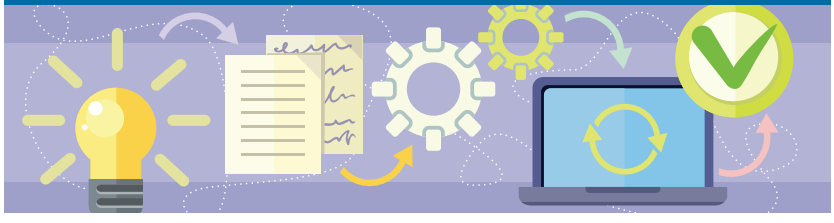
From Incident to Insight

Incident Responders and Software Innovation

Robert Biddle and Judith M. Brown, Carleton University

Steven Greenspan, CA Technologies

// Incident response teams often gain deep knowledge about the experience of using products and services. If this knowledge was made available to designers, significant opportunities for innovation would emerge. //



OVER THE LAST decade, new software processes have appeared that emphasize collaboration among people involved in creating successful software. For example, agile methods stress collaboration between development teams and business clients,[1] and DevOps emphasizes better collaboration between development teams and deployment teams.[2,3]

Digital Object Identifier 10.1109/MS.2017.442103917
Date of publication: 8 January 2019

In our recent work, we have been studying operations centers, with a focus on incident response (IR). We were interested in how IR works in practice and how it might be better supported. We studied seven operations centers and reported our findings in detail elsewhere.[4] Our primary finding was that, above the initial help-desk level, IR is being done with dynamic tailor-made teams, as opposed to fixed teams. These teams involve people with various levels of seniority and expertise, brought together virtually to best address an issue. Their experience has great potential value.

Incident management has some clear terms. In 2000, the Information Technology Infrastructure Library (ITIL) in the United Kingdom defined an incident as "an unplanned interruption to an IT [information technology] service or reduction in the quality of an IT service."[5] The ITIL documents led to the International Organization for Standardization (ISO) standards on IT service management (ISO 20000). In both the ITIL and ISO documentation, the goal is clear: Incidents need to be resolved and then closed. Service continuity is the priority. Of course, the ITIL and ISO processes also include reporting and, where necessary, root-cause analysis. However, the focus is always on technical faults and their resolution, and there is little or no consideration of innovation in products, services, or processes.

In our study of IR teams, we were impressed by the tailor-made teams.[4] We especially admired the knowledge that frequent members of the teams built up over time about the experience of users and customers. In the context of IR, they learned about customer workflows, their motivations, and their priorities. All this information helped resolve the incidents and might even help avoid such incidents in the future. We developed a view that there was an even more valuable opportunity. While agile methods bring the client and developer together, and DevOps extends the collaboration to deployment, we feel that a new collaborative relationship might involve incident responders. The unique and valuable knowledge from incident responders should be used to support the design and development of new releases and new

 0740-7459/19©2019IEEE

software products. Incidents need not only indicate trouble; they might also lead to insight.

Collaboration and System Development

Collaboration among different groups in software development is not a new idea. Even in early processes, for example, there was acknowledgment that collaboration between clients and developers was an important idea. As long ago as the 1960s and 1970s, participatory design (also known by several other names) emphasized the role that users might play in the design process.[6] The misinterpretation of Winston Royce's waterfall description[7] is probably a low point, as it suggested to many people that the parts of the process could be executed in strict sequence, with artifacts passed "over the wall." However, Craig Larman and Victor Basili documented that, even in the early history of software development, iterative and incremental processes involved customer feedback.[8]

As computer usage increased, the role of users became more widely acknowledged, leading to the field now known as *human–computer interaction*. One of the key early books, *User Centered System Design*, emphasized that software development needs an interdisciplinary collaborative team.[9] This led to the practices now common in user experience (UX) work.

