Improving Documentation for eSOA APIs through User Studies

Sae Young Jeong[1], Yingyu Xie[1], Jack Beaton[1], Brad A. Myers[1], Jeff Stylos[1], Ralf Ehret[2], Jan Karstens[2], Arkin Efeoglu[2], and Daniela K. Busse[3]

[1] School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA 15213
[2] SAP, AG
Walldorf, Germany
[3] SAP Labs, LLC Palo Alto, CA 94304
tooth2@gmail.com, clare.xie@gmail.com, jackbeaton@cmu.edu,
bam@cs.cmu.edu, jsstylos@cs.cmu.edu,
{ralf.ehret,jan.karstens,arkin.efeoglu,daniela.busse}@sap.com

Abstract. All software today is written using libraries, toolkits, frameworks and other application programming interfaces (APIs). We performed a user study of the online documentation a large and complex API for Enterprise Service-Oriented Architecture (eSOA), which identified many issues and recommendations for making API documentation easier to use. eSOA is an appropriate testbed because the target user groups range from high-level business experts who do not have significant programming expertise (and thus are end-participant developers), to professional programmers. Our study showed that the participants' background influenced how they navigated the documentation. Lack of familiarity with business terminology was a barrier we observed for developers without business application experience. Participants with business software experience had difficulty differentiating similarly named services. Both groups avoided areas of the documentation that had an inconsistent visual design. A new design for the documentation that supports flexible navigation strategies seem to be required to support the wide range of users for eSOA. This paper summarizes our study and provides recommendations for future documentation for developers.

Keywords: Usability, API Design, Service-Oriented Architecture, Web Services, Documentation, Business Solution Architects.

1 Introduction

"Service-Oriented Architecture" (SOA) is a way to structure large and distributed software systems, where services communicate over a network with the client and with other services, and can be combined into composite applications. Enterprise SOA (eSOA) is focused specifically on supporting business processes across an enterprise by reusing existing services. When an eSOA application is being planned and developed, many kinds of people are involved, some of whom are end-user

V. Pipek et al. (Eds.): IS-EUD 2009, LNCS 5435, pp. 86–105, 2009.

developers (EUD). For example, business process experts, who might be titled "Solution Architects," are knowledgeable about the business context but may not necessarily be professional programmers, and are often responsible for identifying and selecting which services will be used. Specifications they write will then be passed to professional programmers, who are responsible for writing code that uses the actual services. Therefore, the documentation and some of the tools must be accessible to both EUDs and professional programmers.

In a service-oriented architecture, code on the user's machine communicates with a remote service using messages across the internet. The communication is usually encoded in XML, and the format of the messages is usually described using a WSDL (Web Services Description Language) file, which has been formalized by the World Wide Web Consortium (see, for example, http://www.w3.org/TR/wsdl).

As part of the "Natural Programming Project" [11], we are interested in a whole range of usability issues around programming. Recently, we have begun to focus on the usability of libraries, frameworks, toolkits, and other application programming interfaces (APIs) [6, 14, 17]. APIs are crucial to professionals and EUDs alike, since most of modern development of all kinds involves finding, understanding, and connecting pre-built items, from small library calls to large-scale components. SOA APIs are particularly interesting to study, because they are often large and complex, and therefore expose interesting issues of scale, and because they often target a wide range of developers. As one typical example, we studied a sales order scenario from SAP's SOA services. SAP provides a large number of SOA services (over 2000) with interdependencies between services, and each service has many parameters, with interdependencies about which parameters are optional and required and what values are allowed based on values of other parameters.

Our previous research has shown that programming using eSOA APIs is not simple if APIs are providing access to powerful business functionality [1, 2]. Some barriers we identified included long names for services, different behaviors of services due to different business behavior, parameters of the services as hierarchies of objects with complex dependencies driven by internal, not exposed, business logic, and lack of example code [1, 2]. The current paper presents the results of a new user study of the usability of the online documentation provided by SAP for their SOA product.

In summary, our results are that when navigating eSOA API documentation, users with business backgrounds did better, and they experienced the most benefit from process component documentation. The process component documentation provides diagrams showing the architecture of an eSOA API in terms of service interfaces, the service operations they contain, and the business objects to which they are connected. All users avoided sites with visual designs that were inconsistent with their starting pages. Developers without business application experience were unfamiliar with business terminology and so they focused on searching and scanning for individual terms with limited success. Based on these results, we recommend that documentation provide flexible ways to navigate for different users with different backgrounds.

2 Related Work

Some of the first work on applying usability principles to APIs comes out of Microsoft, focused on specific APIs [4]. Inspired by this, we began working on the usability of

API design patterns [6, 15, 17], and the barriers to programming faced by EUDs, which includes the difficulties of identifying and understanding the appropriate resources in the documentation, which we called "Selection Barriers" [10].

We also reported on our previous studies of the usability of eSOA APIs for programming. We identified many barriers for installing and using the eSOA development environments, including issues with generating the stubs that will interface between the user's code and the XML messages that are required to communicate over the web, and issues with the long and confusing names of the services [2]. In a second study, we asked experienced programmers to use four services which had already been identified for them [1]. The current study complements these other works by focusing on the task of finding the services in the first place.

Many other people have provided recommendations and guidelines for APIs, but most of these are just based on the writer's intuitions or personal experience, rather than usability studies with users. For example, API designers with experience building the Java [3] and Microsoft .NET [5] APIs have published API design guidelines. For SOA, Jones lists anecdotal common mistakes made when developing SOA architectures, such as problems caused by service hierarchies that are either too fine-grained or too coarse-grained [9].

Focusing on documentation in general, Purho adapted Nielsen's 10 heuristics to apply to documentation [13]. Friendly [8] applied an informal methodology of user testing to JavaDoc, which is automatically generated from a Java project, and derived clear and succinct recommendations for future API documentation designers. Unlike JavaDoc, the eSOA documentation we studied contains a large amount of hand-created content and addresses a larger, more complex framework.

Others have focused on the internal documentation for projects, focusing on the software developers themselves (e.g., [7]), but this is not relevant for understanding how documentation for external users should be designed.

