Understanding Project Sociology by Modeling Stakeholders

Ian Alexander and Suzanne Robertson

Ian Alexander and I have been collaborating on ways to improve how we involve stakeholders in projects. Here, we summarize the approach we've been taking to discover the right stakeholders and maintain their involvement. —Suzanne Robertson

A ll the projects that we work on seem to have one thing in common: developers are concerned about the difficulty of involving the right stakeholders to discover, specify, and test requirements. We ran the Stakeholder Concerns Survey (see Table 1) to better understand what people mean when they say, "We have a problem with our stakeholders." We asked each participant to choose one stakeholder concern that causes the biggest problem in his or her environment. The objective was to discover which concerns are the most frequent and where we should focus our improvement efforts.

Table 1 summarizes some specific problems:

■ *Commitment*. People with authority (business and technical managers) don't commit enough time or budget for requirements.
■ *Skill*. The stakeholders (this includes all types of stakeholders, both business and technical) don't have the skills necessary to participate in gathering requirements. They describe solutions rather than requirements, and they change their minds.
■ *Discovery*. We can't discover the appropriate stakeholders. Finding all the people who must be involved in discovering, inventing, and specifying requirements is difficult.
■ *Maintaining*. We can't keep the stakeholders interested and involved throughout the entire project.
■ *Other*. Of the 5 percent who answered "other," the most common reason was lack of communication between stakeholders and developers.

On reflection, we realized that a formal understanding of what we mean by a stakeholder was the first step toward addressing the stakeholders' concerns.

Who are the stakeholders?

A project stakeholder is someone who gains or loses something (could be functionality, revenue, status, compliance with rules, and so on) as a result of that project. In other words, a stakeholder is much more than a product's

Table I

Stakeholder concerns survey

	Commitment	Skill	Discovery	Maintaining	Other	Total responses
Germany, Nov. 2001	18	40	14	3	7	82
Australia, Nov. 2001	11	5	0	2		18
Australia, Dec. 2001	10	14	5	4		33
Denmark, Jan. 2002	37	39	4	1	7	88
UK, Feb. 2002	14	6	4	1		25
Denmark, Jan. 2002	4	7	5	1		17
Netherlands, Mar. 2002	10	12	4	4	1	31
Australia, Apr. 2002	13	3	16	8	2	42
Australia, Apr. 2002		6	5	5		16
US, Apr. 2002	26	27	4	5	6	68
Italy, May 2002	3	4	3	4		14
Finland, May 2002	40	45	8	10	4	107
Total	186	208	72	48	27	541
Percentage of total	34%	38%	13%	9%	5%	100%

Figure I. An onion model of stakeholder relationships. Each circle represents a different stakeholder zone.

eventual "user." A stakeholder can be someone who finances the project; someone whose skill is needed to build a product, such as a network specialist or a usability expert; an organization whose rules developers must obey, such as a tax lawyer firm or a standards institute; or an external organization that can influence project success, such as an environmental group or a competi-

tor. Given this wide and complex project sociology, it helps to have some way of modeling the stakeholders and their differing involvements.

We can do this using an onion diagram (see Figure 1). This deceptively simple-looking model actually conceals a wealth of project information, both structurally and metaphorically. First, it presents a view of the project that centers on what we're building. We call this the *kit*, the *equipment*, or the *product* to distinguish it from *our system*, which includes generic *slots* for the people who operate and maintain the equipment and deliver its results, as well as any standard rules or procedures they use to operate it. (Even an automatic product such as a missile only works when people test, install, and launch it.)

These directly interacting roles don't begin to exhaust the range of stakeholder responsibilities in a development project. Systems invariably operate in a *containing system* that is outside the project's control and includes our system plus any of its nonoperational beneficiaries. Normally, the people who benefit from the product work in the containing system— otherwise, the product is running only for its own benefit! We call these people *functional beneficiaries* to distinguish them from operators. In the special case where you buy a device (say, a handheld music player) for your own benefit or enjoyment, and you operate it yourself, you are a (*device*) *user*, which we consider a hybrid role that crosses the "our system" boundary. The term properly applies (in our view) only to such hybrid roles in mass-market product developments. We think unsurprising that developers who habitually think about "users" and "end users" (the end of what?) often encounter confusion.

