DOI:10.1145/1866739.1866753

Article development led by acmqueue
queue.acm.org

Talking with Julian Gosper, Jean-Luc Agathos, Richard Rutter, and Terry Coatta.

ACM CASE STUDY

UX Design and Agile: A Natural Fit?

FOUND AT THE intersection of many fields—including usability, human-computer interaction (HCI), and interaction design—user experience (UX) design addresses a software user's entire experience: from logging on to navigating, accessing, modifying, and saving data. Unfortunately, UX design is often overlooked or treated as a "bolt-on," available only to those projects blessed with the extra time and budget to accommodate it. Careful design of the user experience, however, can be crucial to the success of a product. And it's not just window dressing: choices made about the user experience can have a significant impact on a software product's underlying architecture, data structures, and processing algorithms.

To improve our understanding of UX design and how it fits into the software development process, we focus here on a project where UX designers worked closely with software engineers to build *BusinessObjects Polestar* (currently marketed as SAP BusinessObjects Explorer), a business intelligence (BI) query tool designed for casual business users. In the past, such users did not have their own BI query tools. Instead, they would pass their business queries on to analysts and IT people, who would then use sophisticated BI tools to extract the relevant information from a data warehouse. The Polestar team wanted to leverage a lot of the same back-end processing as the company's more sophisticated BI query tools, but the new software required a simpler, more user-friendly interface with less arcane terminology. Therefore, good UX design was essential.

To learn about the development process, we spoke with two key members of the Polestar team: software architect **Jean-Luc Agathos** and senior UX designer **Julian Gosper**. Agathos joined BusinessObjects' Paris office in 1999 and stayed with the company through its acquisition by SAP in 2007. Gosper started working with the company five years ago in its Vancouver, B.C., office. The two began collaborating early in the project, right after the creation of a Java prototype incorporating some of Gosper's initial UX designs. Because the key back-end architecture is one that had been developed earlier by the Paris software engineering team, Gosper joined the team in Paris to collaborate on efforts to implement Polestar on top of that architecture.

To lead our discussion, we enlisted a pair of developers whose skill sets largely mirror those of Agathos and Gosper. **Terry Coatta** is a veteran software engineer who is the CTO of Vitrium Systems in Vancouver, B.C. He also is a member of *ACM Queue*'s editorial advisory board. Joining Coatta to offer another perspective on UX design is **Richard Rutter**, a longtime Web designer and a founder of Clearleft, a UX design firm based in Brighton, England.

Before diving in to see how the collaboration between Agathos and Gosper played out, it's useful to be familiar with a few of the fundamental disciplines of classic UX design:

Contextual inquiry. Before developing use cases, the team observes users of current tools, noting where the pain points lie. Contextual inquiry is often helpful in identifying problems that users are not aware of themselves. Unfortunately, this often proves expensive and so is not always performed.

Formative testing. Formative testing is used to see how well a UX design addresses a product's anticipated use cases. It also helps to determine how closely those use cases actually cleave to real-world experience. In preparation, UX designers generally create lightweight prototypes to represent the use cases the product is expected to eventually service. Often these are paper prototypes, which the test-group participants simply flip through and then comment upon. For the Polestar project, the UX team used a working Java prototype to facilitate early formative testing.

Summative testing. In summative tests, users test-drive the finished software according to some script. The feedback from these tests is often used to inform the next round of development since it usually comes too late in the process to allow for significant changes to be incorporated into the current release.

Although the Polestar team did not have the budget to conduct contextual inquiry, it was able to work closely with the software engineer who built the research prototype responsible for spawning the project. This allowed the team to perform early formative testing with the aid of a working UX design, which in turn made it possible to refine the user stories that would be used as the basis for further testing. Working with the software engineer responsible for the initial design also made it possible to evaluate some of the initial UX designs both from a performance and a feasibility perspective, preventing a lot of unwelcome surprises once development was under way in earnest.