3 Methodology

3.1 Participants

Based on the success of our earlier studies [15], we decided to use an informal lab study with users who were representative of the target populations for the eSOA API. Since we were told by SAP that the target API was designed for developers with a wide range of expertise and background, we selected some experienced programmers with little business background, and some experienced business EUDs with little programming background, and some in between. We had 8 participants, all of whom were male Masters students at our university, although all but one of them had work experience before returning to school (see Table 1). The age ranged from about 25 to 35. None of the participants had ever seen the specific documentation web site we were testing, and none had used the API that the documentation was for. Of the 8 participants, 4 had significant experience with business application development using business software such as SAP, PeopleSoft and Oracle. All four of these participants had experience with Enterprise Resource Planning (ERP), which is one of the major areas of business software. Participant p2 had the most business application experience, having used the SAP development environment and SAP's programming

language called ABAP. The other 4 participants had no experience with business applications. Three of the participants had moderate to extensive programming experience with Java (4 years or more), and the others had some experience. All of the participants except p2 were enrolled in a Web Services course, but our study was performed before they had gotten very far along. This means that the participants all had an interest in SOA and had been introduced to some of the terms. Thus, we feel that subject p5 could be representative of new hires who might be assigned to do SOA work, p6-p8 might be representative of system integrators, p2 represents an all-around expert, and p1, p3 and p4 be representative of Solution Architects who have moderate knowledge of both business and programming. The experiment lasted about two hours, and the participants were each paid $20.

Table 1. Characteristics of the participants in the user study, and whether the search feature was available when they performed the study

	p1	p2	p3	p4	p5	p6	p7	p8
Years of Work Experience	3	3	3	1	0	2	1	2.5
Business Application Experience	yes	YES	yes	yes	no	no	no	no
Years Programming Experience	2	3	3	4	2	5	4	2.5
Were able to use Search				yes				yes

3.2 Tasks

The tasks for this study were to find the specific services necessary to perform a "Create Sales Order" using the documentation. The participants did not use any programming tools such as an Interactive Development Environment (IDE) and did not have to produce any code. They were shown the introductory page of the documentation web site and were given a brief tutorial (about 10 minutes) describing the document layout including the various ways to navigate away from the front page. We told participants that they should consider themselves to be high-level architects in a company that was planning to implement a new sales order system using an existing ERP system. They should find the services needed to create a sales order, starting from the string names of a buying company, a selling company, and a product. They did not need to actually implement an application; they only needed to identify the correct services so that another developer could later implement the system. They were given about 2 hours to finish all tasks.

One challenging part of this task was that when participants read the "Create Sales Order" service documentation, they would discover that this service does not take string names for the seller, product and buyer, but instead takes IDs, which the participants had to find other service calls to look up. Therefore, successful task completion required finding four services we refer to here as "Create Sales Order", "Find Customer", "Find Supplier", and "Find Product" (although the actual names were much longer and less clear). Participants were not told in advance about the need for multiple inter-related services. Discovering that inter-related services were necessary from the documentation was an essential part of the task.

During the study, we used the "think-aloud" protocol, in which participants are encouraged to talk to themselves and the moderator, because we were interested in gaining insights as to what participants were thinking while performing the task. In order

to be able to gain as much useful information from each participant as possible, after seeing enough confusion to confirm that the participant was experiencing a usability breakdown, we would offer help so the participant would not remain stuck on one problem for the entire session. We wanted to know why problems occurred, not the length of delays. However, explicit help was relatively rare because it was difficult to give advice without giving away the whole solution.

3.3 Context – SOA Documentation

The participants used the actual then-released (February, 2008) online documentation of SAP's eSOA product. Based in part on the results of our study, SAP has since improved the site significantly, and many parts now no longer match what the participants saw, which is described below.

There are several different paths that participants could use to navigate from the starting page down to the pages of individual service operations (see Fig. 1). One path grouped services into Enterprise Service (ES) "Bundles" that collected together a set of services that are often used together. The ES Bundles navigation path was implemented using a user-editable Wiki, so that users of the documentation could add and update the bundles to show what services were actually used together in practice. Since this navigation path led to a Wiki, the visual formatting of these pages was quite different from other parts of the documentation web site. The bundles contained a list of service operations, and from there, users could eventually navigate down to the individual services, at which point they would leave the Wiki and return to the previous "normal" format.

A second path used the Enterprise Service Index, which listed business "process components" in alphabetical order. The Process Component pages each contained a

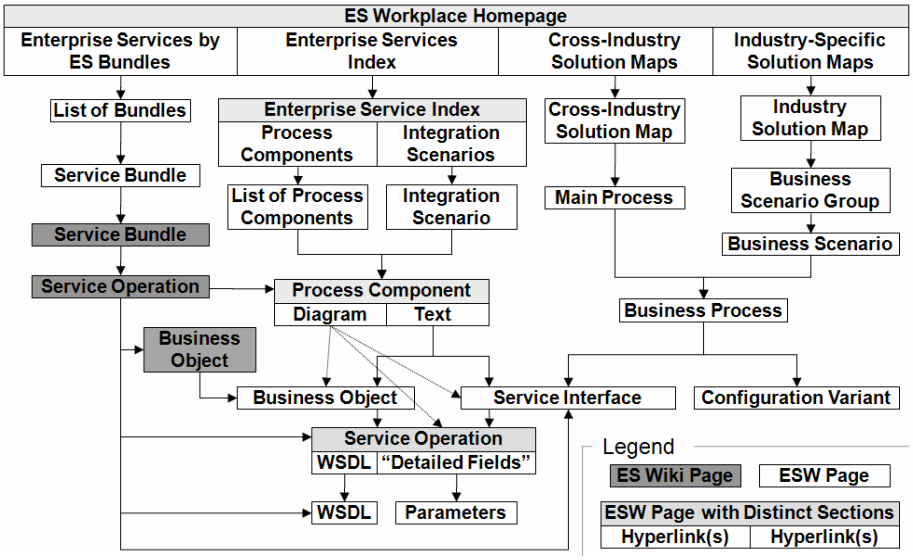


Fig. 1. Different possible navigation paths

Table 2. Descriptions of webpage content for pages shown in Figure 1

Documentation	Content
Solution Map	Business value chains displayed as colorful diagrams. Colored bars hyperlink to processes and scenarios. May apply across industries (ERP, CRM, etc.) or for one industry. (Oil, Retail, etc.) (Fig. 2)
Scenario Group	Similar to a Solution Map, but specific to a part of one industry.
Main or Business Process, Scenario	Text description of a business process or scenario. Hyperlinks lead further down, but often do not link to Service Interfaces.
Configuration Variant	Text description of business use cases that may not be intended for developers, but rather business analysts. Hyperlinks go only to other Configuration Variants, and upwards.
Process Component	This page contained both a diagram and text. The diagram linked to a group of Business Objects, and all Service Interfaces and Service Operations using those Business Objects. Text links below the diagram went to the Objects and Interfaces only. (Fig. 3a)
Service Interface	Text hyperlinks to some Service Operations sharing a Business Object, which may have one or more Service Interfaces.
Service Operation	Description of a service operation. Hyperlinks to the service WSDL and parameters. (Fig. 3b)
Business Object	Description of a distinct business "entity" (such as a sales order, supplier, etc.) with links to Service Operations acting upon it.
WSDL	XML file that describes the service in machine-readable form.

process component diagram and a textual description of the process component (see Fig. 3a). From the process component diagram, participants could then navigate to the relevant business objects and service interfaces. Participants could navigate using hyperlinks located in the process component diagram or in a table below the diagram listing the contents of the process component as text.

