Requirements Eng (2002) 7:152–171

Requirements
Engineering

Large-Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering

Mark Bergman[a], John Leslie King[b] and Kalle Lyytinen[c]
[a]Department of Information and Computer Science, University of California, Irvine, California, USA; [b]School of Information, University of Michigan, Ann Arbor, Michigan, USA; [c]Department of Information Systems, Case Western Reserve University, Cleveland, Ohio, USA

This paper addresses the political nature of requirements *for large systems, and argues that requirements engineering theory and practice must become more engaged with these issues. It argues that large-scale system requirements is constructed through a political decision process, whereby requirements emerge as a set of mappings between consecutive solution spaces justified by a problem space of concern to a set of principals. These solution spaces are complex socio-technical ensembles that often exhibit non-linear behaviour in expansion due to domain complexity and political ambiguity. Stabilisation of solutions into agreed-on specifications occurs only through the exercise of organisational power. Effective requirements engineering in such cases is most effectively seen as a form of* heterogeneous engineering *in which technical, social, economic and institutional factors are brought together in a current solution space that provides the baseline for construction of proposed new solution spaces.*

Keywords: Functional requirements; Heterogeneous engineering; Political requirements; System failures; System requirements

1. Introduction

Large-scale software development has remained a high-risk proposition despite huge advances in computing and telecommunications technologies. Large projects in particular continue to fail at an unacceptable rate [1–6]. While some troubled projects are turned around, many projects seem to be successful only at random. Improved software tools, modelling methods and process technologies for performing requirements engineering (RE) have yet to provide sufficient progress in avoiding failures in such initiatives. Advances in technologies and modelling techniques alone are inadequate to save large and complex projects. We must turn our attention to a wider set of issues and try to understand how technological, organisational and institutional changes are inherently interwoven in such initiatives. Only then can we expect to make progress in understanding how system developers might successfully state and manage requirements for such systems.

The following example by Drummond [7] provides a helpful starting point. On 11March 1993, the financial world was shocked by the sudden cancellation of the London Stock Exchange Taurus project. Taurus had been under development for more than six years, following over 20 years of deliberations about how to restructure and organise the exchange's trade settlements. Taurus was expected to enable a radical restructuring of the securities trade by forming a backbone system for a fully automated settlement process involving digitalisation of bonds and certificates. By then the project had cost the exchange $130 million, while securities companies had invested and additional $600 million [2]. The expected transformation of the exchange, which was heralded as the Big Bang in promotional literature, went out with a whimper. Taurus was cancelled before a single module was implemented when management discovered that the functionality and performance required of the system could never be delivered in an acceptable time frame.

Correspondence and offprint requests to: M. Bergman, Department of Information and Computer Science, University of California, Irvine, CA 92697-4650, USA. Email: mbergman@ics.uci.edu

Taurus was a technically complex project,[1] involving the use of several novel technologies. It also required massive changes in organisational and institutional operations of the financial industry. These operations were not well understood and were constantly changing. The project was also plagued with ineffective governance and poor financial controls that allowed specifications to change continuously. Management support had remained strong – so strong, in fact, that it was nearly a matter of religious faith. Taurus was literally seen as a vital tool in the challenge to increase London's competitiveness in the global capital market. Top management ignored clear warnings in 1989 and 1991 that organisational and technical risks were escalating. The thought of failure was simply too frightening to bear for the project's powerful backers, who included large institutional investors, the Bank of England, and the UK Department of Trade and Industry.

Taurus failed largely due to the fact that the system's purpose and design were unclear. The principals were not in agreement regarding what stakeholders the system should serve and how it was supposed to serve them. This was not a failure of 'method' or 'process'. The system's design followed well-established software engineering methodologies, and was documented at a detailed level. However, there was no clear understanding among project staff regarding the interaction of technical, business and institutional requirements, which seemed to change constantly. In place of sound factual knowledge, the Taurus advocates held an almost superstitious faith that the project would succeed. They dismissed criticism and recommendations for change on the grounds that years of argument over such matters had not accomplished anything ([2], p. 352).

Looking back it is easy to see that Taurus might have been more successful had the project been based on a more functional and evolutionary approach, but this does not go far enough. Taurus failed, in large measure, because (1) the political, business and technical issues on which success depended could not be clearly apprehended during requirements capture, and (2) the unforeseen interactions among these issues created cascades of new requirements and continuous shifts in the focus of the project. Although the project followed contemporary requirements engineering approaches, these were inadequate to the needs of a project of such size and complexity. Functional and evolutionary strategies are important, but for projects like Taurus the task of requirements engineering requires a strategy of 'heterogeneous engineering' [8–10]. The underlying concept of heterogeneous engineering is that requirements are actually networks involving technical elements and a deep understanding of the application domain. According to John Law, 'The argument is that those who build artifacts do not concern themselves with artifacts alone, but must also consider the way in which the artifacts relate to social, economic, political, and scientific factors. All of these factors are interrelated, and all are potentially malleable' ([11], p. 112).

This paper examines the problem of stating and managing requirements for large system development in terms of heterogeneous engineering, focusing especially on issues of power and interest amongst principals involved in a project. Our aim in this paper is to expand research agenda of requirements engineering by incorporating the concept of *political ecologies* in which requirements emerge [8,11]. We extend the contemporary strategy of requirements engineering, which we characterise as a 'low-level' *functional ecology* focus, to embrace a 'high-level' understanding of the political ecology in which the functional ecology becomes meaningful in practical terms. The resulting high-level requirements engineering strategy incorporates factors drawn from the social sciences, such as political mobilisation and negotiation and stabilisation in heterogeneous socio-technical networks. Following this strategy, we assert successful high-level requirements engineers must have the skills to incorporate both functional and political ecologies in their work.[2]

The idea that requirements engineering is a political process is already well established, although it is often overlooked in the prescriptive literature. The information systems literature has developed a robust discussion of this issue (e.g. [12,13]), and the software engineering and requirements engineering communities have joined the discussion [14–16]. A well-established stream of research in organisational theory that draws upon the politics of organisational change is also relevant to this discussion [17–19]. However, the RE literature as a whole contains little discussion of the need to incorporate such perspectives into the practice of RE.

This remainder of the paper is organised as follows. In Section 2, we introduce the concept of the *functional ecology of requirements*. Section 3 abandons the functional ecology assumption of goal congruence among stakeholders to develop a *political ecology of requirements*. Section 4 extends the narrow conception

[1]Project manager John Watson claimed in 1992 that it was the single largest software project in Europe at that time [7].

[2]Law comments on Edison's success in establishing the electrical systems as follows: 'Edison's problem … was simultaneously economic (how to supply electric lighting at a price that would compete with gas), political (how to persuade politicians to permit the development of a power system), technical (how to minimize the cost of transmitting power by shortening lines, reducing current, and increasing voltage), and scientific (how to find a high-resistance incandescent bulb filament)' ([11], p. 112).

of politics and political negotiation found in the functional view to include multi-party negotiation. Section 5 analyses the emerging hybrid of functional and political ecologies and its potential for RE practice. The paper concludes by drawing some consequences of the political models of RE on both research and practice of requirements engineering.

2. The Functional Ecology of Requirements Revisited

Two key assumptions frame traditional RE [20–24]. One is that requirements exist 'out there' in the minds of stakeholders (users, customers, clients), and they can be elicited through various mechanisms and refined into complete and consistent specifications. The second is that the key stakeholders operate in a state of goal congruence, in which there is widespread and coherent agreement on the goals of the organisation. The main problems from this perspective have been a lack of consistency in requirements elicitation and specification process that results in either missing or erroneous information about requirements. The strategy of RE research has been to improve the process of *formalising* the *content* of system functionality and constraints, and mapping the results into an appropriate *format* that enables control and automatic manipulation of solutions. It is hoped thereby that the objective of creating complete, consistent and implementable representations can be achieved.

An important contribution to understanding the problems in this construction appears in the literature on *emergent functional* perspectives in RE [25]. The term *functional* denotes RE as a task for achieving practical objectives, such as to make project accomplishment more efficient and/or effective. The term *emergent* captures an evolutionary view of RE in which organisational objectives, problems and solutions are dynamically constructed rather than simply elicited from stakeholders and documented. The dynamic perspective recognises the inherent complexity of the socio-technical systems, which results in their resistance to being 'engineered' by human beings subject to bounded rationality. Complexity is an outcome of (1) the equivocal nature and size of the search spaces entailed in any consideration of options [26] and (2) the fact that the parameters of the project (and its attendant search spaces) can grow or change in a non-linear manner [26,27]. The emergent functional ecology of RE seeks to understand the sources of this complexity, and offers a means to address the complexity challenge that leads to the functional ecologies perspective.

2.1. Requirements Analysis Framework

We draw on two sources to understand how large-scale RE processes unfold. The first is Simon's theory of bounded rationality [28,29], which suggests that it is virtually impossible to find optimal solutions to complex problems in a reasonable amount of time due to the inherent limits of human information processing. At best, such problems are amenable to 'satisficing' – solutions that are satisfactory but not optimal. Complex processes such as deciding on organisational goals or establishing organisational procedures are dominated by heuristics that limit rather than expand search spaces. This is recognised in a way by contemporary RE methods, which seek to reduce complex social processes to simple models. This strategy actually works reasonably well when the problem and the solution spaces under consideration have already been fixed within a relatively limited size. However, this strategy scales up poorly when the problem space is inherently large and the solutions spaces are not already bounded. The result is the tendency for an explosion in the size of the search space due to uncertainties related to the difficulty of knowing the full extent of the problem, its solution spaces and the relationships between them.

Our second source of insight is behavioural theories of decision-making [30,31] that suggest that complex organisational decisions are not discrete events in which pros and cons are systematically weighed in an effort to achieve optimal decisions. Rather, organisational decision-making occurs through what has been called the 'garbage can model', a protracted process of iteration in which problems search for solutions, solutions search for problems, and decision-makers, e.g. principals, search for decisions to be made [30,32].[3] The traditional notion of 'problem → solution' that underlies most established RE models is transformed into an evolutionary cycle of iterations in the form 'solution → problem → solution'. The importance of this for high-level RE cannot be overstated, because it explains why many large-scale RE decisions occur through a process of solutions searching for problems rather than the other way around. In conditions where the search space of both the problem and the solutions is large and perhaps unbounded, RE specialists assert order by imposing their own competencies, irrespective of whether their competencies are really appropriate to the situation.