Normal operators operate equipment so as to deliver the results (for which the equipment was designed) to functional beneficiaries, who in turn work in the containing system for other stakeholders in the *wider environment*, which includes the containing
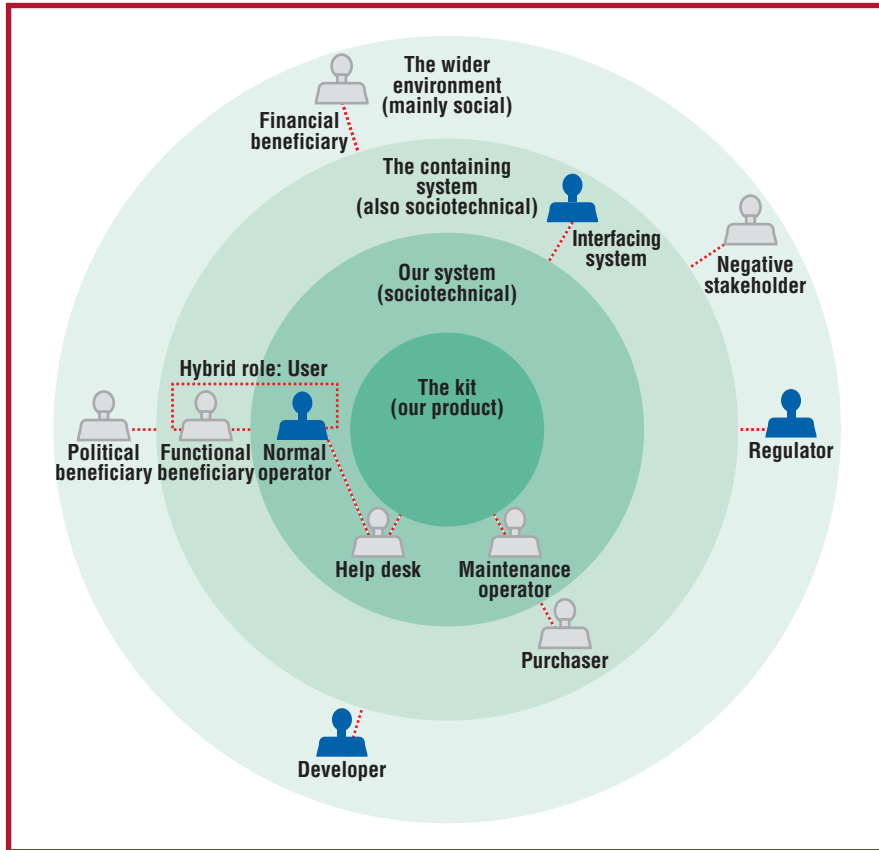
system plus any other stakeholders who affect decisions we make about our system. We can view the containing system as a tool. For instance, if the product is a business information system, then the containing system might be an investment business. Shareholders might own this business and benefit financially from it, and a board and president might run the business and derive power from it (of course they are probably also financial beneficiaries). So, a chain of responsibility exists, and we use the simple metaphor of a straight dotted line of linked icons (left side of the diagram) to indicate this.

Stakeholder metaphors

Incidentally, the onion model uses several more metaphors (not to mention the name onion itself and the title of a tutorial we recently ran, "Stakeholders without Tears," shown in Figure 2).

Supporting roles are toward the bottom of the diagram, where they can carry the project's weight on their shoulders. The head-and-shoulders icon is a simple silhouette, empty of individual facial features. (We also chose it to differentiate it from the UML Actor icon, but that's another story.) A stakeholder who is involved only indirectly—the *regulator*, for example—is both distant from the center and off to the side. Conversely, closeness to the center indicates direct operational involvement. A solid icon shows the presence of a defined stakeholder role (in our model, there is at least one in each slot), while graying out (indicating insubstantiality) demonstrates absence (empty slots). A slot is a generalized role—for instance, regulator must be specialized to aviation authority, radio regulator, safety regulator, food safety authority, and so on, as appropriate to the domain.

Finally, the term "stakeholder" itself apparently alludes to mine owners and is represented by a map boundary and (literally) wooden stakes hammered into the ground to delimit the claimed area. It is an engagingly powerful metaphor for the need to be clear about how to share project responsibilities.
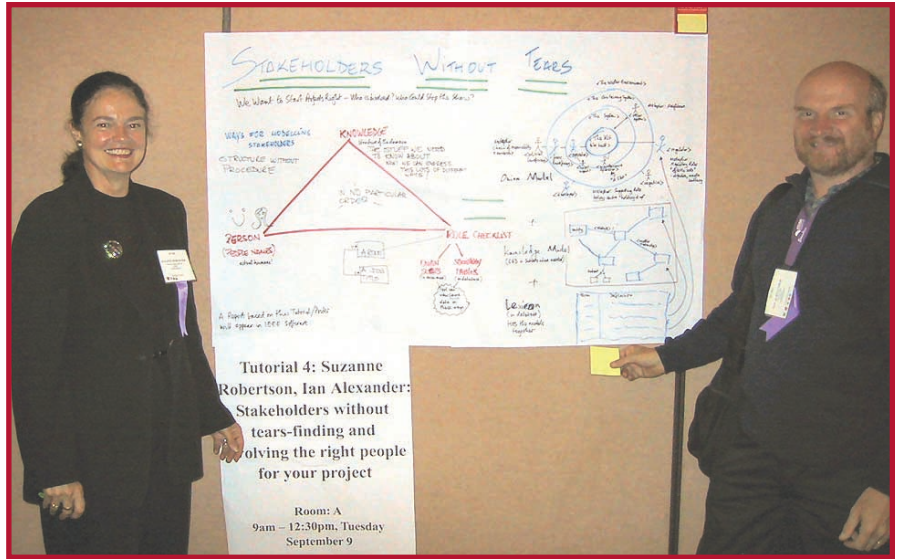