Collaboration has always been vigorously emphasized in agile software development. The Agile Manifesto, for example, says the signatories value "Customer collaboration over contract negotiation."[10] One of the 12 principles is "Business people and developers must work together daily throughout the project." Another is "The best architectures, requirements, and designs emerge from self-organizing teams." Alistair Cockburn suggested software development should be a cooperative game:

> The team, which consists of the sponsor, the manager, usage specialists, domain specialists, designers, testers, and writers, works together with the goal of producing a working and useful system. In most cases, team members aim to produce the system as quickly as possible, but they may prefer to focus on ease of use, cost, defect freedom, or liability protection. …The game is cooperative because the people on the team help each other to reach the goal. The measure of their quality as a team is how well they cooperate and communicate during the game. This measure is used because it affects how well they reach the goal.[1]

In addition to stressing team self-management and collaboration between business interests and developers, agile processes emphasize rapid and frequent iteration. Other principles underpinning the manifesto are "early and continuous delivery of valuable software" and the need to deliver working software frequently. These are also some of the motivations for DevOps:[2,3] If we are going to frequently transition software from development to production, the transition process needs to be smooth. In particular, development teams and operations teams need to work effectively together and be supported by appropriate tools.

One reason quickly becomes clear: when systems evolve rapidly, incident responders need to be kept informed about new features to make sense of problems that arise. However, while IR is often included as part of "operations," there is very little consideration of how it might play a collaborative role in innovation. For example, in the 2011 DevOps papers that Patrick Debois introduced, IR is barely mentioned and does not participate fully, if at all.[2] Greater involvement of incident responders, therefore, will make deployment of increments and changes smoother, steadier, and faster. Another benefit is less immediate but represents a significant opportunity because IR can provide key information to guide system design. Statements about DevOps can be clear about the need for a continuous cyclic approach, but there is little consideration of IR.

Figure 1 illustrates collaboration across activities in system-development processes. The rows show the roles according to the timing of their main emphasis, from earlier (top) to later (bottom). The columns display collaborative processes in a similar way, with UX earlier (left) and incident response later (right). Collaboration is emphasized between temporally adjacent roles.

In an iterative approach, however, the later roles should also collaborate with the earlier roles. For example, UX processes emphasize learning from users to inform design. So, while UX design comes early and designers study users and work practices before beginning design, they also study users of the system later to inform design of a later release.

Our proposal is that a better process would also involve learning from incident responders, connecting incident to insight: I2I. We do not suggest it can be a complete substitute for learning from real users. However, it has added value because incident responders are part of the operations team. They have insights into the lives of many users having difficulties and into the root causes of and solutions

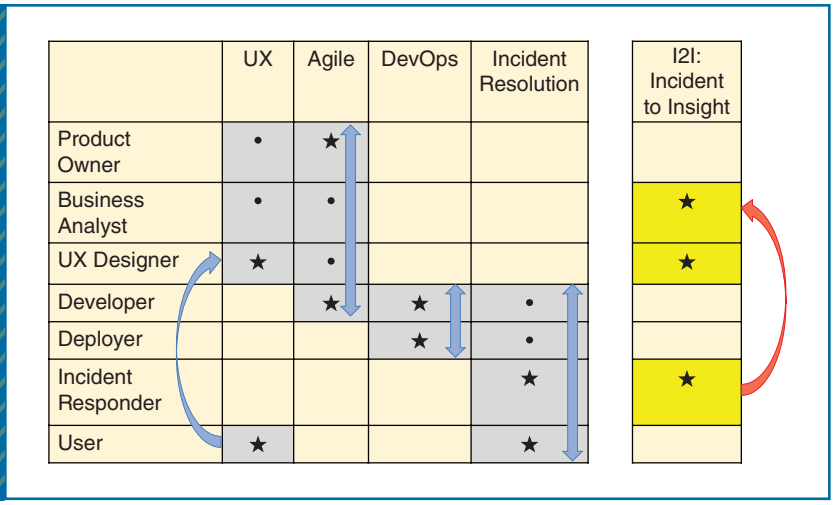


FIGURE 1. Collaboration in system-development processes. The rows show various roles; the columns show particular processes, indicating the roles that collaborate. Stars indicate primary roles; bullet points indicate secondary roles. The straight double-headed arrows indicate ongoing collaboration; the curved single-headed arrows indicate learning to inform later design. The column on the right illustrates the authors' proposal.

to the problems and requests. Moreover, because they are part of the operations team and are often in the same organization, working with them would be easier and more continuous. This new connection would augment existing processes and need not replace or change them.