To summarise, bounded rationality suggests that human capacity to understand and act rationally upon

[3]It is called the 'garbage can model' because of the relatively permanent nature of the cans in which different solutions, problems and decisions are 'thrown' within an organisational arena.

the challenge posed in large-scale system efforts results in the abandonment of optimisation and the adoption of satisficing strategies. This is not a problem as long as the RE specialists realise when they have abandoned optimisation in favour of satisficing, but it is a major problem when that shift is not recognised because the RE process proceeds under the false assumption that the problem is actually understood. This often results in the fatal mistake that important factors impinging on the project's success must have been considered when, in fact, they were not, simply because the process itself does not reveal that anything was overlooked. Behavioural decision theory suggests that decision-making in complex problem domains seldom follows a straightforward 'problem → solution' cycle in which experts identify a problem and then identify an appropriate solution. Often, experts search for complex problems that seem amenable to the solutions with which they are most competent, and then attempt to impose the solution on the problem. Again, this is not necessarily a bad thing – sometimes there is simply no other alternative available, and it makes sense to experiment and see what might work. But when the shift is unnoticed, it is easy to mistakenly assume that the imposition of the expert's solution on the problem was the result of a systematic explication of the problem that 'led' to the appropriate solution. The key to understanding the functional ecology of high-level RE is therefore found in the concept of 'solution' and 'problem', and the *iterative 'walk'* between them.

2.2. Functional Ecology Model of Heterogeneous Requirements

We now clarify several key parts of the functional ecology model with short examples from Taurus adopted from Drummond [7]. The examples appear in italics.

The functional ecology model views the concept of 'solution' as a *solution space*, which embodies a specific socio-technical configuration of computer- and communication-related technologies, skills, organisational resources (e.g. data, financial, physical), processes and aligned stakeholder groups, capable of exhibiting a stable set of behaviours and responses within an institutional environment. Any RE activity can be defined in terms of the following set of solution spaces:

- **S** – A *local solution space* is the current solution space and all locally accessible solution spaces that can be reached from the current solution space using available skills and resources offered by the principals. Any such move is itself a system: a new socio-technical ensemble that exhibits, enables and reinforces a set of organisational behaviours. A system at any point in time can be partitioned into a set of desired solution spaces that increase the utility value of all or most stakeholders, and a set of undesired solution spaces, which fail to increase utility value of all or most stakeholders.

  The local solution space of Taurus included the current settlement system inherited from the 1970s and all of the subsequent change initiatives to resolve the back office settlement problem between 1970 and 1990. The attempts to resolve the settlement problem under different projects involved over 15 different specifications of such systems ([7], pp. ii–xxviii). The final instantiation of the local Taurus solution space (i.e. the 'final' design) was backed by the Stock Exchange and the large institutional investors, but resisted by small investors who felt insecure over the outcomes of the Big Bang and were worried that their costs would increase.
- **GS** – A *global solution space* is the space of all feasible solution spaces, including those not currently accessible from the local solution space, that require mobilisation of all principals and technologies to be accomplished. All local solution spaces exists within relevant global solution spaces. That is, S is a subspace of GS.

  Taurus represented, in the minds of its most ardent supporters, the evolution of settlement activities not only for the City of London, but for all other stock markets from all over the world, over the last 50 years and beyond.
- **S_t** – The *current solution space* embodies the history of solved social, technical, economic and procedural problems that constitute the legacy of previously solved organizational problems at current time *t*. The current solution space exists within a local solution space. That is, S_t is a subspace of S.

  The status of the settlement process and its stakeholders in 1986 when the development of Taurus is started as part of the City's big bang included the solutions and processes carried within institutional investors, brokers and banks.

  Every RE process starts within an S_t. Using the 'garbage can model', we see that S_t forms a sourcing opportunity for an alternative space that is perceived to be more desirable, and which justifies the need to move into that new solution space, i.e. $S_t \rightarrow P \rightarrow$ New S (e.g. S_{t+1}). This other space, needed to mobilise the resources to carry out the move, is called a *problem space* (P). The 'garbage cans' here can be seen as collections, organised over time and according to

varying rationalities, of solution and problem space pairs that are seen as well matched by a set of diverse organisational stakeholders.

In Taurus, the 'garbage cans' included all of the solution–problem space pairs used to try to solve the settlement problem between 1970 and 1992.

A problem space is built within any S_t through a process called *problemisation*, where a space of anomalies is derived from S_t by some stakeholder.

- **Anomaly** – An inconsistency, observed by some stakeholder, between the current solution space and a desired solution space believed by the stakeholder to be achievable within the current solution space. An anomaly is not necessarily a 'problem', and most anomalies are not considered by stakeholders to be problems that need to be solved. However, some anomalies take on the status of problems over time.

Typical anomalies in the case of Taurus were cost of settlement, time to carry out the settlement, the competitiveness of the City and Stock Exchange, competition, changing agendas within banks and financial industry, increases in the private ownership of stock, and the emergence of huge financial risks if the system should fail.

- **Anomaly space** – The set of all anomalies that can be associated with the system under consideration.

A *principal*, for his or her own reasons, can transform one or more anomalies into problems:

- **Principal** – A stakeholder with standing. Standing refers here to the power to define and legitimise an anomaly as a problem to be solved by collective action, plus the demonstrated capability to mobilise means to address a defined problem. A principal is thus assumed to hold organisational power, or the access to means by which she or he can influence others to set up and sustain 'garbage cans' in the organisational arena.

In the case of Taurus, principals included the Stock Exchange, Bank of England, large institutional investors (through a committee called SISCOT – Securities Industry Steering Committee on Taurus) and the Department of Trade and Industry (DTI).

- **Problem** – An anomaly that is observed and acted upon by a principal and thereby placed into organisational agenda of 'need-to-solve' anomalies, thus possibly creating a new garbage can, or integrating the problem with an existing garbage can.

In the case of Taurus, one problem was the observed need to remove the antiquated physical settlement system.

A principal has normally more than one problem he or she deems important, and complex institutional domains can host many principals. The collection of all problems entertained by the set of all principals is called a *problem space*:[4]

- **P** – A *problem space* is the space of all problems implied by a current solution space S_t by all its principals.[5] A problem space, P, is a subspace of an anomaly space.

In real life, there are always too many problems in the problem space to be addressed by any single new system. Moreover, problems can be in conflict, or they can be too expensive to address with given resources. Sometimes even serious problems do not receive the attention they deserve by principals (i.e. due to limits imposed by organisational resources or lack of strong championing), or are refused the status of 'problem' by other principals (i.e. their desired spaces are exclusive). Hence, only a subset of problems is addressed at any time by any system. This selected set of problems to be addressed by a new system is called the *proposed system problem space*:

- $\mathbf{P_t}$ – A *proposed system problem space* is the space that contains all the recognised problems chosen by the principals at time *t* that justify the proposed system.[6]

In Taurus the proposed system problem space was the set of problems identified in the mid-1980s as the financial, legal and business risks due to the limitations of the existing settlement system that led to the specification of so-called Taurus 1. Taurus 1 was abandoned because of its high cost (estimated at £60 million) in 1986, but was revitalised on a larger scale in 1988 as a result of the sour experience of 'Black Thursday' in autumn 1987, which left £12 billion in unsettled trades in the hands of trading houses.

- $\mathbf{p_n}$ – A problem (one of at least *n* problems) within P_t.

The set of initial problems chosen by principals to define the system is called a *primary problem set*. By implication, the primary problem set can incur other problems that need to be addressed. These new problems are due to bounded rationality (i.e. missing them initially), non-linear changes in the solution space due to uncontrollable internal or external organisational changes, or emergence of new princi-

[4]A sharp-eyed reader may notice a circular definition around problems, principals and systems. This not a mistake but a purposeful choice as such circularity is a typical feature of complex requirements engineering tasks (e.g. [33]).

[5]A collection of principals with problems can exist intra- or inter-organisationally. Hence, a problem space covers problems that exist within as well as between organisations.

[6]Note that this is in most cases an ideological statement by principals: actual problems solved may not necessarily coincide with the proposed problem space due to goal displacement or uncertainty concerning what solution is achieved de facto.

pals as the implications of problems become clearer. The process of identifying these new problems is called *problem blossoming*:

- **Problem blossoming** – A process of recursive problem determination and expansion as determined by a set of principals. The process of problemisation often uncovers other anomalies that are deemed problems by some stakeholders. This can lead to explosive expansion of the current solution space, where recursion continues until all related problems are surfaced. It can also be prematurely stopped by one or more of the principals with sufficient power to halt the process.

In Taurus, problem blossoming took place when the developers observed that the move towards digital settlement required them to solve a host of additional problems. Among these were a lack of established legal frameworks for digital trade and identification of stock, uncertainty in deciding whether trading through a digital system would be mandatory, confusion over responsibilities and procedures related to shareholder information, difficulty in resolving who had the right to carry out trade in the stock, problems with maintaining required security and privacy, and inability to agree on how to establish a trace of trade.

Problem blossoming can uncover new *solution space sources* in S*t*:

- **Solution space source** – The source of one or more problems.
- $\mathbf{s_m}$ – One of at least *m* solution space sources in S_t.

The main solution space sources are principals, but problem blossoming can cause any part of the solution space that becomes a source of a problem in P_t to become a solution space source as well. This might include current applications in whole or in part, new software or related technologies, business processes, business or other rules, contracts and commitments, budgets, organisational structure, and so on. Solution sources are therefore usually heterogeneous, and problems that originate from heterogeneous solution sources will be heterogeneous as well.

In the case of Taurus, the current legacy systems of the Stock Exchange, the banks, the financial institutions, the rules of the trading, business processes around trading and the culture of the City were all new solution space sources.

The purpose of creating and deploying a new system is to solve all problems in P_t. The new or changed system and related changes instigated in the current solution space give rise to a new solution space:

- $\mathbf{S'}$ – A *future solution space* is an alternative local solution space postulated from the problem space analysis for the problem set P_t. Normally there are several alternative *future solution spaces.*

In the case of Taurus, 15 different versions of the plan emerged over time to enhance or expand the current system, change the responsibilities between different players, shift the burden of settlement to someone else, and so on. A member of the SISCOT committee described it thus: 'We started walking through water, then it became mud, then it became honey, then it became glue, and in the end, it was quick-drying cement' ([7], p. 55).

- $\mathbf{S_{t+1}}$ – A *proposed solution space*, or more simply *proposed solution*, is a subspace of the future solution space, S′, which includes the reconciliation of S_t to P_t, i.e. $S_t \rightarrow P_t \rightarrow S_{t+1}$.

Taurus involved an automated settlement in real time, which would also remove the concept of 'physical' bonds and stocks and replace them with a digitally mediated ownership of stock.

- $\mathbf{s_{t+1,i}}$ – A future solution source piece is one of at least *i* pieces from the future solution space, S_{t+1}. Each $s_{t+1,i}$ is a solution to one or more problems in P_t. Like solution space sources, future solution space pieces are usually heterogeneous.

