# Chapter 2.4
# A Taxonomy of Stakeholders:
## Human Roles in System Development

**Ian F. Alexander**
*Scenario Plus, London, UK*

## ABSTRACT

Systems engineers have often paid too little attention to the nature of the so-called "users" of products under development. These are better called stakeholders, as many roles are involved, and few of those are in direct contact with the developed products. A simple and robust conceptual framework for classifying development stakeholders—a taxonomy—is proposed. The taxonomy is product centric, with concentric "circles" denoting broad categories of stakeholders. Within these, generic "slots" describe typical classes of stakeholders; these are subdivided into "roles," which are expected to vary at least in name with the domain. Examples are given, and a popular template is reanalysed using the framework. The taxonomy has immediate value in identifying and validating stakeholder roles in requirements elicitation, helping to ensure that key viewpoints are not missed, and hence reducing the risk of instability and failure during development.

## INTRODUCTION

### Motivation

The structure of stakeholder roles and their relationships such as surrogacy have been very little investigated in the requirements world (though much more extensively in the political, ethical, and information systems worlds—one reason for believing that an attempt at an interdisciplinary look at stakeholders may be worthwhile). Requirements work almost inevitably involves dealing with stakeholders of widely varying kinds, and hence demands a commensurately wide range of elicitation techniques. The first step in identifying which techniques should be applied is therefore to identify the stakeholder composition for a new project, and this in turn demands a suitable taxonomy of stakeholders.

Too many projects focus their attention too closely on the product—perhaps especially when that is software—to the exclusion of nonoperational roles, and often even of secondary operational roles such as maintenance. I suspect this is

due to "inside-out thinking" where the system is seen as important, and the user as secondary. Such thinking is a hangover from the past. When I was at university, an IBM 360 mainframe occupied the only air-conditioned tower on the campus. Students were permitted to approach only the card reader with a deck of punched cards; only trained operators were allowed upstairs to see the computer itself. This was truly a priestly hierarchy (in Greek, *hieros* is holy, *arches* is ruler) of operator roles. As Christopher Locke writes, "Even the word 'users' is an artefact of the [command-and-control] mentality" (Levine, Locke, Searls, & Weinberger, 2000). It is time to move on from treating "the user as a computer peripheral" (in Julian Hilton's words). The system is made for man, not man for the system.

Many industrial development problems seem in practice to be caused not so much by a failure to write requirements, as by a failure to perceive that specific stakeholders' viewpoints are relevant. That failure causes whole groups of requirements, typically those related to scenarios involving the missing stakeholders, to be missed.

A similarly unhappy result is obtained when one stakeholder, for example, a software developer, assumes one scope for a product, while another stakeholder, for example, a purchaser, assumes another. For instance, when a developer assumes that it will be sufficient to design, code, and test a piece of software, but the purchaser hopes to have everything set up and that the operators are trained, then the points of view of the installer, the trainer, and to some extent that of the operators have not been adequately considered and made explicit. Legal disputes and financial losses are then likely.

It seems likely that stakeholder composition is a good predictor of project risk; hence, it should be cost effective to characterize projects at their initiation according to their likely stakeholder impact (and to other variables, such as safety relatedness, technological innovation, similarity to previous projects, and so on).

In addition, maintaining a model of stakeholders throughout a development allows changes in stakeholder composition to be modeled explicitly, leading to appropriate changes in requirements.

Stakeholder surrogacy has powerful and paradoxical connotations in requirements engineering. It is almost a dogma that projects should seek out ever-closer dialogue with stakeholders—consider the current fashion for integrated project teams, facilitated workshops, rapid prototyping, agile development with user stories, and so forth. Yet all the time, the obvious truth is glossed over: that it is remarkably rare to be able to talk to many stakeholders in the flesh. Every requirements engineer knows that the basic answer to the client organization's boss who says, "I know everything that happens in my department; ask me," is, "Well, that's fine, but can I see if the people on the shop floor know of any small issues?" To put it more formally, standardized procedures, no matter how critical[1] and how carefully defined in writing, are always modified when operationalized "on the shop floor." Therefore, it is essential to talk to stakeholders directly—without intermediaries—to find out what actually happens. Yet, requirements engineers are themselves intermediaries! Stakeholder surrogacy is accordingly discussed at some length below.

Worse, many kinds of stakeholders are inaccessible: They may be distant geographically, separated by contractual and procedural barriers, hidden within organizations (with cultural barriers), simply unaffordably expensive to contact given scarce project time and resources, or not yet in existence (for future products).

The naive "go and talk to the users"—whoever may be meant by that phrase—is therefore far from helpful as advice. This paper considers what we mean by stakeholder roles in development projects, and it offers both a theoretical framework for classifying them and some practical suggestions for making use of that knowledge.

It may be that the approach can be applied outside system development, for example, to model