Identifying and Addressing Risks in the Early Design of a Sociotechnical System through Premortem

Briana Bettin, Kelly S. Steelman, Charles Wallace, Dana Pontious, Elizabeth S. Veinott;
Michigan Technological University
bcbettin@mtu.edu, steelman@mtu.edu, wallace@mtu.edu, dpontiou@mtu.edu, eveinott@mtu.edu

Anticipating risks in software development is always challenging, but particularly so when the software application is part of a novel sociotechnical system with various human and physical components. Our interdisciplinary team of software engineering and human factors researchers is designing such a system. In order to identify and mitigate the risks latent in this previously unexplored space, we have used the premortem method at an early stage in system design. In the premortem, the team ideated failure scenarios across the range of system use, then collaborated on ways to eliminate, mitigate, or monitor the risks of these failures. We have found the premortem method valuable in recognizing and mitigating previously unanticipated risks and in enriching team communication.

INTRODUCTION

As sociotechnical problems and solutions become increasingly pervasive and complex, human-centered design has become more important than ever to the success of innovations and the well-being of those affected by them. Cross-collaboration, or codesign, between user experience or human factors designers and software developers utilizes the diversity of knowledge from the actors in respective disciplines to create innovative designs and products. Difficulties can arise, however, when there is a lack of shared understanding between the domain experts in codesign teams. Kleinsmann (2006) describes shared understanding as individuals having similar perceptions about design concept and process. Barriers to shared understanding stem from actors on the teams having different background knowledge, levels of experience, and understanding of goals (Dougherty, 1992). These differences cause gaps in communication and understanding that can be difficult to bridge. Dong et al. (2013) have found that high quality mental models are associated with high quality enactment of a design, but differences among individual mental models and the team mental model leads to relevant information not being considered. Interdisciplinary teams need to ensure shared understanding and high-quality mental models in order to have successful collaboration. Pirinen (2016) finds that when designing services, actors on a team find common ground regarding communication and project goals and translate these findings into client needs. This is a foundational aspect of human-centered design among interdisciplinary teams.

Shared understanding among an interdisciplinary team during early design, when basic system architecture is being laid out, is instrumental for codesign. Committing time to creating shared understanding can reap benefits for a design team by reducing the time spent later on reworking (Kleinsmann *et al.*, 2010). But capturing the shared understanding of a team can be as difficult as creating that understanding. Dong *et al.* (2013) suggest a method to evaluate mental models, but it cannot be done in a short period of time and requires good team communication. This method also does not consider what is causing mental models to fail or be successful. Likewise, Kleinsmann & Valkenburg (2008) used the learning history method to analyze the design process, but this can only happen after the design process has commenced (Roth & Kleiner, 2000). Pirinen (2016) suggests the use of early prototypes and flexible design iterations, but this assumes prior shared understanding among team members at the time of planning.

Engaging the entire team broadly in risk identification is important for a number of reasons. When contemplating risks, team members may gravitate to their areas of specialization—software developers to issues with technical implementation or deployment, human factors experts to unexpected ways in which human participants may use the system. But there is also a risk that by focusing on a single component of the system, experts may develop unstated assumptions about that component—for instance, a software developer charged with implementing the "happy path" through a use case may lose sight of ways in which external circumstances could divert a user from that path. Finally, some failure scenarios may involve both technical and human factors and hence may not be apparent to specialists in any single area.

PRACTICE INNOVATION

Our interdisciplinary team is developing a sociotechnical system called *Illuminated Devices* with the goal of providing technology training to individuals with little to no computer experience in rural communities. The system's components include a portal device and a social subsystem comprising the human tutors, community, procedures, and tasks associated with the established community-based tutoring program. The portal is a lightweight application that connects a user via video to a human tutor. **Figure 1** describes the key actors within this sociotechnical system and their relationships to each other.