The process of creating S_{t+1} from S_t is called *solution space transformation* (see Fig. 1):

- **Solution space transformation** – The process of reconciliation in which S_t is changed into S_{t+1} by solving for P*t*.

S′ contains all of the possible solution spaces such that when solving for P_t, S_{t+1} becomes *the* desired new solution space as selected by the principals. Altogether, the results of RE are an initial mapping, or transformation, from S_t through P_t to S_{t+1}, i.e. a set of mappings

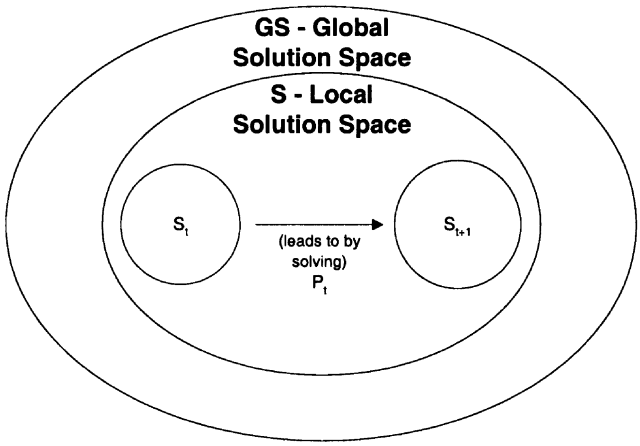

Fig. 1. Functional ecology solution space transformation.

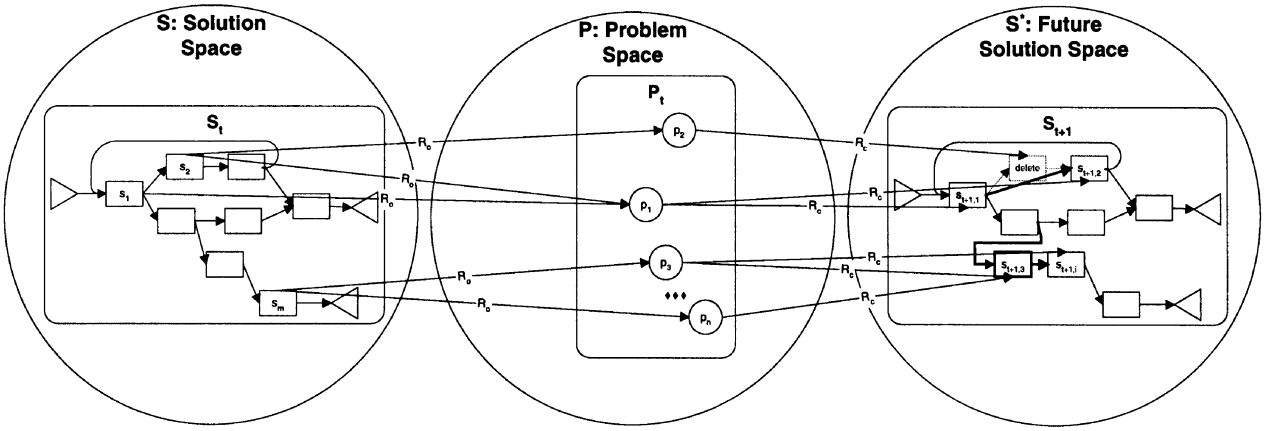

Fig. 2. Functional ecology model of heterogeneous requirements.

from a current local solution space to a new solutions space, which is backed by a set of problems heeded by principals.

The critical part of the proposed solution space transformation within the requirements is *whether the problems will be ameliorated when the move takes place into the new solution space*. This does not always happen: a proposed transformation can lead to inclusion of the wrong problems to solve [33]. There are well-known examples of this from information systems development, such as Markus [34] and Oz [35]. Another problem can occur in which there is no means to carry out the solution space transformation for technical, business or political reasons. Clemons et al. [36] discuss failures due to business reasons (e.g., the business cannot be transformed into a new solution space). Political reasons for failure are discussed in Section 3 of this paper.

In Taurus the solution space transformation emerged from within the Stock Exchange and SISCOT. There is no account of other solutions offered by financial institutions and consulting houses before the establishment of SISCOT. One of them, a combination of Taurus 3 and Taurus 8, was chosen by the Stock Exchange and SISCOT in 1989 as the primary solution space transformation because of its expected capability to address their major problems in settlement, and an estimated cost that was expected to be lower than the original Taurus 1 solution developed in 1986. These expectations were not met.

Overall, a *requirement* is a specific contextualisation representing a connection (e.g. mapping) between a solution space source s_m and a problem p_n or a problem p*n* and its solution $s_{t+1,i}$. A mapping from s_m to p_n is called an *objective requirement* that contextualises the objectives of the principles, denoted R_o in Fig. 2. A mapping from p_n to $s_{t+1,i}$ is called a *constraining requirement* or, more simply, *constraint* that contextualises what the future solution to a problem must abide by, denoted R_c in Fig. 2. There are many possible mappings that could become requirements. As with problems, a mapping must be supported by one or more principals to become a requirement.

Requirement arcs are built over time based on learning-by-doing and ongoing negotiation between principals as they strive to understand what is a problem and what is needed to solve the problem. Requirements continue to be built until either the principals believe there are enough of them to be sure their intentions are well covered, or one or more of the principles stops the construction process for other reasons. This search for requirements is constrained by bounded rationality and is dependent on the heuristics such as scenarios or participatory design that actors (i.e. requirements engineers, system analysts) use to construct the requirements arcs. Requirements mapping complexity is significantly increased by the fact that each principal can suggest more than one requirement, per solution–problem or problem–future solution pairs. In fact, any solution source can incur multiple requirements per s_m–p_n or p_n–$s_{t+1,i}$ pairs. There is thus a many-to-many mapping of requirements between S_t and P_t as well as P_t and S_{t+1}, as shown in Fig. 2.

Constructed requirements reflect their constructors. In other words, if solution sources or problems used in defining requirements are heterogeneous, then the resulting requirements are heterogeneous as well. Figure 2 represents a functional ecology of heterogeneous requirements such that, *if any of the components change, the requirements set will change too, leading to a spiralling problem-blossoming process.* The larger the $S_t \rightarrow P_t \rightarrow S_{t+1}$ spaces covered by a proposed new system,

the larger the requirements mapping. Even minor changes to a large requirements mapping can result in a huge change in the mapping.

Problem blossoming and the highly heterogeneous nature of requirements construction in Taurus led to a spiralling process of requirements expansion over a period of nearly six years (1986–1992). New technical features were added to the system to address observed business and institutional constraints. Problems with the chosen software package (Vista) and other technical constraints[7] *resulted in delays as well as new business requirements for business partners. These, in turn, led to new rounds of technical requirements and so on. Loose control over the development budget, as well as a lack of a clear business vision and strong political leadership, caused the functional ecology between different Taurus elements to explode in a non-linear way.*

3. The Political Ecology of Requirements

Functional perspective assumes that the principals proposing a system are pursuing legitimate organisational goals, and that all actors are in substantial agreement with the goals and the strategies for achieving them. This is a state of *goal congruence.* In many situations where the system is large and involves many principals, the prevailing situation is *goal incongruence*, in which principals disagree on goals or at least the means of achieving goals. Under such circumstances, requirements analysis inevitably becomes a *political process* that selects whose goals will be addressed and whose will not, and determines the means by which the goals will be reached. In such cases, the functional walk ($S_t \rightarrow P_t \rightarrow S_{t+1}$) is transformed into a finessed political dance. In conditions of goal incongruence it can be very difficult to arrive at stable heterogeneous solutions. Some actors will resist the proposed emerging order and use everything under their control to destabilise the emerging heterogeneous system. Implementation success requires political as well as functional stability.

3.1. Negotiation and Political Ecology in the Literature

When there is disagreement that cannot be easily settled and autocratic solutions ruled out, political issues are usually resolved by negotiation. Negotiation has been recognised and addressed in a variety of organisational and strategy design techniques, including Mason and Mitroff's SAST (strategic assumption surfacing and testing) [33,37,38], and Ackoff's S3 (social systems sciences) [39,40]. The role of two-party negotiation and reconciliation of different interests in proposed system changes was recognised in the early work on socio-technical design literature [41,42], but they focused mainly on relatively circumscribed areas of change until soft systems methodology (SSM) emerged [43,44]. SSM focuses on differences in stakeholder goals and values and suggests an adaptive means to analyse and debate such differences through the use of abstract, system-based models. Nevertheless, SSM does not provide a clear strategy to identify, analyse and model the underlying political dynamics that generate such differences.

The concept of negotiation and conflict resolution in software engineering is implicit in Boehm's theory W [14,15]. The negotiation cycle [16] and the unified software development process [45] recognise the need for negotiation and set conditions for successful negotiation outcomes under specific conditions, but they do not analyse dynamic interactions among stakeholders. Truex et al. [25] provide an important step towards better understanding the dynamic processes in the functional ecology with semi-ambiguous requirements in the notion of continuous requirements analysis for systems in emergent organisations. These developments touch upon aspects of the political and functional ecologies in software development, and foreshadow the explication of those ecologies pursued in this paper.

Negotiation and conflict resolution have not been explicitly recognised in requirements engineering, although recognition of the need for attention to this area is growing. For example, Mylopoulos et al. observed in 1999 that: 'We are only beginning to understand the importance and depth of the conflict analysis problem' [46]. The RE tradition assumes or specifies that the analyst must rectify conflicting requirements during requirements analysis [47,48]. When negotiation is unavoidable, the requirements analyst must be the mediator who controls the negotiation process [23]. There is little discussion of whether the analyst is sufficiently empowered to carry out these duties, and if not, how the analyst should proceed to get the requisite negotiation and conflict resolution addressed by those who do have the power to act. More detailed assessment of this need is beginning to appear in the RE literature [20,49], but only a few of these discussions have fully engaged the problem in its complexity. Lamsweerde et al. suggest a set of formal

[7]Consider the following example: 'You realized that you were increasingly operating at the edge of available technology. We were constantly moving out to a point where not even the major suppliers could guarantee that they were supplying would work' (Member of the Taurus Monitoring group ([7], p. 138).

techniques to resolve stakeholders' goal conflicts [50,51]. Robinson and Pawlowski examine how modelling of goals can help deal with goal inconsistency [52]. Karlsson and Ryan outline a cost-value framework for carrying out requirements analysis negotiation [53]. Mylopoulos et al. examine the use of a goal-oriented model to reveal stakeholders' different interests and concerns [46,54]. These provide a good start in revealing conflicting goals, but they do not deal with the political ecology that underlies and generates conflict. They proceed instead to the need to further formalise requirements specifications. We suggest that this strategy cannot succeed in the end because no amount of formalisation will rectify unsettled goal conflicts. One must look at the structure underneath to understand the source of the conflicts.

