Towards Normal Organisational Problem Solving

Jon G Hall
Lucia Rapanotti
Karl Cox
Steven J Bleistein

9th February 2007

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk
Towards Normal
Organisational Problem Solving

Jon G. Hall\(^1\), Lucia Rapanotti\(^1\), Karl Cox\(^2\), Steven J. Bleistein\(^2\)

\(^1\) Centre for Research in Computing, The Open University, UK
\(^2\) Empirical Software Engineering, National ICT Australia, Australia

\{J.G.Hall, L.Rapanotti\}@open.ac.uk, \{Karl.Cox, Steven.Bleistein\}@nicta.com.au

Abstract. The paper illustrates a sophisticated analysis and retrospective justification as adequate of the design of a solution of a real-world organisational problem. The approach integrates two extant frameworks defined by the authors, problem orientation and B-SCP, which is able to elaborate, transform and analyse the problem requirements, support the alignment of IT and business strategy, reason about the effect of partially detailed candidate solution architectures, and traceably audit design rationale through iterative development. The joint approach brings together many non-formal and formal aspects of organisational development, providing a structure within which the results of different development activities can be combined and reconciled.

Keywords: Problem Oriented Software Engineering, B-SCP, organisational engineering

1. Introduction

Design is predominantly about making and detailing choices. In most design endeavours, most choices are easily made—even predetermined—through appeals to normal design practice [15]. Many others others are parametrisations of normal design to cope with small, local variations in the context of the implemented and commissioned design. Although such observations were first made in the context of aeronautical engineering [15], they are much more widely applicable.

An analysis of normal design practice is not trivial, post hoc, as one must separate components—the ‘functional units’ normal design—from the glue—by which they are assembled—as well as determining the parameters that can be varied. In this paper, we make an initial attempt to place in their correct juxtaposition the components, glue and parameters for a complex socio-technical system: the retail chain that is Seven Eleven Japan.

Our approach to is to combine Hall, Rapanotti and Jackson’s Problem Orientation (PO; see, for instance, [6]), with Bleistein and Cox’s B-SCP [2][3]—two orthogonal approaches to problem solving. The first focusses on the transformational solution of engineering design problems, the second on the alignment of developmental and organisational goals. This paper illustrates how, through the combination of B-SCP and PO, a sophisticated analysis of complex real-world normal design can be begun.
The paper is organised as follows: background and related work are presented in Section 2. The basics of the POSE framework are described in Section 3. Section 4 demonstrates the use of POSE on the case study. Section 5 contains a discussion and conclusions.

2. Background and Related Work

2.1. Problem orientation

Problem orientation brings together many non-formal and formal aspects of development to provide a structure within which the results of different development activities can be combined and reconciled. Essentially, the structure is that of the progressive solution of a development problem; it is also the structure of the adequacy argument that must eventually justify the developed solution.

Problem Orientated Software Engineering (POSE), an application of problem orientation, regards a software problem as requirements in a (real-world) context. This view of a problem takes its inspiration from the work of Jackson [7] and is shared by all approaches to problem orientation.

A context is a set of (possibly) interacting domains described in terms of their indicative properties; each domain description captures a part of the real-world which is of interest in the problem; a requirement is a statement – written in the optative mood – of what should be true (or what should turn out to be true) of the context given an operating solution to the problem. A solution is simply the description of a domain (a machine and its software) whose behaviour is effected or constrained through design to solve a problem. Thus, a (software) problem challenges us to find a (software) solution that, in the given context, brings about the requirement.

POSE does not prescribe any particular development process; rather it identifies steps of development which may be accommodated within the development process chosen. Under POSE, problems requiring solution are transformed into other problems that are easier to solve, or that will lead to yet other problems that are easier to solve. Problem transformations capture discrete steps in the solution process. The following classes of transformation are recognised in the framework: representation (the initial identification of the major component parts of a problem); interpretation (capturing increased knowledge of the real-world and designed artefacts in problem descriptions); expansion (adding structure to a problem, its context or its solution); progression (simplifying a problem); and solution (providing justification that a solution description adequately satisfies the problem’s requirement). Each defined problem transformation transforms problems in a way that respects solution adequacy, and is accompanied by a justification for the transformation.

