Are Object Protocols Burdensome?
An empirical study of developer forums

Ciera Jaspan
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ciera@cmu.edu

Jonathan Aldrich
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA 15213, USA
aldrich@cs.cmu.edu

Abstract

Object protocols are a commonly studied research problem, but there is little known about their usability in practice. In particular, there is little research to show that object protocols cause difficulty for developers. In this work, we use community forums to find empirical evidence that object protocols are burdensome for developers. We analyzed 427 threads from the Spring and ASP.NET forums and discovered that 69 were on a protocol violation. We found that violations of protocols result in unusual runtime behavior rather than exceptions in 45% of our threads, that questions took an average of 62 hours to resolve, and that even though 54% of questions were repeated violations of similar protocols, the manifestation of the violation at runtime was different enough that developers could not search for similar questions.

Categories and Subject Descriptors D.2.m [*Software Engineering*]: Miscellaneous

General Terms Languages, Human Factors

Keywords community forums, usability, object protocols, collaboration constraints

1. Motivation

Object protocols have formed the basis for a large body of research. In only the last five years, there have been multiple projects which infer protocols [6, 12, 13, 17] or specify and check them [4, 5, 7, 10, 11, 14, 15]. The SAVCBS workshop has even hosted several contests to specify and verify canonical protocols, such the Iterator protocol [2], the Subject-Observer protocol [16], and the Database Access protocol [8].

Though there is an abundance of work on solving the problem of object protocols, there is little work on the problem itself. Recently, Beckman, Kim, and Aldrich showed that at least 7.2% of classes in several large open-source codebases require that their clients follow a protocol, and they found that 13% of classes in another set of codebases were clients of one of these protocols [3]. While this tells us how prevalent protocols are in code and how frequently protocols are used, it does not tell us whether these protocols cause significant challenges for developers. After all, a File protocol is not particularly complex; perhaps these protocols are so trivial that developers don't have difficulty understanding and following them.

In this work, we will provide evidence that object protocols are burdensome for developers in the ASP.NET and Spring frameworks. By analyzing the postings on developer community forums, we will show that protocols:

- cause unusual runtime behavior that is difficult to debug,
- take developers hours and days to solve, even with expert help, and
- recur for several developers over time.

Our study involves 427 threads of conversation from the ASP.NET and Spring community forums. Of these, 69 were relevant for our interest, as explained by our methodology in Section 2. Section 3 presents our results on these 69 threads, which shows that 45% of the problems resulted in unusual runtime behavior without an exception, the average time to solution was 62 hours, and 54% of the problems were from protocols that were violated in multiple ways by different developers. Section 4 discusses the benefits and pitfalls of using forums as a source more generally and hypothesizes about what other data could be extracted in this way.

In this work, we will be studying an expanded form of object protocols known as *collaboration constraints*. A collaboration constraint is a protocol over how multiple objects may collaborate together, rather the traditional proto-

OOPSLA '11 October 22-27, Portland, Oregon
Copyright © 2011 ACM Copyright is held by the author/owner(s). This paper was published in the Proceedings of the Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU) at the ACM Onward! and SPLASH Conferences. October, 2011. Portland, Oregon, USA. . . $5.00

col that considers the usage of a single object. Additionally, these collaborations may span beyond traditional program artifacts (ie: C# or Java) into non-traditional, declarative artifacts, such as XML, JSP, or ASPX. Further information on collaboration constraints can be found in [9].

As an example of a multi-object constraint found in the forums, consider the forum thread in Figure 1, which breaks the protocol by which controls may be dynamically added to a web page. In this thread, the poster "strange_will" describes a problem where the controls that were dynamically added to the web page disappear after a postback. The poster also describes an attempt to correct this problem by removing the if-statement, but this resulted in the controls appearing without any data. Two experts explain the issue: since the controls are dynamically generated, they must be recreated on every load of the page (ie: the if-statement needs to be removed). Additionally, the controls must be generated before the page restores its state from the session. As this happens before the `Page_Load` callback, the controls must be generated in a prior callback, such as `Page_Init`.