The third and fourth paths used two different kinds of graphical tables called "Solution Maps." The cross industry solution maps (see Fig. 2) provided categories such as ERP (Enterprise Resource Planning), CRM (Customer Relation Management), SCM (Supply Chain Management), and then at the next level, groups of services such as "analytics," "financials," and "sales and service.". The industry solution maps provided categories like "Retail," "Airlines," or "Oil & Gas", and then groups of services such as "Sales & Marketing" and "Vehicle Maintenance". As shown in Fig. 1, all of the links in the Solution Maps lead to "Business Process" pages. Unfortunately, some of these Business Process pages linked only to Configuration Variants, and not to the Service Interfaces that link to Service Operations pages and the technical information necessary for implementation. The Configuration Variant pages were dead ends and apparently not intended for use by developers. Instead they linked to other variants or back to Business Processes, so this path proved useless to our participants.

Once participants had navigated down to the "service operation" page (Fig. 3b), they could find out information about the specific service, including the WSDL files to download. On each of the service operation pages, there was a hyperlink to a separate "Detailed Field" page with collapsible tree hierarchies of the input and output parameters for calling the service. Since the web services can be accessed from a variety of programming languages, coding examples were not provided in the API

documentation. Instead, a browser-based service "testing jig" was available. By showing all available fields of the complex input and output parameter structures of the web services in a tree view, this testing jig allowed users to test service consumption with real values. However, at the time, the only link to the testing jig was provided inside a "Handbook" PDF guide hyperlinked from the main starting page of the documentation. This guide provided an end-to-end walkthrough of the documentation site and screenshots of pages along the navigational paths.

When we began this study, the web site had a search box, but it appeared to be inoperable, in that all searches returned no results. By the time we ran the last two subjects (p5, p8 – see Table 1), the search seemed to be fixed and began working.[1]

In summary, the documentation provided several architectural description pages to help understanding of the overall architecture. Table 2 shows some of the different architectural description pages, and their content.

4 Results

Table 3 shows a summary of the overall results – only two of the 8 participants (25%) were able to find all of the services during the two-hour session. Three of the four participants with business backgrounds (75%) were able to find the correct first service ("create sales order"), however one was not sure that he had found the right one. Similarly, two of the participants without business backgrounds found the right sales order service, but were not sure, and none found any of the other services. Since there are such a small number of participants, we are not able to establish statistical significance between the two groups, although the trend is striking. From our observations and the think-aloud comments of the participants, we were able to understand the participants' strategies and barriers at a much more detailed level.

4.1 Paths through the Documentation

Given the four starting entry points for navigating from a home page (Fig. 1), participants were confused with which one to use, and spent significant time reading text on the home page to try to figure it out. The main page did not explain the motivations and goals of the four different paths, leaving participants confused about why there were multiple choices and which might be the most useful. This confusion made participants feel frustrated right at the beginning. Table 3 summarizes where the participants started.

An interesting observation was the use of what we call *rally points* by participants while navigating through unfamiliar areas (see Table 3). Participants would choose a path, go down that path until they decided whether or not it was worth continuing, and then return to an earlier page multiple times. Participant's selection of a rally point indicated a level of certainty that the navigation up to that point, at least, was correct.

Fig. 4 summarizes the paths of all of participants when trying to find the Create Sales Order Service. Each row represents a type of web page, as described in Table 2. Each circle represents web pages that the participant visited, with the size of the circle

[1] A hazard of using a commercial on-line system for a study – one cannot guarantee all participants will have the identical system!

representing how long the participant stayed at that page. In Table 3 and Fig. 4, we can see that the page at which the participant started was a natural rallying point at first, but participants would move the rally point around as they gained and lost confidence in the usefulness of various paths through the documentation.

Most participants showed a tendency to choose the Solution Maps as a starting point (as shown in Table 3), but five of the participants changed to the Enterprise Service Index after failing to use the solution maps. The Enterprise Service Index page only provided process component lists and integration scenario lists in alphabetical order. In the process component lists, there were prominent business software categories such as CRM, ERP, SCM and SRM. Participants with business application backgrounds used the "Enterprise Service Index" pages as a rally point, and when they found the "Sales Order Processing" component in the ERP and SRM category, they felt they were on the right track. Most participants were frustrated by new and unfamiliar terms and acronyms, but participants without business application backgrounds were particularly confused by the large number of prominent acronyms such as ERP, CRM, SCM, SRM and other business-specific terms that they did not understand.

When participants navigated to the Enterprise Service Bundles page (which was a Wiki), they were surprised by the different look and feel of this part of the web site, and felt they must have gone astray, so they quickly back-tracked. None of the participants made use of the Bundles pages, so they do not appear in Fig. 4.

