How to Get a Paper Accepted at OOPSLA (PANEL)

Ralph E. Johnson, *University of Illinois at Urbana-Champaign,* (moderator)
Kent Beck, *First Class Software*
Grady Booch, *Rational*
William Cook, *Apple Computer, Inc.*
Richard Gabriel, *Lucid, Inc.*
Rebecca Wirfs-Brock, *Digitalk*

1 Ralph Johnson: General Comments

The last few OOPSLA's have had acceptance rates of around 12by anyone's standards. The acceptance rate for OOPSLA'93 was 9due in part to a large number of submissions, and in part to a program committee with very high standards. But it is also due to a rapidly growing community that does not all understood the standards.

Although the set of standards is not widely understood, there *is* a set of standards. There are many areas of disagreement, but they are outweighed by the areas of agreement. The purpose of this panel is to try to explain how OOPSLA papers are judged so that it will increase the odds that your paper will be accepted.

Alan Snyder's appendix to the OOPSLA'91 proceedings on "How to get a paper accepted at OOPSLA" is very accurate. However, abstract rules like "explain the contribution of your paper" and "convince the program committee that your work is correct" can be interpreted differently for different kinds of papers. That is why most of the panel members will describe the way that standards are applied to a particular topic. Kent Beck will give his extremely concrete rules for writing papers.

One piece of advice that often seems to be ignored is to have your paper reviewed by colleagues. Two of the most important rules are that papers should be understandable and that they should be of interest to the OOPSLA community. In both these cases, you will need outside readers to tell whether the paper is acceptable. I have a hard time telling whether a paper I wrote is easy to understand - I never have any trouble understanding my papers. Most people are like me. Since most authors are new to the OOPSLA community, they have a hard time telling whether the paper is of interest. Many problems can be avoided by having someone else read your paper and criticize it. The best reviewers will be people who are active in the area and who often attend OOPSLA, but who are from a different institution and so do not know the work you are reporting on. Authors of papers you are citing are good prospects, especially if the papers were presented at OOPSLA. They can tell you whether you are ignoring some related work and whether they are convinced by your argument.

A low acceptance rate is hard on the program committee, just as it is hard on people submitting papers. Members of the committee read thirty to fifty papers, trying to write careful reviews while knowing that all but a handful of the papers will be rejected. Eventually they fall into the state of expecting to reject papers, and consequently sometimes overlook good ones, or ones that with a little improvement would be good. We would all be better off with fewer papers submitted and a higher acceptance rate. The best way to do this is for au-

thors to have a better idea of what it takes to get a paper accepted at OOPSLA and not to submit papers that obviously are not suitable.

2 William Cook: Theoretical Papers

When writing a theoretical paper for OOPSLA, you must remember that OOPSLA is not a theory conference. Special efforts must be made to motivate and analyze the significance of the theory being presented. Here are some tips on getting a theory paper accepted.

Give your paper a clear focus. It is far better to have a single idea that has been thoroughly discussed than a set of interesting but unfocused ideas. This is one of the easiest ways to get into trouble: you may want to say too much. Remember that only so much can be said in twenty pages.

State your thesis carefully and be sure to support it. It must be clear that the body of the paper supports the thesis. A paper that makes a claim but does not support it is far worse than a paper that makes lesser claims but supports them well. All too often a paper will claim to provide a better or more efficient solution to some problem, but then offer no analysis or evidence to support the claim. Reviewers spend much of their time scanning for your thesis statement and will read anything that looks like a claim as one. They will then be sure to check for supporting evidence.

Theory for its own sake is not enough. The theory must prove (or disprove) some proposition that is of interest to the community. It is not enough to simply give a x-theoretic account of concept-y. You must answer the question: what does an x-theoretic analysis provide that was lacking in previous analyses? Explain how things can be done differently as a result of the understanding the theory gives.

As with all papers, comparison to related work is essential for an acceptable theory paper. Although it can be difficult, a formal analysis of the difference between your new theory and previous attempts is very convincing.

Test the power of your theory against accepted practice of object-oriented programming, languages, and systems. Take a real example and analyze it with your theory. Or prove that the theory is related to an existing system. An untested theory is not worth much, but a good theory should be easy to apply to practical examples.

3 Rebecca Wirfs-Brock: Experience Papers

There have been few experience papers presented at OOPSLA. The OOPSLA program committee would like to see more. This is an exciting time for object technology. Accounts that present case studies or intelligently discuss the issues of developing object-oriented applications are a vital contribution to the advancement of our field. From our small sample of accepted papers, , we have gleaned ideas about what makes good experience papers.