TERRY COATTA: You mentioned you worked early on with a software engineer to iterate on some lightweight paper prototypes. What was the role of the engineer in that process?

JULIAN GOSPER: Adam Binnie, a senior product manager who had conceived of the project, brought me in to elaborate the interaction design of the early Polestar prototype. Davor Cubranic, who had produced that initial proof-of-concept for research purposes, was looking to work some of the user experience ideas I had begun to collaborate on with Adam back into his original design. Davor saw value in creating Java prototypes of some of those new concepts so we would have not only paper prototypes to work with, but also a live prototype that end users could interact with and that development could evaluate from a technical perspective. I really pushed for that since we already had this great development resource available to us. And it didn't seem as though it was going to take all that long for Davor to hammer out some of the key UX concepts, which of course was going to make the live prototype a much better vehicle for formative testing.

Generally speaking, as an interaction designer you don't want to invest a lot of time programming something live, since what you really want is to keep iterating on the fundamentals of the design quickly. That's why working with paper prototypes is so commonplace and effective early in a project. Typically, we'll use Illustrator or Visio to mock up the key use cases and their associated UI, interactions, and task flows, and then output a PowerPoint deck you can just flip through to get a sense for a typical user session. Then various project stakeholders can mark that up to give you feedback. You also then have a tool for formative testing.

Collaborating closely with development at that stage was appealing in this particular case, however, because some of the directions we were taking with the user interface were likely to have serious back-end implications—

Top to Bottom: Jean-Luc Agathos, Terry Coatta, Julian Gosper, and Richard Rutter.

for example, the ability of the application to return and reevaluate facets and visualizations with each click. Having Davor there to help evaluate those proposed design changes right away from a performance perspective through rapid iterations of a lightweight Java-based prototype helped to create a nice set of synergies right from the get-go.

JEAN-LUC AGATHOS: Even though I didn't get involved in the project until later, it seemed as though Davor had produced a solid proof-of-concept. He also figured out some very important processing steps along the way, in addition to assessing the feasibility of some key algorithms we developed a bit later in the process. I think this was critical for the project, since even though people tend to think about user experience as being just about the way things are displayed, it's also important to figure out how that stuff needs to be manipulated so it can be processed efficiently.

GOSPER: That's absolutely correct. For this product to succeed, performance was critical—both in terms of processing a large index of data and being able to evaluate which facets to return to a user to support the experience of clicking through a new data analysis at the speed of thought. The capabilities for assessing the relevance of metadata relative to any particular new query was actually Davor's real focus all along, so he ended up driving that investigation in parallel to the work I was doing to refine the usability of the interface.

I do recall having discussions with Davor where he said, "Well, you know, if you approach 'X' in the way you're suggesting, there is going to be a significant performance hit; but if we approach it this other way, we might be able to get a much better response time." So we went back and forth like that a lot, which I think ultimately made for a much better user experience design than would have been possible had we taken the typical waterfall approach.

COATTA: Are product engineers typically excited about being involved in a project when the user experience stuff is still in its earliest stages of getting sorted out?

AGATHOS: Yes, I think developers need to be acquainted with the user stories as early in the process as possible. To me, user experience and development are essentially one and the same. I see it as our job as a group to turn the user stories into deliverables.

Of course, in development we are generally working from architectural diagrams or some other kind of product description that comes out of product management. We're also looking to the user experience people to figure out what the interaction model is supposed to look like.

The interesting thing about the first version of Polestar is that Julian essentially ended up taking on both roles. He acted as a program manager because he knew what the user stories were and how he wanted each of them to be handled in terms of product functionality. He also had a clear idea of how he wanted all of that to be exposed in the UI and how he wanted end users ultimately to be able to interact with the system. That greatly simplified things from my perspective because I had only one source I had to turn to for direction.

COATTA: You used Agile for this project. At what point in the process can the software developers and UX designers begin to work in parallel?

