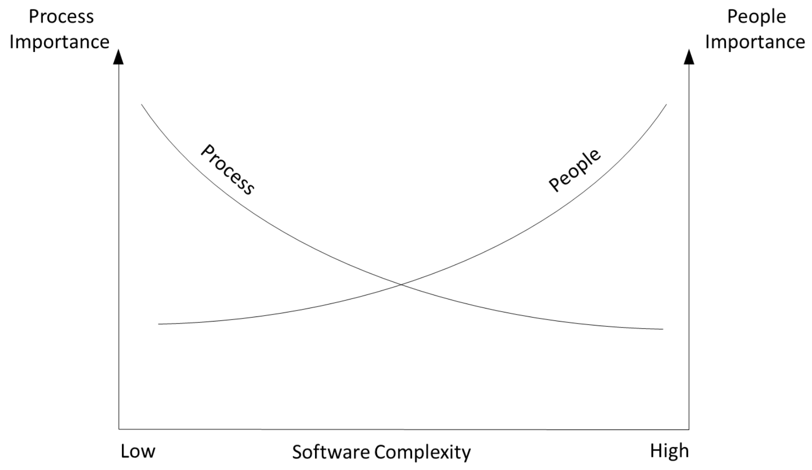


Figure 2: Increasing Software Complexity Driving Paradigm Shift

• Systems Thinking	*Think holistically* to understand “big picture,” relationships, and interdependencies
• Associative Thinking	Exploit metaphors and analogies to simplify software architectures, and circumvent constraints
• Storytelling	*Engage all stakeholders* in upfront software engineering to ensure their timely participation, contributions, and acceptance
• Visual Analysis	*Discern patterns* and trends that can be exploited in software simplification and implementation
• Abstractions	*Abstract details* to develop a mental representation that informs development of scalable and extensible software
• Tradeoff Analysis	*Place right emphasis* on conflicting objectives to create responsive software that meets stakeholder needs while satisfying schedule, budget, technical, and legacy constraints

Table 2. Unique Human Capabilities that Bear on Software Quality

risks) but also availability (or lack thereof) of the required talent and skillset in the development team [12]. The maturity and the experience of the team members and leadership play a pivotal role in defining use cases, specifying architecture, and developing the right set of abstractions.

From Process-Driven to People-Driven, Process-Enabled Development

People-driven development is more than stakeholders influencing and agreeing on what is being created. It is more than empowering engineering teams and the activities they perform to develop software. And it is more than directing software users in the use of software. It is in fact all of the above. People-driven development means humans playing an active role in software-related trade-offs, designing the software, managing the software development process, and even distributing software development activities to the development team members. People-driven development is also influenced by culture and power distance [13]. Compounding the problem is the "clash of values" between developers and program managers [14]. And, of course, human behavior exhibits nonlinearity and variability [2, 15]. These factors influence both the development process and the software product. Cockburn [15] and Madni [2, 4] identify specific factors that influence the outcome: humans are social beings who perform best in face-to-face collaboration; human are inconsistent and inconsistency shows up over time; humans exhibit variability from day-to-day and place-to-place; human generally want to do the right thing for their organizations.

These characteristics bear directly on process. It is important to recognize that process enforcement can vary from strict to loose. In light of human characteristics and ever-growing system complexity, loose process enforcement is preferable to strict enforcement. In cases, where strict process enforcement is required, there is a need for performance support for humans to behave consistently.

Software lifecycle processes provide a structured disciplined means to guide the development of complex, real world software [16]. This software spans: primary processes (acquisition, supply, development, operation, maintenance); supporting processes (documentation, configuration management, quality assurance, reviews and audits, problem resolution); and organizational processes (management, infrastructure, maintenance, improvement, training). The question that needs to be asked in where do lifecycle processes benefit software design and where do they become an impediment. For most support and organizational processes, following software life cycle process is a benefit. Also, periodic architecture and design reviews help to ensure design quality, and traceability between requirements and design elements to ensure design completeness. However, there are times where strict process enforcement becomes a hindrance to creativity and innovation [17]. In this case, humans can "dial back" on strict process enforcement and adopt loose process enforcement. This shift puts people in charge of the process, making it people-driven, process enabled software. This recognition is at the heart of adaptive software development in general, and agile development in particular.