An experience paper ideally tells an informative story on some aspect of the development, delivery, design or architecture of a significant object-oriented application. The best papers present a single focused topic or case study and then reflect on broader issues. On the other hand, research papers disquised as "experiences gathered in an academic setting" and papers discussing the implementation of a commercial product that are thinly disguised advertisements are inappropriate.

Many potential authors with interesting stories don't know how to write a technical paper for a technical conference. The key to a good experience paper is to present ideas clearly and succinctly. Experience papers needn't mimic the style of a research contribution. For example, extensive citations or discussions on areas for future research aren't appropriate. However, they should include relevant facts, sum up key points, and cite related ideas when appropriate. They also must present enough details so that others can understand and relate to the experience.

Experience papers should assume that their audience understands object technology, current

methods and practices. It is appropriate to assume that the audience does not know much about about the application area discussed in the paper, and that they don't wish to. The audience is more interested in the important aspects of application development that are affected by the use of object technology. Ideally, the paper should present lessons learned that can be easily compared with others' experiences. Presenting a balanced view of both positive and negative aspects of the experience can be illuminating.

Many submissions fall short of their potential simply by attempting to cover too much ground. For example, when presenting a case study, it isn't neccesary to include coding examples. When discussing a project history, it is appropriate to sumarize the highlights and perhaps amplify a few key points. Including too many low level technical details is generally not interesting. Papers discussing detailed language specific techniques are more appropriate for a language specific conference or journal.

4 Grady Booch: Methods

A good paper about object-oriented methods should achieve one of the following results: * enable the practitioner to apply the notation or process described in the paper to a real project without requiring unnatural leaps of faith or gaping simplifying assumptions. * enable the researcher to unambiguously compare and contrast the novel, subtle, or perhaps even profound ideas in the paper to previous work in the area * enable the reader to understand the purpose, activities, measures, and milestones of its process

Above all, a good methods paper should have clear goals, be focused, be respecting of previous work, and defend its statements without question.

A good paper about object-oriented methods should also: * be much more than an enumeration of a process without explaination * practice good science, meaning that, like a proof, its description of the method must be clear, follow from previously-discussed foundations, and unambiguous * provide a balance treatment: if benefits are suggested, they must be defended and tempered by a discussion of the method's limitations.

Method comparions papers are a special topic because 1) they have been done so many times already and 2) are often very poorly done, because they often provide no tangible, repeatable measures for comparison. A good method comparion paper must state its criteria clearly, so that others can repeat the comparision in the context of their own environment.

5 Richard Gabriel: Programming Language Papers

A major aspect of object technology is object-oriented programming languages, but getting a paper accepted that is about programming languages is very difficult. I break up programming language papers into a few categories. The more clearly you can place your paper in one of these categories, the more likely your paper is to be accepted – this is a special case of the general advice to focus on one topic and make your presentation of that single topic as clear as you can.

Whole Cloth Languages Primary Audience: experienced languages designers

Papers in this category have been rare since 1990.

A paper in this category presents a completely new programming language, either in its entirety or in summary. The language would need to be quite small to fit into the page limitations of the OOPSLA proceedings, and so my advice is as follows:

- focus on the novel aspects of the new language – people already understand that a programming language probably has to provide a means to do arithmetic
- explain the syntax as briefly as you can, perhaps with one or two program examples. Usually the flavor of the language's syntax can be gotten across more briefly than you expect. A

good rule of thumb is that no one cares about syntax.

- compare and contrast the language with existing ones, but only in the language's novel aspects
- explain clearly what there is about the new language that makes it worthy of use, for example:
  - does it solve a new problem?
  - does it solve existing problems better?

Your chances of getting the paper accepted improve if the new language has at least a prototype implementation.

- if the language has implementation difficulties or would seem to the casually informed reader to have performance problems, explain briefly implementation techniques you have used or would use to overcome or mitigate those difficulties.
- if the language has clear implementational or performance problems, you must explain in detail why these are not fatal flaws
- if the prototype language is sufficiently mature and already written up elsewhere, consider writing a language implementation paper.

Language Extensions Audience: experienced language designers

Papers that discuss extensions to an existing programming language are still popular. If the extensions you are proposing solve new problems or solve old problems in new ways, you should treat this paper like a Whole Cloth Language paper and talk about the extensions as if they were the new language. If the extensions solve implementation problems or focus on reducing complexity or improving performance, you should probably treat this paper like a Language Implementation paper.

Language Implementations Audience: language implementors

Though there are conferences that focus on language implementation, there is also a core of people in the OOPSLA audience who care deeply about implementation techniques. Here are some rules:

- you must answer these questions
  - does the technique address a current inadequacy?
  - does the technique enable something to be implemented that couldn't be before?
- explain what problem your technique solves
- explain the technique or algorithm well enough that a reasonably expert implementor could implement it
- explain the technique simply enough that a general audience with average CS credentials could understand the idea
- proof of concept: show that your technique is effective by doing one of these:
  - proving the technique or algorithm correct
  - or showing benchmark results

  if you are claiming a portable implementation of something, it must run on two *different* computers or operating systems (depending on the level of portability you are claiming).