GOSPER: If you have a good set of user stories that have been agreed upon by the executive members of the project and include clear definitions of the associated workflows and use cases, then the Agile iterative process can begin. At that point you are able to concretely understand the functionality and experience the product needs to offer. On the basis of that, both UX interaction designers and the development team should have enough to get going in parallel. That is, the developers can start working on what the product needs to do while the UX guys can work on use-case diagramming, wireframing and scenarios, as well as begin to coordinate the time of end users to supply whatever validation is required.

The important thing is that you have lots of different people involved to help pull those user stories together. Clearly, the UX team needs to be part of that, but the development team should participate as well—along with the business analysts and anybody else who might have some insights regarding what the product requirements ought to be. That's what I think of when we

> “
> JULIAN GOSPER
> **For this product to succeed, performance was critical—both in terms of processing a large index of data and being able to evaluate which facets to return to a user to support the experience of clicking through a new data analysis at the speed of thought.**
> ”

talk about starting development on the basis of some common "model."

For the most part, development of Polestar's first release went smoothly. Gosper's collaboration with the prototype developer helped iron out some of the technical challenges early on, as well as refine the main user stories, which helped pave the way for implementing the product with Agathos. The user interface that emerged from that process did face certain challenges, which were verified during summative testing, after development work for release 1 was essentially complete. Contributing to the problem was the general dearth of query tools for casual business users at the time. The blessing there was that Gosper and his UX team had the freedom to innovate with their design. That innovation, however, meant the software would inevitably end up incorporating new design concepts that—even with many iterations of formative testing—faced an uncertain reception from users.

Further challenges arose during the development of subsequent releases of Polestar, once Agathos and Gosper were no longer working together. In response to user feedback, the new UX team decided to make fundamental changes to the UI, which forced the engineering team to rearchitect some parts of the software. On top of these challenges, the new team learned what could happen when there is confusion over the underlying conceptual model, which is something Agathos and Gosper had managed to avoid during the first phase of development.

Of course, that's not to say the initial phase was entirely free of development challenges, some of which could be ascribed to pressure to prove the worth of the new endeavor to management.

RICHARD RUTTER: What were the most challenging parts of this project?

AGATHOS: One big challenge was that right after we received the original Java POC (proof of concept), we received word that we needed to produce a Flash version in less than two weeks so it could be shown at Business Objects's [now SAP BusinessObjects] international user group meeting. Actually, that was only one of many constraints we faced. The POC itself imposed certain constraints in terms of how to approach processing the data and exposing appropriate facets to the user. There already were some UI constraints driven by the user stories. *Then* we learned we had only a couple of weeks to get something ready to show to customers. And finally, it was strongly suggested that we use Adobe Flex—something we had not even been aware of previously—to get the work done.

So, our first job was to learn that technology. Initially, we were pretty frustrated about that since it wasn't really as if we had a choice in the matter. Instead of fighting it, however, we quickly realized it was a really good idea. Flex is so powerful that it actually allowed us to come up with a working prototype in time for that user conference.

GOSPER: There was a special aspect to this project in that it was earmarked as innovation—a step toward next-generation self-service BI. Prior to this, the company had never productized a lightweight business intelligence tool for business users, so we didn't really have much in the way of similar products—either within the company or out in the market—that we could refer to. As a result, the design we ended up pushing forward was sort of risky from a user adoption perspective.

The first version of Polestar, in all honesty, tended to throw users for a loop at first. Most of the users we tested needed to spend a few minutes exploring the tool just to get to the point where they really understood the design metaphors and the overall user experience.

AGATHOS: That was definitely the case.

GOSPER: That sparked a fair amount of controversy across the UX group because some of the methodologies around formative testing back then differed from one site to another. The next designers assigned to the project ended up coming to different conclusions about how to refine the interaction design and make it more intuitive. Some questions were raised about whether we were fundamentally taking the right interaction development approach in several different areas of the interface. We employed faceted navigation in a fairly unique way, and the interactions we created around analytics were also pretty novel. [Since users often have only a fuzzy idea about some of the parameters or aspects of the information they're seeking, *faceted navigation* is a UI pattern that facilitates queries by allowing users to interactively refine relevant categories (or "facets") of the search.]