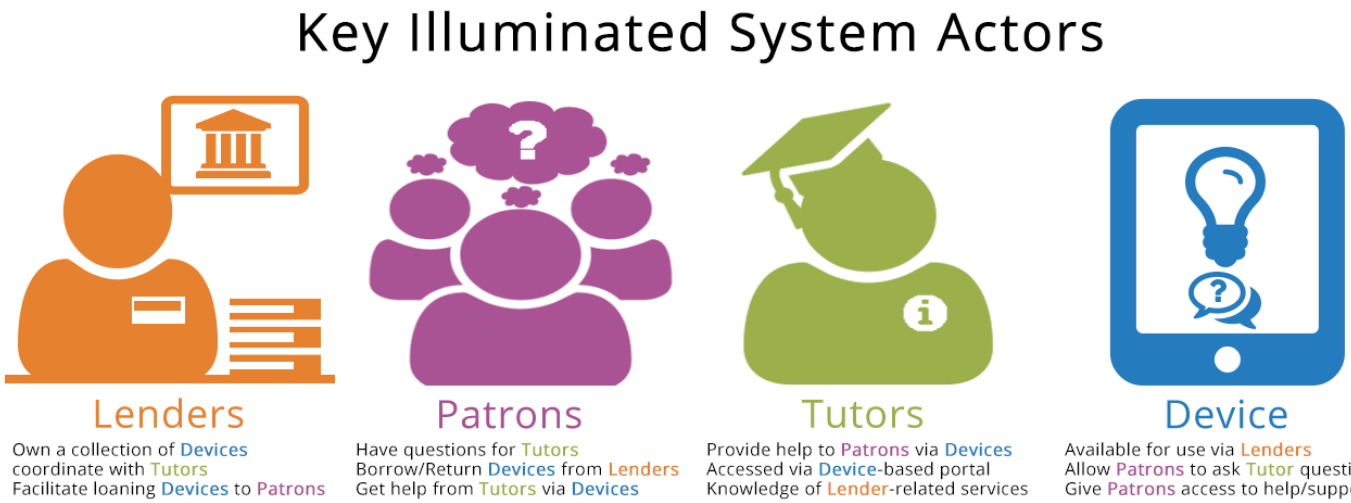

Figure 1. Key Illuminated System actors (Lenders, Patrons, Tutors, Device) and their relationships to other actors within the system are identified.

An important design consideration for the portal is minimizing barriers to access; the path to tutoring support must be available and understandable even to digital first-timers. Portals will be available for checkout at local libraries and other community organizations, with staff members acting as caretakers of the devices and guides for patrons of the system.

The development and human centered design (HCD) teams were both tasked with designing considerations or approaches for the back end design of our sociotechnical system. In viewing the design documents of different teams, we realized that focuses differed largely across team member types. This required considering approaches to synergize communicating concerns. Our team needed a method for re-scoping and redefining how we viewed the problem space so that everyone could communicate from a common playing field. We recognized that both the software developers and HCD team appeared to be considering disjoint areas of risk within the system through their team conversations. To bring all team member's ideas forward together, as well as generate new ones, we investigated ways to design a risk management discussion.

Our team employed the *premortem* method (Klein, 2007) to develop a broad shared view of system risk across all six team members. The premortem method is a planning evaluation technique where team members critique the current plan under the assumption of project failure. The premortem uses prospective hindsight, which reduces uncertainty and helps facilitate team brainstorming (Mitchell *et al.*, 1989). Prospective hindsight saves the team from spending time arguing about the likelihood of failure, but instead creates an environment that values critical judgment of the current plan based on its failure. Premortems allow for independent viewpoints from all team members as well as an environment to create novel critiques (Klein, 2007; Veinott et al., 2010). Our team decided to use the premortem technique to evaluate our design plan to improve the quality of the design produced. The method is also fast to complete, low cost, ensures the participation of the entire team, and can be performed early on in the design process. It has been demonstrated to lead to more actionable ideas compared to other evaluation methods (Veinott *et al.*, 2010; Peabody & Veinott, 2017).

The HCD team members planned the premortem activity prior to its execution, and identified two key goals in using the premortem method:

- To identify and mitigate risks before product development
- To support interdisciplinary team communications

PRACTICE APPLICATION

Following identification of the goals, the HCD team members planned a premortem meeting schedule based on the expected process design (Klein, 2021).