Participants spent a lot of time trying to use the solution maps (Fig. 2). Some participants selected cross industry solution map, possibly because they were not told about any specific industry in the task instructions, but others guessed an industry they thought might be appropriate, and used an industry-specific map. However, the

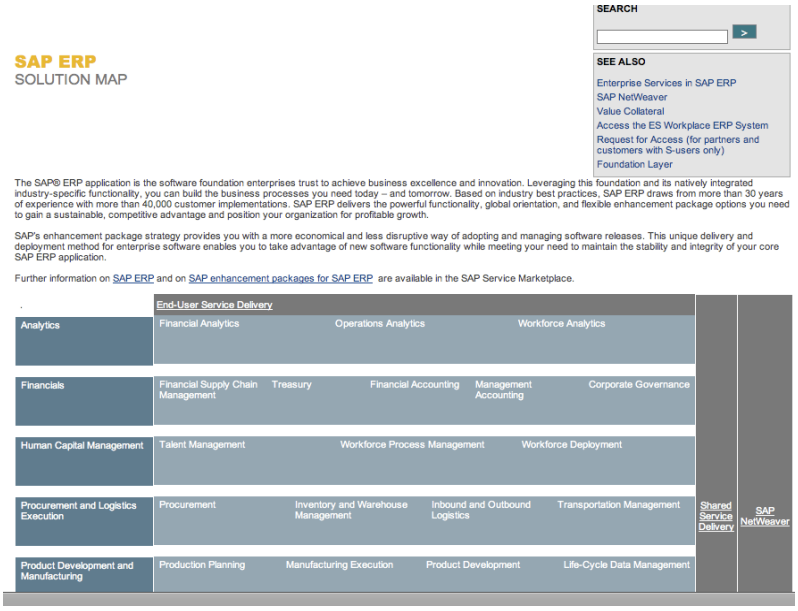
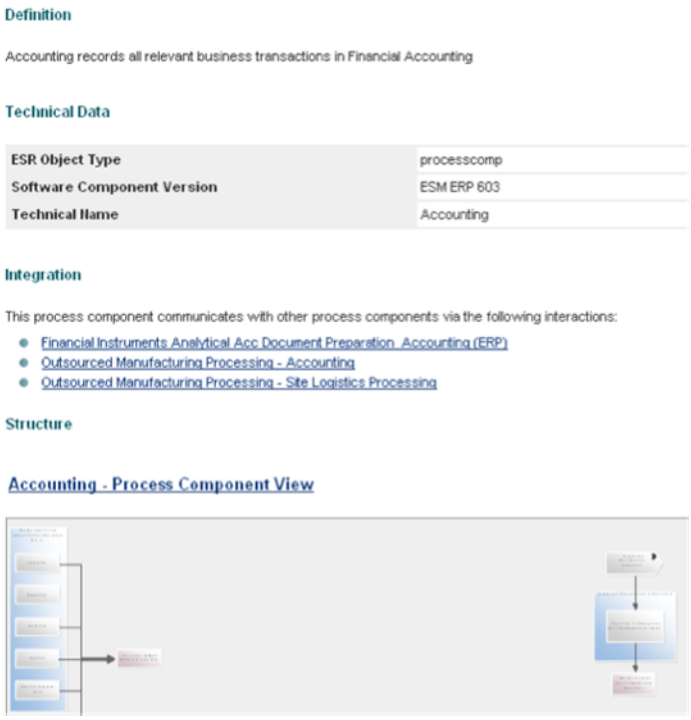
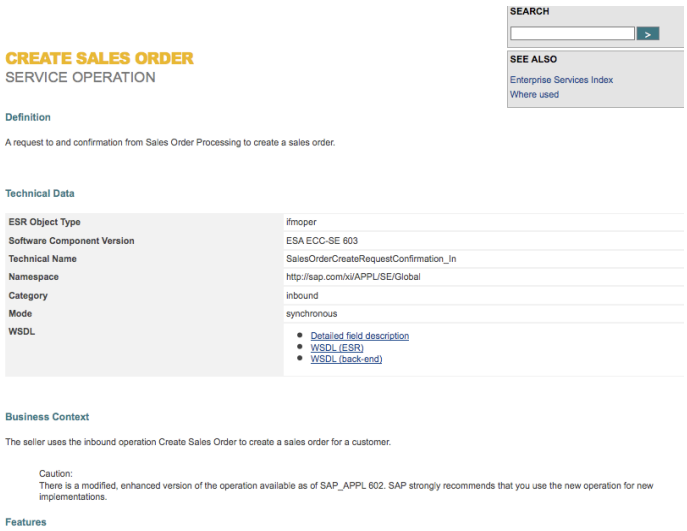


Fig. 2. Cross industry solution map for ERP



(a)



(b)

Fig. 3. (a) Process Component View Page for the Accounting process component which includes a diagram and text below the diagram (not shown). (b) Service Operation page for Create Sales Order.