A lot of those initial design directions ended up being re-explored as open topics as the UX team in Paris worked on the second version of the product. They ended up coming to some different conclusions and began to promote some alternatives. That, of course, can be very healthy in the evolution of a product, but at the same time, it can really challenge the development team whenever the new UI choices don't align entirely with the code you've already got running under the hood.

COATTA: Another area where I think there is huge potential for trouble has to do with the feedback process to UX design. I've been through that process and have found it to be extremely challenging. As an example, suppose you need to find two business objects that are related to each other. Let's say we know that one of those objects can be shared across multiple business domains. One possibility is that they share a unique ownership, which would have huge ramifications for the user experience. When we have run across situations like that, we've often had trouble communicating the semantic implications back to the folks who are responsible for doing the UI. I wonder if you've run into similar situations.

AGATHOS: Actually, Julian had already worked all of that out before joining us in Paris, so we didn't have any problems like that during our initial round of development. With later versions, however, we had an issue in the administration module with "infospace," which is a concept we exposed only to administrators. The idea was that you could create an infospace based on some data source, which could be a BWA (Burrows-Wheeler Alignment) index or maybe an Excel spreadsheet. [*BWA* is a fast, lightweight tool that aligns relatively short queries to a sequence database. These sequences are usually indexed in the FASTA format.]

Before anybody can use the system's exploration module to investigate one of these information spaces, that space must be indexed, resulting in a new index entity. That seems straightforward enough, but we spent a lot of time trying to agree on what ought to happen when one person happens to be exploring an infospace while someone else is trying to index the same space. Those discussions proved to be difficult simply because we had not made it explicit that the entity in question was an index. That is, we talked about it at first strictly as an information space. It wasn't until after a few arguments that it became clear what we were actually talking about was an index of that information space.

Whenever the model can be precisely discussed, you can avoid a lot of unnecessary complexity. When a model is correctly and clearly defined right from the outset of a project, the only kind of feedback that ought to be required—and the only sort of change you should need to add to your sprints in subsequent iterations—has to do with the ways you want to expose things. For example, you might decide to change from a dropdown box to a list so users can be given faster access to something—with one click, rather than two. If you start out with a clear model, you're probably not going to need to make any changes later that would likely have a significant impact on either the UI or the underlying architecture.

In developing a product for which there was no obvious equivalent in the marketplace, Agathos and Gosper were already sailing into somewhat uncharted waters. And there was yet another area where they would face the unknown: neither had ever used Agile to implement a UX design. Adding to this uncertainty was the Agile development methodology itself, where the tenets include the need to accept changes in requirements, even late in the development process.

Fortunately, the Polestar team soon embraced the iterative development process and all its inherent uncertainties. In fact, both Gosper and Agathos found Agile to be far more effective than the waterfall methodology for implementing UX designs. This doesn't mean all was smooth sailing, however. Agile requires close collaboration between team members and frequent face-to-face communication, often between stakeholders with vastly different backgrounds and technical vocabularies. It's therefore no surprise that one of the team's biggest challenges had to do with making sure communication was as clear and effective as possible.

TERRY COATTA

An area where I think there is huge potential for trouble has to do with the feedback process to UX design. I've been through that process and have found it to be extremely challenging.

COATTA: I understand that some UX designers feel Agile is less something to be worked *with* than something to be worked *around*. Yet, this was the first time you faced implementing a UX design using Agile, and it appears you absolutely loved it. Why is that?

GOSPER: In an ideal world, you would do all your contextual inquiry, paper prototyping, and formative testing before starting to actually write lines of code. In reality, because the turnaround time between product inception and product release continues to grow shorter and shorter, that simply isn't possible. If the UX design is to be implemented within a waterfall project, then it's hard to know how to take what you're learning about your use cases and put that knowledge to work once the coding has begun.