Agile processes (or agile, for short) are a prime example of people-driven, adaptive development. Agile relies on process acceptance by the development team, not process imposition by management [6-8, 12]. In other words, only developers themselves can choose to follow an adaptive process. This is especially true of extreme programming (XP), which requires disciplined execution, with developers making all the decisions and generating all time estimates. This is a huge cultural shift for management in that it requires sharing of responsibility between developers and management [12].

Measuring software productivity is a challenge with adaptive processes. In this regard, Robert Austin distinguishes between measurement-based and delegatory management in software development. Measurement-based management is best suited to repetitive work with minimal knowledge requirements and easily measured outputs. For software development, the delegatory style of management is appropriate. Delegatory management calls for developers to decide how to do the work. In fact, this approach is central to the agile philosophy. This does not mean that developers have to do it all. In fact, developers rely on management for guidance when it comes to business needs. Finally, in adaptive development, change is an expected and frequent occurrence. Consequently, people need to be kept apprised as they continually adapt the process to fit changing needs [18].

Recent Trends

Several recent developments make a people-driven view of software both attractive and eminently viable. Three of the more compelling advances that bear on a people-driven view are: Model-Based Engineering, Experiential Design and Visual Analytics; and Interactive Technical Storytelling in Virtual Worlds [1,9]. Each is discussed next.

Model Based Engineering transforms traditional approaches in a number of ways. First, it replaces document-centric engineering with software models at the center of the development process. The model serves as the sole source of truth, from which documents can be created on demand. Second, model-based software engineering assures consistency among the different perspectives embodied in the model. Third, model-based software engineering can provide different lenses for different stakeholders allowing them to explore the consequences of changes in assumptions, constraints, and resource/data availability.

Experiential Design and Visual Analytics is the combination and use of context-sensitive visualization interfaces and analytical reasoning methods to enable visual debugging, simplification, and redesign of both systems and processes. As importantly, visual analytics appeals to all stakeholders because it transforms calculation results into easy-to-assimilate visuals, patterns and trends [9]

Interactive Technical Storytelling in Virtual Worlds is a means to engage all stakeholders in upfront engineering to ensure that the inputs and concerns of all stakeholders are known and addressed when conducting trade-off analysis [1]. By providing each stakeholder with an appropriate "lens" into story execution, meaningful inputs and concerns from all stakeholders can be elicited and resolved through timely, multi-stakeholder collaboration [9]. Virtual Worlds are simulated environments within which stories unfold and stakeholders explore the consequence of "what-if" assumptions, decisions and tradeoffs. By providing appropriate "lenses" for the different stakeholders, instrumented virtual world offers a convenient means for knowledge acquisition, data collection, and uncovering surprising behaviors.

What it Takes

To realize this shift in mindset, requires advances on several fronts: a) a persuasive value proposition of people-driven development; b) demonstration of how people-driven software delivers superior value than process-driven software; and c) a risk-mitigated, staged process to gradually make the transition from process-driven to people-driven development.

Value Proposition: Software development is a collaborative process that involves multiple stakeholders who need to jointly explore tradeoffs and reach consensus. Expanding stakeholder participation is critical. An experiential, stakeholder-oriented interface [9] is the key to ensuring full stakeholder participation early and throughout software development.

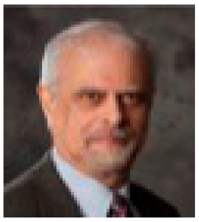
Demonstration of the value proposition: The demonstration should highlight the key elements of a people-centric approach to developing high quality software. The approach should highlight: an experiential interface with "lenses" for various stakeholders; an interactive storytelling capability to engage the various stakeholders from their respective perspectives; an instrumented virtual world that supports story execution and that can be collaboratively and individually explored by the different stakeholders under a variety of "what-if" assumptions, parameter values, and technical and programmatic tradeoffs [9].

Staged transition: The transition from a process-driven view of software development to a people-driven, process-enabled view has to be accomplished in stages. It is a cultural change for both management and developers. In this regard, the first stage is the transition from traditional use cases to stories. The second stage is the introduction of storytelling in virtual worlds with stakeholder-oriented "lenses" that allow stakeholders to explore the software tradespace and understand the CONOPS when the stories execute in the virtual world. The third stage is story-enabled collaborative trade-off studies supported by sensitivity analysis and comparative evaluation.