In general you cannot write a successful paper based solely on coding tricks, but if they are part of the solution, show them if you can do it briefly enough. One possible exception to this rule is a technique that uses the computer's architecture or OS capabilities in a very novel way to achieve effects that were either prohibitively expensive or impossible before.

In considering the level of "proof of concept" mentioned above, make sure you are informed about the state of the art in presentations for the category of implementation you are reporting on. For example, garbage collection papers have

reached the point where merely describing the algorithm, even formally, or determining its complexity are not enough. You must either prove the algorithm is correct rigorously or describe a running implementation that is essentially in production-level use. For persistent objects, simply describing clearly and compellingly a new technique without proof or providing benchmark results is acceptable – there is no requirement of rigorous proof or a production-quality implementation.

Language Comparisons Audience: language designers, people making language choice decisions

A language comparison paper takes two or more languages and compares them narrowly or broadly. A narrow comparison must focus on the key areas of the languages. This can be either a research or experience paper, depending on the means of comparison – it could be at the semantic level (research) or at the use level (experience). A language comparison paper should be a paper that the designers of each of the languages in question would agree with. You should have people who are ardent users of each of the languages read you paper; a successful paper is one that each of the readers finds at least thought-provoking as well as informative regarding all the languages and accurate with respect to the favored language.

- focus only on key areas – syntax is not one of them
- show strengths and weaknesses of each language
  - expressiveness

    performance
  - adequacy
- if reasonable, show or explain how to achieve the same thing in each language where doing it is easy in one language and hard in another

You really have a lot of responsibility here: Imagine your paper convinced an important project – such as a project to provide intensive care assistance – to use the wrong language and the project failed. Would your name on a cute OOPSLA paper be worth it?

Language Critiques Audience: language designers, people making language choice decisions, general

I draw a distinction between critiques, pans, and accolades. All three of these types of papers focus on one language and present arguments about their adequacy, usefulness, design quality, or expressiveness.

A critique contains both positive and negative statements about the language. A critique is not a flame session.

- present design, implementation, or other principles that you believe should govern language design and implementation
- justify those principles
- analyze the language against those principles

The caveats about responsibility mentioned in the Language Comparison section apply here.

Language Pans or Accolades

Audience: language designers, people making language choice decisions, general

A language pan is a critique that is entirely negative; a language accolade is a critique that is entirely positive. It is extremely hard to get such papers accepted unless there is extraordinary scholarship involved. Such a paper must have a clearly objective basis and must be very compelling. You probably have to be one of the well-regarded language designers to pull this off.

The best way to approach this paper is as an experience paper in which an application failed or succeeded because of the language.

- show what about the language was crucial to the outcome of the project
- if you are writing a pan, explain what features or capabilities would have helped
- it has to be a good experience paper

In general, you cannot get a paper accepted unless people learn something from it; people don't want to learn what language you like or dislike. People don't like ads.

Your best bet is to submit this sort of paper to one of the trade magazines and not to OOPSLA.

The caveats about responsibility mentioned in the Language Comparison section apply here.

Language Semantics Audience: Theoretical programming language people

Such a paper presents the semantics of parts or all of one or more programming languages. Here are some rules:

- choose a formalism that is well-known and accepted
- present the semantics deeply enough that an expert would be able to complete it
- present the semantics clearly and simply enough that a general audience can follow the idea
- explain the reason for presenting the semantics - it is not enough to present formal semantics just for the fun of it, there must be an existing or perceived ambiguity, unclarity, or confusion about the characteristics explicated

Language Reflection Audience: General

A paper in this category explains something about languages that is either not clear, not apparent, or poorly understood, or it presents a new way of thinking about things that is compelling. There are few papers like this today, and many of them are formal or theoretical. An example would be a paper that showed that abstract data types and classes were orthogonal ways of expressing or understanding modularity. There is but one rule:

- present a new insight compellingly

A paper in this category, if well done, is highly valuable to the community.

/bf A note on writing

Referees don't have a lot of time to spend on reading and reviewing conference papers. Although every single one of them is devoted to doing the best possible and most thorough job, qualified referees have a hectic and demanding schedule. A referee has limited time to read your paper and frequent distractions to confusion him or her. Often, your paper will get exactly one reading by each referee. Wouldn't you like that one shot to have the best effect it can?

Over 2300 years ago Thucydides wrote:

A man who has the knowledge but lacks the power clearly to express it is no better off than if he never had any ideas at all.