In contrast, if you are embedded with the development team and you're acquainted, tactically, with what they're planning to accomplish two or three sprints down the road, you can start to plan how you're going to test different aspects of the user experience in accordance with that.

COATTA: In other words, you just feel your way along?

GOSPER: Yes, and it can be a little scary to dive right into development without knowing all the answers. And by that I don't just mean that you haven't had a chance to work through certain areas to make sure things make sense from an engineering perspective; you also can't be sure about how to go about articulating the design because there hasn't been time to iterate on that enough to get a good read on what's likely to work best for the user. Then you have also got to take into account all the layers of development considerations. All of that comes together at the same time, so you've got to be very alert to everything that's happening around you.

There is also a lot of back-and-forth

in the sprint planning that has to happen. For example, Jean-Luc would let me know we really needed to have a certain aspect of the UI sorted out by some particular sprint, which meant I had essentially received my marching orders. Conversely, there also were times when I needed to see some particular functionality in place in time to use it as part of a live build I wanted to incorporate into an upcoming round of formative testing. That, alone, would require a ton of coordination.

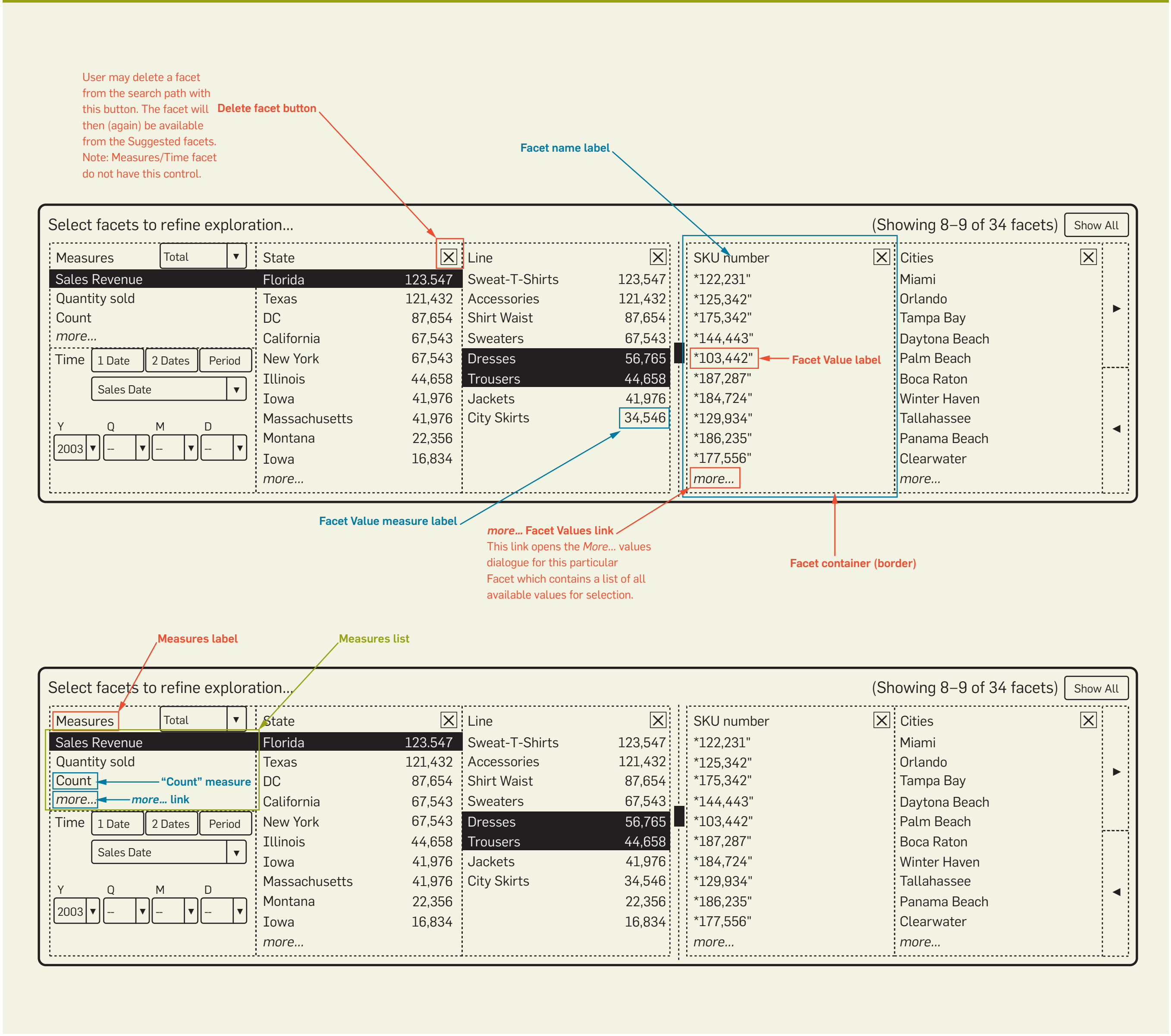
The other important influence on sprint planning comes from product management, since they too often want to be able to show off some certain capabilities by some particular date. With all three of these vested interests constantly vying to have certain parts of the product materialize by certain points in time, there's plenty of negotiating to be done along the way.