The IR Experience

In our study of IR in operations centers, we employed rigorous ethnographic and qualitative-analysis methods. We conducted extensive observations, interviewed 38 people in depth, and shadowed 139 people for several hours. We studied seven operations centers covering a range of sizes (from three people to more than 100). Each center was part of a different organization, and the organizations provided services in different domains (finance, health, education, military, and security). We also met with managers and executives responsible for the operations centers. As we conducted the study, we gained a good understanding of the processes involved, how they compared with standards, and how they might be better supported.

All this is documented in our paper "Incident Response Teams in IT Operations Centers: The T-TOCs Model of Team Functionality."[4] We documented how current incident management practices use dynamic ad hoc IR teams—that is, tailor-made to the circumstances. The response process may be highly structured and under tight time constraints. However, over time, these professionals construct a detailed mental model of when and how users experience the product or service when things go wrong.

As our study progressed, we also developed an understanding of the people involved, their knowledge, and their reflections on their work. There was considerable diversity. We talked to young people relatively new to work, having come from training in the last year or two, older people who had spent their entire career at one organization, and others who had moved from another organization specifically because their expertise in some special context was wanted for IR.

Another aspect of the diversity was that, above the tier-one or help-desk level, many people involved in IR did not have that as their sole or even primary role. For example, where an incident concerned a specific software system, it was common to directly involve developers who created or maintained that software. Where hardware or networking elements were involved in the incident, it was common to get experts on those aspects, sometimes from other organizations, such as hardware vendors or infrastructure providers. When the business customer involved was known to be critically important, it was common to directly involve higher-level managers or sales representatives. All these people and those from other specialties served directly on the tailor-made teams we observed.

Finally, we observed that some incident responders were regularly on teams and experienced the incidents from beginning to end, while others were there on an occasional and as-needed basis. Those who were regularly on IR teams served as the coordinator of the team, the technical liaison with the customer, the technical expert capable of resolving the issue at hand, or in some other role. These and others, such as operational managers, developed a broad perspective on incidents.

We observed these teams and watched incidents unfold, typically using conference calls, dedicated multiperson chat windows, and a variety of one-on-one communication. We also were able to discuss the situation with team members, sometimes while they waited for information

and at other times after the incident was resolved. What we appreciated more and more as our studies progressed was the breadth and depth of understanding these people had for their customers and users.

It seems obvious: When your role is to help people with problems, especially complex problems, you learn a lot. First there is the problem itself, which typically requires some workplace and workflow context to even describe. Then there is the priority (how important this is), what the time frame is, why, what the consequences of failure are, and for whom.

This all forms a rich picture. It often goes beyond workplace functionality and efficiency: It can be personal, emotional, and even heartbreaking. Business success and failure can be at stake, as can personal careers. Some incidents can take hours or even days to resolve, and people get to know each other with small details that illuminate lives on the other end of the phone line and the other side of the country or the world. With long-term business relationships, incident responders and customers begin to recognize names, people, their context, and their history. Moreover, they become familiar with long-term patterns, when the system load is typically heavy or light, the performance characteristics that are manifested, and which times are critical for a customer's business.

All this has great potential to inform system refinement and redesign. Depending on the nature of what is learned, a broad range of system issues may be affected. For example, the knowledge might inform user-interaction design. Even improving documentation or context-sensitive help might significantly improve UX. The knowledge might also inform the business model of the system because that model can create the foundation for problematic behavior. For example, users may overuse parts of a system that do not incur costs and thus fail to use more appropriate but costlier mechanisms. Furthermore, the knowledge may affect architecture or even hardware, when unforeseen issues indirectly lead to incidents. Incident responders may learn about such patterns, which might otherwise go undetected.

We observed one example involving IR in a system for a health-care organization. The actual incident involved authentication, and the resolution was straightforward: a password reset. This might be seen as a request, rather than an incident, but the issues arising are similar, as they both can represent significant learning opportunities.

The context was rich, because in health care, as in banking, authentication is important to not only protect access to services and resources but to also record who took what actions, thus providing an audit trail. In health care, however, the imperative for care can be seen as especially important, and workers may sidestep authentication or use a colleague's credentials to immediately proceed with their work. Password reset requests, despite seeming of low importance, can be frequent and may involve several kinds of stress for the user: lives, as well as regulatory compliance, may be at stake. To their credit, the incident responders we studied detected this pattern and formed a team involving the responders and managers to explore what was happening in more detail by inspecting incident logs.