Figure 2. The "Stakeholders without Tears" tutorial.

The onion diagram shows explicitly only our model's top two (circles and slots) levels. Ian's onion model tool shows further details via pop-up windows.

How to …

To use the onion model approach, you can start either with an empty set of circles or with typical slots already present. Either way, you ask stakeholders (in workshops or interviews) what their roles are, and you populate the model accordingly—down to the contact details of the people filling each role. Then you can investigate any slots that remain unfilled and ask if anyone

knows who might fill roles in those areas. Finally, you can check for possible conflicts (which we describe further in the next section).

Figure 3 shows part of a stakeholder analysis template (see www.volere.co.uk). You can use this model, as you would an onion model, to help discover missing stakeholders. The template also approaches stakeholder discovery by asking an alternate question: "What sorts of knowledge are we looking for?" leading to, "Who is the source of that knowledge?" The two approaches are complementary because people who have been using one model might overlook a role that the other model immediately suggests. The template provides a formal tool for recording stakeholders' names and responsibilities and is especially useful when you have several different stakeholders concerned with the same knowledge or multiple stakeholders occupying the same role.

Learning from stakeholder models

One attractive result of modeling project stakeholders this way is that some simple and effective metrics immediately come to mind (see Table 2).

The number of circles, slots, and roles in a model mostly indicates its size, although if fewer roles exist than slots, clearly some slots must be empty. This

> The term "stakeholder" is an engagingly powerful metaphor for the need to be clear about how to share project responsibilities.

Stakeholder role (the job title, department, or organization that indicates a stakeholding)	Stakeholder name (the name(s) of the responsible stakeholder(s))	Necessary involvement (estimate of when and how much time)	Classes of Knowledge						
			Business Goals	Technical constraints	Functionality	Look and feel	Usability	Performance	
Client									
Customer(s)									
Business/ subject experts									
Future idea specialists									
Current system specialists									
Clerical user									
Technical user									
Potential user									
Sales specialist									
Marketing specialist									
Aesthetics specialist									
Graphics specialist									
Usability specialist									
Safety specialist									
Security specialist									
Cultural specialists									
Legal specialists									
Environmental specialists									
Maintenance specialists									
Packaging designer									
Manufacturer									
Product installer									

Figure 3. A fragment of a stakeholder analysis template. The complete template contains many more stakeholder roles and knowledge classes.

leads at once to the idea of checking for empty slots—general stakeholder categories that haven't been specialized into project-specific roles. Similarly, roles that aren't filled by named persons aren't going to serve the project.

Table 2

Metrics from the onion model

Metric	Quantity	Notes
Circles	4	
Slots	11	
Roles	5	Shown only by solidity of slots (and in pop-ups); can have >1 role per slot, as with our model
Empty slots (roles are missing)	7	Grayed out in onion model
Slots with multiple stakeholders	1	
Slots linked to circles or other slots	11	
Slots not related to anything else	0	
Persons named	0	
Roles without named persons	5	Not visible in onion model (need pop-ups)
Roles linked to use case actors	2	
System or operator roles not linked to actors	1	

Conversely, a slot that is filled with multiple roles suggests at the very least a fine division of responsibilities; at worst, it can indicate that serious stakeholder conflicts are possible. It is entirely reasonable for an industry sector to be subject to several regulators, for example. An aircraft project must comply with normal health-and-safety regulations governing workplaces, the aviation authorities must certify that the airframe is airworthy, and the avionics and radio equipment must comply with radio regulations. Conversely, a product development with two departments each believing they have purchasing authority over the project is certain to run into difficulties. After all, "no man can serve two masters."