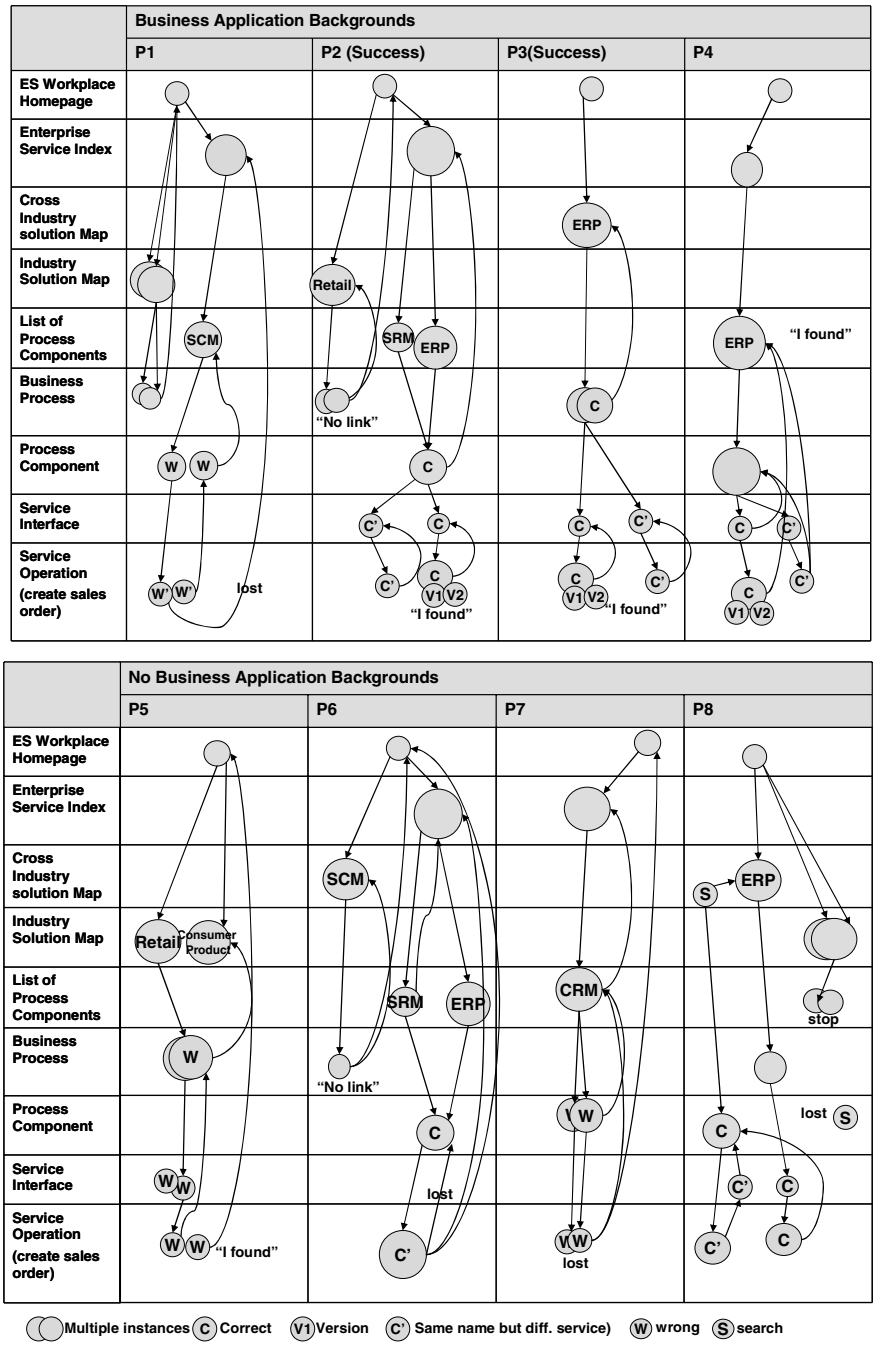


Fig. 4. Summaries of the navigational paths of all of the participants when trying to find the Create Sales Order service. The sizes of the circles represent the amount of time spent at web pages of the type in the first column.

Table 3. Starting and rally points for the participants (using page categories from Fig. 1), and success of participants on finding the 4 services.

Key: **M**=Solution **M**ap; **S**=**S**earch; **I**= ES **I**ndex, **P**=**P**rocess Component, **L**=**L**ist of Process Components, **B**=**B**usiness Process
✓=Success; √- =Success but not sure; **X**=Failure

	p1	p2	p3	p4	p5	p6	p7	p8	Total
Was able to use Search				yes				yes	
First Entry Point	M	M	M	I	M	M	I	S	
Other Rally Points	I,P	I,P	B	L,P	M,B	I,P	L,P	M,P	
Found Correct Service Operation:									
Create Sales Order	X	√	√	√-	X	√-	X	√-	62.5%
Find Customer Service	X	√	√	X	X	X	X	X	25%
Find Supplier Service	X	√	√	X	X	X	X	X	25%
Find Product Service	X	√	√	X	X	X	X	X	25%

participants without business backgrounds had difficulties in using the any of these solution maps to navigate further due to the unfamiliar terminology and the large number of choices making a brute-force systematic search difficult. However, half of the participants without business backgrounds used a map page as a rallying point (see Fig. 4). In the think-alouds, the participants expressed a desire to understand the "big picture," and the solution maps seemed to provide a good overview. The business-background participants understood the category names such as ERP and SRM, and their sub-grouping such as "financials," "retail," etc., but even these participants often only had experience with some of the categories and sub-groupings. However, all participants were confused by classifications with similar names such as: "sales", "sales execution", "sales order" and "sales & service" in the solution maps.

4.2 Process Component View

The Process Component view shows one or more related business objects and services (see Fig. 3a). For example, in the "Sales Order Processing" process component view, the user can navigate to the "ordering in" and "ordering out" service interfaces and the "sales order" business objects. The page was composed of two parts: a diagram, and a table. The diagram displayed business objects as small blocks and service interfaces as large blocks that held groups of smaller blocks representing service operations. The service operations were connected to the business objects they acted upon with arrows. The titles of the blocks acted as hyperlinks to the appropriate business object, service interface, and service operation pages. Due to the large number of objects to be shown in the diagram, the font of the elements was extremely small and yet horizontal and vertical scrolling was still needed.

In spite of these barriers, some participants spent an extensive amount of effort trying to understand the diagrams. Many of the participants found this view to be a good rallying point, since it provides a well-organized collection of related items to explore. However, some of the participants who were familiar with UML (Unified Modeling Language) notation mentioned that they would have preferred UML class diagrams, which have a standard notation for classes and their relationships.

Another cause for confusion was that the system provided many similar-sounding services in the process component view, and even multiple versions of the same service with similar names such as “create sales order v1”, “create sales order v2”, and “check sales order creation”. Participants could not find any relevant information to differentiate those three different versions of services from the process component view. The participants had to drill down to the service operation level for each, to try to determine which should be used. If the user could recognize the differences among these different services at the process component view, this would have saved significant time and confusion.

Beneath the diagram, a table contained text descriptions and hyperlinks to many of the same locations as the diagram, with the exception of the service operations.

4.3 Service Descriptions

In the tasks we gave the participants, it was important to investigate the input and outputs of the various services. However, this was difficult to verify from the detailed service pages. Only three participants were able to find the “buyerID”, “sellerID” and “materialID” parameters for the “Create Sales Order” service operation, which was crucial to determining what other services were needed.

Other problems with understanding the services included unfamiliar technical terms such as synchronous and asynchronous mode and inbound and outbound messages. The participants did not find any explanations of these terms in the documentation, although they are pervasive throughout all services. Some of the details of the operation, such as which fields were required versus optional, were actually not documented anywhere except in the generated WSDL XML files themselves, which was too long and difficult to read to be effective documentation.

The detailed page for each service listed three classes of messages: input message, output message and fault message, which participants did not understand. In fact, only input messages are relevant (messages that go “in” to the server), but this was not explained anywhere.

In general, many participants found the correct target service, but then were unsure whether it was correct or not, and continued searching. For example, Table 3 and Fig. 4 show that participants p4, p6 and p8 found the right service operation for Sales Order, but then navigated away and kept looking. Participant p4 eventually decided that a *different* service was actually right, and p6 and p8 were never confident of which service should be used.

4.4 Using Search

As mentioned above, the search box was present for all participants, but only began working for the last two (Table 1). All of the participants expressed a desire to use search to try to find the services. In general, if the participants knew the name of what they wanted, they preferred to use search, and the participants for whom search worked often returned to try searching when they were lost. Participants often tried to search for phrases we used in the instructions, such as “create sales order”, “selling company”, “buying company” and “product”, but these were not helpful, and then participants tried related terminology such as “agency”, “supplier”, “customer,” etc.

In general, participants were not successful at using search because there were always either no results or too many matching results. Even the most experienced participants had difficulty mapping the product in the instructions with the actual parameter name of "material".