The justification for any transformation will not, in general, be formal and so the transformations of the framework need not be sound in any formal sense: the informality of the subject matter precludes fully formal treatment of some transformations. Illustrations of the forms of justification we admit are given later in the paper.

As well as novel solutions, solution interpretation, one of PO’s interpretation steps, allows known or likely adequate solutions or solution components—such as architectures—to be applied [6]. This leads the way to capturing the componentry and glue of normal design which, in turn, leads to extremely fast design cycles.
POSE has so far been applied to mission- and safety-critical software problems [9][8][11]. Problem orientation is sufficiently generic, however, to apply to more general types of problem. In particular, in this paper, we apply it to business problems in which the nature of the solution is socio-technical, i.e., composed of interacting technical and human subsystems. To do so, we refer to the ideas and notation of [5] where a problem-oriented view of socio-technical problems was first suggested.

The basic notation is summarised in Table 1. A Solution domain is the object to be designed to solve the problem. A Given domain represents a part of the world which is given but which is not the object of design. Typically, a solution domain interacts with a number of given domains in order to satisfy the requirement, i.e., the Requirement is what a solution needs to bring about through interaction with its context.

Solution domains have three flavours: a Machine domain represents a technological solution (part), e.g., a computer for which a program needs specifying; a Knowledge domain represents a solution (part) that requires some form of human flexibility – training, for instance, extra competencies or knowledge – to be designed as part of the solution. An example could be the training that a store assistant will need to be able to operate a Point of Sales system. An Organisational domain represents an organisation (or part thereof) that will need to be designed to solve an organisational problem.

In this paper, we give an example of an organisational solution to the business problem of supply chain management. As, organisational solutions typically consist of human resources and machines working together, we will show how organisational problems can be transformed into problems involving machines and people.

**Table 1. Basic notation for (socio-technical) problem diagrams**

<table>
<thead>
<tr>
<th><strong>Domain</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Machine domain</strong></td>
<td>A machine or technological grouping whose behaviour and properties have to be designed.</td>
</tr>
<tr>
<td><strong>Knowledge domain</strong></td>
<td>A person or social grouping whose behaviour and properties have to be ‘designed’</td>
</tr>
<tr>
<td><strong>Organisation domain</strong></td>
<td>An organisation whose behaviour and properties have to be designed</td>
</tr>
<tr>
<td><strong>Given domain</strong></td>
<td>A part of the world which is not subject of design and whose behaviour and properties are (more or less) known</td>
</tr>
<tr>
<td><strong>Requirement</strong></td>
<td>What we want to be true when the solution has been designed</td>
</tr>
</tbody>
</table>

As an example of the notation, Figure 1. provides an initial representation of the problem of designing a new franchise store for the 7/11 Retail Chain. In it, we see the
given domains—Customer, Logistic Ptnr and Supplier—that form the context of the franchise store, the requirements for the new store (here just labelled Satisfy Customers’ Needs). Links between solution and contextual domains capture relevant shared phenomena (information flow; e.g., entities, values, events, commands or operations). For instance, in Figure 1, 7/11 Organisation shares phenomena in set a with Customer; that those phenomena are controlled by 7/11 Organisation is indicated by the ‘!’ after the domain’s name abbreviation on the arc annotation; similarly phenomena set b is controlled by Customer.

The figure consists of a context diagram (all elements with solid lines) and the requirement (the ellipse labelled Satisfy Customers’ Needs) that makes it a problem diagram. The requirement is linked to domains whose properties and phenomena are referred to or constrained by the requirement. For instance, the dotted line between Satisfy Customers’ Needs and Supplier indicates that Satisfy Customers’ Needs refers to phenomena in set c, while the dotted arrow between Satisfy Customers’ Needs and Customer indicates that Satisfy Customers’ Needs constrains phenomena in sets a and b. Appropriate descriptions of solution and given domains, requirement and phenomena are associated with a problem diagram in the course of analysis, as we will see in the case study of the next section.

