End-User Debugging for E-Commerce

Henry Lieberman Earl Wagner

MIT Media Lab
20 Ames St
Cambridge, MA 02139 USA
{lieber, ewagner}@media.mit.edu

ABSTRACT

One of the biggest unaddressed challenges for the digital economy is what to do when electronic transactions go wrong. Consumers are frustrated by interminable phone menus, and long delays to problem resolution. Businesses are frustrated by the high cost of providing quality customer service.

We believe that many simple problems, such as mistyped numbers or lost orders, could be easily diagnosed if users were supplied with end-user debugging tools, analogous to tools for software debugging. These tools can show the history of actions and data, and provide assistance for keeping track of and testing hypotheses. These tools would benefit not only users, but businesses as well by decreasing the need for customer service.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *user issues*

General Terms

Human Factors

Keywords

End-user debugging, debugging e-commerce.

1. THE PROBLEM: E-COMMERCE WHEN THINGS GO WRONG

"Your call is important to us…

Please enter your 15-digit card number, your PIN...

if you're calling from a touch-tone phone, please press 6 for customer service now,

... Hi, this is customer service. Can I have your card number?"

Electronic commerce is great – you log onto a Web site, type in what you want to its search engine, fill out a simple form, give your credit card number, and Fedex brings you the stuff the next day. It's simple, painless, efficient, and effective.

Except when it doesn't work. People inevitably mistype numbers, misunderstand directions, or just click on the wrong buttons. Vendors lose orders, or confuse names, and computer systems crash. Then what?

More often than not, customers have to pick up the phone. And they probably reach a phone tree where they're forced to navigate through complex menus, a tedious process that might or might not give the right choice of problem to be solved. It might give another phone number to be dialed. It might ask for card numbers or transaction numbers that aren't readily at hand, and have to looked up offline. If someone in customer service is successfully reached, that person (often a low-paid worker in a high-pressure call center) may specify a tedious process to be performed. They may not be empowered to actually understand or fix the problem themselves. Customers find themselves bounced endlessly from one support person to another. All of us have had these kinds of experiences.

Customer service problems are incredibly frustrating. Not only do they cause frustration about the immediate transaction, they also poison the relationship between customers and vendors. Customers feel like they are being deflected, that they are not being listened to, and that vendors do not care about their time or about their relationships. This casts a pall over future interactions.

Nobody expects technology to be perfect, and people are tolerant of mistakes and occasional failures, but what drives people crazy is when they get "stuck" – left with no recourse, no obvious way to systematically go from having a problem to understanding what is causing the problem and how to resolve it.

But it isn't all the fault of the vendors. The problem is difficult to fix simply by businesses spending more money on call centers, since human time, even if low paid, is expensive. Many businesses claim that, as soon as a customer picks up the phone to call about a problem, they have already lost money on that customer's relationship.

This problem is now becoming a major obstacle to further adoption of e-commerce itself. Many people who now choose brick-and-mortar shopping over e-commerce do so simply to "have a face to talk to" in case of problems. Again, the impact of the problem-resolution experience has a disproportionate effect on the trust relationship between a customer and vendor, and nowhere is the trust issue more important than in electronic commerce.

2. A PROPOSED SOLUTION: E-COMMERCE DEBUGGING

Our proposed solution is to provide users with E-Commerce Debugging Tools, that enable consumers to diagnose and solve many simple problems themselves from their computers. The E-Commerce Debugger is analogous to a programming language debugger used in software development. It allows the user to systematically investigate the possible causes of an error by examining processes and data at varying levels of detail until the problem is discovered.

IUI'03, JANUARY 12–15, 2003, MIAMI, FLORIDA, USA.

ACM 1-58113-586-6/03/0001.

Debugging is detective work. It requires generating hypotheses, and testing them by a process of elimination. In the software development area, we have developed some novel interactive tools, described below, for supporting the investigation involved in debugging.