While we did not see how this issue was resolved, the conflict involved in health-care authentication has been well documented elsewhere.[11] It also seems possible that developing an understanding of the problem might lead designers to suggest solutions. For example, health-care authentication should be so fast and easy that the conflict seldom arises. Examples might include usable biometrics or authentication tokens. Where conflict still arises, perhaps in emergencies, there might be remedial actions available to investigate and approve the actions taken retroactively. In this way, while password reset might seem like a simple resolution, the potential insight might lead to better design with a much greater impact for both hospital workers and information integrity.

Another example from the same site stems from an interview with one responder who reflected on a pattern spanning many incidents. These concerned users, patients, and interactions with insurance providers. The responder suggested that these issues could be either simplified or eliminated if the users themselves were able to enter the system and access some information online through the already-existing portal. This would ease the evident frustration that users felt when having to request help for things they might be able to do themselves. It would also save time for the responders. Another idea from the same responder was that the resolution of user problems might be much easier with some simple screen-sharing capability.

These examples highlight two important points. One is that some knowledge stems not from a single incident but from many. This suggests that simple recording of data from incidents may not be sufficient, as detecting patterns involves experience over time. The other point is that, although the responder's suggestions seem valuable, we must be cautious, as other factors might be involved. For example, there might be health-care, legal, or business reasons

for keeping responders in the loop. Accordingly, the ideas from the responder would need careful consideration by design or business analysts. In this context, we should not expect responders themselves to design solutions. Rather, their major value is to help identify problems and patterns.

Making the Connection

So if IR can inform design, how might the connection be made? On the basis of our time spent studying IR teams, we can identify several possible approaches.

IR Management

One potential approach might be for IR management to take responsibility. In our study, it was common for incidents to be logged and for descriptive reports to be written upon closure. Reports were later reviewed and analyzed by managers to appraise performance and assist root-cause analysis to avoid future incidents. We never saw IR managers considering implications for future design or business opportunities. Moreover, incident reports tended to be dispassionate and did not include the rich understanding gained by those handling the incident.

Accordingly, having management take the role of informing design would require a different approach. Incident reports would need to be much richer, thereby requiring expertise in the team for creating rich descriptions. Alternatively, IR management could play a larger direct role in incidents. Either way, there would need to be a new pathway for IR managers to inform design, perhaps through meetings with product design teams or by comprehensive reports.

IR at the Table

Perhaps the most extreme approach to making a connection would be to bring incident resolution to the table: to include IR personnel on design teams. This would begin to approximate participatory design, where users are deeply involved. This would certainly bring the experience into the design process, but the practical issues would be significant. For example, design teams cannot include all IR personnel lest the team become unmanageable. Also, the more someone becomes involved, the less time he or she has for IR. Moreover, it is unclear whether the design team needs involvement from IR all the time, and other kinds of contribution will require other skills and other knowledge.

One interesting practice emerging in DevOps is called *ChatOps*, where developers and deployment personnel share ongoing chat sessions to speed communication and foster situation awareness.[12] We admire the practice in that context, and in small startup organizations it might also connect incident responders to designers and developers. In sufficiently small organizations, such connections have always been close anyway. In larger and more established organizations, however, we are doubtful that this approach can scale well because it could include incident responders as well as business analysts, designers, and developers.

Some IR groups do, however, maintain ongoing chat sessions among themselves to develop situation awareness across all incidents. For example, they would briefly relay issues arising or nonstandard solutions that might help others. IR is very much a team activity, and while this intrateam communication has a focus on understanding and resolving incidents, it may well be a very valuable source of information for improving products or services.

IR, Data Mining, and Ethnography

Related to the management approach described previously is the data-mining approach. Several managers suggested such an approach for analyzing incident reports. Most analyses we saw involved simple counts of various categories of incidents by different dimensions: the time of day, week, month, platform involved, or impact. But there was a desire for something more: something that would somehow link together data to identify patterns that might otherwise escape notice. To accomplish this in a way that might identify patterns that go beyond faults would again require more extensive data collection or reporting and the involvement of many more factors relevant to the incident. Automatic capture of vast amounts of data is now possible: from applications, from networks and other infrastructure, and from a variety of external sources, such as social media.