As our team was designing a novel sociotechnical system with several components, we also wanted to leverage the premortem as an opportunity to ideate on failure across core system segments. Premortems typically would focus on the entire design and let key failure points emerge, but for this activity we focused on key components of the design instead by dividing our project into three key segments of the systems execution:

- Checking out the device
- Device session
- Returning the device

Elizabeth S. Veinott, not on the design team, facilitated the premortem. The entire team completed two back-to-back premortems which is unusual and focused on different aspects of the design. The first premortem centered on the plan for the "device session", during which the portal device is used by a patron to access tutoring services virtually. The second

premortem encompassed the checking out and return of the device, centering the larger system of the device obtainment lifecycle. For both, we went through the following steps:

1. Individuals brainstormed for two minutes. The team assumed the system had been built, and it catastrophically failed. The team members then individually generated reasons why it failed.
2. Next, each person went around in the circle and shared their top reason for failure. For each, we elicited a show of hands from others who also noted this potential failure. This continued until all reasons had been exhausted from all members.
3. Next, we grouped failures categorically to sort "problem areas and types" within the system.
4. Finally, we spent two minutes individually brainstorming potential solutions to the list of failures: how might we design out these failure opportunities?
5. Each person shared their top potential solution. As before, these were added to the white board, with a tally of the number of individuals who identified a similar solution. This continued until all solutions had been exhausted from all members.
6. Finally, we reviewed the lists of failures and solutions, identifying which items were novel and hadn't been considered prior to this activity.

The entire process took approximately 1.5 hours to complete.

FINDINGS

As shown in **Table 1**, the Device Session premortem generated 17 unique failures from our team, and 15 unique solutions. As shown in **Table 2**, the Device Checkout and Return premortem generated 13 unique failures and 18 unique solutions. From the two premortems, 15 solutions (bolded in the two Tables) were identified as novel ideas not previously considered by any member of the team.

Throughout the two premortem activities, our team had a number of key moments that allowed for fruitful discussion of the project. These moments created opportunities for the vision of the project's process and priority development items to grow and change through mutual team understanding.

Device premortem (Table 1):

- Failures 1, 2, and 5 revealed opportunities for failure that are largely outside of our control (e.g. video conferencing software failure, server outages, cell service interruption), but that could lead to frustration and eventual disuse by our patrons. Ideation of solutions focused on mitigation strategies, detecting these external events and informing the user in our language that something has gone wrong, that it is a temporary situation, and that it is not their fault.
- With Failure 2, we recognized the need to step far back in terms of assumptions about patrons throughout the tutoring session (e.g. will a patron even know how to plug in the device if it loses power?) We needed to consider how the design could help users who are anxious about digital technology overcome perceived "simple" failures like this.

Table 1. Output of Premortem 1: Failures and Solutions related to the device

Device Failures	*Potential Solutions*
1. No cell services	***1. Detect external problems (e.g., zoom is down) and present message to user***
2. iPad dies	
3. App doesn't work	
4. Connected to app but camera fails/ hot mic	*2. Bring in second tutor for hard problems*
5. Zoom is down	*3. Provide extra charger and instructions for charging*
6. Can't get to app	
7. Too many users in queue	*4. iPad coverage map*
8. User's problem not in database[/tutor training]	***5. Demo mode for librarians***
	6. Crash course training (tip sheet)
9. Outside of operating hours/users don't know this	***7. Manage expectations for learners, tell users that problems are "not their fault"***
10. Server down/gets joining message but never joins	***8. Talk through initial steps on the phone/call library hotline***
11. Not willing to use device without someone to help	***9. Disable user control of mic and camera***
12. Connection drops	
13. iPad is updating	***10. Override default app permissions***
14. Tutor can't solve problems	*11. Preflight checklists for librarians*
15. Break a connection at home	*12. Clear UX*
	13. AI agent to walk through when no human tutors are available
16. No tutors log on	
17. Patron breaks the device	*14. User testing with true never-evers*
	15. If Zoom outage, send message to development team

Checkout/return premortem (Table 2):

- Failures 1 and 4 raised possibilities that the patron using the device may not behave as a neutral actor within the system, but may instead act in bad faith. For example, the patron may break something on the device (whether intentionally or not) and try to hide it or cover up the damage. Further complicating the issue, patrons may "break" the device in invisible ways, e.g. knowingly or unknowingly installing malware. The team addressed these risks with a combination of risk avoidance (e.g. locking down the device, allowing only critical software on it) and mitigation (enabling tracking and remote disabling of device).
- Through Failures 5 and 7, the role of lenders within the system, and considering designing for them, became apparent. Reflecting on our experience to date, we realized that we have only worked with lender stakeholders who are heavily invested in our existing community-based tutoring program, and hence full of the confidence and motivation to make the system work. Thinking more broadly, we acknowledged the need to design for lenders who are unfamiliar with our current program, who may not share the enthusiasm of our lender colleagues, and who may not feel confident explaining digital

technology. This yielded ideas for risk mitigation such as creating resources for lenders and a demo mode for the device that lenders can use as a prop when explaining the device to patrons.