When search began working, the results were presented grouped by the various API documentation types shown in Table 2, such as solution map pages, process component view pages, and service operation pages. This grouping proved helpful to participants, and made it easy to find the appropriate process components and business objects when they recognized them in the results. However, since there were often too many search results, and the listing was in alphabetical order, often participants missed the answer even when it was included.

4.5 Individual Strategies

By performing a detailed time analysis of each participant, we were able to break down their activities into various categories. We identified four categories of activities, with two opposing strategies in each:

- Focusing on scanning textual descriptions ("scan text") vs. focusing on scanning process diagrams ("scan diagrams").
- Trying to understand how to use the web site by reading the provided PDF documentation ("PDF overview"), or just by looking through the web site itself, relying on the documentation to be self-explanatory ("Self-explanatory"). Five participants found the PDF document but three of them did not use it, because it was a separate document.
- Browsing the documentation with a single specific key word in mind from the task instructions, such as "buyer" ("Single word"), or else using a set of interrelated synonyms ("Synonyms").
- Skimming the documentation focusing on only the prominent text, such as the headers ("Skim"), or systematically reading the pages line-by-line ("Line-by-line").

We then analyzed each of the participants, looking for whether they tried to use each of these strategies, and whether it worked for them. Note that each participant might have used different strategies at different times. Fig. 5 provides a radar chart averaged over all participants for the strategies. The opposing strategies are shown at opposite ends of each line. The outer black line (connecting the circles) shows the average of whether this was used or not (where 1 would mean everyone used it, and 0 would mean no-one used it). The inner red line (connecting the squares) shows our estimate of how successful this strategy was.

Fig. 5 makes it clear that participants were split on using text and diagrams, they strongly preferred the documentation to be self-explanatory, rather than using the PDF overview, more tried single words rather than synonyms, and everyone skimmed, but only a few systematically read line-by-line. As for the success of these strategies, by-and-large, the success seemed to mostly correlated with participants' expectations (they used a strategy about as much as it was successful, so the two lines go in and out together), with the notable exception of the diagrams – as discussed above, many participants wanted to use these, but they did not work out for them. Another notable result is that the PDF overview was surprisingly un-helpful.

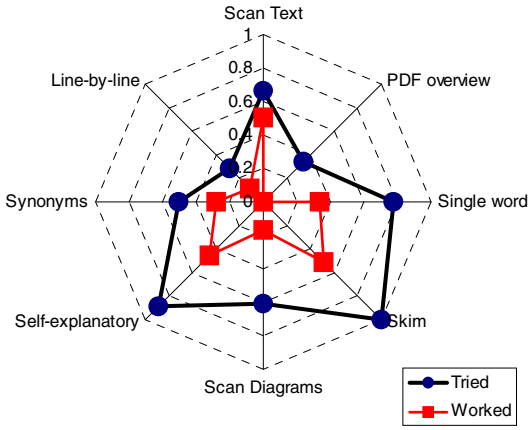


Fig. 5. Strategies the participants tried, and how well each strategy worked

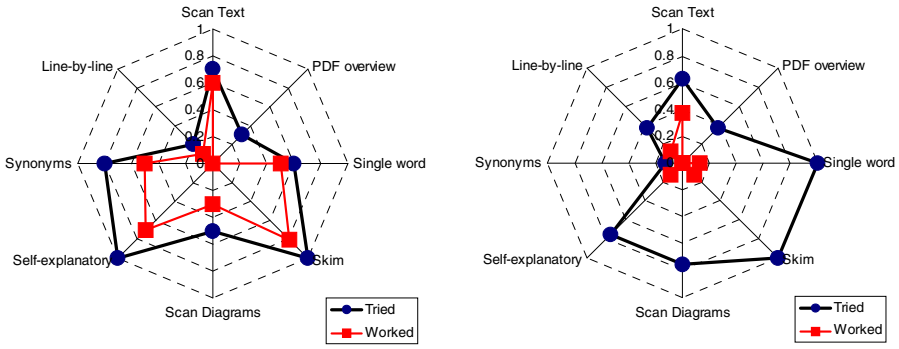


Fig. 6. Breakdown for participants with (a) and without (b) business backgrounds

Fig. 6 provides the same data broken down by whether the participants had business background or not. In terms of what they tried, it is clear that the non-business participants did not use synonyms (because they did not know the other terms that might be related), and the non-business participants were more systematic, trying to extract more meaning from the pages (whereas the business participants were more likely to be able to pick up the meaning from skimming). It is clear that few strategies worked for the participants without business backgrounds, and only scanning the text was overall successful.

5 Threats to Validity

There are many reasons why the results of this study may not generalize. First, we only used a small number of participants, and we were not able to get statistically significant results about their different behaviors. Most of the results reported here are impressions and informal analyses based on our observations of their behaviors,

barriers and successes. The participants are also not necessarily representative of the target population for the documentation. eSOA APIs may be used only by people with some business background or people who have specific, relevant training. For example, SAP offers various training courses that would have explained many of the fundamental terms about which the participants were confused. Our participants were all completely unfamiliar with the documentation or the API.

The experimental set-up may have also biased the results. In real-life, users would have more than two hours to perform tasks, and they would likely go to more experienced colleagues for help when they were stuck, which was not an option in this study. Also, our task was much simpler than real-world eSOA tasks.

6 Discussion

In spite of these concerns, we feel that useful information can be learned from our study. As with other usability analyses [12], when multiple user-study participants have difficulty with something, it is highly likely that at least some of the target audience will also have trouble, so the documentation is likely to be improved by eliminating the barriers reported here.

The differences in strategies and success between the people with business experience and those without are also interesting. These can mainly be attributed to the differences in their ability to understand the many terms and acronyms used in the documentation. Participants with business backgrounds were aware of interrelated business concepts and terminology, and so understood more explanations on the web site. The navigational strategies were also very different between the two groups.

Of the four ways to navigate to the service operation, the ES index was found to be most useful to many developers, who then used the process component diagrams as a rally point. The graphical solution maps were frequently used by all participants, but tended to lead developers to the wrong services. The frequent use of the maps and process diagrams strongly suggests that a good diagram of the system is important to users. The presence of the many alternative navigation paths was itself a barrier to participants, since they had to investigate which one to use.