2. Methodology

In this work, we studied two software frameworks for web applications, ASP.NET and Spring. In both studies, our goal was to find forum threads that showed a developer who broke a collaboration constraint. We used different methodology to gather the threads for these studies due to differences in the activity level and response rates in the developer communities. However, we evaluated the resulting threads using the same criteria.

2.1 ASP.NET

In the ASP.NET study, we performed an archival analysis of the postings in the Web Controls sub-forum of the ASP.NET help forums. At the time of the analysis (spring of 2007), this was the most popular of the 104 sub-forums, with over 87,000 conversation threads since 2003. Our analysis was on the threads that had their last activity during the first week of October 2006. As the analysis itself was conducted many months later, each of these threads can be considered closed (that is, we expect no further helpful responses).

There were 271 threads with their last activity during this period. We removed any threads which met one of the following properties:

- The question was not about Web Controls.
- The poster or responder used extremely poor English, to the point of not being understandable.
- The poster needed compilation help or otherwise did not understand basic syntax.
- The poster described the problem in such a vague way that it could not be reconstructed.
- There was no response at all or no response that solved the problem.

This left 66 threads which were on topic and were understandable enough to answer. Of these, 50 were requests for tutorials or documentation for a specific task. This left 16 threads for the study, which we have archived [1].

2.2 Spring

When we went to study Spring, we found a community with lower activity on the forum and a lower response rate. While we attempted to use the same process to gather threads, we found that there were many fewer threads and that the vast majority had no response.

In order to find examples effectively, we created an automatic filtering system that would scan threads for specific properties and only return those that met our criteria. While this filtering mechanism will miss some relevant threads, we found that it was a more effective method to find interesting threads. The criteria we used are:

- *Has a* `<pre>` *tag.* To ensure that there was a specific example being discussed and filter out requests for tutorials and documentation, we accepted only threads where there was code posted within an HTML `<pre>` tag (for pre-formatted text, commonly used for displaying code). This might miss threads where people did not use the `<pre>` tag to display code.
- *Uses words "exception" or "error".* Again to filter out requests for tutorials and documentation, we accepted only threads where the words "exception" or "error" appeared somewhere. This unfortunately misses many issues where the error was unexpected run time behavior, rather than an exception.
- *Responded to by a top-poster.* We accepted only threads where one of the responders is in the top-25 of all posters. We found that these posters are requently experts such as consultants and framework developers themselves and that they are more likely to provide a solution. This filter misses threads that were correctly solved by a user with a lower post count.
- *Has an affirmation.* To ensure that there actually was a solution presented, we accepted only threads where the *original* poster had a secondary post with one of the following strings: "solved", "that work", "works", and "working". This is meant to capture threads where the original poster returns to say "Thanks! That worked for me." This filter misses threads with solutions where the original poster either did not return or did not respond in this way.

Additionally, we limited the first post to be before October of 2007 as a new version came out at this time. The analysis was performed in January-February of 2011.

The automated filtering system yielded 156 threads. Of these 103 were not useful because they did not meet the original criteria from Section 2.1. That left 53 threads for the study, which we have archived as well [1].

strange_will: So I have a table control, I have a control I built in a class that uses a control as it's base:
`public class Addons : System.Web.UI.WebControls.TableRow`
So I add properties to it, run it as a table row, shows FINE on the first load, but after you hit submit, or any other navigation that is a postback the data dissapears!

```
protected void Page_Load(object sender, EventArgs e) {
   if (!Page.IsPostBack) {
      Database.Connect(sql);
      Database.obj_Reader = Database.obj_Command.ExecuteReader();
      while (Database.obj_Reader.Read()) {
         Addons obj_Addons = new Addons(...);
         tbl_Addons.Controls.Add(obj_Addons);
      }
      Database.Disconnect();
   }
}
```