3.2. Political Dynamics in Requirements Analysis

Determining an adequate set of functional requirements forms an iterative walk between problem and solution spaces, when both spaces are continuously shifting [25,55]. The establishment of a workable political ecology that permits closure on the issues of concern to principals and the stabilisation of the resulting socio-technical network requires iterations between the *political* and *functional* ecologies. This is seldom recognised in conventional requirements analysis, with some unfortunate consequences as evidenced by the Taurus case.

Taurus experienced ongoing flip-flop of the technical and the functional that catapulted requirements size and scope dramatically. The political interests of different parties could not be reconciled within a chosen requirements set, and the surrounding political battles continued to change the requirement set. Therefore, political and functional ecologies never stabilised during the course of the project.

It is convenient to assume that the recognition of an initial political ecology implied by what is called a 'stakeholder analysis' determines conclusively who the key actors with standing are, what they want, and how to meet their expectations [55,56]. In fact, this is usually not the case. In addition, the initial political ecology can change as goals, problems and solutions are modified and new ones are discovered as the process proceeds [57,58]. This happens in part because the political ecology is extremely difficult to stabilise when the functional ecology is destabilised, which it is during system planning. The reverse is true as well: destabilisation in the political ecology will destabilise the functional ecology. Political positions shift during functional iterations between problem spaces and solution spaces as the principals become more cognisant about the implications of any particular construction of requirements, and make more sense of the situation [59]. Likewise, the shifts in political positions result in requests for changes in articulations of problem spaces and solution spaces. When problem blossoming occurs, it is common for principals to rethink their commitments to the system and change them, if need be. Sometimes, agreements that seemed to be secure at the start begin to erode, leading to the demise of the whole system [60].

Dynamics between the functional and political in Taurus were at the root of the escalation in requirements, which led to delays in project delivery. This could only be stopped when the power bases were reorganised and powerful actors saw an opportunity to intervene. The CEO of the Stock Exchange garnered enough power to intervene, first by cutting Taurus's budget, thereby increasing its internal difficulties. He subsequently obtained outside expert evaluations of the credibility of the original requirements to create a mapping between the current solution space and the desired future solution space, and the status of the effort to transform the current space into the future solution. These assessments supported his position to resist the change, but required that he build a new 'reality' of the problems that the Stock Exchange faced and downplay the validity of the original interpretation of the critical nature of the automatic settlement [7].

Constructing requirements among a diverse set of principals requires an agreement on key functional issues, *and* a process of social contracting and solicitation of commitment.[8] Social contracts must be stabilised and enforced over time in order to move all the elements from the current solution space to the future solution space. Current RE theory addresses this through the development of *specifications*, which constitute a kind of contract between the system developers and stakeholders [36]. In principle, this is a sound strategy; a contract should align the understandings among all the stakeholders regarding what is to be done by whom, with clear delineation of recourse among the stakeholders when expectations are not met. Unfortunately, in practice, specifications are often so badly flawed due to poor input from the requirements process that they cannot serve as effective contracts.[9] In the worst cases, each stakeholder interprets ambiguities or confusion in the specification in its own favour, while the collective interpretation is entirely in conflict, thus encouraging the

[8]The key issues being embodied by a requirement set that everybody agrees upon and the agreement to the problems and the goals implied by them.

[9]In Taurus, this was in fact totally outsourced and the technical capability and skill of the builders were never questioned.

project to proceed until the resulting system manifestly fails to meet some stakeholders' expectations. This helps explain why some projects go on for so long in a state of emerging disaster without any effort from the stakeholders to intervene and stop the process.

During the contracting mode, principals, including the requirements engineer, establish themselves as capable subjects for political action, and at the same time develop a shared understanding of whose interests are going to be served by the system. They also determine the *quid pro quos* and other negotiated deals that are usually necessary to obtain an agreement among those who must agree for a change to proceed. Any alteration in the principals' understanding will destabilise the earlier contract.

Contemporary RE processes do not necessarily decrease conflicts and increase principals' understanding of each other's positions and the nature of the proposed system [55]. Sometimes the political ecology becomes so complicated and unstable that no progress is possible. Few analysts are equipped to deal with such situations. In the absence of effective intervention, the process devolves to the point where no stakeholder seeks a common solution, and all or most manoeuvre for positions in which they can act as principals and dictate which anomalies are to be seen as problems and acted on accordingly. This problem is particularly dangerous to project success when stakeholders are seeking to gain control of the project, while appearing to be engaged in a cooperative effort to find common solutions.

The political process in Taurus was complicated in that everybody acted as though they had agreed on the principles of Taurus because it was so important, but in fact they had not done so. No wonder that some participants later described the process of managing Taurus requirements as 'the Mad Hatter's tea party' [7].

3.3. Categories of Politics in Requirements

Politics relevant to RE can be divided into concerns about the functional intent of the resulting system (*functional politics*) and concerns about the flow of resources going into the system (*resource politics*). Functional politics arises from the initial set Ro, the requirements that contextualise the objectives of the principals. Functional politics concerns those whose interests will be served by the implementation of these requirements: (1) who is entitled to govern and (2) what anomalies deserve attention [12,34,61–63]. Functional politics will determine whether the governing coalition of principals will have its desires met, and can be seen at the heart of most large system development efforts. Ineffective handling of functional politics is frequently responsible for system project failures.[10]

Functional politics in Taurus revolved around attempts to develop a system that would benefit everybody, when that was impossible. Nobody wanted to lose the opportunity to benefit, even though not everyone could. Drummond correctly identifies this as a form of Hardin's 'tragedy of the commons' [64]. A more careful consideration of the political ecology of the requirements might have revealed such problems in advance, reducing the possibility of downstream failure.

*Resource politic*s derives from the fact that organisations seldom have sufficient resources to meet all the desires of their constituents, and choices must therefore be made among which desires will be met [13,32,65]. A pure form of this is a zero-sum game, in which resources spent on one thing must be taken from another. Still, this issue appears any time opportunities go begging due to resource constraints. Resource politics should appear in budgetary processes surrounding or preceding RE, wherein organisational subunits compete for resources necessary to carry out their mandates. If the organisation has sound processes for prioritising requests, resource politics can be handled directly and effectively. Unfortunately, it is not uncommon for resource politics to be hidden as proponents of a new system deliberately downplay or 'low-ball' the expected costs of the system in order to improve the chances that the system will be funded [13]. This increases the likelihood of an expectation failure when budget overruns occur. It also makes the system development effort vulnerable to escalation and opportunistic attacks once the overruns begin. The worst case of such failures arises when the political backlash becomes so strong that everyone near the project (even the proponents) disowns it and no effort is made to capture the organisational learning benefits from the failure [66].

The Taurus governance structure failed to manage resource politics. The governance body did not have responsibility over the cost of the system, and Taurus did not compete in a direct and obvious way with the principals' other opportunities. Principals could thus keep their options on the benefits of Taurus at no cost, reducing their incentive to insist on careful management

[10]One of the largest system efforts in history, the multi-billion dollar World Wide Military Command and Control System (WWMCCS) of the 1970s and early 1980s, arguably failed due to the reluctance of the US armed services to subordinate their command authority to the system. A similar claim can be made about the ongoing difficulties of the effort to replace the US national air traffic control systems, now far over budget and past schedule. This is due to disagreements between key actors concerning the functional responsibilities and profiles of different actors like the Federal Aviation Administration, the main hubs and the major air carriers.

of the project's costs relative to progress. The resources needed to complete the system were never questioned within the Stock Exchange until the bitter end.

Political ecology problems seldom show up as clear disagreements regarding control or controversies about resource allocation because it is too difficult from the level of top decision-makers to get a clear view of problems with control or resources. Political ecology problems show up in endless rounds of systems change orders that are sometimes described as *requirements creep* (cf. Baier and March [67]). It is common for problems of political ecology to appear on the surface as problems in the functional ecology, triggering repeated efforts to reconcile the problem and solution spaces. This cannot solve the problems, and almost always adds delay and cost to the project. The functional ecology is ultimately dependent upon the political ecology, and the latter must be fixed and stable before the former can be. Ongoing problems with the political ecology produce functional outcomes that cause disaffected principals to restate their requirements or resurface resource politics concerns.

Functional politics and resource politics are linked, of course, but they cannot be addressed by the same strategies for management. The requirements engineer must address functional politics by considering the ways in which the proposed system will affect power relationships between the different principals who have the standing to affect development and implementation processes. Effective handling of functional politics is usually achieved by having the principals confront and compromise upon the truly difficult trade-offs in the project leading to a Pareto-optimal solution in which no principal's interests are damaged when one principal's interests are served.[11] In some cases, an essential principal cannot be convinced to compromise in ways that enable the project to move forward. In such situations, either the project must be turned over to the uncompromising principal at the expense of other principals, or the objecting principal must be rendered powerless to affect the process [13]. If this cannot be accomplished, the project is almost sure to fail and, in the organisation's broader interests, should be abandoned [61].

The requirements engineer has a different challenge in handling resource politics. The requirements engineer's special expertise is the relationship between the requirements and the characteristics of the system. This is not really of interest to those opposed to spending the resources the system will require.[12] The vital issue is whether the organisation should invest the required resources in the project. This problem usually cannot be resolved without bringing in an authority at a higher organisational level who has the power to determine who wins and who loses in the struggle over resources – a person Keen calls a 'fixer' [13]. The requirements engineer's role becomes critical in the process when a fixer attempts to find an intermediate solution rather than to simply killing the project or approving it at full funding. There can be a great temptation to assume that costs can be cut while still retaining functionality. The requirements engineer must explain the close linkage between specific costs and specific functional capabilities in order to ensure that cost reductions match explicit modifications and deletions of functionality from the system design. Although this is an undesirable outcome for the system's proponents because they do not get the full system they desired, it is a far better outcome than proceeding with the false belief that the system can still be built at the original expectation of functionality without the resources required. At least it makes clear in advance that certain requirements will not be met, and therefore no resources will be devoted to them.

3.4. Decision Models for Handling Political Ecologies

There are many possible articulations of decision models, but three main alternatives dominate political theory: autocracy, pluralism and two-party contests. Autocracy is simple: an individual or close-knit group decides the political question and all others comply. This model works for decisions of modest consequence or when the choices are obvious for all concerned, but it is ineffective for decisions that are important and complex. System designers sometimes assume autocracy in for-profit firms with executive rule, but real autocracy is relatively rare. Effective executives usually consult with relevant parties before making such decisions to find workable compromises that increase chances for success, and they leave their options open to change course when they see things working out differently than planned. For large and complex system projects it is unwise to assume the operation of autocratic decision-making.