**Figure 1.** $P_{initial}$: The initial representation of an SEJ retailing problem.

### 2.2. B-SCP

To address alignment of requirements with competitive business strategy, Bleistein et al. [2][3] present B-SCP, a requirements analysis approach for the verification and validation of requirements according to alignment with and support for business strategy. B-SCP is based on the three themes of business strategy, context, and process. For each of the themes a requirements analysis technique is used: i* goal modelling [16] for strategy, Jackson context diagrams [7] for context, and role-activity diagrams (RADs) [10] for process. (That the context description is a context diagram forges the link with problem frames, and hence problem orientation.)

A cross-referencing technique is used to connect each theme with the other two to form an integrated model [2][3]. B-SCP combines use of business strategy analytical tools and requirements engineering techniques. Goal modelling is used to represent business strategy as requirements, and Jackson context diagrams to represent business and system model context. The strategy and context parts are integrated using a problem diagram framework defined in [2][3]. Strategy is elicited using VMOST [14], an organisational alignment analysis technique. The model for business strategy is constructed according to the Business Motivation Model (BMM)

[4], first proposed by the Business Rules Group and currently under consideration for adoption by the OMG. BMM presents a mechanism consisting of Means and Ends in the context of an organisation, where Means strive to achieve Ends. Means describes processes, tasks and activities in terms of Mission, Strategy and Tactics, while Ends are goals (soft and hard), which are achieved by using these Means; Ends include Vision, Goal and Objectives. B-SCP uses i* goal notation [16] to represent the model of business strategy. The i* operationalisation of BMM is illustrated in Figure 2.

*Figure 2. BMM Operationalised using i* [2][3]*

In this paper, we show how B-SCP modelling effort can be channelled within a problem oriented framework that guides the development of realistic high-level strategic goals through to lower-level detailed requirements. Problem orientation provides the benefit of traceability throughout.

3. Case study - Seven-Eleven Japan

7/11 (Seven-Eleven) Japan (or SEJ) is a retail convenience store. In this study we structure the decomposition of business goals as they are reflected in the design of a new franchise store. The structuring is retrospective, and based on a published B-SCP analysis (including [1][12]). We add value through the breaking up of the monolithic goal tree into chunks that allow the focus on the franchise store to be easily seen.

Such post-rationalisations are not just mind games; learning can be achieved through the codification of patterns in the problem and solutions domains, with the discovery of their interrelationship being particularly valuable.

3.1. Initial problem representation

In essence, the SEJ Organisation domain of problem $P_{Initial}$ (Figure 1) has been designed to address a well-known retailing problem: the company positions itself in the centre of a value chain that includes suppliers and third-party logistics partners, all of whom are independently-owned companies, yet all of whose objectives are to maximise throughput of products ultimately sold to end-customers. SEJ’s macro-level business model includes the participants mentioned above with which it shares the
phenomena involved in transactional flows of information and products with details of phenomena given in Table 2.

**Table 2. Shared phenomena of problem $P_{\text{Initial}}$**

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Phenomena</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>sell(goods)</td>
<td>Goods sold to customers</td>
</tr>
<tr>
<td>b</td>
<td>demand(goods)</td>
<td>Goods demanded by customers</td>
</tr>
<tr>
<td>c</td>
<td>order(goods)</td>
<td>Goods ordered from suppliers</td>
</tr>
<tr>
<td>d</td>
<td>pickUp(goods)</td>
<td>Goods acquired by logistics partners for distribution</td>
</tr>
<tr>
<td>e</td>
<td>deliver(goods)</td>
<td>Goods delivered by logistics partners</td>
</tr>
</tbody>
</table>

This initial representation is justified as follows:

$J_{\text{Initial}}$: at a macro-level, SEJ can be regarded as the solution to a generic retailing problem with the organisation being at the centre of a value chain that includes Suppliers and Logistics Ptnrs.

In the analysis that follows, there are a number of what might be called B-SCP steps and problem oriented (PO) steps. Each complements the other: a B-SCP step is a way of ensuring that structures are consistent with organisational goals and this can be used to justify a PO interpretation; a PO step is a way of assigning a particular goal decomposition to a design choice that must be made.

### 3.2. A B-SCP step: determining a single business goal

B-SCP allows the detailed consideration of an organisation’s objectives to determine goal analysis and decomposition so that objectives and, ultimately, specific tasks for the various socio-technical components of the organisation are aligned with those of the organisation.