RUTTER: Yes, but I think you have to accept that some reengineering of bits and pieces along the way is just an inherent part of the process. That is, based on feedback from testing, you can bank on sprints down the line involving the reengineering of at least a few of the things that have already been built. But as long as that's what you expect...no problem.

AGATHOS: I think you're right about that: we have to develop a mind-set that's accepting of changes along the way. But I actually think the biggest problem is the tooling. The development tools we have today don't help us with all those iterations because they don't provide enough separation between the business logic and the UI. Ideally, we should be able to change the UI from top to bottom and revisit absolutely every last bit of it without ever touching the underlying logic. Unfortunately, today that just isn't the case.

GOSPER: That's why, as a UX interaction designer, you have to be prepared to demonstrate to the product developers that any changes you're recommending are based on substantive evidence, not just some intuitive or anecdotal sense of the users' needs. You need to make a strong case and be able to support design changes with as

Polestar Faceted Navigation.



User may delete a facet from the search path with this button. The facet will then (again) be available from the Suggested facets. Note: Measures/Time facet do not have this control.

This link opens the *More...* values dialogue for this particular Facet which contains a list of all available values for selection.

Measures	State		Line		SKU number	Cities
Sales Revenue	Florida	123.547	Sweat-T-Shirts	123,547	"122,231"	Miami
Quantity sold	Texas	121,432	Accessories	121,432	"125,342"	Orlando
Count	DC	87,654	Shirt Waist	87,654	"175,342"	Tampa Bay
more...	California	67,543	Sweaters	67,543	"144,443"	Daytona Beach
	New York	67,543	Dresses	56,765	"103,442"	Palm Beach
	Illinois	44,658	Trousers	44,658	"187,287"	Boca Raton
	Iowa	41,976	Jackets	41,976	"184,724"	Winter Haven
	Massachusetts	41,976	City Skirts	34,546	"129,934"	Tallahassee
	Montana	22,356			"186,235"	Panama Beach
	Iowa	16,834			"177,556"	Clearwater
	more...				more...	more...

much quantitative and qualitative end-user validation data as you can get your hands on.

COATTA: So far, we've looked at this largely from the UX side of the equation. What are some of the benefits and challenges of using Agile to implement UX design from more of an engineering perspective?

AGATHOS: The biggest challenge we faced in this particular project—at least after we had completed the first version of Polestar—had to do with changes that were made in the overall architecture just as we were about to finish a release. Even in an Agile environment you still need to think things through pretty thoroughly up front, and then sketch out your design, prototype it, test it, and continue fixing it until it becomes stable. As soon as you've achieved something that's pretty solid for any given layer in the architecture, you need to finish that up and move on to the next higher layer. In that way, it is like creating a building. If you have to go back down to the foundation to start making some fairly significant changes once you've already gotten to a very late stage of the process, you're likely to end up causing damage to everything else you've built on top of that.

