Aligning the Map Requirements Modelling with the B-method for Formal Software Development

Abdul Babar1,2, Vladimir Tosic1,2,3, John Potter1
1 School of Computer Science and Engineering, University of New South Wales, Australia
2 Empirical Software Engineering, NICTA, Australia
3 Department of Computer Science, University of Western Ontario, Canada
abdul.babar@nicta.com.au, vladat@computer.org, potter@cse.unsw.edu.au

Abstract

We present a software development approach that aligns a requirements elicitation technique with a formal method of software specification abstraction. The goal/strategy modeling technique Map augmented with Jackson’s context diagrams (representing environment) is used to elicit requirements and the B-method is used to translate Map requirements into formal specifications. Comprehensive tool support allows the B-method to refine and implement the specification correctly. Our approach brings improvement to an approach that uses generic requirements for rigorous software development. The resulting specification model bridges the gap between software requirements and formal specifications and supports automatic refinement of strategic requirements into software code.

To illustrate how our approach bridges this gap, we discuss the Point of Sale (PoS) requirements model of Seven Eleven Japan (SEJ).

1. Introduction

Strategic alignment of business and Information Technology (IT) exists when a business organization’s goals, activities, and processes are in harmony with the information systems that support them [1]. Effective alignment positively influences IT effectiveness [2] and leads to superior business performance [3]. Getting alignment right means getting requirements right and ultimately getting the executable software right. Despite this, issues of business strategy and strategic alignment are all but ignored in requirements engineering (RE) research literature.

Two requirements engineering research approaches, Map and B-SCP, have delivered alignment between organizational intentions (on one side) and operational level requirements models (on the other side) through the formal refinement process. B-SCP presents an explicit approach of top-down refinement and bottom-up traceability between strategic and operational level models [4]. Map is a strategy driven business process modelling technique that presents a close relationship between goals and strategies [5]. However, B-SCP and Map do not materialize determined requirements into their executable software implementation, so further integration with formal software engineering methods, such as B-method [6], is needed. Thus, we set a research question for this paper:

RQ: How can we provide alignment between strategic requirements model and formal methods for software development that leads to the implementation of software system?

We chose to work with Map as a strategic requirements model and B-method as a formal method for software development. We find that Map has advantages over B-SCP:

(1) Due to the process-oriented nature of Map, it is easier to translate into B-method specifications, compared to the concept of goals as snapshot-in-time presented in B-SCP [4]. Map presents a clear view of what to achieve and how to achieve it using the concepts of goals and strategies, respectively.

(2) Map offers a mechanism that addresses evolution of requirements explicitly. Evolution of requirements in Map allows smooth transition of new requirements into the current software systems.

However, Map by itself does not address context explicitly, preferring to leave this to the judgment of the requirements analysts. To address the problem, we have presented a detailed approach of integrating Map with Jackson’s context diagrams in [8] and we reuse this solution in the work presented in this paper.

B-method [6] is a formal method that uses mathematical notations to specify requirements and has comprehensive tool support to animate, refine and implement the specification model into software.
Figure 1 shows our objective by presenting a step-by-step approach to preserve the alignment. Steps 1 and 2 represent refinement of a contextualised strategic Map to a detailed level where it can be translated in B-method in step 3. On the other hand, Steps 4 and 5 represent refinement and implementation steps of the specification model supported by B-toolkit. Our contribution is in Step 3 connecting Map with B-method. In this way, we present a software development approach that aligns the resulting product with organizational strategic intent more rigorously than the previous development approach that uses the B-method for specification and refinement of generic requirements obtained through domain analysis [9].

To present our alignment of Map with the B-method steps to implement a software system, we use the Point of Sale (PoS) case of Seven Eleven Japan (SEJ).

The rest of the paper is organized as follows. Section 2 presents background on Map, problem frames (including context diagrams), and B-method. Section 3 presents our alignment approach and Section 4 illustrates our approach on the Point of Sale (PoS) case. Section 5 discusses evaluation, while Section 6 draws conclusions and outlines future work.