The goal model of Figure 4 shows the top 5 B-SCP determined business goals ($G1$-$G5$) for SEJ, all of which are supported by (i.e., are discharged by a solution to)

$$G6: \text{Enable SEJ to provide products that consumers want when they want them according to changing needs}$$

also shown in the detailed decomposition of $G6$ through objectives to detailed tasks.

### 3.3. A problem oriented step: factoring in SEJ’s ‘organisational architecture’

SEJ manages, without ownership, a national network of convenience franchise stores. To better understand customer demand, SEJ actively gathers and analyses purchasing information in real time, and correlates this with other social and environmental factors including neighbourhood demographics; planned local events like festivals; and the weather. In other words, SEJ’s business strategy for competitive advantage as summarised in $G6$ rests on an extremely high level of competency at anticipating consumer purchases store-by-store, item-by-item, hour-by-hour, to provide customers with products they want when they would appear to want them.
SEJ’s strategy relies on a high level of store automation: SEJ’s active information management allows sophisticated supply chain management to reduce inventories, lower costs, and increase sales: SEJ shares information between with its partner companies via an ISDN network, using a just-in-time delivery system to meet demand.

Figure 3. The B-SCP determined goal model for SEJ [?]
Figure 4 provides an interpretation of \( P_{\text{Initial}} \), in the PO style, leading to \( P_{\text{Interpreted}} \) in which SEJ’s organisational architecture—consisting of interacting Head Office and Franchise Stores—is used to detail the SEJ Organisation domain, and leading to problem \( P_{\text{Interpreted}} \). The additional shared phenomena for this problem are given in Table 3.

That franchise stores exist might be considered an element of ‘normal organisational design practise,’ i.e., a design that is repeated throughout the sector. The particular details of a franchise store’s interaction with Head Office can either be regarded as a parametrisation of the generic retail solution based on a ‘franchise architecture,’ and designed as such or, as we do, as but another element of normal design for SEJ: in support of this we note that, although SEJ does not own the franchise stores, the SEJ business model prescribes very closely how a franchise store should conduct its business, in particular as to information gathering for forecasting customer demand.

The justification for this interpretation of the initial problem is:

\[ J_{\text{Interpreted}}: \text{The given SEJ architecture captures our knowledge of the organisation from the literature, contained in the literature.} \]

Table 3. Additional shared phenomena of problem \( P_{\text{Interpreted}} \)

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Phenomena</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>notify(purchase), notify(delivery), request(restock)</td>
<td>Flow of information on purchases, delivery and stock level from Franchise Store to Head Office</td>
</tr>
<tr>
<td>g</td>
<td>notify(trends)</td>
<td>Flow of information on expected trends from Head Office to Franchise Store</td>
</tr>
</tbody>
</table>

3.4. A B-SCP Step: operationalisation

The B-SCP analysis operationalises \( G6 \) into a single objective for SEJ, that to:

\[ O1: \text{deliver to Franchise Stores just-in-time} \]

Reflecting this decomposition in problem \( P_{\text{Interpreted}} \) leads to a problem with the same shape and domains as that of Figure 4, but in which \( G6 \) is replaced by \( O1 \).
3.5. A PO step: uncovering a co-design problem

With the introduction of two organisational components—a Franchise store and Head Office—we identify the following co-design problem [6]:

Franchise Store problem: to design a Franchise Store that interacts with the Head Office, Customers, and Logistics Ptnrs to satisfy the organisation’s goals;

and

Head Office problem: to design a Head Office that interacts with Franchise Stores, Suppliers and Logistics Ptnrs to satisfy the organisation’s goals.

Co-design problems are, typically, difficult to solve: they are to design problems what simultaneous linear equations are to equations of a single variable. In this case, however, we are analysing a current design and not synthesising a solution. Thus, we know Head Office is already designed, with standard protocols for interaction with a Franchise Store; the co-design problem is, therefore, reduced to the first of those identified, the Franchise Store problem. This problem is shown in Figure 5.

