A Better Fit – Characterising the Stakeholders

Ian F. Alexander
Scenario Plus
ian@scenarioplus.org.uk

Abstract. The foundations of support for any business process lie in understanding the needs of people, the stakeholders in business processes and support systems. Stakeholder analysis has been neglected to a surprising extent, yet it may hold the key to understanding and supporting people in their business processes. This paper proposes a simple but powerful way of modelling stakeholders. Free tool support is described.

1. Motivation

There is often a mismatch between the needs of a business for support for its processes, and the hardware and software 'kit' that is provided for the purpose. All too often, the requirements are captured from just one or two points of view, neglecting other stakeholders whose viewpoints may be very different, even hostile. For example, the current fashion for modelling processes as Use Cases [5] with (possibly human) Actors tends to focus attention strongly on operational roles (and on the design of software and its user interfaces). The writing of 'Business Use Cases'[7] widens the focus slightly to look at business processes in the absence of software, but still essentially excludes all non-operational stakeholders and goals.

A little reflection suggests that despite this narrowness of vision, there are stereotypical roles that are typically (but not always) involved: and that these could be arranged in a set of concentric circles. This led to the development of a simple approach to modelling and representing Stakeholders and their Viewpoints in a requirements database, with a graphical front end.

2. The Onion Model

A development's stakeholder sociology can be modelled graphically on an Onion Diagram (**Figure 1**).

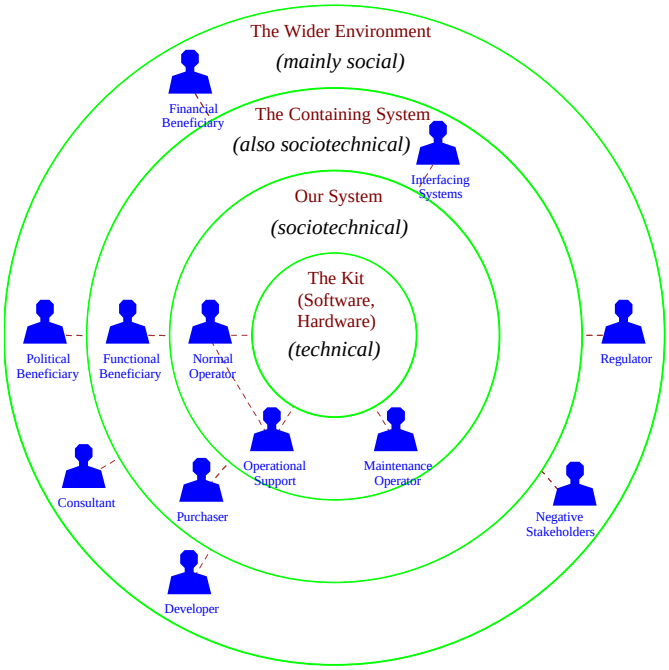

Figure 1: Onion Diagram of System Stakeholders**.** The Onion Diagram displays a customisable set of named ‘slots’ (silhouette icons), containing stakeholder roles, in 3 or more 'circles' centred on the 'kit' or product. E.g. ‘Helpdesk’ is a possible role (not shown) in the ‘Operational Support’ slot in the 'Our System' circle.

This deceptively simple-looking model documents and indexes a wealth of information about a development. The stakeholders in the outer rings do not participate in day-to-day operations – in business processes – but contribute to or receive benefits (or harm) from them. The web of relationships creates the potential for many kinds of conflict of interest and much misrepresentation of viewpoints: it is the job of stakeholder analysis to untangle this web.

Structure of the Onion Model

The Onion Model consists by default of a set of three concentric **Circles**. Each Circle logically contains the Circles drawn inside it. (The term ‘**Annulus**’ may be used to refer to the contents of a Circle without the Circles it contains.) Other Circles may be added.

A ‘**Circle**’ denotes a subset of entities in the world relevant to a development project. Those entities are represented as (stakeholder) Slots and Circles.