I'm obviously making sure that I'm not deleting it by using !postback, if I get rid of that, it will delete all data that I assigned to the controls, any ideas?
vivek_iit Note that Page class is stateless, it will lose all data which is generated dynamically unless you use other persistent medium like Sessions etc. So you need to regenrate your controls on each PostBack. See this post for details:
http://forums.asp.net/t/1024956.aspx
strange_will: Can I give it a state? So table rows are stateless? If I build it from a CheckBox control it will keep it's state?
The problem is that the control is a checkbox and a textbox inside of a tablerow. I need on page submit for the data to be accessable.
The post you gave me is sort of confusing. Being as I'm not sure which solution in that thread I should be reguarding.
TonyAlicea This has to do with the Page lifecycle. Remember that controls *always* lose their data across postback - it is .NET that is examining the form post and putting the data back in again for you. If you want dynamic controls to hold their values across postback then you must declare those controls in the Page_Init, not the Page_Load, because it is in the page initialization where the form post is processed and controls are refilled.
vivek_iit Yes, you can store the state of dynamic controls in Session or similar. Make sure you regenerate all dynamic controls in the Page_PreInit() or Init() methods and load their state from Session.
Please go through the Page lifecycle to understand the details

Figure 1. Condensed version of the ASP.NET thread 1030504, which shows a broken protocol across multiple objects.

3. Preliminary Results

From studying these threads, we have found significant evidence of burden for developers using these APIs. The complete results are in Table 1 and Table 2. In several cases, a single thread had more than one problem discussed in it. This happened for many reasons: sometimes the original code had multiple problems, sometimes the original poster asks additional questions after the first is solved, and sometimes new posters hijack the thread on a related topic. In these cases, we used the primary issue that was causing the error for the original poster.

3.1 Unusual manifestations of errors

In Tables 1 and 2, the column "Run time error" described whether the error seen by the original poster was an exception or unexpected behavior. As seen in Table 1, only 7 of the faults from ASP.NET threads resulted in a runtime exception; the remaining 9 resulted in incorrect behavior at run time. In Spring, 22 of the faults (41.5%) resulted in incorrect behavior at run time. As seen in Figure 1, incorrect behavior at run time, such as missing data and unusual redirects, can be very difficult to debug as there is no information to lead the developer to the correct solution. The developer "strange_will" even attempted a change to the code, but without the any understanding of the problem, the change resulted in more unexpected behavior. Unfortunately, even exceptions may not be that helpful, as many of the exceptions were simply null pointer exceptions or class cast exceptions and did not provide a useful message.

To make matters more difficult, many of the errors were not local to the fault that caused the problem, as shown in the column "Run time local?". We counted an error as "local" if the run time error led developer to the right method (or the right element, in the case of JSP, XML, or ASPX). The example in Figure 1 was counted as local since the developer was able to identify the problem method. In non-local threads, the developer posted code that did not contain the actual source of the fault. There were four non-local errors

Table 1. Archival analysis of ASP.NET forum postings. These postings were understandable, solvable, on topic, and were not requests for a tutorial. The URL for each thread is http://forums.asp.net/t/NUMBER.aspx. Data was originally collected in spring 2007 on threads with last activity in the first week of October 2006.

Protocol	Number	Run time error	Run time local?	#Questioners	#Responders	Response time (H:MM)
1	1030504	Unexpected Behavior	Yes	1	3	162:10 (over 6 days)
	1027694	Unexpected Behavior	No	1	1	381:39 (over 2 weeks)
	1032187	Unexpected Behavior	Yes	2[‡]	1[*]	18:36
	1033046	Unexpected Behavior	Yes	1	1[*]	1:46
2	1032991	Exception	Yes	1	2	7:43
	1033020	Unexpected Behavior	Yes	1	2	3:02
	1031946	Exception	Yes	1	3	117:21 (over 4 days)
	1033217	Unexpected Behavior	No	1	2	3:13
3	1031139	Exception	Yes	1	1	0:47
	1032020	Exception	Yes	1	0[†]	24:44 (over 1 day)
	1032624	Exception	Yes	2[§]	1	2:10
	1031123	Exception	No	1	1	3:23
	1031804	Unexpected Behavior	Yes	1	1	9:13
	1031933	Unexpected Behavior	No	1	1	12:44
	1032278	Exception	Yes	1	1	16:18
	1033450	Unexpected Behavior	Yes	1	1	260:22 (over 10 days)