Of course, software debuggers are used by expert programmers and have interfaces designed to be used by expert users. For e-commerce debugging tools to be effective, they have to have user interfaces that are designed for use by non-expert users, with simple interfaces and ample explanation and visualization facilities. While we don't expect consumers to understand software debuggers like experts do, we believe that most people have enough understanding about cause-and-effect relationships to be able to use the kinds of tools we propose.

Expert programmers are confident that no matter where a bug might be, systematic investigation will almost always find the problem if a fine enough level of detail is examined. That gives them confidence in using the technology. We hope to instill that level of confidence in consumers who participate in e-commerce transactions. We also expect, where the software tools prove inadequate, that it will be possible to at least partially automate the human-to-human communication that may be required to resolve problems.

In the design of the interfaces, we also expect that work in the area of intelligent software agents will be useful. Context and learning [6] can be used to automatically infer what information is necessary, automatically find it when available, and put relevant choices at the user's fingertips, streamlining the user interface.

In the past, it would have been impossible to provide practical e-commerce debugging tools because the raw material that such tools would need – information about transactions, user identification, vendor specifications, payments, and so on, would exist only on paper. Now, however, since much of it is accessible via user accounts on Web sites, it has become practical to capture that information via Web browsers, and to provide debugging facilities that operate through the browsers.

2.1 Tools For Customer Support

Sometimes, it won't be possible to completely solve a problem from the user's perspective alone, because the answer might depend on details of the process or data that are internal to the vendor and that the user does not have access to. In this case, he or she must resort to communicating with a customer support person, via e-mail or phone.

But similar tools may be made available to the customer support person, so that that person can employ a similar debugging strategy. The user's agent could communicate automatically to the support person's agent, which could instantly access the details of the problem, and see the investigation the user has done thus far. Valuable phone support time would not be wasted reciting account numbers and retelling the story to different personnel in the case that more than one support person needs to be involved. The support person's interface could be far more detailed and specialized to the industry and to the company than the general public would be willing to tolerate. Typically, the user does not want to need to know the details of the company's internal process, but such knowledge might be essential to solving the problem. The customer support person could again benefit by breaking down the problem into steps, verifying each independently, tracking the history of individual data items, and so on.

3. A BROADER CONTEXT: END-USER DEBUGGING

We actually see this work in the context of a broader goal of developing end-user debugging tools for all sorts of computer interactions, not just for e-commerce transactions. Much lost time and frustration is experienced by computer users dealing with the intricacies of installing software, system maintenance, and problem solving with specific applications. Why can't the computer itself know what it is doing and help us solve the problems we are having with it?

Why can't we ask a computer in any situation, "What are you doing right now?", "Why are you doing that?" , "What does this error message mean?", and other questions that you might reasonably ask a knowledgeable human assistant? If something goes wrong, why can't we trace the problem back through the steps that led up to it? Why can't we formulate hypotheses about what might have been wrong, and test them until we get to the answer?

Many of the cognitive processes involved in diagnosis and repair of e-commerce situations have analogies in many kinds of general system interaction as well. So we think our results will generalize to many other areas of computer interaction.

But e-commerce is a good place to start, because the transactions themselves are procedurally very simple, compared to the operation of an operating system or specialized application. People understand the basic concepts of financial transactions, and since everyone participates in such transactions, interfaces to deal with them will find a wide audience.

The economic value of improving the situation is enormous, and the business community has not so far tackled the problem in a comprehensive way. Because many actions involve multiple parties, such as an online purchase involving the vendor, the consumer's credit card company and the shipping service, consumers need help from tools that operate on their behalf, rather than on behalf any particular vendor.

Further, we think the interfaces developed to support end-users in debugging could also influence future debugging tools for professional software developers. Since debugging accounts for approximately half of all software development costs, the potential economic payback is enormous here, too. Because development of debugging tools has so far been limited to expert professionals, we believe that insufficient attention has been paid to the usability of such tools, and we believe that a focus on supporting the needs of non-programmers could benefit in these situations as well.