The innermost ring denotes the '**Kit**', the hardware and software supporting the business process. It does not contain Stakeholders.

A '**Slot**' is a class of Roles expected to be significant. For example, 'Normal Operator' is a default Slot in the 'Our System' Circle. The existence of the slot is a prediction that most systems will need one or more 'Normal Operator' Roles.

A '**Role**' is a class of Stakeholder with a distinct relationship to the Kit under development. For example, in an order-processing system, both Order Clerk and Accounts Clerk could be Roles within the 'Normal Operator' Slot.

A Role is said to be a '**Surrogate**' if it is conducted on behalf of another Role which cannot speak directly for itself. For example, a Product Manager acts as the purchaser of a new type of Product until such time as the Product can be sold on the mass market; then the Consumers buy instances of the Product (or not) for themselves. (A mass-market Product is a special kind of 'Kit'.). Surrogacy is important as it introduces a chance of misrepresentation of stakeholder viewpoints, and hence loosens the fit between business processes and support systems.

A Role is said to be '**Negative**' if it is associated with Viewpoints opposed to the business process and its successful implementation in a support system. Viewpoints – statements of the points of view of stakeholders in a Role with respect to the Kit – do not appear in the Onion Model as such, but can be documented as text attributes of Role objects in an associated database (Figure 2).

ID	Stakeholder Slots and Roles	Viewpoint	Polar Angle
SH-37	**2.1 The Wider Environment**		
SH-55	**2.1.1 Consultant**	gives advice	150
SH-56	**2.1.1.1 Tools and Techniques**	advises on efficient use of technology	
SH-65	**2.1.1.2 Environmental Impact**	advises on likely impact of product on the environment	
SH-39	**2.1.2 Financial Beneficiary**	stands to profit from system	240

Figure 2: Viewpoints as Attributes of Slot and Role Objects. In this example, 'Consultant' is a Slot in the 'Wider Environment' Circle, containing two currently-defined Roles. Attributes include 'Viewpoint' text.

A 'Stakeholder' is an individual person or other legal entity able to act like a person (e.g. a Limited Company, an Industry Regulator, a Registered Charity) playing one or more Roles.

Circles

The four default Circles (others can be added) used in the Onion Model are:

1) '**The Kit**' (or 'The Product'): the hardware and software under development.
2) '**Our System**': 'The Kit' plus its human Operators and the rules governing its operation.
3) '**The Containing System**': 'Our System' plus any human Beneficiaries of Our System (whether they are involved in operations or not).
4) '**The Wider Environment**': 'The Containing System' plus any other Stakeholders.

The Slots of the Onion Model

The Onion Diagram by default displays the following 'Slot' icons in the named Circles. These are what we consider to be the bare minimum of Slots.

Our System

Normal Operator: roles that involve giving routine commands and monitoring outputs from the product, whether these are via a human-computer interface or not.

Maintenance Operator: roles that involve maintaining the Kit, such as servicing hardware, and diagnosing and fixing faults. (So-called maintenance of software involves changing the design of the Kit, and is generally the responsibility of our Developer Slot; it is not maintenance in our sense.)

Operational Support: roles that involve advising Normal Operators about how to operate the Kit. These roles are very close to operations but support rather than conduct productive use of the Kit itself.

Containing System

Functional Beneficiary: roles that benefit from the results or outputs created by the System. For example an astronomer benefits from the astronomic data captured by a space telescope though he or she cannot operate the instrument directly. Since Systems are or should be designed to produce results, this is an important Slot. Note that with the tendency of many systems to become interactive, these roles can be combined with Normal Operator (see Discussion of 'users' below).

[Responsibility for] **Interfacing System**: roles responsible for neighbouring systems that have electronic or other interfaces to/from the Kit. Such systems behave much like human Operators in terms of demanding specific capabilities from the Kit, but naturally the interfaces are precisely defined as protocols, etc.

Purchaser: roles responsible for having the Kit developed. These can include Project Manager (working within an agreed budget and timescale), Product Manager (with knowledge of what can be sold) and Procurement (responsible for obtaining a contract with a supplier).