* None of the responders actually gave the correct response.

† Poster ended up "answering" own question, but actually got it slightly wrong.

‡ This thread had an additional questioner over 4 years later: "I must have read 10 or more post on how to do this but they were all so complicated I spent hours trying to understand one of them. Yours was great, I figured it out in a few minutes. Thank you for simplest example possible."

§ And another one, also about 4 years later: "this is precious...i did not know that...Perfect..saved me a lot of frustration :)"

in the ASP.NET threads and 22 in the Spring threads. As Spring requires plugins to have more interaction with configuration files, it is not surprising that there were relatively more non-local errors. This usually occurred because the error led the developer to think their Java code was incorrect when their configuration file actually contained the fault.

3.2 Hours or days to fix

Another property that shows burden for developers is the amount of time it takes to solve these problems. On ASP.NET, the average time from original posting to answer, was 64 hours (about 2.67 days), and the median time was 11 hours. The Spring forums were slightly less skewed with an average time of 61.5 hours and a median time of 22.25 hours. Given that forum threads frequently go unanswered, we can assume that this is a method of last resort for developers; asking a work colleague for help or searching the internet for a solution would be more likely to provide a solution in faster time. Anecdotally, some developers even mentioned the amount of time and the frustration it had caused. The footnotes of Table 1 showed two people who found these postings later through a successful search and specifically mentioned the amount of time they had already spent looking for an answer. Had these developers had to post their question, they could have expected several more days of waiting for an answer, assuming one appeared.

3.3 Repeating problems

Finally, this data shows that these are not unusual protocols that developers will rarely run into. There are three protocols that comprise 11 of the 16 threads in ASP.NET, and ten protocols comprise 26 of the 53 threads in Spring. This is particularly surprising from the ASP.NET threads given the short time range that these threads had last activity, so many of them were active while a similar problem was being posted. In fact, there were only two threads, both in Spring, where a second poster came in while a thread was actively being discussed to say that they were having the same problem.

The problem is that even for a single protocol, there are multiple possible ways to break the protocol that result in multiple possible errors at runtime. As an example, consider protocol 2 in Table 2. The first two threads described a non-local exception, and both were due to null pointers, but they were coming from different parts of the code. The third thread described an example with no exception that worked fine until a user entered an invalid value, at which point, the error handling did not work as expected. While all of these posters had a problem with the same protocol, none of them would have been able to easily search the forum for their error to find existing solutions.

As seen, there were also three threads where a second poster appeared years later. In these cases, the poster came on simply to say that they had the same problem and that a search led them to this very helpful thread. All three noted how much time this had saved them. Again, this implies that developers will indeed search for a solution first and only post when a search turns up no useful answers. There are likely more developers who found these posts helpful and did not post in this manner.

Table 2. Archival analysis of Spring forum postings. These postings were understandable, solvable, on topic, and were not requests for a tutorial. The URL for each thread is http://forum.springsource.org/showthread.php?NUMBER. Data was collected in January 2011 on threads with their first posting before October, 2007.