- With Failure 9, patrons losing enthusiasm or motivation became a different way of thinking about failure. The system might work in a purely functional sense, but if it is slow or doesn't meet expectations, the patrons may give up on it. This highlighted the mitigation strategy of setting expectations for patrons regarding the system and potential failures/limitations during checkout.

Table 2. Output of Premortem 2: Failures and Solutions related to the check-out and return process

Device Failures	*Potential Solutions*
1. Entire fleet of devices damaged/not returned	*1. Weekly check by research team on tutor note keeping*
2. ID doesn't scan/system down and can't check out	***2. Redesign case or Drop box to make it hard to use***
3. Library doesn't not enter user info into database	***3. Remotely disable device at end of check out period***
4. User breaks device in subtle way that is hard to detect (virus/misuse)	*4. Make resources for device available to librarian via QR code*
5. Librarian unfamiliar with ID system	***5. Wipe device upon return to eliminate personally identifying information***
6. User puts iPad in dropbox/crushed	*6. Scan device for virus*
7. Librarian makes the device sound difficult to use or provide incorrect info/not following protocol	*7. Prevent new installations/ Keep device bare bones*
8. User doesn't sign waiver/isn't qualified to check out device/doesn't have an ID (and is angry)	*8. If devices can't be scanned, add a way to connect to devices (stickers)*
9. User on waiting list/gives up/forgets	***9. Enable Find my iPad, track device, don't allow this feature to be disabled***
10. Library closes for period of time	*10. Ensure that devices remain in case and avoid damage*
11. Nobody knows about device, so they don't use it	*11. Provide (on site) help services while you wait*
12. Charger lost	*12. Create a marketing campaign to get the word out*
13. Tutor session notes never entered	***13. If library closes, push notification out to app***
	14. Force device to only use cell and not wifi
	15. Have enough devices to reduce wait time
	16. Write the waiver in simple language
	17. Add function to allow tutor to remotely clear data
	18. Add "if found, return to" label

DISCUSSION

This activity was illuminating for our team in several ways. By centering failure, our team shared a common ground in discussion which served as a bridge for communication. With failures and solutions centered on the board as well, we all were considering the same common problem set in our approaches. This facilitated discussion across the team of what is within our control and what is not.

The concept of risk management is not new within software engineering, but the application of the premortem, in this context—especially with an interdisciplinary team—appears to be. Traditional risk management in software engineering grew out of applications in safety-critical and high-hazard industries, where existing approaches to risk in mechanical, civil, and electrical engineering have been transferred to software. Failure mode and effects analysis (FMEA) is a bottom-up technique for identifying and quantifying risk, starting with elementary components and proceeding to subsystem and finally full system analysis (Reifer, 1979). Fault tree analysis (FTA) is a complementary top-down technique that breaks down system failure events into Boolean combinations of lower-level events (Ovstedal, 1991). These techniques have never found widespread use in mainstream software development, for a number of reasons. FTA can give estimates of system resilience to faults (sources of failure) but is not well suited to clearly identifying them. FMEA can provide an exhaustive catalog of component-level risks, but it is not feasible to generate such a catalog at an early architectural phase, when components are not even known; moreover, there is no straightforward way to model system-external events (Haigh, 2021).

Boehm (1988, 1991) and the Software Engineering Institute (Higuera & Haimes, 1996), both initially motivated by high-risk, safety-critical military systems, developed risk-driven processes for software development (Boehm & Ross, 1989; Boehm, 1991). The risk identification techniques in both approaches involve matching project characteristics against predefined taxonomies of risk (Boehm & Ross, 1989; Keil *et al*., 1998). These taxonomies offer entry points into discussion of project risk but may fail to take into account the particulars of a given project, especially one without well established precedents. As our application of the premortem has shown, it can exist as a complementary method that is effective in identifying particulars of even novel projects.