We feel, however, that this is unlikely to capture or reveal the kind of specific contextual information that becomes apparent to incident responders. Data-driven deep learning may miss what is obvious in person-to-person communication. Moreover, for the responders to reflect and document all this would be difficult. They already have strict obligations to record many details, and we frequently observed incident responders working hard to record events even after incidents were resolved. This pressure was often intense when service-level agreements were involved. There was little or no time for reflection in the moment.

Of course, there are other sources of data that can be captured automatically. A wealth of data about the patterns of usage is now available, especially where software is used online, which is common now with software

as a service. Aapo Koski and his colleagues discussed the opportunities for monitoring and analyzing these data to understand and improve UX.[13] This is a potentially valuable approach, but we think that IR teams would provide additional insight. These teams engage with the users themselves and gain an understanding of context and motivation that is difficult or impossible to glean from log files.

If IR personnel are not expected to join the design team, then the critical issue is how to get their knowledge to the design team. This suggests that they document their knowledge or that someone else should document their experience, and the design team should have access to this knowledge as needed. Writing rich descriptions of experience and working with others to gain their experience require significant skill and experience. These activities also require significant time, and spare time is scarce among IR teams and management.

These skills and experience involve ethnography, which has a long tradition in software development, especially in interaction design and requirement analysis. The role of research is also important. Helen Sharp and her colleagues discussed the issues that arise in the software context.[14] They identified the various roles for ethnography in the software process, and what we propose is an additional role. However, knowledge of ethnographic methods is typically already present in the business-analysis and product-design teams. On the business side, interviewing has long been regarded as a core systems-analysis skill. On the design side, it is a core element of UX research. Normally, this kind of work involves customers and users, and of course this is always critical.

Applying these skills and practices also with IR personnel has some advantages. First, IR personnel will typically be more available, and many will work for the same organization as the researchers and design team. Second, while IR personnel will not understand the whole world of the customer and user, they do have knowledge from some critical pain points. Finally, this approach has the potential to change attitudes within IR, as the work changes from preventing loss of service to potentially adding value. IR might be seen as a contributor to profit rather than a necessary cost. In all these cases, there is vast potential for new insight as well as confirmation (or repudiation) of ideas already considered.

Looking Forward

There may well be new approaches that combine some elements from all of the strategies discussed here. Interestingly, social media may provide some inspiration. Social media practices are already becoming common in software development.[15] Most of these practices involve developer teams and communities, but we suggest that new approaches involving IR experience are possible.

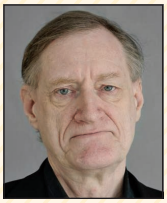
Practices that have arisen in social media platforms seem relevant. Brief messages are followed by a wide range of messages from other users, creating an open and evolving classification using hashtags that, in turn, help identify trends worth further investigation. We speculate that this model might support one practical way of connecting knowledge from IR to the wider system design and development team. Responders do not have time to create extensive documentation, but they might have time for brief messages (tweets) that easily could be seen, instantly or later, by many others. While fixed classification systems do not cope well with emerging issues in context, experience shows people develop a flair for creating and adapting hashtags. All this would create situation awareness not only among incident responders but also for anyone in the organization.

Finally, social media has significant potential for both immediate and longer-term data analysis: trends, correlations, and social connectivity. Even so, social media analysis cannot substitute for ethnographic and contextual studies. However, such analysis can be a starting point to show it is needed, if only perhaps in a modest way. Of course, we do not suggest using public social media platforms, but a similar, smaller, intranet facility might be transformative.

System development is a collaborative process: Many perspectives are necessary for success. Throughout the short history of system development, several collaborative partnerships have been stressed: designers and users, clients and developers, and developers and operations. But no system is perfect, and imperfections are often exposed when incidents occur. The professionals who handle such incidents have a wealth of knowledge. This knowledge applied to resolving such incidents also has the potential to identify new opportunities. The renewed emphasis on iterative design and development means that, more than ever, such opportunities can be acted upon quickly.