Protocol	Number	Run time error	Run time local?	#Questioners	#Responders	Response time (H:MM)
1	13320	Unexpected behavior	Yes	1	1	46:07 (almost 2 days)
	21751	Unexpected behavior	Yes	1	2	121:03 (over 5 days)
	33139	Unexpected behavior	Yes	1	1	0:25
	33456	Unexpected behavior	No	3*	4	98:42 (over 4 days)
	36333	Unexpected behavior	Yes	1	2	15:07
2	26787	Exception	No	1	2	13:59
	36109	Exception	No	1	1	21:27
	43182	Unexpected behavior	No	1	2	68:39 (over 2 days)
3	23199	Unexpected behavior	No	1	2	51:22 (over 2 days)
	30206	Unexpected behavior	No	1	1	22:16
	43402	Unexpected behavior	No	1	2	20:56
4	17321	Unexpected behavior	Yes	1	1	20:51
	31120	Unexpected behavior	No	1	2	0:46
	33825	Exception	Yes	3*	3	46:07 (almost 2 day)
5	16150	Exception	No	1	1	114:51 (almost 5 days)
	28951	Unexpected behavior	No	1	2	146:39(over 6 days)
6	16766	Unexpected behavior	Yes	2*	2	3:09 (over 13 days)
	24527	Exception	Yes	1	1	22:29 (over 4 days)
7	28603	Exception	No	1	2	313:31
	39209	Exception	No	1	1	103:10 (almost 10 days)
8	33873	Exception	Yes	1	2	17:55
	36551	Exception	Yes	1	2	238:49
9	31927	Unexpected behavior	No	1	1	20:46
	43643	Exception	Yes	1	2	18:14
10	32429	Unexpected behavior	No	1	1	18:01
	39040	Exception	No	1	2	197:01 (over 8 days)
	13692	Exception	Yes	2*	1	57:27 (over 2 days)
	15048	Exception	Yes	2*	3	119:39 (almost 5 days)
	17967	Exception	No	4*	4	335:39 (almost 2 weeks)
	18245	Exception	Yes	1	2	33:07 (over 1 day)
	20688	Exception	Yes	1	1	72:06 (over 3 days)
	20932	Exception	No	1	1	16:48
	21764	Exception	No	1	1	21:56
	22020	Exception	No	1	1	2:51
	29849	Exception	No	3	3	40:47 (over 1 day)
	31857	Exception	No	2	2	90:39 (over 3 days)
	31989	Unexpected behavior	No	1	1	7:39
	33126	Exception	No	2*	2	6:43
	33168	Unexpected behavior	Yes	1	2	14:26
	33282	Unexpected behavior	Yes	1	2	15:20
	33317	Exception	No	1	1	172:43 (over 7 days)
	33491	Unexpected behavior	No	1	1	204:03 (over 8 days)
	33799	Exception	Yes	3*	2	119:23 (almost 5 days)
	34479	Exception	Yes	1	1	28:05 (over 1 day)
	34760	Unexpected behavior	Yes	1	1	20:30
	36891	Exception	No	1	2	68:38 (over 2 days)
	37090	Exception	No	2†	1	2:03
	38940	Exception	Yes	1	1	1:14
	39418	Exception	Yes	1	1	2:12
	39480	Unexpected behavior	No	1	1	4:38
	39725	Exception	No	1	1	1:37
	43259	Unexpected behavior	No	1	1	3:05
	43610	Exception	No	3*	6	30:36 (over 1 day)

* These additional questioners hijacked the thread on tangentially related issues.

† Posted over two years later: “Thanks guys, this saved me a hassle.”

4. Discussion

This work uses forum postings as evidence that these protocols are burdensome. However, it is difficult to actually prove this; we have to make assumptions about the behavior of developers before they post (eg: that they have exhausted all other resources) and assumptions about the generalizability of the developers who post on forums and on these protocols. This work can be used as initial evidence of a problem, and further studies should be done on both the complexity of typical protocols and the level of complexity that a typical developer would find difficult.

This work also did not describe why these protocols might be burdensome. [9] described several properties that are common to these protocols and could be contributing factors. In particular, most of violations described are of multi-object protocols and utilize non-traditional artifacts such as ASPX, JSP, and XML. These both contribute significantly to the complexity of the protocol in question, the ability to document the protocol in a clear way, and the ability of the system to provide more helpful error messages at run time.