2. Background

To address the huge challenge of competitiveness in a constantly changing software system environment, Map presents a strategy driven business process requirements modelling technique that represent close relationships between the “whys” and the “whats” [5]. The “whys” identify strategic goals of an organization and whereas the “whats” identify how they are achieved through tasks carried out by actors. A Map is a process model expressed in intentional terms. It provides a representation based on non-deterministic ordering of intentions and strategies [5]. Generally, a Map is composed of several sections/facets. A facet is an aggregation of two kinds of intentions, source and target, linked together with a strategy. An intention is a goal that can be achieved by the performance of a process. For example, “open a bank account” is an intention and “visiting a branch of the bank” is a strategy (see Figure 2). Each Map has two special intentions, start and stop, to begin and end the process respectively. In addition, Map is considered to have advantages over traditional AND/OR goal modelling techniques because Map captures variability through requirements analysis. Here variability means the ability to change, customize or configure software systems according to the user’s requirements [11]. As the evolution of a system creates movement from one state to another, the current state is captured as an As-Is model and the future one as a To-Be model. Map offers a Gap modelling technique to address differences between As-Is and To-Be models in a system [12].

Figure 2. Map modelling artefacts

Problem frames capture, structure and classify software development problems [13]. Problem frames are often derived via problem diagrams, which provide a larger analytical framework for requirements based on real-world physical entities and their observable interactions and behaviours. A problem diagram consists of two major components: a requirements part and a domain context diagram, as a requirement can only be understood in the context in which it occurs [13].

Context diagrams, illustrated in Figure 3, describe the properties of a problem diagram. They contain real-world physical domain entities, such as people, companies and hardware devices, sometimes also called domains of interest, as well as the phenomena two or more domains of interest share. Shared phenomena consist of observable behavioural phenomena that occur between entities in a context diagram. Context diagrams always contain exactly one entity called the machine, which is a general-purpose computer that we program and that may consist of multiple devices or computers. The requirements part of a problem diagram describes the effects in the real world that the machine should guarantee [13].

The B-Method [6] provides a model for the formal development of software through different phases of
specification, refinement and implementation. With tool support (described in [10]), the B-method allows production of code that an proven correct with respect to a high-level specification. The B-method encourages very high-level abstract specifications, and furthermore, allows the specification to model the environment (context) of the system, as well as the software itself. A specification in B-method is decomposed into a number of different modules (called abstract machines in B), each of which models some assumed context, abstract state, and a collection of operations representing behaviours of the system.

3. From Contextualised Map to B-method

Specification of Map requirements into B-method (Step 3 in Figure 1) provides a bridge between these two formal development techniques. The B-method itself supports the development of refinements and implementation from given specifications. Therefore, we focus on the translation of functional (detailed) requirements specified in contextualised Map into a specification written using the Abstract Machine Notation (AMN) of B-method. (For the rest of this section, we use italics for keywords used in AMN.) The reasons that support our approach are that B-method allows specifications to be written very abstractly. It offers a variety of ways of composing specifications, explicitly dealing with different forms of sharing between parts of specified sub-systems (B-machines). In this section, we evaluate underlying logic for the notations used in Map and B-method and discuss how particular constructs in Map extended with problem diagrams and B-method relate to each other. This will allow us to validate and verify requirements artefacts of a contextualised Map and a B-method specification against each other.

Requirements captured with a combination of problem diagrams and Maps represent a number of desired behaviours for the system, which refer to and constrain the domain context. One of the features of the B-method is that high-level specifications may incorporate a model of the surrounding context (environment). This context can be gradually refined, so that implementations only focus on the actual software and not the context in which it operates.

Every problem diagram in a requirements document provides a context. In translating low-level requirements into the B-method, we expect to represent a problem diagram with a Machine typically with the same name as the given context. Each machine defined in B has its own state within a specified environment, and an associated collection of operations.