Champion (aka '**Sponsor**'): role responsible for initiating development of the Kit, for obtaining funding for it, and for protecting the development from 'political' pressures and funding cuts. The role requires positional power within the organisation. The Champion is perhaps the best person for the Requirements Engineer to meet with first; an effective Champion can indicate the scope and purpose of the development, the opportunities and threats, and can suggest who the key stakeholders are. All of this helps to cut the risk of mismatch between process and support system.

Wider Environment

Negative Stakeholder: any role that could be harmed by the System physically, financially, etc. For example, employees finding their decision-making abilities reduced by 'intelligent' software might oppose its introduction.

Political Beneficiary: any role in public office or private business that can benefit in terms of power, influence and prestige through the success of the System.

Financial Beneficiary: any role that can benefit financially from the success of a System.

Regulator: any role responsible for regulating the quality, safety, cost or other aspects of the System, e.g. a Financial Services Authority.

Developer: any of the many roles (requirements engineer, analyst, designer, programmer, tester, safety engineer, security engineer, electronics engineer, metallurgist, human factors engineer, etc) involved directly in development. Note that none of these roles are operational unless tied into operations via a maintenance contract – in which case the affected people have hybrid Developer / Maintenance roles.

Consultant: any of the many roles (marketing expert, software expert, business analyst, management specialist, etc) involved in supporting some aspect of System development, characteristically from outside the development organisation. Internal consultancy is possible but problematic, as it is hard to speak out in the face of 'political' pressure within the organisation (without the help of a Sponsor, see above).

3. Discussion

'User' – a Hybrid Role

One Slot that is not provided is 'User'. I consider this term to be both dangerously overloaded and confusing. It has many loose meanings in colloquial engineering parlance, including:

- 'all stakeholders';
- 'all stakeholders other than us, the developers' (this verges on the derogatory, and should be avoided);
- 'any stakeholder who gets any benefit from the System';
- 'any stakeholder who operates the Kit'.

Perhaps its main meaning is a hybrid of Normal Operator and Functional Beneficiary, as when the 'user' of a mass-market product operates it for personal enjoyment. Terms like 'user', 'end-user' have little value, and may harmfully divert attention away from non-operational stakeholders.

Greater care in choosing names for stakeholder roles should provide a rapid reward through better coverage of stakeholders, especially those not involved in day-to-day operations, and hence through more complete representation of viewpoints and business processes, and more accurate requirements.

Overlapping Taxonomies

Clearly, this model is focussed mainly on people. It is possible to create process-centric models around the Developer and development activities, the business processes and system usage. Most of the requirements literature is tightly system-centric to the exclusion of most kinds of stakeholder.

The Volere template [11] contains an interesting and useful list of stakeholders – certainly far more comprehensive than most others in the literature. The list is developer-oriented: most of the roles fall into the Developer and Consultant slots. This makes sense from the point of view of the Volere classification of non-functional requirements: perhaps the thinking ran "here are some Usability and some Security requirements, we better consult some appropriate Specialists about these". Thus the Volere template is perhaps more specifically requirement-centric than developer-centric. On the other hand, the Negative and Champion Slots are rightly stated to be 'project / product'-centric.

The gaps in Volere – useful as it is – hint at the probably far more serious gaps in other, more sketchy attempts to deal with stakeholders. A stakeholder analysis can quickly identify Onion Model Slots that are unpopulated, leading to urgent questions like 'Are you sure that no regulator is involved in this business process?' – and hence to a better fit between systems and reality.