We suggest that the best way to make this happen is to bring this knowledge into design and analysis through an I2I process. Alfonso Fuggetta and Elisabetta Di Nitto, in their paper on the future of software process, identified the need to reconsider established boundaries: "In general, the classical distinction among design, implementation, and operation tends to disappear or to be

ABOUT THE AUTHORS



ROBERT BIDDLE is a professor cross-appointed to computer science and cognitive science at Carleton University. His research interests include human factors in software design, and his recent projects involve human issues in cybersecurity. He received a Ph.D. in computer science from the University of Canterbury. He is a Member of the IEEE and the Association for Computing Machinery and a fellow of the New Zealand Computer Society. Contact him at robert.biddle@carleton.ca.

JUDITH M. BROWN is a research associate at Carleton University's School of Computer Science. She received a Ph.D. in psychology with a specialization in human–computer interaction from Carleton University. She has been a user-experience researcher and is an advocate for agile methods. She is a member of the Association for Computing Machinery and the Special Interest Group for Human–Computer Interaction. Contact her at mmjbrown@gmail.com.

STEVEN GREENSPAN is a lead research scientist at the Strategic Research Labs at CA Technologies. His research interests include human factors in decision making, IT operations, and visualization. He received a Ph.D. in experimental (cognitive) psychology from the State University of New York at Buffalo. Contact him at sgreenspan@gmail.com.

radically redefined."[16] We suggest IR has a place in any new understanding. We can use new approaches to collaboration, such as the use of social media, and the skills and experience of ethnographers, such as business analysts or UX researchers. Incidents should lead to not only resolution but also insight.

References

1. A. Cockburn, *Agile Software Development: The Cooperative Game*, 2nd ed. Reading, MA: Addison-Wesley, 2006.
2. P. Debois, "DevOps: A software revolution in the making?" *Cutter Inform. Technol. J*, vol. 24, no. 8, pp. 3–4, 2011.
3. C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, 2016.
4. J. M. Brown, S. Greenspan, and R. Biddle, "Incident response teams in IT operations centers: The T-TOCs model of team functionality," *Cogn. Technol. Work*, pp. 1–22, 2016.
5. C. Agutter, *ITIL Foundation Handbook*, 3rd ed. London: U.K. Stationery Office, 2012.
6. S. Bødker, "Creating conditions for participation: Conflicts and resources in systems development," *J. Human–Computer Interaction*, vol. 11, no. 3, pp. 215–236, 1996.
7. W. W. Royce, "Managing the development of large software systems," in *Proc. IEEE WESCON*, 1970, pp. 1–9.
8. C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.
9. D. A. Norman and S. W. Draper, eds., *User Centered System Design: New Perspectives on Human-Computer Interaction*. Boca Raton, FL: CRC Press, 1986.
10. M. Fowler and J. Highsmith, "The Agile Manifesto," *Softw. Develop.*, vol. 9, no. 8, pp. 28–35, 2001.
11. K. Hedström, E. Kolkowska, F. Karlsson, and J. P. Allen, "Value conflicts for information security management," *J. Strategic Inform. Syst.*, vol. 20, no. 4, pp. 373–384, 2011.
12. G. V. Hulme. (2014, July 16). ChatOps: Communicating at the speed of DevOps. *DevOps.com*. [Online]. Available: https://devops.com/chatops-communicating-speed-devops
13. A. Koski, K. Kuusinen, S. Suonsyrjä, and T. Mikkonen, "Implementing continuous customer care: First-hand experiences from an industrial setting," in *Proc. 42nd Euromicro Conf. Software Engineering and Advanced Applications (SEAA 16)*, 2016, pp. 78–85.
14. H. Sharp, Y. Dittrich, and C. R. B. de Souza, "The role of ethnographic studies in empirical software engineering," *IEEE Trans. Software Eng.*, vol. 42, no. 8, pp. 786–804, 2016.
15. M.-A. Storey, L. Singer, B. Cleary, F. F. Filho, and A. Zagalsky, "The (R) evolution of social media in software engineering," in *Proc. Future of Software Engineering (FOSE 14)*, 2014, pp. 100–116.
16. A. Fuggetta and E. Di Nitto, "Software process," in *Proc. Future of Software Engineering (FOSE 14)*, 2014, pp. 1–12.