This study particularly focused on identifying services based on their input/output characteristics, but this turned out to be surprisingly difficult to determine from the documentation. Our previous study showed that understanding the dependencies among the parameters is also a key barrier to developers [1], since which parameters are required and which are optional depends on the values supplied for other parameters. This means that more attention is needed on documenting the parameters of services, where it is possible[2].

A consistent look and feel for the documentation was found to be important. When participants encountered the different format of the Wiki, they immediately

[2] Some services deal with highly customizable business processes. Customer can set up the system for their special business needs and therefore the behavior of the services can change from customer to customer. So a “create sales order” service can be used in a simple retail scenario, where you just buy 100 pencils, as well as in the aerospace industry when you order 20 Airbus A380 airplanes, which have quite different requirements.

backtracked without studying the new location. As a result, the grouped services on the wiki went unseen and unused by all of the participants.

The names of services and their types were found to be a problem. We observed one participant confused with several versions of "create sales order v1" and "v2", the differences between the same service name with "in" and "out" appended, and also with the difference between the "synchronous" and "asynchronous" versions of the same service. Another problem was the length and construction of the names themselves, which some participants found confusing (for example, SalesOrderERPCreateRequestConfirmation_In_V1).

7 Implications for Design

How can the documentation be designed to best serve developers across the whole spectrum? Improvements in the usability of the documentation are clearly necessary if users such as our participants are to succeed. We are happy to report that many of these recommendations have been implemented in the current version of the SAP documentation, and others are being investigated for future versions.

Based on our observations and user study results, we recommend the following as documentation guidelines:

- **Consistent Look-and-Feel.** Overall, the entire documentation web site should have a consistent, yet unique, format, so that developers who leave the path know it instantly, and developers who find a useful area do not backtrack. This may mean that more developer participation in a Wiki might occur if its format is not visibly different from the rest of the API documentation.
- **Provide an Overall Map.** When we were trying to understand the SAP documentation ourselves before we ran the user study, we created early versions of Fig. 1 and Table 2, which we found very helpful. Having such information at the front of the documentation web site would likely benefit users.
- **Explain Starting Points.** Make the purpose of various starting points clear. It seems that some of the paths on the eSOA documentation may be targeted at different classes of users, such as Business Experts versus developers. Alternatively, they might be used for different tasks. Users would benefit from a better explanation of *why* there are multiple paths, and how they are intended to be used.
- **Provide "bread crumbs" the documentation structure.** Users were often lost in the documentation. Providing a trail that shows where they are in the documentation structure, and what are the main nodes along the path to that page, would be helpful. However, the documentation is a graph and not a tree because some paths are in multiple paths (e.g., the same service may be used by multiple industries). This means that the trail will have to be careful to differentiate multiple possible paths to the current page, hopefully highlighting the path actually used.
- **Directly support rally points.** In addition to the bread crumbs, there should be other support for users to backtrack to well-known pages that are serving as "rally points". For example, we created a prototype which included an always-visible bookmark list into which the user could easily save pages while continuing navigation, and then these could serve as shortcuts for navigating back to a rally point. Another idea is to provide pages that *other* users or the system designers have identified as useful rally points.

- **Integrate "How-To-Use" Information.** We discovered that although a PDF guide explained how the documentation could be used, users were reluctant to leave their browsing to read a document in an external format, so the explanations should be in html format. Even better would be if the documentation was self-explanatory, with explanations integrated with the main documentation content, so there would be no need for separate documents explaining how to the API documentation. For example, pop-ups or special hyperlinks might explain "What is this?" for items that users may not be familiar with.
- **Effective Search.** The participants for whom search did not work were unhappy, so a good search facility needs to be part of all documentation. Since participants tried to search on all aspects of services, all parts of the API should be included in the search, including the parameter names and types, and the documentation of the names and types. It should be easy to navigate from data types to the fields that use those types. In order to reduce the size of the answers, the search should allow users to qualify what they are looking for (e.g., limit the answers to service operations). The grouping of the search results into categories is a good idea, but each result should be presented in a way that is easy to understand, so the user does not need to navigate into each result item to see if it is the desired one or not.
- **Provide Familiar Diagram Formats.** Our participants expected UML Class diagrams or other well-known architectural presentations to help them understand the services. Users should be surveyed on what formats they will find familiar before the decision is made to create new formats.
- **Balance of Diagrams and Text.** Some participants focused on diagrams showing the relationships among services, so these need to be clear and concise, with appropriate labels that are understandable yet not too big. At the same time, other users skipped the diagrams in favor of text, so both should be supported.
- **Curtail User Focus on Esoteric Terminology.** Specialized terminology for specialized users and use cases is absolutely necessary in API documentation. However, we observed that participants who are exploring tend to focus on unfamiliar terms, even if they are unnecessary as part of their task, and so waste time while increasing their level of confusion and frustration. However, most users will (at least eventually) be familiar with the terminology, so it is important that any definitions or other help not interfere with expert use. It is also important that users be able to quickly tell what parts of the document are important to them, so they can skip large parts (and any unfamiliar terminology in those parts).
- **Explain Crucial Terminology.** Participants could not find the correct services in our study without understanding the difference between synchronous and asynchronous services, or the meaning of "in" and "out" services. To the extent that all users must understand certain "esoteric" terminology, make sure it is clearly explained, or even better, use more generally-understood terms so less explanation is needed.
- **Make the Parameters for Services More Prominent.** Participants cited the parameters of the service signature as the main indicators of the usefulness of a service. Therefore, parameters should be given a prominent position in the description of a service operation. Our previous research showed that the distinction between optional and required parameters, and parameters used to call the service and those filled in by the service as return values was not clear to developers [1]. This needs

to be particularly well explained, and certainly not left to be deduced from the WSDL files.