Surrogacy

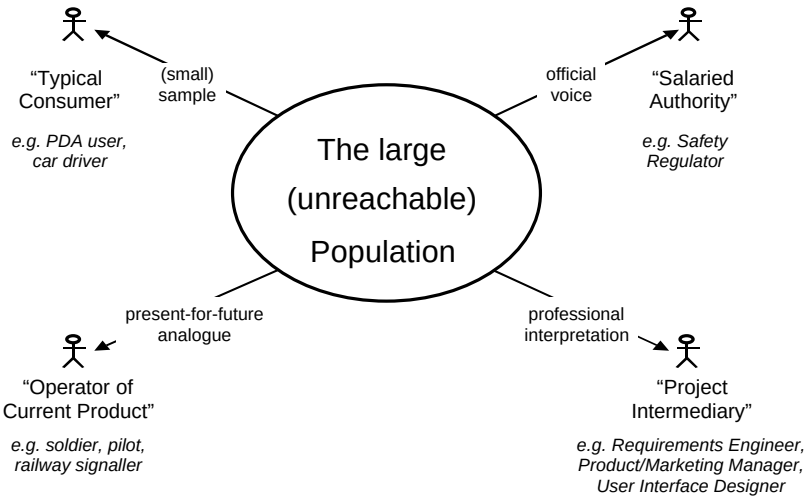

Figure 3: Four Kinds of Stakeholder Surrogacy

One aspect of stakeholder sociology that is brought out clearly on an Onion Diagram is surrogacy – the representation of one stakeholder's viewpoint by some other person. Surrogacy has (when it has been considered at all) always been seen as a dangerous obstacle to successful requirements work. It threatens to loosen the 'fit' between business processes and the tools and systems developed to support those processes. It is therefore remarkable that stakeholder surrogacy (Figure 3) is central to requirements engineering, unavoidable in at least some forms, and yet scarcely discussed in the literature.

We may at least hope that an awareness of surrogacy in general, and a detailed analysis of the forms it takes in specific businesses, will lead to a better fit. This might happen in practice through:

- increased attention and sensitivity to the risk of hearsay requirements – 'I believe that the ABC operators would probably want XYZ' masquerading as 'the ABC operators want XYZ';
- increased precision in the selection of surrogates, e.g. with properly-randomised samples of consumers, or by obtaining a second professional interpretation, etc.;
- seeking out non-surrogate stakeholders, where appropriate.

4. Tool & Template Support

The taxonomy presented here has, as described above, several possible practical applications. A graphical tool has been implemented in a requirements traceability tool environment (Telelogic DOORS), along with a document template in formats suitable for word-processing and spreadsheet tools. Both are available for free download [9].

5. Related Work

Stakeholder Analysis

[8] distinguishes clients, customer, and other stakeholders including subject matter experts, marketing people, product managers and so on (pp356-358) and attempts to prioritise them: 'the principal stakeholders are the users, clients, and customers' (p35). However the general effect is of a flat unstructured list of many interested parties.

[3] contains a chapter on identifying stakeholders (pp 19-26). It distinguishes 'users' from clients, suppliers, managers 'who are concerned for the system to succeed', and regulators, and briefly discusses the role of people in development organizations (p7).

There is much pragmatic wisdom on dealing with stakeholders in Peter Checkland's many writings, e.g. [4]. He does not on the whole focus on the development of products or services, and indeed (p312) he seems to believe that that is the easy part of the problem! Despite Checkland's fame and the undoubted power of his 'soft systems methodology', the approach has made little impact on the quality of stakeholder analysis in systems and software development.

Goals and Viewpoints

Two areas of requirements engineering related to Stakeholder analysis are Goal Modelling and Viewpoint Analysis. There is an extensive literature on Goals; a good starting point is [12]. This, the Toronto school, believes that stakeholders' needs can be effectively handled by modelling goals, whether 'hard' (functional) or 'soft' (qualities of service). The i* notation permits the graphical description of both

dependencies between stakeholders and the relatively private intentions of stakeholders. However, the approach is complex, which is perhaps why it has remained unfamiliar to the vast majority of analysts for more than a decade since it was first described. It is also an academic framework, without the backing of a major software corporation behind it.

A readable introduction to Viewpoints is [6]. It describes a method of 'Viewpoint-Oriented Requirements Development' (VORD), explicitly a stakeholder-centred approach. It distinguishes 'direct' and 'indirect' stakeholders, roughly our 'operational' and 'non-operational' categories, but again it does not cover most of the Onion Model Slots, and has made little headway among practitioners.