Pluralism accommodates the diverse interests of various principals with standing in the decision process. Ideally, interests are considered on their individual merits, and efforts are made to accommodate all reasonable interests in the solution. All principals have standing to have their interests considered and participate in the decision, but pluralism does not assume that

[11]This indicates the possibility of applying the Nash equilibrium to requirements analysis [68].

[12]A common place for resource politics is military logistics.

all interests carry equal weight. Those with more to win or lose might be given more authority than those only peripherally involved.

Pluralism represents an ideal of participatory strategies for system development. Involving users as well as designers in the process will make the system more successful by ensuring that all relevant interests are served and by incorporating detailed domain knowledge of users into the system's design [69–71]. In practice, however, genuine pluralism is difficult to maintain in system development due to resource politics. When there are not enough resources to satisfy all user expectations, principals must decide on which interests will be served or eliminated from the project. It is also a fact that system developers tend to pay attention to user participants at the beginning of the analysis process, and to marginalise them as the development process unfolds [58]. Technical demands come to preoccupy the analysts as work proceeds, and it becomes more difficult to accommodate the shifting demands and 'requirements creep' of the pluralistic participants. In addition, the objectives and constraints among the pluralistic interests conflict with one another and must be resolved authoritatively or by negotiation. When there is no effective authority and negotiation breaks down, the conflicts become hidden and the problems surface later when the costs of attending them are much higher. Thus, sustained pluralism does not work very well for managing the political ecology of requirements. *In the case of Taurus, the major decision-making body, SISCOT, had a representation from all major constituencies who were involved in trading on the Stock Exchange. As long as everyone was comfortable in the false security that the project was proceeding according to plan, there was no problem with this pluralistic governance structure. However, when the evidence of trouble began to mount, the pluralistic structure quickly broke down and governance defaulted to the Stock Exchange, especially its CEO (Peter Rawlins).*

There is increasing evidence to show that pluralistic political ecologies often devolve into two-party contests in which diverse interests come together behind either a proponent or opponent position on a particular set of requirements representing and contextualising the current possible future solution (S_{t+1}). A growing literature suggests that pluralistic coalitions tend to be less stable in handling controversial political issues over time than two-party structures.[13] This theory helps to explain the 'winner-take-all' outcome of many large system projects, and suggests the application of dialectic to the process of negotiation in requirements analysis [78].

4. Modelling the Political Ecology

4.1. The Political Ecology of Requirements

The political ecology of requirements can be modelled just as the functional ecology can be modelled. We begin our discussion of modelling with the assumption of a two-party contest because it is more tractable to model than a pluralistic decision process, and because many if not most system development efforts devolve to two-party contests even if they start out as pluralistic. The basic conceptual structure of the analysis is a dialectic among participants in a socio-technical network [78,79]. The system proponents offer the *thesis* under which their desires for the new systems will be met. The thesis is challenged by an *antithesis* argued by the system's opponents. Presuming that neither side wins its points unilaterally, the resulting agreement will be a compromise or a *synthesis* of the thesis and antithesis.[14]

4.1.1. Proponent Coalition

The political scientist Robert Dahl notes: 'Influence is a relation among actors such that wants, desires, preferences, or intentions of one or more actors affect the actions, or predispositions to act, of one or more other actors.' [81]. Principals in system development efforts seek influence over others in system decisions that deal with functional and political ecologies. Each principal's influence defines his or her political strength. Requirements engineers are principals in the process whose primary goal is presumed to be the establishment of a stable and constructive political coalition that will enhance the chances of project success [82]. This is the *proponent coalition* (PC), which seeks to have its problem space legitimised and its requirements specification adopted by the organisation. The PC seeks also to overcome resistance in functional and resource politics through political means. The PC usually emerges from within the organisation adopting the new system, but it can also be made up primarily of outside contractors [55], or other external constituencies (*as was true in the case of Taurus*) [7].

The problem space derived by the PC is the proponent problem space, denoted P_P, which is a subspace of P_t that is defined, owned and controlled by the principals in the

[13]This work has been done primarily in electoral politics, but the principles derived from game theoretic analysis suggest the underlying theory of Duverger's law is extendable to organisational politics [72–77].

[14]This is a contemporary adaptation of Hegel's phenomenology, but does not presume the determinism implied in Hegel's theory of history and progress [80].

PC. The importance of the P_P cannot be overstated at this point in the discussion, because it is the hinge point on which the entire process of requirements engineering must turn. As noted earlier, it is often difficult to agree upon a complete set of requirements due to bounded rationality, non-linear functional complexity and time pressures [55]. In the immediately preceding discussion, we have added to the difficulties of RE the challenges from functional and resource politics. Taken together, these factors make it very difficult to agree on the contents of P_P in any instance where the system under design will be large and complex. In some cases, it is so difficult to reach agreement that no agreement is reached, and the project is either abandoned or it is continued in the naïve hope that the problems generated by this ambiguity will somehow sort themselves out. It might seem surprising that intelligent people would pursue the latter course of action, but it happens quite easily through the simple mechanism of defining P_P at such an abstract level that all can agree on it [67]. Most members of PC are unaware of the true consequences of the ephemeral P_P until the expectation failures around the functional ecology begin and political conflicts among the principals erupt. Another key problem with an ambiguous P_P is the devolution of undue influence to system designers, who are the only ones operating at a sufficiently concrete level to decide among the ambiguities and implement the system's features. In such a case, important stakeholders lose control of the project and emerge later as opponents to continuing the project because their needs are not being met.

The proponent coalition in Taurus was established with powerful actors including the Stock Exchange, the Bank of England, and committees in the securities and finance industry. The strength of this coalition became a weakness in that this powerful group of stakeholders came to see itself as unstoppable. It was felt that the strength of the coalition alone would guarantee success. The coalition stood on quicksand, however, in that it failed from the start to agree on the contents of PP and the requisite mappings to a future solution space.

It is far better to acknowledge that no agreement has been reached among the principals than to pretend that it has and proceed. Acknowledgement of failure to agree enables the problem space (P_t) and the existing solution space (S_t) to be redefined with clarity, thereby redefining the proposed solution, S_{t+1}. We call this process *reprobleming*. It can be costly, because it often incurs at least one new round of problem blossoming, but it is preferable to stopping the project simply because parties cannot agree and become intransigent in their demands [13]. The PC works best when it understands the difficulty of managing the functional and political ecologies and the necessity of deriving stable, unambiguous solution and problem spaces. At least then, the project has a chance of overcoming severe challenges without disintegrating into internal pressures that can tear a project apart even before more organised opposition begins.

4.1.2. Opponent Coalition

The *opponent coalition* (OC) is a collection of people who oppose the proposed system. The problem space of the opponents' coalition is the opponent problem space, P_O, and is a subspace of P_t, which is defined, owned and controlled by the principals in the OC. The P_O is rooted in the perceived negative consequences that are exposed but not resolved by P_P. In principle, for any S_t, there are likely to be opponents. Some principals like the *status quo* and want to maintain rather than change S_t. Some will want to start from a different S_t that more directly meets their objectives. And some will prefer a different P_t. Opposition usually arises from concerns about possible loss of power, influence or welfare resulting from the success of the PC's plans. Opposition can arise in either functional politics or resource politics (see also [36]). In larger systems, opposition often arises from both although, as mentioned earlier, the issues that trigger opposition in the early stages often arise from functional politics. Opposition sometimes arises from protocol failures wherein principals with standing to influence the project are not adequately consulted, and therefore oppose the project simply because they believe their interests will be damaged by it [36].

The major opposition in Taurus emerged first from the small investors, whom one observer characterised as 'The old lady from Sheffield with 100 ICI stock', who feared loss of control and increasing cost. There was also initial opposition within the Stock Exchange itself that was amplified over time when the CEO shifted his support away from Taurus and initiated a set of measures that eventually led to the demise of Taurus.

The OC usually arises from within the adopting organisation when the system affects only that organisation. The advent of inter-organisational systems has increased the likelihood that others such as consumers, competitors and regulators will join the OC. A particularly important phenomenon is the tendency of members of the PC outside the adopting organisation, and especially contract developers, to fail to recognise the emergence of an OC or its importance to system success. It is often wrongly assumed that the members of the PC understand the nature of the emerging OC, and are dealing with it effectively. Members of a PC often discover to their surprise that the OC can mobilise

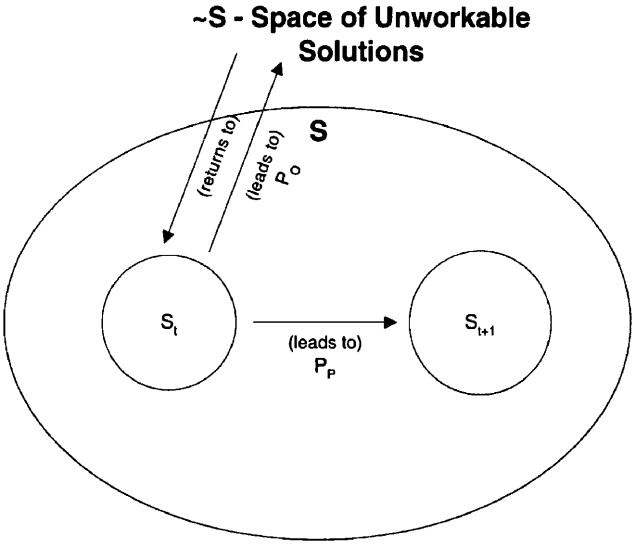

Fig. 3. Political ecology view of the solution space transformation.

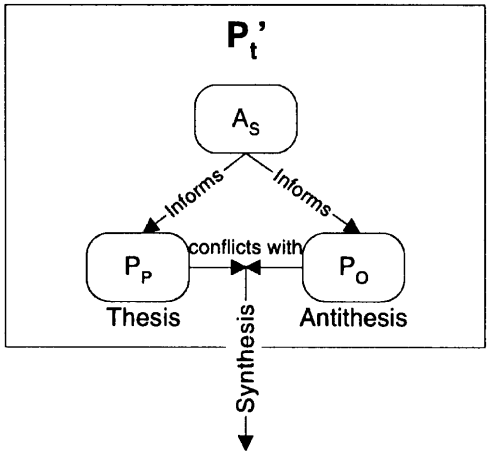

Fig. 4. Problem space synthesis (P_t').

powerful opposition based on concerns that were not even envisioned by the PC, such as fear of invasion of customer privacy or risks due to weak system security.