Within a problem diagram, there will be many domains. Each domain can be represented in several ways within a B-machine. One way is to introduce into the B specification an entity (a Set or Constant in AMN) with associated Properties that provide assumed relationships between the entities. Entities modelled in this way are assumed to be fixed and immutable.

Other domains that denote changeable, dynamic entities need to be represented either as Variables or as other Machines. Domain entities represented as Variables form part of the state of the current Machine. A machine Invariant allows relationships amongst these variables to be expressed. This, in part, allows us to model the sharing that may be represented between domain entities of a problem diagram. Domain entities represented as other Machines denote other parts of the system that the current machine depends on.

In AMN there are a variety of dependencies that may be expressed between machines. The Sees dependency is used when a machine Operation refers to the information provided by another machine. We expect to use this in capturing the referred dependency in our requirements documents. The other machine dependencies are Includes, which allows a machine specification to be extended, and Uses, which allows a strong dependency between machines (though the state of the used machine remains encapsulated, so that it can only be modified with its own operations).

Those strategies identified by a Map requirements model that are intended to be achieved by a software system itself need to have Operations specified for them in the B-specification. Each high level Operation represents a single atomic update of state of the Machine that it belongs to. When there are constraints between a Map and a problem diagram, then we would normally expect them to be traceable through to particular aspects of the B specification of operations. For example, an operation may explicitly update a variable that represents a domain constrained by the strategy corresponding to the operation.

The B-method explicitly avoids interference between effects of operations specified in different ma-

![Figure 3. Anatomy of domain context diagram](Image 240x643 to 288x669)
chines. In translating from requirements to B-specification, we need to model the problem diagram abstractly enough to achieve this separation and non-interference. In particular, it is often the case that the B-method will encourage us to model explicitly any shared effects in a system. Identifying these possible points of conflict and sharing between different parts of a software system as early as possible is one of the benefits of system specifications based on B-method.

4. Example: Seven Eleven Japan (SEJ)

Seven Eleven Japan (SEJ) manages a national franchise of independently owned convenience stores that provide solutions to the daily needs of consumers on an hourly basis [14]. SEJ uses IT to leverage information to coordinate a supply chain of business partners, which help stores to stock products based on hourly-based consumer needs [15]. Business partners of SEJ include suppliers, who produce the products, combined delivery centres, companies with warehouses and fleets of trucks that are used for deliveries to the franchise stores; these are the direct customers of SEJ and provide services to consumers. SEJ is responsible for the IT and marketing strategy of all franchise stores; this provides a competitive edge to the convenience stores over other businesses. While land is a precious commodity in Japan, stores have very limited space in which to hold inventory. Therefore SEJ’s business strategy is to provide an appropriate order for each store based on its demography, local special events, weather forecast, etc. SEJ is able to deliver fresh products such as boxed lunches, rice balls, and the like, four times a day with a just-in-time delivery philosophy to meet the actual demands of consumer [15].

4.1. Map Refinement Approach

Refinement of a Map means to select a facet <source intention, strategy, target intention> and expand it to another Map. The entire refined Map represents the facet of the higher Map. By employing this technique, we have developed four Maps with Jackson context diagrams: MA, MB, MC and MD. We present an alignment of these four Maps in Figure 4. The first three Maps (top down) have been adapted from [7], where these Maps were part of the integration approach with B-SCP. By refining the third Map (MC), we achieve the detailed level fourth Map (MD) with its domain context. Map MA captures SEJ’s strategic requirements and its facet <Start, By using IT systems, Manage Convenience Store> is refined to achieve Map MB. The facet <Start, By using store computer, Build Customer Purchase Model> of Map MB is refined to achieve Map MC. We refined the facet <Start, Using PoS, Collect Consumer Profile and Purchase Data> of Map MC to achieve Map MD.

4.2. Identifying Map MD

The franchise store computer relies on Point of Sale (PoS) for gathering consumer profile and purchase data. DD presents domain context for PoS machine, which performs share phenomenon (p) with Cash Drawer. The cash drawer opens only when the clerk (r) uses the PoS to enter (r2) consumer’s profile (age and gender) and scan (r1) barcode of the product (q).