PRACTITIONER TAKEAWAYS

Our experience with premortem indicates that it shows promise as a risk identification technique for novel software applications operating in complex and varied contexts. The collaborative nature of the process means that team members share their perspectives openly with the full team. Discussion is open-ended and not constrained by predetermined categories or checklists—though we can imagine a variation where participants can consult lists of potential risks to supplement their own brainstorming.

- The premortem centers the discussion on ideas and gives everyone on the team an equal voice and ownership over identifying potential problems and solutions.
- Team members achieve broadened perspectives through collectively hearing and discussing sources of failure or risk.

- The activity is low cost, high yield. A short time frame and no additional overhead costs allowed for generation of important ideas and perspectives.
- The premortem crosses boundaries by encouraging shared perspective, allowing ideas to be explored that no one team member would likely have thought of.
- Premortem does not provide guarantees of completeness. For high-hazard application areas, it can serve as a complement but not a substitute for more deliberate methods. A hybrid approach that accompanied individual brainstorming on failure scenarios with considerations of predefined risk taxonomies may offer the best of both approaches.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. BCS- 2122034. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Boehm, B.W. (1988). A spiral model of software development and enhancement. *Computer* 21(5), 61-72.

Boehm, B.W. (1991). Software risk management: principles and practices. *IEEE Software* 8(1), 32–41.

Boehm, B.W. and Ross, R. (1989). Theory-W software project management principles and examples. *IEEE Transactions on Software Engineering*. 15(7), 902–916.

Dong, A., Kleinsmann, M. S., & Deken, F. (2013). Investigating design cognition in the construction and enactment of team mental models. *Design Studies*, *34*(1), 1–33. https://doi.org/10.1016/j.destud.2012.05.003

Dougherty, D. (1992). Interpretive barriers to successful product innovation in large firms. *Organization Science*, *3*(2), 179–202. https://doi.org/10.1287/orsc.3.2.179

Haigh, F.D. (2021) SW failure modes and effects analysis. NASA Software Engineering Handbook, 8.5. https://swehb.nasa.gov

Higuera, R.P. & Haimes, Y.Y. (1996). Software risk management. Tech Report CMU/SEI-96-TR-012, Software Engineering Institute.

Keil, M., Cule, P.E., Lyytinen, K., and Schmidt, R.C. (1998). A framework for identifying software project risks. *Communications of the ACM* 41(11), 76–83.

Klein, G. (2007). Performing a project premortem. *Harvard Business Review*, 85(9), 18-19.

Klein, G. (2021). The Pre-Mortem Method. https://www.psychologytoday.com/us/blog/seeing-what-others-dont/202101/the-pre-mortem-method

Kleinsmann, M. S. (2006). *Understanding Collaborative Design*. PhD Thesis. Netherlands, Delft University.

Kleinsmann, M., Buijs, J., & Valkenburg, R. (2010). Understanding the complexity of Knowledge Integration in collaborative new product development teams: A case study. *Journal of Engineering and Technology Management*, *27*(1-2), 20–32. https://doi.org/10.1016/j.jengtecman.2010.03.003

Kleinsmann, M., & Valkenburg, R. (2008). Barriers and enablers for creating shared understanding in co-design projects. *Design Studies*, *29*(4), 369–386. https://doi.org/10.1016/j.destud.2008.03.003

Mitchell, D. J., Russo, J. E., & Pennington, N. (1989). Back to the Future: Temporal Perspective in the Explanation of Events. Journal of Behavioral Decision Making, 2, 25-38

Ovstedal, E.O. (1991). Using fault tree analysis in developing reliable software. IFAC Proceedings, Volume 24(13), 77-82.

Peabody, M., & Veinott, E. (2017). Focus shift: Differences in reasons generated using Premortem and Worst Case Scenario plan evaluation methods. *Naturalistic Decision Making*, 259–261.

Pirinen, A. (2016). The Barriers and Enablers of Co-design for Services. *International Journal of Design*, *10*(3), 27–42.

Reifer, D. J. (1979). Software failure modes and effects analysis. *IEEE Transactions on reliability*, *28*(3), 247-249.

Roth, G., & Kleiner, A. (2000). *Car Launch: The Human Side of Managing Change*. Oxford University Press.

Veinott, E. S., Klein, G., & Wiggins, S. (2010). Evaluating the effectiveness of the PreMortem technique on plan confidence. Proceedings of ISCRAM, Seattle, WA.