When P_t is split into P_P and P_O, the original solution space transformation changes as the implied system design is transformed from S_t to S_{t+1} (see Fig. 3). The negotiation model now incorporates the possibility of failure due to political pressures. That failure can occur functionally if a stable S_{t+1} (and S_t) cannot be constructed, P_t or P_P cannot be agreed to, or P_t is not fully addressed by S_{t+1}.

4.1.3. The Support Network

It is common in successful projects that a *support network* (SN) is established to provide a network of commitments, supporting technologies, processes and people vital to the system's success. A project SN is usually created by the PC. A project SN is built on top of the general organisational support networks that exist in the organisations either building the project or supporting the delivered system. A general SN has organisational inertia when it is grounded in well-established norms of work employed by S_t [83]. Hence, a general SN is usually reliable in its applied capabilities, but is resistant to changes that a proposed system would impose on itself. A common example of a support network is the formal information systems department in a large organisation. The SN employs instrumentalities to assist in the definition, selection and/or creation of the problem space or solution space under consideration. These include *support technologies*, *support processes* and *support personnel*. Support technologies and support policies cannot align politically with the PC or OC, but support personnel can. It is often assumed that support personnel have no standing to create or change P_t, but in fact they often do, and their influence can grow over time as they become more essential to completion of the project. Support personnel can inform the PC and OC of likely impacts on their interests from the system, and both the PC and the OC can use support personnel to inform their actions. These impacts create the support network anomaly space, A_S. Any anomalies in the A_S can be promoted to problem by principals in either the PC or OC, and are thus placed in the P_P and P_O, respectively. Altogether, anomalies in A_S can be used to inform, augment or support the positions of principals who control the problem spaces (e.g. P_P and P_O). Figuratively, A_S informs P_P and P_O as shown in Fig. 4.

4.1.4. Problem Space Synthesis of PC and OC

In some cases the OC does not arise, or it emerges and is not effective in altering the course of the project. This is a victory for the PC. A protracted struggle between a PC and an OC can result in victory for the OC in the form of a complete halt to the development. Complete victory for either the PC or OC is not that common, however, and a struggle between them often creates a new common problem space (P_t'). As shown in Fig. 4, P_t' embodies the conflict between PP and PO as informed by AS. P_t' thus represents a synthesis of these conflicting problem spaces. A viable P_t' is found when the mechanisms of problemisation and problem blossoming identify a subset of a solution space that is technically and socially stable. This adjusted current solution space is deemed S_t'. Still another possible outcome occurs when no stable solution reconciles the competing interests into $<P_t', S_t'>$ and a mapping from S_t into S_t'. (*This is what happened in Taurus*.) Updating Fig. 3 to include the detail from Fig. 2

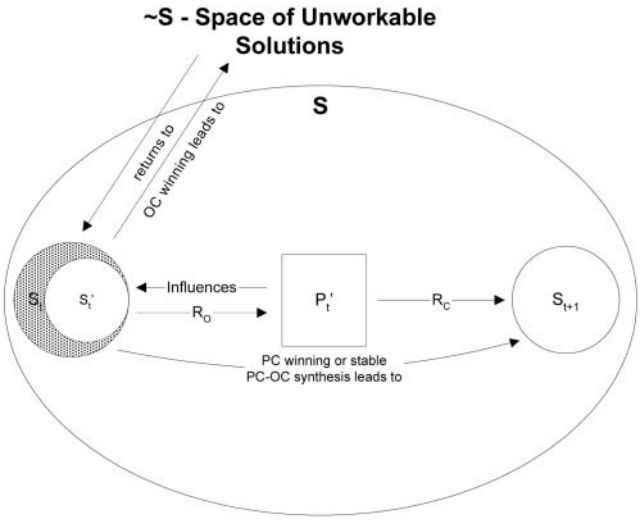

Fig. 5. Synthesis solution transformation space.

(with the requirements being represented by RO and RC), and applying S_t' and P_t', produces the synthesis solution transformation space shown in Fig. 5.

A synthesis (P_t' and S_t') is achieved through one or more political techniques. These include control of the agenda, display of charisma, establishment and execution of *quid pro quo*, control of the decision criteria, rationalisation, legitimisation of control, incurring of obligation, dispensing of rewards, use of outside experts and co-opting of the opposition members [84].[15] These are drawn from Dahl's [81] political theory of influence and sanction of non-compliance, based on rational persuasion, manipulative persuasion, inducement, power, coercion, physical force or domination [82]. For the purposes of requirements analysis they can be considered means of *negotiation*. Successful negotiation can produce win–win outcomes as well as win–lose outcomes [85]. Unsuccessful negotiation results in a lose–lose outcome, i.e. no agreement.

4.2. The Political Ecology Model

A careful examination of the synthesis model (Fig. 5) in relation to negotiation reveals some of the missing intricacies in the current RE literature. First, during synthesis, the coalitions PC and OC, and thus the problem spaces P_P and P_O, can change with concomitant changes in goals and solutions. In a manner similar to problem blossoming, a change in P_t' (e.g. a change in P_P or P_O) can trigger non-linear changes in the current solution space S_t (and any resulting, stable S_t'). Moreover, any change in P_t' can change S_{t+1}. This non-linearity can produce dramatic shifts in the relative influence of the PC and the OC due to addition, change or removal of problems and the requirements that must attend them. This can be seen in cases when a new or changed system affects a part of S_t that has consequences foreseeable by the OC. Similarly, the more a proposed system will affect the SN, the greater the burden on the PC to persuade those in the SN that the changes will be beneficial to them. A failure on the part of the PC to attend to the interests of the SN can result in the SN moving to the side of the OC.[16] Altogether, there are four possible outcomes to the political negotiation process:

- *Outcome 1.* The PC can overwhelm the OC by the direct exercise of power, producing S_{t+1} and a heterogeneous set of requirements that are slanted towards the objectives and constraints of the PC. These requirements may be underdeveloped functionally, especially if problem blossoming was curtailed prematurely, or the functional concerns of the OC were not appropriately accounted for. 'Heavy-handed' politics, or an inadequate requirements specification, can destabilise this political ecology and force reprobleming or failure. However, if requirements specification is done well and/or the PC has enough power to enforce political stability, the design of S_{t+1} will continue based on the PC's specification.
- *Outcome 2.* The PC and OC can successfully negotiate a mutually acceptable agreement, thus re-establishing functional stability and a heterogeneous set of requirements for S_{t+1} as redefined. This can produce the most stable P_t' and S_{t+1}, but if not done with sufficient attention to detail it can result in solving some of the wrong problems via 'necessity creep' [33]. It is difficult to reach an optimal functional ecology due to bounded rationality and other factors discussed earlier, so it is often the case that a satisficing solution will be adopted. Progress can still be threatened by political instability from factors like unrestrained problem blossoming, requiring the PC to either reproblem or face failure.
- *Outcome 3.* The OC can overwhelm the PC using political techniques that impose escalating costs and strangle development. This outcome can also occur when negotiation between PC and OC fails, or when the PC fails to produce a viable, stable set of requirements that can survive the opposition. Reprobleming is the only way for the PC to save the project

[15]This is a partial list from *Organizational power politics*, p. 41, Table 4.1, 'Twenty-two power tactics' [84].

[16]The desire not to change current work practices by the SN is a form of organisational inertia. In general, organisational inertia tends to accrue advantage to the OC.

from failure. Celebrated failure in one project can change the functional and resource politics in future projects.

- *Outcome 4.* The PC chooses to reproblem, changing P_P enough to obtain a more favourable P_t'. Reprobleming has an uncertain outcome and incurs at least one new round of problem blossoming with associated costs and risks. Together, reprobleming and problem blossoming can result in changes in the PC, which are frequently accompanied by changes in the OC. In some cases the project reverts to pluralistic competition, reopening the search for alternative S_{t+1}'s. For the project to continue, a stable, viable P_t' and S_{t+1} must be found. If this cannot be done, the project can be split up in to multiple projects (if supportable by the resulting political and functional ecologies), with each new project forming its own solution–problem space alignment (i.e. $S_t^X \rightarrow P_t^{X'} \rightarrow S_{t+1}^X$), where X represents a new project. Each project space would likely overlap significantly with other project spaces with their functional and political ecologies. This new set of projects would have to be accounted for and managed, just as the original project would. For this reason, it is often less costly to stay with the original project as reproblemed, even though that can be difficult. If no new, stable and viable spaces can be constructed, then the project fails.

The foregoing observations come together in the political requirements engineering (PRE) process model, explicated in Fig. 6. Options 1 and 2 allow for requirements specification ($R_C + R_O$). If the political ecology stabilises around this specification, the design of S_{t+1} can continue in earnest. Outcomes 3 and 4 do not allow for the construction of a viable requirements specification, and