![Figure 5. The Franchise Store sub-problem (PFS)](image)

Under PO, from a problem such as $P_F$ in which requirements are expressed far into the world, we should progress to a problem in which the requirements are expressed closer to the solution. There are many explored ways of doing this: see [6][11] for formal or semi-formal definitions. Here our progression is eased by knowledge of how the organisation normally operationalises its objectives. In particular, there are three (sub-)objectives—$O_2$, $O_3$ and $O_5$. $O_2$ is an objective that concerns the relationship between Head Office, its Suppliers and Logistics Ptnrs:

$O_2$: co-ordinate the supply chain participants via data networking;

Those concerning the relationship between Head Office and Franchise Stores:

$O_3$: supply stock ordering decision support to Franchise Stores;

$O_5$: forecast Customer demand.

We focus on $O_3$ and $O_5$ as driving the design of the Franchise Store; $O_2$ will already have been discharged by the current Head Office domain.
O3 and O5 mention neither Logistics Ptnrs nor Suppliers, and so these can be progressed (removed) from the context of $P_{FS}$. Moreover, $O_6$ refines $O_5$.

![Figure 6. Progressed $P_{FS\text{progressed}}$ (without Supplier and Logistics Ptnr)](image)

The justification for this progression is:

$J_{FS\text{progressed}}$: The sub-problem directly derives from expansion of the 7/11 architecture, with Franchise Store as the subject of design. Its requirement is the result of a refinement of the B-SCP goal model.

### 3.6. A PO step: Interpretation and expansion of the Franchise Store sub-problem

Knowledge of the SEJ business model allows us to fill in some detail of the way a Franchise Store is organised; this detail is added through Solution Interpretation in Figure 7. Here, POS stands for Point Of Sale, a system that monitors customers' purchases, while GOT stands for Graphical Order Terminal, a device that allows Shop Assistant to track and report on sales and shop stock levels. Note that the interpretation of the Franchise Store sub-problem includes two targets for design: the Shop Assistant, a knowledge domain who deals with customers' requests and operates the store equipment, and the Store Computer, which is part of the formidable information system of SEJ and interfaces directly with the Head Office. GOT and POS are assumed to be two standard hardware devices, acquired from third parties rather than being designed. The new shared phenomena (sets $h$ to $l$) are detailed in Table 4.

![Figure 7. Interpretation of Franchise Store (problem $P_{SA}$)](image)

**Table 3.** Additional shared phenomena of problem $P_{SA}$

---

1 Actually, the removal of domains through progression is a sophisticated transformation requiring the requirement to be reinterpreted. We omit the details as the reinterpretation turns out to be trivial, even if the justification of why it is trivial is not itself trivial.
<table>
<thead>
<tr>
<th>Set</th>
<th>Phenomena</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>check(stock), check(trend), initiate(restock), record(delivery)</td>
<td>Basic actions of Shop Assistant for stocking goods and checking customers’ trends</td>
</tr>
<tr>
<td>i</td>
<td>record(purchase)</td>
<td>The action of Shop Assistant of recording a purchase through the system</td>
</tr>
<tr>
<td>j</td>
<td>initiate(restock), record(delivery)</td>
<td>Actions of GOT to relay Shop Assistant requests to Store Computer</td>
</tr>
<tr>
<td>k</td>
<td>update(trends)</td>
<td>Action of Store Computer to relay new trends to GOT</td>
</tr>
<tr>
<td>l</td>
<td>record(purchase)</td>
<td>Action of POS to relay to Store Computer a purchase entered by Shop Assistant</td>
</tr>
</tbody>
</table>

J64: The given Franchise Store architecture captures our knowledge of this organisational component from the literature. Again, the addition of detail in the design of the Franchise Store allows us to progress from requirements expressed on contextual domains to those expressed closer to the knowledge domain to be designed, i.e., the Shop Assistant.

The effect of the progression is to identify the Shop Assistant as the agent that enables O8: the continuous collection of data

The justification for this is:

J64Progressed: The sub-problem directly derives from expansion of the Franchise Store architecture, with Shop Assistant as the subject of design. Its requirement is the result of a refinement of the B-SCP goal model.