A third interesting metric is whether any slots exist without defined relationships (shown as dashed lines in the onion diagram). At the least, each should have a legitimate interest in the product—that is, a working definition of stakeholder. Interactions with other stakeholders will also likely occur, and identifying these explicitly early in a project is often helpful. For example, computer operators will consult a software help desk when they encounter difficulty with the software, so in general you can expect the help desk and normal-operator slots to interact.

Finally, stakeholder roles in the "our system" circle are directly involved with the product. In our terms, they operate the kit, equipment, or product that the project will produce. As will be obvious to UML aficionados, these roles will often correspond directly to the project's use case actors—assuming that this is a product development and that we are talking about product rather than business use cases. Where tools handle both the stakeholder model and the product use cases, we can copy and link "our system" roles to actors in the use case model (www.scenarioplus.org.uk) and to the product use cases (www.volere. co.uk). Any roles in the "our system" circle that are not so linked could have been overlooked.

Maintaining stakeholder involvement

Once you find the appropriate stakeholders, you can do more detailed work. You model the context, partition the problem, define the requirements, and build the product according to whatever combination of cyclic, incremental, or agile approach you are using. But regardless of the approach, one thing is common: appropriate participation of the necessary stakeholders directly influences your work's success.

Suppose that you have thoroughly analyzed your project's sociology and have identified the appropriate roles and the stakeholders to occupy them. And all the work you're doing maps directly to your stakeholder models. All will be well as long as nothing changes—and there's the rub. The world your stakeholder models reflect is guaranteed to change during your project's lifetime, and these changes are likely to affect your stakeholder involvement. Certain indicators signal the need to do some work on stakeholder involvement.

Role changes

Any reorganization in your contained system or anywhere in the wider environment might take stakeholders away from your project. Changes in job titles, department names, repartitioning of the organization, and new geographical locations all signal potential changes to your stakeholders. This might require you to add new roles.

Hiring and firing

When someone leaves the organization, gets promoted, or changes jobs, you'll probably need to make some changes to your stakeholder roles. We have often seen this slip through the cracks because when an occupant leaves a role, he or she often forgets to inform the new role occupant of his or her stakeholder responsibilities.

Boundary changes

If the context of your investigation changes, new or different knowledge boundaries will probably exist. This might happen because extra functionality is introduced to the project or a new external influence could exist, such as a new law. This often introduces new roles and hence new stakeholders. At the start of a project, we map the context of investigation to the stakeholder analysis and monitor the mapping to ensure that the world we are investigating is still in sync with our stakeholders and vice versa.

Missing feedback loops

Finding the appropriate stakeholders is one thing; keeping them involved throughout the project is another. Everyone is increasingly busy—unless you can keep your project in the limelight, people lose interest and treat it with less urgency. You need to design feedback mechanisms so that the stakeholders know what's going on and why it's relevant to them. Part of our project planning involves building an agreed-on project knowledge model with the stakeholders—specifically what we mean by each project deliverable and who is interested in and responsible for which deliverables. Then we design frequent feedback loops to make the agreed deliverables visible to the appropriate stakeholders.
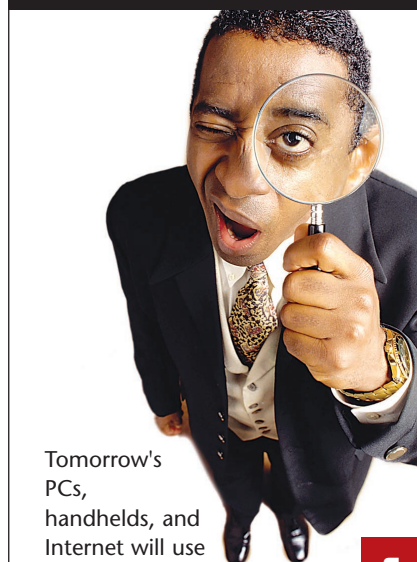
M issing stakeholders result in missing requirements and escalating project costs. Once you've analyzed your project using a combination of onion models and stakeholder analysis templates, your project sociology models become a tool for monitoring the continuing involvement of the appropriate stakeholders and dealing with changes. ⟨sw⟩

Ian Alexander is the director of Scenario Plus and co-author of *Writing Better Requirements* (Addison-Wesley, 2002). Contact him at ian@scenarioplus.org.uk.
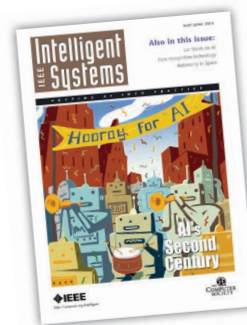
Suzanne Robertson is a principal of The Atlantic Systems Guild and coauthor of *Managing the Requirements Process* (Addison-Wesley, 1999) and *Complete Systems Analysis* (Dorset House, 1994). Contact her at suzanne@systemsguild.com.