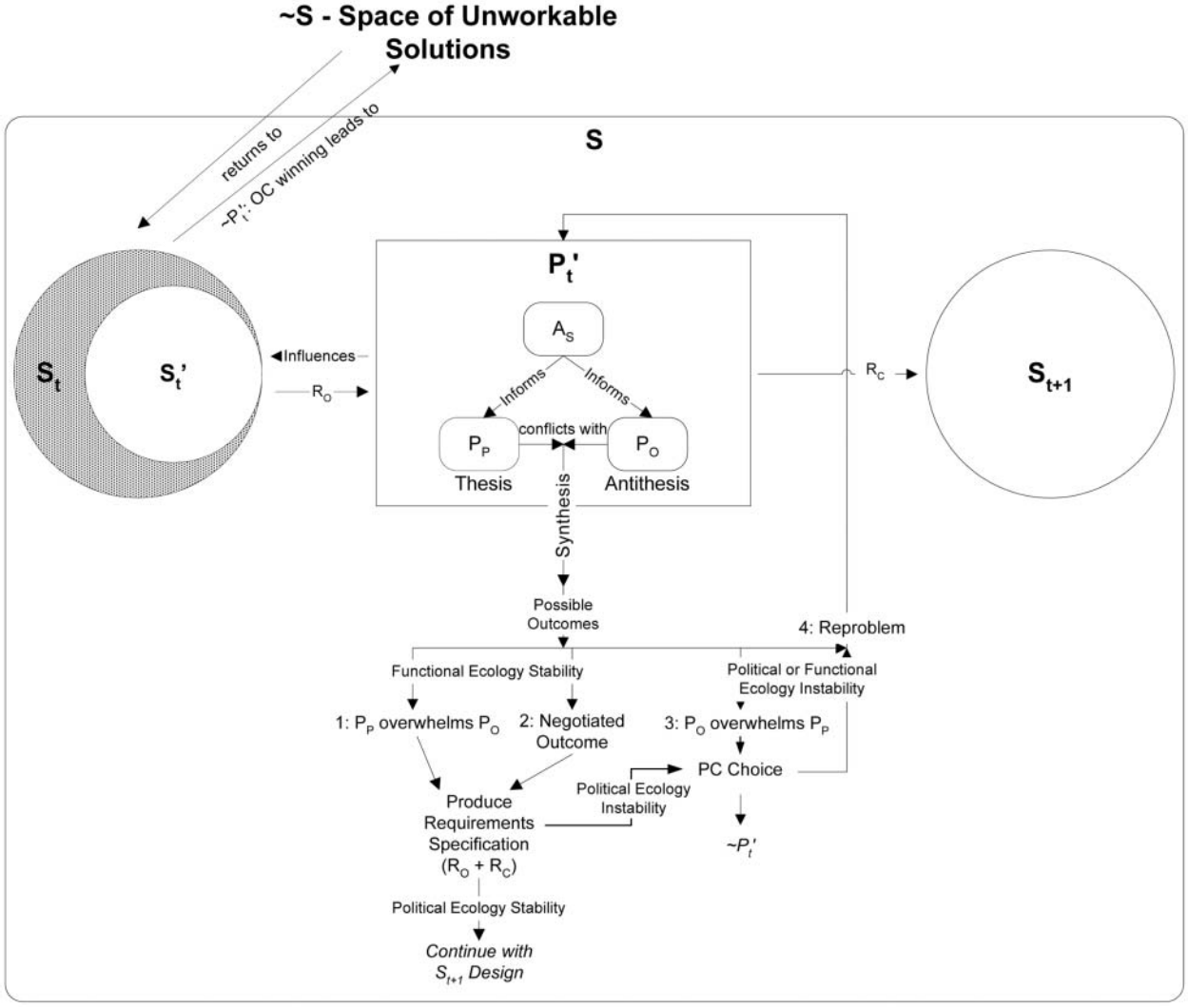

Fig. 6. Political requirements engineering (PRE) process model: updated solution transformation.

demand yet more changes to the problem and solution spaces. Any workable synthesis that results in the construction of the system can be considered a victory for the PC, while a 'win–win' synthesis [14] can be a victory for both the PC and the OC (this would be a version of outcome 2). A project is not doomed when the OC wins substantial concessions from the PC (outcome 3), but the system can be badly compromised, and the project can ultimately fail due to instability of the resulting functional and political ecologies. *This is basically what happened with Taurus. The weakness of the requirements process was its multiple, vicarious agreements that led to outcome 4. The PC chose to reproblem over a long period of time, producing over 15 different formulations of Taurus. The political instability became too great, and in the end defaulted to the fatal outcome 3. The project went through many loops between outcome 2 and outcome 4. Although at times there seemed to be a stable functional ecology, in fact, there never was.*

5. Implications for Requirements Engineering

There is a complex web of requirements for each stable solution–problem space,[17] and each requirement is owned by at least one principal. Principals own requirements as part of a proponent coalition (PC) arising from political alliances that allow for stabilisation of the solution–problem space to overcome resistance. Alliances construct and pursue a complex web of requirements characterised by political interests. The resulting political–functional constellation is a consequence of the need to reach a politically and technically stable solution–problem space.[18]

5.1. The Rise of the *Political* Requirements Engineer

Any change in the solution or problem spaces will alter a political–functional constellation and render it unstable. The construction of stable solution–problem spaces and resulting political–functional constellations is difficult. Instability works to the advantage the OC by facilitating charges of mismanagement. Protracted instability can lead to an escalation in cycling through the intra-problem space, P_t', and repeated reprobleming in search for a more viable S_t' and P_t'. Cycling can bring high costs in time and money. If the costs become too high, then either there is project failure and the support is dropped [60], or the process is 'prematurely' stopped. The problem with 'premature stopping' is the likelihood of creating dysfunctional political–functional constellations and their attendant requirement mappings, and simply declaring them to be 'good' when they are not. Even in cases where a politically stable and functionally workable arrangement is eventually achieved, the costs can be so high that future development efforts are under an enduring cloud of suspicion.

Requirements engineers will probably never have the political power to manage the negotiations and conflict resolution processes that emerge in the solution–problem space (S_t'–P_t') [86]. However, this does not mean that requirements engineers should stay out of the political process. On the contrary, a great many project failures can be ascribed to just this mistake. In any case, where there is an artificial split between the 'policy' and the 'technical' that places organisational leadership and users on one side and the technical specialists on the other [20,47], the chances of project failure loom. This is because unsolved problems in the political ecology almost always turn up in the functional ecology as technical due to problem blossoming. Indeed, policy-makers will tend to see all problems as political, while engineers will tend to see the same problems as technical. Those on the policy side cannot see the technical implications of unresolved political issues, and those on the technical side are unaware that the political ecology is creating serious problems that will show up in the functional ecology. This creates a skewed and unstable S_t'–P_t' that is reflected in a constantly changing requirements set. *This was one of the major problems with Taurus. None of the negotiations in SISCOT involved technical experts to clarify the functional implications of the political solutions for the resulting new solution space.*

It is widely recognised that the requirements engineer is usually the best party to represent the functional ecology during the requirements phase [26]. The detailed knowledge of systems required for this purpose suggests that it will not be practical to assume that non-technical senior principals will be able to do this job well. The question, then, is who should handle the political ecology. The tradition has been to assign that responsibility to the political principals with the expectation that they can sort out the issues and provide a clear and complete statement of requirements to the requirements

[17]The complex web of requirements is discussed in detail in Bergman et al. [26].

[18]This can been viewed as two networks (graphs) connected by principal–requirement pairs (s_m–R_O or s_m–R_C). It is also possible for a set of principals (under an alliance) to own a requirement or one principal to own many requirements. Thus, it is possible for a set of principals to own many requirements. Also, a principal can enter into many alliances, even with the same other principals. To manage alliances owning a requirement, an arc connecting a principal–requirement pair should be annotated with its associated principal–principal alliance arc, if any. As such, each alliance arc needs to be identified by an 'named' alliance.

engineer. This model is deeply flawed, as we have discussed, and cannot be mended by increasing effort to formalise the processes of requirements engineering. In fact, formalising may well make this situation worse by further separating the political and functional sides, especially if it is done too soon in the design process. The more formal the specifications, the less likely those without technical expertise can understand them and their implications. This can skew the negotiation process. Political principals will likely make poorer decisions based on that which they do not understand. In addition, since the formalisation process is quite laborious and is the result of a protracted analysis [22,47], the more the specification has been formalised, the more resistant it will become to change. Formalisation should occur only after a stable S_t'–P_t' has been obtained.

The political ecology must be embraced and pulled into the process. Given that the political principals cannot be expected to come over to the functional side (except for whatever technical expertise they may already have) and bring their political knowledge with them, the only remedy is to have the requirements engineer embrace the political ecology and merge the functional and the political. To accomplish this, requirements engineers must be trained to manage inherently political processes that are necessary to construct stable solution space–problem space pairings and requirements mappings. They also need to learn how to explain the functional and political risks involved in different negotiated solutions. This is a process of political and technical sense-making [59], and is best supported by the view of the requirements engineer as a practitioner of heterogeneous engineering.

5.2. Requirements Engineering as Heterogeneous Engineering

Requirements engineering in large and complex systems is inherently political, requiring the establishment of stable networks of social and technical components in the midst of conflict over the resources and goals. This process has been called *heterogeneous engineering*, emphasising the need to pull together a wide array of sometimes conflicting problems and reconciling them towards a workable solution [11]. Heterogeneous engineering shows up in many contexts, including the process of setting technical standards in software platforms or telecommunications [87]. System design requires allocation of scare resources to effect a move from the current solution space S_t to a new solution space S_t'. This requires the application of political power and influence in order to overcome opposition [81,88,89]. Political techniques are necessary to inform, direct and improve RE. The analysis provided in this paper explicates the importance of finding a workable solution–problem space pair (S_t'–P_t'), and explains how this process is unavoidably both politically and technically. This view is explicitly at odds with RE approaches claiming requirements elicitation must occur *after* the business decisions about the system have already been made [20,47]. We assert that this strategy is the *cause* of most failures in large and complex system projects.

We believe that one source of opposition to explicit engagement of the political side of RE is the sense that politics is somehow in opposition to rationality. This is a misconception of the nature and role of politics. Political action embodies a vital form of rationality that is required to reach socially important decisions in conditions of incomplete information about the relationship between actions and outcomes [81]. Since it is impossible to see the future, most complicated decisions fall into the class of issues that must be decided politically, while informed with technical analyses. Political solutions are used simply because no other solutions are available, due to bounded rationality [28]. Moreover, well-constructed political processes are highly effective as a kind of heterogeneous engineering. They enable a systematic process of experimentation, in which the parameters of problems and their desired outcomes are clarified, and various proposals are weighed. The resulting political decision is a social agreement to pursue a particular strategy with the intent of examining the outcomes and, if they fall short of expectations, changing the strategy to improve the outcomes. It is not only naïve to view politics as at odds with rationality, but it is irrationally dysfunctional to proceed on proposed technical requirements without engaging the political processes required to stabilise them. It is only by engaging political issues at the *beginning* of project design that requirements engineers can succeed.

We do not suggest that the analysis in this paper resolves the problems of preparing requirements engineers for their role in political analysis. Much remains to be done in this respect. The paper makes the argument that requirements engineers must engage the politics of system design to do their jobs well, and it provides an example of the ways that the political ecology of requirements can be explicitly modelled as part of the RE process. There is a further need for sustained and deep empirical investigations of the dynamics of functional and political ecologies. Research is also needed in the nature of negotiation tactics and their efficacy for improving the management of evolving requirements in the midst of dissention within the PC or

opposition from the OC. A promising direction of work is further exploration of negotiation tactics and equilibrium models for two-party negotiations using game theoretic strategies. Effective modelling of functional and political ecologies in RE will provide a basis for improvements in RE practice and, by extension, in the development of cost-effective systems that are both large and complex.