The relevance of this requirement is that it constrains the interactions of the Store Assistant with the Customer, and with the POS and GOT. This indicates a need for Store Assistant training: both technical training for the technical part of the role, and interpersonal training for the social part of the role. There is an important difference between the former and the latter: for the former, both POS and GOT are standard across the franchise network, which translates to standard training rolled out across the franchise. In the case of the latter, training for interpersonal interaction will necessarily be contextualised on the differences in the cultural characteristics of the customer base. This points to two things being important in the roll out of a new franchise store:

- a careful analysis of the nature of the Customer domain, in terms of its behaviour (for example, regional variations in traditional foods), intentions (including how they are expressed), expectations (for example, of service), and differentiation between sub-groups. This analysis needs to be on-going to track Customer changes
- a careful reflection on the interpersonal training required to reflect the discovered cultural characteristics.
Indeed, it transpires that SEJ routinely institutes careful analyses of the Customer group within the scope of new franchise store [13] (page 48).

4. Discussion and conclusions

In our view, organisational system engineering includes the identification and clarification of system requirements, the understanding and structuring of the problem world, the structuring and specification of a socio-technical solution that can ensure satisfaction of the requirements in the problem world, and the construction of arguments, convincing both to developers, customers, users and other stake-holders that the system will provide the functionality and qualities that are needed. System development is, thus, a complex, iterative process made more difficult by the need to relate human and physical domains to the formal world of the machine. An effective approach to system development must therefore deal adequately with the informal, the formal, and the relationships between them.

In this paper, we have illustrated how problem orientation applies to socio-technical problems. Figure 8. summarises the transformations which have been applied in the case study. The figure indicates the transformed problems at each step, and their corresponding justification. The analysis has demonstrated how the artefacts produced under problem orientation can be used to evolve the business level requirements such that they are directly related to social and technological components. It also illustrates how problem orientation can be used to record the structure of the development and of the argument of the adequacy of the proposed solution, including the rationale for important design choices (e.g., why the business architecture is appropriate) properly situated within the documentation of the development. Indeed, under problem orientation, each step in the requirements transformation process can be audited, the rationale for making a particular decision validated, and the traceability of the process demonstrated.

Note that the analysis we have conducted has shown only one possible path, and, of course, one which is not exhaustive of the problem: because of space limitations, we have omitted, for instance, the whole analysis of the Head Office sub-problem. Also, there was no illustration of iteration through the process, which is, of course, an essential characteristics of problem solving.
Figure 8. The path of analysis followed from the initial problem representation ($P_{Initial}$) to the reduced sub-problem for the design of the Shop Assistant ($P_{SAProgressed}$)

As a result of the study we are encouraged to believe the set of transformations defined in POSE may be sufficient for the analysis of complex socio-technical problems. There are, however, some validity threats to this claim. One threat is the realism or otherwise of the example. We argue that this is a well-documented case, taken from numerous sources in the literature. Our view of the SEJ problem is taken from that of industrialists, including SEJ’s CEO, as well as of leading academics. We can thus draw on a number of informed views on aspects of the case.

We use the notion of design of a knowledge domain to indicate the skills and competencies we should assume of a trained person performing a well-defined role within an organisation. We understand that it makes no sense to ‘design’ a human user of a machine and we are well aware that employees receive training to use the machines to do their work. However, we assert that it is critical for employees to be aware of exactly what their job entails and, even more significant for us, as engineers, that developers be very aware of the role of employees like a Shop Assistant have in the success or failure of the entire business. As an example, the Shop Assistant may be expected to record, say, each Customer’s age and gender at each transaction. This information can then be used by SEJ to implement its strategy for competitive advantage – providing the customers with the goods they want when they want them. SEJ examines each sales transaction and customer profile on a real-time basis to understand the demographics and customer purchase patterns of the catchment area for each store. If this requirement is not implemented correctly on the machine, or the Shop Assistant does not do their job properly, then the risk of failure is high. Without a detailed domain decomposition of the problem, in terms of domains and requirements, providing traceability from the business strategy to low-level requirements, this potential show-stopper might get overlooked.

5. Acknowledgements

The first and second authors are are pleased to acknowledge the financial support of IBM, under the Eclipse Innovation Grants, and the EPSRC, Grant number EP/
6. References