Negative interactions can be modelled as negative scenarios or Misuse Cases [1]. This technique has been applied in industry for security, safety, and trade-off analyses, and in principle is quite a powerful way of analysing conflicts and intentions. It also employs the fashionable Use Case notation, which may be both its strength and its weakness: it may be noticed, but it contributes mainly to the elicitation of operational threats and hence of operational requirements.

Human Aspects of System Development

There is an extensive literature on human aspects of system development. It seems however to focus quite naturally on the 'user' at the expense of all other roles. This is no place for a full literature survey; a good starting-point is [10]. This distinguishes customers, users, managers, software engineers, system testers and system maintainers (p17). These roles are notably concentrated around 'Our System', but to be fair they are said to be those that can write requirements. Stakeholders are further classified (p56-7) as Primary, Secondary, or Tertiary, roughly our 'Normal/Maintenance Operators', our 'Functional Beneficiary', and our 'Political Beneficiary' respectively. This is sensible and pragmatic, but still fails to consider most of the default Slots of the Onion Model.

Surrogacy

Surrogacy has apparently scarcely been researched as a requirements engineering issue. It is mentioned briefly in some of our own earlier work on Stakeholders [2] and in a practical way on the Scenario Plus Stakeholders template [9].

6. Conclusion

The trend in system specification away from the machine towards human "users" leads to a natural end-point in modelling and analysing the nature, goals, and viewpoints of human stakeholders. While there has been some academic interest in goal and viewpoint modelling, the stakeholders themselves, and the dangers of stakeholder surrogacy, seem largely to have been overlooked.

It may be that systems will support and 'fit' business processes better when stakeholder viewpoints and relationships are studied more carefully. The quality of the fit depends on many aspects of systems engineering, including the whole of the requirements process from initial mission-setting to stakeholder identification and analysis through to eliciting and documenting; the translation of requirements into specifications; the completeness of the traceability from specifications to requirements; and not least the thoroughness with which the requirements are verified by test (or other means). However, the stakeholders occupy such a vital position in the whole development process that paying them more attention is almost guaranteed to be a good investment.

References

1. Alexander, Ian, *Initial Industrial Experience of Misuse Cases in Trade-Off Analysis*, IEEE Joint International Requirements Engineering Conference, 9-13 September 2002, Essen, Germany, pp 61-68
2. Alexander, Ian, *Stakeholders – Who is Your System For?*, Computing & Control Engineering, Vol 14, Issue 1, pp 22-26, April 2003
3. Alexander, Ian and Richard Stevens, *Writing Better Requirements*, Addison-Wesley, 2002
4. Checkland, Peter and Jim Scholes, *Soft Systems Methodology in Action*, John Wiley 1999
5. IBM Rational 2004, *UML Resource Center (website):* http://www.rational.com/uml/index.jsp (UML including Use Case documentation for business modeling, etc)
6. Kotonya, Gerald and Ian Sommerville, *Requirements Engineering : Processes and Techniques*, John Wiley, 1997.
7. Ng, Pan-Wei, *Effective Business Modeling with UML: Describing Business Use Cases and Realizations*, The Rational Edge, Nov 2002, (http://www.therationaledge.com) http://www.ibm.com/developerworks/rational/library/905.html
8. Robertson, Suzanne and James Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999
9. *Scenario Plus website*: http://www.scenarioplus.org.uk (onion model tool for Telelogic DOORS, stakeholders template for Microsoft Word, etc)
10. Sutcliffe, Alistair, *User-Centred Requirements Engineering, Theory and Practice*, Springer, 2002
11. *Volere Stakeholder Analysis Template*, http://www.volere.co.uk
12. Yu, Eric and John Mylopoulos, *Why Goal-Oriented Requirements Engineering*, REFSQ 1998, http://www.cs.toronto.edu/pub/eric/REFSQ98.html