References

1. Abdel-Hamid TK, Madnick SE. The elusive silver lining: how we fail to learn from software development failures. Sloan Manage Rev 1990; 32:39–48
2. Drummond H. The politics of risk: trials and tribulations of the Taurus project. J Inform Technol 1996;11:347–357
3. Mitev NN. More than a failure? The computerized reservation systems at French Railways. Inform Technol People 1996;9:8–19
4. Myers MD. A disaster for everyone to see: an interpretive analysis of a failed IS project. Accounting Manage Inform Technol 1994;4:185–201
5. Rosenwein M. The optimization engine that couldn't. OR/MS Today 1997;24:26–29
6. Jones C. Patterns of software system failure and success. International Thomson Computer Press, Boston, MA, 1996
7. Drummond H. Escalation in decision-making: the tragedy of Taurus. Oxford University Press, Oxford, 1996
8. Hughes T. The evolution of large technological systems. In: Bijker W, Hughes T, Pinch T (eds). The social construction of technological systems. MIT Press, Cambridge, MA, 1987, pp 51–82
9. Hughes T. Emerging themes in the history of technology. Technol Culture 1979;20:697–711
10. Hughes T. The electrification of America: the system builders. Technol Culture 1979;20:124–161
11. Law J. Technology and heterogeneous engineering: the case of Portuguese expansion. In: Bijker W, Hughes T, Pinch T (eds). The social construction of technological systems. MIT Press, Cambridge, MA, 1987, 111–134
12. Markus ML. Power, politics and MIS implementation. Commun ACM 1983;26:430– 444
13. Keen PGW. Information systems and organizational change. Commun ACM 1981;24:24–33
14. Boehm BW, Ross R. Theory-W software project management: principles and examples. IEEE Trans Software Eng 1989;15:902–916
15. Boehm B, Bose P, Horowitz E, Lee MJ. Software requirements negotiation and renegotiation aids. Presented at: ICSE 17, Seattle, WA, 1995
16. Robinson WN, Volkov V. Supporting the negotiation life cycle. Commun ACM 1998;41:95–102
17. Pfeffer J, Leong A. Resource allocations in united funds: examination of power and dependence. Social Forces 1977;55:775–790
18. Pfeffer J, Salancik G. The external control of organizations: a resource dependence perspective. Harper & Row, New York, 1978
19. Provan KG, Beyer JM, Kruytbosch C. Environmental linkages and power in resource-dependence relations between organizations. Admin Sci Q 1980;25:200–225
20. Kotonya G, Sommerville I. Requirements engineering: processes and techniques. Wiley, Chichester, 1998
21. Loucopoulos P, Karakostas V. System requirements engineering. McGraw-Hill, London, 1995
22. Davis AM. Software requirements: objects, functions, and states (rev edn). Prentice-Hall, Englewood Cliffs, NJ, 1993
23. Macaulay L. Requirements engineering. Springer, London, 1996
24. Robertson S, Robertson J. Mastering the requirements process. ACM Press, Reading, MA, 1999
25. Truex DP, Baskerville R, Klein H. Growing systems in emergent organizations. Commun ACM 1999;42:117–123
26. Bergman M, King JL, Lyytinen K. Large scale requirements analysis as heterogeneous engineering. In: Floyd C, Klischewski R (eds). Social thinking – software practice. MIT Press, Cambridge, MA, 2002
27. Cilliers P. Complexity and postmodernism: understanding complex systems. Routledge, London, 1998
28. Simon HA. Rational decision making in business organizations. Am Econ Rev 1979;69:493–513
29. Simon HA. Models of bounded rationality, vol 2. MIT Press, Cambridge, MA, 1982
30. March JG, Olsen JP. Ambiguity and choice in organizations. Universitetsforlaget, Bergen, 1976
31. Lindblom CE. Still muddling through. Publ Admin Rev 1979;39:517–526
32. March JG, Heath C. A primer on decision making: how decisions happen. Free Press, New York, 1994
33. Mason RO, Mitroff II. Challenging strategic planning assumptions. Wiley, New York, 1981
34. Markus ML, Keil M. If we build it, they will come – designing information systems that people want to use. Sloan Manage Rev 1994;35:11–25
35. Oz E. When professional standards are lax: the CONFIRM failure and its lessons. Commun ACM 1994;37:29
36. Clemons E, Thatcher M, Row M. Identifying sources of reengineering risks: a study of the behavioral factors contributing to reengineering risks. J Manage Inform Syst 1995;Fall:9–36
37. Mitroff II, Emshoff JR. On strategic assumption-making: a dialectical approach to policy and planning. Acad Manage Rev 1979;4:1
38. Mitroff II, Emshoff JR, Kilmann RH. Assumption analysis: a methodology for strategic problem solving. Manage Sci 1979;25:583
39. Ackoff RL. Creating the corporate future. Wiley, New York, 1981
40. Ackoff RL. The art and science of mess management. Interfaces 1981;11:20
41. Mumford E, Land FW, Hawgood J. The ABACON approach 1978: a participative approach to forward planning and system change. Automation Benefit Appraisal Consultants, Durham, 1978
42. Mumford E. Effective systems design and requirements analysis: the ETHICS approach. Macmillan, Basingstoke, 1995
43. Checkland P. Systems thinking, systems practice. Wiley, Chichester, 1981
44. Checkland P, Scholes J. Soft systems methodology in action. Wiley, Chichester, 1990
45. Jacobson I, Booch G, Rumbaugh J. The unified software development process. Addison-Wesley, Reading, MA, 1999
46. Mylopoulos J, Chung L, Yu E. From object-oriented to goal-oriented requirements analysis. Commun ACM 1999;42:31–37
47. Wieringa R. Requirements engineering: frameworks for understanding. Wiley, New York, 1996
48. Firesmith DG. Object-oriented requirements analysis and logical design: a software engineering approach. Wiley, New York, 1993
49. McGraw KL, Harbison K. User-centered requirements: the scenario-based engineering process. Erlbaum, Mahwah, NJ, 1997
50. Lamsweerde A v, Darimont R, Letier E. Managing conflicts in goal-driven requirements engineering. IEEE Trans Software Eng 1998;24:908–926
51. Lamsweerde A v, Letier E. Handling obstacles in goal-oriented requirements engineering. IEEE Trans Software Eng 2000;26:978–1005
52. Robinson WN, Pawlowski SD. Managing requirements inconsistency with development goal monitors. IEEE Trans Software Eng 1999;25:816–835
53. Karlsson J, Ryan K. A cost-value approach for prioritizing requirements. IEEE Software 1997;14:67–74

54. Mylopoulos J, Chung L, Liao S, Wang H, Yu E. Exploring alternatives during requirements analysis. IEEE Software 2001;18:92–96
55. Pohl K. Process-centered requirements engineering. Wiley: New York, 1996
56. Gause DC, Weinberg GM. Exploring requirements: quality before design. Dorset House, New York, 1989
57. Ulrich W. Critical heuristics of social planning: a new approach to practical philosophy. P Haupt, Bern, 1983
58. Beath CM, Orlikowski W. The contradictory structure of systems development methodologies: deconstructing the IS–user relationship in information engineering. Inform Syst Res 1994;5:350–377
59. Weick KE. Sensemaking in organizations. Sage: Thousand Oaks, CA, 1995
60. Sauer C. Why information systems fail: a case study approach. A Waller, Henley-on-Thames, 1993
61. Keil M. Pulling the plug: software project management and the problem of project escalation. MIS Q 1995;19:421–447
62. Beynon-Davis P. Information systems failure: the case of LASCAD project. Eur J Inform Syst 1995;4:171–184
63. Grudin J. Groupware and social dynamics: eight challenges for developers. Commun ACM 1994;37:92–105
64. Hardin G. The tragedy of the commons. Science 1968;162:1243–1248
65. Danziger JN, Dutton WH. Technological innovation in local government: the case of computers in the US cities and counties. Urban Information Research Group (URBIS) Public Policy Research Organization University of California, Irvine, CA, 1976
66. Lyytinen K, Robey D. Learning failure in information system development. Inform Syst J 1999;9:85–101
67. Baier V, March J. Implementation and ambiguity. Scand J Manage Studies 1986;4:197–212
68. Rao RC. A Nash equilibrium view of market shares, prices, and profits in branded markets. Institute for Research in the Behavioral Economic and Management Sciences Krannert Graduate School of Management Purdue University, West Lafayette, IN, 1981
69. Kensing F, Blomberg J. Participatory design: issues and concerns. Journal of Computer Supported Cooperative Work 1998;7:167–185, 1998
70. Raymond ES. The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary (1st edn). O'Reilly, Cambridge, MA, 1999
71. Noyes JM, Baber C. User-centered design of systems. Springer, London, 1999
72. Riker W. The two-party system and Duverger's law. Am Political Sci Rev 1982;76:753–766, 1982
73. Palfrey T. A mathematical proof of Duverger's law. In: Ordeshook PC (ed). Models of strategic choice in politics. University of Michigan Press, Ann Arbor, MI, 1989, pp 69–91
74. Feddersen T, Sened I, Wright SG. Rational voting and candidate entry under plurality rule. Am J Political Sci 1990;34:1005–1016
75. Shepsle KA. Models of multiparty electoral competition. Harwood, Chur, NY, 1991
76. Osborne M. Candidate positioning and entry in a political competition. Games Econ Behav 1993;5:133–151
77. Cox G. Making votes count: strategic coordination in the world's electoral systems. Cambridge University Press, Cambridge, MA, 1997
78. Jackson MC. Systems methodology for the management sciences. Plenum Press, New York, 1991
79. Churchman CW. Operation research as a profession. Manage Sci 1970;17:837
80. Berthold-Bond D. Hegel's grand synthesis: a study of being, thought, and history. State University of New York Press, Albany, NY, 1989
81. Dahl RA. Modern political analysis (5th edn). Prentice-Hall, Englewood Cliffs, NJ, 1991
82. Bacharach SB, Lawler EJ. Power and politics in organizations (1st edn). Jossey-Bass, San Francisco, 1980
83. Orlikowski WJ, Robey D. Information technology and the structuring of organizations. Inform Syst Res 1991;2:143–169
84. Fairholm GW. Organizational power politics: tactics in organizational leadership. Praeger, Westport, CT, 1993
85. Neale MA, Bazerman MH. Cognition and rationality in negotiation. Free Press, New York, 1991
86. Macaulay LA. Cooperative requirements capture: control room 2000. In: Jirotka M, Goguen J (eds). Requirements engineering: social and technical issues. Academic Press, Boston, MA, 1994, pp. 67–87.
87. Fomin V, Lyytinen K. How to distribute the cake before cutting it into pieces: Alice in Wonderland or radio engineers' gang in the Nordic countries. In: Jakobs K (ed). Information technology standards and standardization: a global perspective. Idea Group, Hershey, PA, 1999, pp 198–222
88. Orlikowski WJ. Learning from notes: organizational issues in groupware implementation. Presented at: CSCW'92, Toronto, Canada, 1992
89. Bowers J, Button G, Sharrock W. Workflow form within and without: technology and cooperative work on the print industry shopfloor. Presented at: ECSCW'95, Stockholm, Sweden, 1995