Conclusion

Software has been a process-driven discipline for quite some time. While "process" will continue to be a key enabler of software development, the people aspect will continue to gain in importance as software continues to grow in scale and complexity requiring greater human involvement. Surprisingly, the growing importance of people in software development has not produced a paradigm shift in software development. And yet it is people that bring ingenuity, imagination, and creativity that can dramatically improve software quality and development efficiency and effectiveness. This paper emphasizes the importance of people-driven, process-enabled software development. Additionally, in light of growing emphases on people, four significant advances are identified as key enablers of this transformation: model-driven engineering, experiential interfaces, visual analytics, and interactive storytelling in virtual worlds. Looking down the line, as various relevant technologies mature, software quality and development will increasingly depend on the "people factor," with process continuing to be an important enabler, but not the sole driver.

ABOUT THE AUTHOR



Dr. Azad Madni is a Professor and Director of the Systems Architecting and Engineering Program in the Viterbi School of Engineering at the University of Southern California. He is also the founder and Chief Scientist of Intelligent Systems Technology, Inc. He received his B.S., M.S., and Ph.D. degrees in engineering from UCLA. His government research sponsors include DOD, DARPA, AFRL, AFOSR, ONR, NAVAIR, NAVSEA, SPAWAR, ARL, ARI, RDECOM, DHS S&T, DTRA, NIST, DoE, and NASA. His research has also been sponsored by commercial companies including Boeing, Northrop Grumman, Raytheon, Hughes, Orincon, and General Motors. He is an elected Fellow of AAAS, AIAA, IEEE, INCOSE, SDPS, and IETE. He is listed in the major Marquis' Who's Who, including Who's Who in America.

Phone: 213-740-9211

E-mail: azad.madni@usc.edu

REFERENCES

1. Madni, A.M. "Expanding Stakeholder Participation in Upfront Systems Engineering", System Engineering, 2015
2. Madni, A.M. "Integrating Humans With and Within Software and Systems: Challenges and Opportunities," (Invited Paper) CrossTalk, The Journal of Defense Software Engineering, May/June 2011, "People Solutions."
3. Cockburn, A. "People and Methodologies in Software Development," Ph.D. Dissertation, University of Oslo Press, University of Oslo, February 2003.
4. Madni, A.M. "Integrating Humans with Software and Systems: Technical Challenges and a Research Agenda," Systems Engineering, Vol. 13, No. 3, pp. 232-245, Autumn (Fall) 2010.
5. Madni, A.M. "Thriving on Change through Process Support: The Evolution of the ProcessEdge™ Enterprise Suite and TeamEdge™," International Journal on Information - Knowledge - Systems Management, Special Issue Vol. 2, No. 1, pp. 7-32, 2000.
6. Cockburn, A. "Agile Software Development: The Cooperative Game, 2nd Edition, Addison-Wesley Professional, October 2006.
7. Madni, Azad M. "AgileTecting™: A principled approach to introducing agility in systems engineering and product development enterprises." JIDPS 12.4 (2008): 49-55.
8. Madni, A. M. "Agile systems architecting: Placing agility where it counts." Conference on Systems Engineering Research (CSER). 2008.
9. Madni, A.M., Spraragen, M., and Madni, C.C., "Exploring and Assessing Complex System Behavior through Model-Driven Storytelling," IEEE Systems, Man and Cybernetics International Conference, special session "Frontiers of Model Based Systems Engineering", San Diego, CA, Oct 5-8, 2014.
10. Austin, R.D. Measuring and Managing Performance in Organizations, Dorsett House Publishing, 1996.
11. Taylor, F.W. "The principles of Scientific Management, New York and London, Harper and Brothers, 1911.
12. Taylor, F. W. "Shop Management," New York and London, Harper and Brothers, 1903
13. Geert, H., and Hofstede, G."Cultures and organizations: Software of the mind." McGraw-Hill, New York (1991).
14. Cockburn, A. "Software development as Community Poetry Writing. Cognitive, cultural, sociological, human aspects of software development." Annual Meeting of the Central Ohio Chapter of the ACM, 1997.
15. Cockburn, A. "Characterizing people as non-linear, first-order components in software development." International Conference on Software Engineering 2000. 1999.
16. Suryanarayana, G., Sharma, T., Samarthyam, G. "Software Process versus Design Quality: Tug of War", IEEE Computing Edge, Aug 2015.
17. Madni, C.C., and Madni, A.M., "Web-enabled collaborative design process management: application to multichip module design." Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on. Vol. 3. IEEE, 1998.
18. Bramble, P. "Patterns for Effective Use Cases, Addison-Wesley Professional, August 2002