- **Support Comparing Services.** There are many similar services, and participants needed to compare services to find out the differences. In the current system, sometimes they needed to open up the low-level WSDL files and try to manually determine the differences. Instead, direct comparisons and explanations should be available to differentiate services. For example, side-by-side visualizations of two services might emphasize the differences in parameters or actions. If a service is an updated version of another service, the modification dates and differences should be clear with cross-links and explanations of when each might be used.
- **Clear Names for Services.** The user should be able to recognize what a service does by its name. If there are multiple versions, it should be clear why there are multiple versions, and whether they are all intended to be useful (vs. some being deprecated, for example), and which one should be selected.
- **Present Related Services.** The documentation should present related services and business objects. For example, to create a sales order required providing three different parameters that were returned by other services. Listing each of these parameters and services could help the user understand and find related services. The Bundles idea in the current documentation may help with this goal, but we were not able to evaluate how well it worked because our participants did not try the Bundles.
- **Provide Code Examples.** While web services are often advertised for their ability to be consumed in any programming language, this does not excuse the provider from showing sample code snippets. Even if it is not possible to provide example code in every target language, then it is still useful to provide some examples rather than none. It should be noted that standardization across similar services will mean that fewer examples need to be provided, because a pattern that works for one service will also work for its "sibling" services.
- **Online Service Testing.** Developers who want to see how a service works before starting to program may benefit from an interactive way to provide parameters and run the service. The current SAP documentation does have a "Test" function, where users can try out a service and see what it returns for various parameters. This kind of online service testing can have a positive effect on developers' understanding of web service consumption. It has the potential to display required and optional parameters, and allow users to verify their understanding of the service. However, without valid test data to use as parameters, the user may never be able to get a useful return value. Therefore, the testing mechanism should be combined with multiple examples, and cross-linked to other services that might return the kinds of values required for the service to operate correctly. Furthermore, once the user has configured a test call interactively, it would be useful if there was some way to generate code in the desired target language that would do the same thing.
- **New Organizations for Hierarchical Browsing.** In their think-alouds, we noticed that various participants, especially the ones without business experience, seemed to be trying to navigate based on different starting points and hierarchies. For example, some participants seemed to be trying to find particular operations (verbs) first (such as create or find), then the objects on which those operations occurred (the nouns), and finally, other parameters of the operation (adverbs such "by what" the find should get the object, or "using what" to create the object). The current

documentation does allow users to start from a business object and find all of its services, or to get a global list of services, but these are always organized alphabetically. Since the services are named based with the affected business object at the front of the name (e.g., SalesOrderERPCreateRequestConfirmation_In_V1), both lists are essentially noun first. Allowing a sort by operation (sorting all the "create" service operations together) might be helpful.

8 Future Work and Conclusions

This informal study is just the beginning of a long investigation into improvements that can be made to API documentation. We are currently working on interesting new designs to see how we can make documentation even easier to search and browse, and how to make the important information more salient (e.g., [16]). For example, more commonly used items in the documentation might use a bigger font, so they stand out compared to the lesser used items.

Meanwhile, SAP is continuing to improve their APIs and the documentation for them. We plan to repeat this study with the new designs to see what problems have been solved, and if there are any new problems introduced. In the new study, it will be interesting to investigate more classes of users, from Business Expert EUDs with little programming experience to experienced programmers, and hopefully get some non-university participants from local businesses. It would also be useful to compare people who are expert users of the system and documentation to the novice users that we focused on in this study. Good documentation should also be efficient for experts, as well as helpful for novices.

In addition to providing insights into how to improve the current documentation, these kinds of studies can provide generalizable knowledge that is useful for all documentation writers for all kinds of systems, since many of the challenges will be similar. Since all programmers, from EUD to novices and to professionals, spend significant time trying to understand and use APIs, improvements to documentation can have significant impacts on the overall usability of the system as a whole.

Acknowledgements

For help with this paper, we thank many people at SAP (especially Paul Hofmann, Dan Rosenberg, Ike Nassi, Claudius Fischer, Bernhard Drittler, and Oliver Schmidt) and the participants for sharing in user study. This research was partially funded by a grant from SAP and partially by NSF under grant ITR-0325273 through the EUSES Consortium, and CCF-0811610. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NSF or SAP.

References

1. Beaton, J., et al.: Usability Challenges for Enterprise Service-Oriented Architecture APIs. In: 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, Herrsching am Ammersee, Germany, September 15-18, pp. 193–196 (2008)

2. Beaton, J., et al.: Usability Evaluation for Enterprise SOA APIs. In: 2nd International Workshop on Systems Development in SOA Environments, SDSOA 2008 (Co-located with ICSE 2008), May 12, pp. 29–34. Leipzig, Germany (2008)
3. Bloch, J.: Effective Java Programming Language Guide. Addison-Wesley, Boston (2001)
4. Clarke, S.: Measuring API Usability. Dr. Dobbs Journal, S6–S9 (May 2004)
5. Cwalina, K., Abrams, B.: Framework Design Guidelines. Addison-Wesley, Upper-Saddle River (2005)
6. Ellis, B., Stylos, J., Myers, B.: The Factory Pattern in API Design: A Usability Evaluation. In: International Conference on Software Engineering (ICSE 2007), May 20-26, Minneapolis, MN, pp. 302–312 (2007)
7. Forward, A., Lethbridge, T.C.: The relevance of software documentation, tools, and technology: a survey. In: DocEng, McLean. pp. 26–33 (2002)
8. Friendly, L.: The design of distributed hyperlinked programming documentation. In: International Workshop on Hypermedia Design, June 1-2, pp. 151–173. Springer, Montpellier (1995)
9. Jones, S.: SOA Anti-Patterns. Jun 19, C4Media Inc.: InfoQ.com (2006), http://www.infoq.com/articles/SOA-anti-patterns
10. Ko, A.J., Myers, B.A., Aung, H.H.: Six Learning Barriers in End-User Programming Systems. In: IEEE Symposium on Visual Languages and Human-Centric Computing, Rome, Italy, September 26-29, pp. 199–206 (2004)
11. Myers, B.: Creating More Natural Programming Languages. In: VL 2000: IEEE Symposium on Visual Languages, Seattle, Washington, September 10-14 (2000) (Invited Keynote Address), http://www.cs.orst.edu/~burnett/vl2000
12. Nielsen, J.: Usability Engineering. Academic Press, Boston (1993)
13. Purho, V.: Heuristic inspections for documentation-10 recommended documentation heuristics. STC Usability SIG Newsletter, 6(4) (April 2000), http://www.stcsig.org/usability/newsletter/0004-docsheuristics.html
14. Stylos, J., et al.: A Case Study of API Design for Improved Usability. In: 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, Herrsching am Ammersee, Germany, September 15-18, pp. 189–192 (2008)
15. Stylos, J., Clarke, S.: Usability Implications of Requiring Parameters in Objects' Constructors. In: International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, May 20-26, pp. 529–539 (2007)
16. Stylos, J., Myers, B.A., Yang, Z.: Improving API Documentation Using API Usage Information (submitted, 2009)
17. Stylos, J., Myers., B.A.: The Implications of Method Placement on API Learnability. In: Sixteenth ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE 2008), Atlanta, GA, November 9-14, pp. 105–112 (2008)