We found that this methodology provides several benefits over doing lab experiments or developer interviews. It allows us to gain access to real developer questions that occur in the field, which would not be possible through user studies. While we could get this information through developer interviews and observations, mining forums allows us to collect data from more users and uses fewer resources to do so. Of course, the information gathered can be hard to interpret as we cannot ask follow-up questions for more detail. The methodology works well for more course-grained questions about how developers work that could be followed with more specific developer observations and studies.

There is a lot of data left to be gathered from these forums. For example, how many questions on object protocols go unanswered and why? Could we use forums as a way to detect the least-usable aspects of an API? What kind of automated systems might help users find posts about similar protocols? It is our hope that the wealth of information present in these communities can be used to answer these questions, as doing so would be a significant step towards improving the usability of object protocols.

References

[1] Archive of the asp.net and spring forum threads. http://www.cs.cmu.edu/ cchristo/docs/forumposts.zip.

[2] Jonathan Aldrich, editor. *Proceedings of the International Workshop on Specification and Verification of Component-Based Systems (SAVCBS '06), Iterator Challenge Solutions*, 2006.

[3] Nels E. Beckman, Duri Kim, and Jonathan Aldrich. An empirical study of object protocols in the wild. In *Proceedings of the25th European Conference on Object-Oriented Programming*, 2011.

[4] Kevin Bierhoff and Jonathan Aldrich. Modular typestate checking of aliased objects. In *Proc. of the Conference on Object Oriented Programming, Systems, Languages, and Applications*, 2007.

[5] Eric Bodden, Laurie Hendren, and Ondřej Lhoták. A staged static program analysis to improve the performance of runtime monitoring. In *Proc. of the European Conference on Object Oriented Programming*, 2007.

[6] Mark Gabel and Zhendong Su. Online inference and enforcement of temporal properties. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 15–24, New York, NY, USA, 2010. ACM.

[7] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In *Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '08, pages 273–284, New York, NY, USA, 2008. ACM.

[8] Marieke Huisman, editor. *Proceedings of the International Workshop on Specification and Verification of Component-Based Systems (SAVCBS '09), Database Library Challenge Solutions*, 2009.

[9] Ciera Jaspan. *Proper Plugin Protocols*. PhD thesis, Carnegie Mellon University, 2011. Technical Report CMU-ISR-11-116.

[10] Ciera Jaspan and Jonathan Aldrich. Checking framework interactions with relationships. In *ECOOP*, 2009.

[11] Viktor Kuncak, Patrick Lam, Karen Zee, and Martin Rinard. Modular Pluggable Analyses for Data Structure Consistency. *IEEE Trans. Softw. Eng.*, 32(12), 12 2006.

[12] Choonghwan Lee, Feng Chen, and Grigore Roşu. Mining parametric specifications. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 591–600, 2011.

[13] David Lo and Shahar Maoz. Mining hierarchical scenario-based specifications. In *Proceedings of the Conference on Automated Software Engineering*, 2009.

[14] Nomair A. Naeem and Ondřej Lhoták. Typestate-like analysis of multiple interacting objects. In *Proc. of the Conference on Object Oriented Programming, Systems, Languages, and Applications*, 2008.

[15] Mangala Gowri Nanda, Christian Grothoff, and Satish Chandra. Deriving object typestates in the presence of inter-object references. In *Proc. of the Conference on Object Oriented Programming, Systems, Languages, and Applications*, 2005.

[16] Arnd Poetzsch-Heffter, editor. *Proceedings of the International Workshop on Specification and Verification of Component-Based Systems (SAVCBS '09), Subject-Observer Challenge Solutions*, 2007.

[17] Michael Pradel and Thomas R. Gross. Automatic generation of object usage specifications from large method traces. In *Proceedings of the Conference on Automated Software Engineering*, 2009.