I won't try to teach you how to write, but I will give you one piece of advice:

- a computer scientist is equally a scientist and a writer – expend the effort to learn the other half of your profession

6 Kent Beck

I will not talk about a topic area, like my distinguished fellow panelists. I will present the process I use as I am writing my papers. You can adapt it for your writing process, or you can use it as a check list for evaluating finished papers (if this is starting to sound like patterns, well, fancy that). Much of what I will say is "common sense", found in any book about writing. Having looked at hundreds of submissions, though, I can state with certainty that most of the authors don't follow this advice.

1. Write to the program committee. Never forget that before you can write to the vast, eager, and appreciative OOPSLA audience you must first get past the program committee. Before I begin I fix in my mind a picture of a harried PC member, desk piled with papers. Mine comes to the top. I have maybe thirty seconds to grab their interest.

   Remember that the program committee is made up of experts in the field. Even if your topic is of broad interest to beginners, there must still be some spark in it to keep and expert reading to the end. If your topic is highly technical, it may not be in an area that they

are familiar with, so it must readably present the novel aspects of the work.

2. One startling sentence. Now that you know you are writing to the program committee, you need to find the one thing you want to say that will catch their interest. If you have been working on the world's niftiest program night and day for five years, the temptation is to include absolutely everything about it, "The Foo System in All Its Glory." It'll never work. I know it's painful to ignore all those great insights, but find the most amazing thing you have done and write it down, "network garbage collection is fast and easy." You want the reader's eyes to open wide when they realize what it is you've just said.

   I think some people are reluctant to boil their message down to one startling sentence because it opens them up to concrete criticism. If you write about the Foo System and someone says it isn't neat, you can just reply, "Is so, nyah!" If you say network garbage collection is easy, it is a statement that is objectively true or false. You can be proven wrong. Wait! You spent five years proving it was easy. Make your case.

3. Argument- problem, solution, defense, related work. Now that you have a startling sentence, your paper must stand as the argument for its validity. You are convincing the by now intrigued committee member of the truth of your amazing statement.

   Divide your paper into four sections. The first describes the problem to be solved. When the PC member is done reading it, they should understand why it's a problem, and believe that it is important to solve. The second section describes your solution. You are convincing the PC member that your solution really could solve the problem. This section is sometimes supplemented with a section between the defense and related work which describes implementation details. The third section is you defense of why your solution really solves the problem. The PC member reading it should be convinced that the problem is actually solved, and that you have thought of all reasonable counter arguments. The final section describes what other people have done in the area. Upon reading this section, the PC member should be convinced that what you have done is novel.

4. Abstract. The abstract is your four sentence summary of the conclusion of your paper. Its primary purpose is to get your paper into the A pile. Most PC members sort their papers in an A pile and a B pile by reading the abstracts. The A pile papers get smiling interest, the B pile papers are a chore to be slogged through. By keeping your abstract short and clear, you greatly enhance your chances of being in the A pile.

   I try to have four sentences in my abstract. The first states the problem. The second states why the problem is a problem. The third is my startling sentence. The fourth states the implication of my startling sentence. An abstract for this paper done in this style would be:

   The rejection rate for OOPSLA papers is near 90%. Most papers are rejected not because of a lack of good ideas, but because they are poorly structured. Following four simple steps in writing a paper will dramatically increase your chances of acceptance. If everyone followed these steps, the amount of communication in the object community would increase, improving the rate of progress.

   Well, I'm not sure that's a great abstract, but you get the idea.

   I always feel funny writing an abstract this way. The idea I thought was so wonderful when I started writing the paper looks naked and alone sitting there with no support. I resist the temptation to argue for my conclusion in the abstract. I think it gives the reader more incentive to carefully read the rest of

the paper. They want to find out how in the world you could possible say such an outrageous thing.

There are my four steps to better papers. You can use them sequentially to write papers, or you can use them to evaluate papers you have already written.

7 Biographies

Ralph Johnson is the OOPSLA '93 program chair.

Kent Beck was the OOPSLA '89 program chair and on the OOPSLA '88 and OOPSLA '90 program committees.

Grady Booch was on the OOPSLA '86, OOPSLA '89, OOPSLA '91, OOPSLA '92, and OOPSLA '93 program committees.

Richard Gabriel was on the OOPSLA '89 and the OOPSLA '93 program committees and has reviewed over 400 submissions to OOPSLA.

William Cook was on the OOPSLA '90, OOPSLA '91, OOPSLA '92, and OOPSLA '93 program committees.

Rebecca Wirfs-Brock was the OOPSLA '92 program chair and on the OOPSLA '91 and OOPSLA '93 program committees.