4. DEBUGGING IN SOFTWARE DEVELOPMENT

Debugging is the dirty little secret of computer science. Despite the fact that studies of programmers show that half of the costs of software development are consumed in debugging activities, there is precious little work in computer science that explicitly seeks to make the debugging process easier and more effective. Tools available in today's commercial programming environments are essentially unchanged from

those that appeared in programming environments thirty years ago: function trace, breakpoints, line-by-line stepping.

Some of our past work has been aimed at understanding the cognitive processes involved in debugging. These insights have guided the development of environments that provide operations to help programmers ask and answer questions about the dynamic behavior of the systems they're working with [4]. In contrast to the roles of current tools, debugging from the perspective of the programmer consists of the following cognitive processes:

Visualization: What's happening? How do I get an overall feel for what's going on?

Localization: Where is the problem? What specific step or piece of data is responsible?

Instrumentation: Trace or "audit" the history of a particular event or piece of data.

Repair: What are the consequences of a proposed fix?

We think this breakdown between current tools and the parts of debugging described above is largely a user interface problem [5]. Many of the principles that we have identified in constructing debugging environments for software will be useful in developing e-commerce debugging environments, though it will be necessary to take into account the essential differences between the programming domain and the e-commerce domain.

ZStep [5] is one of a series of debuggers we have built that aim to support these cognitive processes, especially visualization and localization. ZStep is a reversible debugger – it records every step of the execution of a program and lets you run the program forward and backwards.

5. THE ZSTEP REVERSIBLE SOFTWARE DEBUGGING ENVIRONMENT

Some of ZStep's innovations include:

- An animated view of program execution, using the code editor
- An inspector window that displays values corresponding to the stepper's focus
- A complete, incrementally-generated history of program execution and output
- Controls to run the program in forward and reverse directions
- Incremental control of the level of detail displayed
- One-click access from graphical objects to the code that drew them and vice versa.

Many analogues of ZStep's features are essential for e-commerce debugging tools,

- Animated views of execution of e-commerce actions
- Display of action data correlated with focus in the action history
- Incrementally-generated history of action execution
- Controls to run the action history in forward and reverse directions
- Incremental control of the level of detail displayed
- One-click access from action values to the action history that produced them
- One-click access from action history to the corresponding action data.

Again, we stress that, although inspired by approaches for software debugging, e-commerce debugging tools should be developed for non-expert users, and very close attention should be paid to usability and accessibility for the general public.

Also in this collection, we demonstrate an early version of an e-commerce debugging tool. The software agent, called Woodstein, records user actions such as purchases on the Web. It interactively matches these actions with generic models and allows users to monitor and investigate the execution of their actions. It also tracks the changes that user action data items undergo, automatically generating audit trails, and illuminating users as to the state of their actions and data.

6. REFERENCES

1. Ackermann, M. and MacDonald, D. Answer Garden 2: Merging Organizational Memory with Collaborative Help. In *ACM Conference on Computer-Supported Collaborative Work*, 1996.
2. Dyche, J. *The CRM Handbook: A Business Guide to Customer Relationship Management*, Addison-Wesley, Reading, MA, 2001.
3. Klein, M. A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. In *Journal of Computer-Supported Collaborative Work*. Special Issue on Adaptive Workflow Systems. Vol 9, No 3/4, August 2000
4. Lieberman, H., Ungar, D., Fry, C. Debugging and the Experience of Immediacy. In *Communications of the ACM*, April 1997.
5. Lieberman, H., Fry, C. ZStep 95: A Reversible, Animated, Source Code Stepper. In *Software Visualization: Programming as a Multimedia Experience*, John Stasko, John Domingue, Marc Brown, and Blaine Price, eds, MIT Press, Cambridge, MA, 1997.
6. Lieberman, H., Selker, T. Out of Context: Computer Systems that Learn About, and Adapt to, Context. In *IBM Systems Journal*, Vol 39, Nos 3&4, pp.617-631, 2000.
7. Rich, C., Sidner, C.L., Lesh, N.B. COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. In *Artificial Intelligence Magazine*, Winter 2001
8. Wagner, E., and Lieberman, H. An End-User Tool for E-commerce Debugging. In *Proceedings of IUI*, 2003.