The nice thing about Agile is that it allows for design input at every sprint along the way—together with some discussion about the reasoning behind that design input. That's really important. For example, when we were working with Julian, he explained to us that one of the fundamental design goals for Polestar was to minimize the number of clicks the end user would have to perform to accomplish any particular task. He also talked about how that applied to each of the most important user stories. For us as developers, it really helped to understand all that.

I don't think we have exchanges like that nearly enough—and that doesn't apply only to the UX guys. It would also be good to have discussions like that with the program managers.

GOSPER: In those discussions for the Polestar project, one of the greatest challenges for me had to do with figuring out just how much depth to go into when describing a particular design specification. Sometimes a set of wireframes supporting a particular use case seemed to be good enough, since most of what I wanted to communicate could be inferred from them. But there were other times when it would have been helpful for me to break things down into more granular specifications. It was a bit challenging in the moment to sort that out, because on the one hand you're trying to manage your time in terms of producing specifications for the next sprint, but on the other hand you want to get them to the appropriate depth.

The other challenge is that each domain has its own technical language, and it can sometimes prove tricky to make sure you're correctly interpreting what you're hearing. For example, I remember one of the sprints where I was very concerned with a particular set of functionality having to do with users' abilities to specify financial periods for the data they might be looking to explore. I therefore became very active in trying to get product management to allocate more resources to that effort because I was certain it would be a major pain point for end users if the functionality was insufficient.

During that time I remember seeing a sprint review presentation that referred to a number of features, a couple of which related to date and financial period and so forth. Next to each of those items was the notation "d-cut." I didn't say a word, but I was just flabbergasted. I was thinking to myself, "Wow! So they just decided to cut that. I can't believe it. And nobody even bothered to tell me." But of course it turns out "d-cut" stands for "development cut," which means they had already implemented those items. There are times when you can end up talking past each other just because everyone is using terms specific to his or her own technical domain. Of course, the same is true for product and program management as well.

COATTA: Don't the different tools used by each respective domain also make contributions to these communication problems?

AGATHOS: I couldn't agree more. For example, when you program in Java, there are some things you can express and some you cannot. Similarly, for architects, using a language such as UML constrains them in some ways. They end up having to ask themselves tons of questions prior to telling the system what it is they actually want.

COATTA: Since we're talking about how engineers and designers live in different worlds and thus employ different tools, different skill sets, and different worldviews—what do you think of having software engineers get directly involved in formative testing? Does that idea intrigue you? Or does it seem dangerous?

GOSPER: Both, actually. It all depends on how you act upon that information. At one point there is an internal validation process whereby a product that is just about to be released is opened up to a much wider cross section of people in the company than the original group of stakeholders. And then all those folks are given a script that allows them to walk through the product so they can experience it firsthand.

What that can trigger, naturally, is a wave of feedback on a project that is just about finalized, when we don't have a lot of time to do anything about it. In a lot of that feedback, people don't just point out problems; they also offer solutions, such as, "That checkmark can't be green. Make it gray." To take all those sorts of comments at face value, of course, would be dangerous. Anyway, my tendency is to think of feedback that comes through development or any other internal channel as something that should provide a good basis for the next user study.

RUTTER: UX design should always involve contact with lots of different end users at plenty of different points throughout the process. Still, as Julian says, it's what you end up doing with all that information that really matters. When it comes to figuring out how to solve the problems that come to light that way, that's actually what the UX guys get paid to do. C

Related articles on queue.acm.org

The Future of Human-Computer Interaction
John Canny
http://queue.acm.org/detail.cfm?id=1147530

Human-KV Interaction
Kode Vicious
http://queue.acm.org/detail.cfm?id=1122682

Other People's Data
Stephen Petschulat
http://queue.acm.org/detail.cfm?id=1655240