The processes of Map MD attempt to achieve two main intentions, (i) Enter Consumer Profile Data on PoS and (ii) Perform Payment Transaction. The (i) intention is achieved by executing two sub-processes, Scanning products (S11.3.1 and S11.3.2) and by entering consumer’s age and gender (S11.3.5 and S1.13.6). Both processes have the same source and target intention. It achieves changed state of the intention (i) after the product is scanned.

The processes involved in achieving the intention Perform Payment Transaction are executed once the intention (i) is achieved completely. Cash drawer for payment transaction is not meant to be open until the consumer’s age and gender is entered by the clerk at PoS. The payment process requires consumer to pay to the clerk. The clerk adds this money to the cash drawer and the PoS terminal displays any required change with receipt to the consumer. The process ends with the customer’s departure and the PoS system transmits data to the store computer.

Thus, three processes: (i) scanning products, (ii) entering consumer’s profile (age and gender), and (iii) payment transaction are executed to achieve the two main intentions. The processes also show explicit involvement of domain entities in them.

4.3. Point of Sale B-machine

Starting from the requirements identified for the Point of Sale system in Map MD of Figure 4, we have developed the specification, in B-method, of an abstract PointOfSale machine, presented in Figure 5. This specification has been validated using the animation facilities of the B-Toolkit demonstrates the state transition of the specification achieved through its operations invoked interactively through user-provided inputs.
Figure 4. Refinement of Map models and domain context for the SEJ example
MACHINE PointOfSale

SETS PRODUCT; RECEIPT;
STATUS = {opened, closed};
GENDER = {male, female}

CONSTANT empty_bag

PROPERTIES empty_bag ∈ BAG ∧ empty_bag = _⊥ pp. (pp ∈ PRODUCT | 0)

VARIABLES items, till, price

INVARIANT till ∈ STATUS ∧
items ∈ BAG ∧
price ∈ PRODUCT → N

INITIALISATION till := closed ||
items := empty_bag ||
price := PRODUCT → N

OPERATIONS
checkIn (item) =
PRE till = closed ∧ item ∈ dom(price)
THEN items(item) := items(item) + 1 ENDS;
cost ← checkOut (gender, age) =
PRE till = closed ∧ gender ∈ GENDER ∧
age ∈ N
THEN till := opened || cost := total END;
receipt, change ← pay (cash) =
PRE till = opened ∧ cash ∈ N ∧ cash ≥ total
THEN till := closed || items := empty_bag ||
receipt ∈ RECEIPT ||
change := cash − total END

DEFINITIONS
BAG = PRODUCT → N;
Total = \( \sum \) item. (item ∈ dom (price)) | items
(item) × price (item))

Figure 5. B-specification of Map requirements

We identify three state variables till, items and price for the PointOfSale machine. The till may simply be opened or closed. At this level of specification we have not attempted to model the contents of the till; indeed it is unclear whether this responsibility lies with the PointOfSale machine. The items keep track of the number of each type of PRODUCT that have been checked in and placed in the shopping bag. The price is the per item cost; this information should be retrieved from the store. To keep our example simple, we have included price as an abstract updatable function in this machine, whereas, in a proper B-method development, it would be specified as part of a separate Store machine, and simply used by the PointOfSale machine. An auxiliary definition is used to calculate the total cost of items in the bag in terms of the price per item. The PointOfSale machine has just three operations checkIn, checkOut and pay. In Table I, we show these three B-specification operations and explicitly present their association with requirements of Map MD. The table also shows the involvement of domain entities for each operation and the state of the B-machine variables during the execution of each operation.

The first two operations, checkIn and checkout, meet the intention of the Enter Consumer Profile Data requirement and associated strategies S1.1.3.1, S1.1.3.2, S1.1.3.5 and S1.1.3.6 of Map MD. The checkIn operation accepts an item (which will be identified by a scanner or by direct entry) and requires both the till to be closed, and the entered item to have a price. The checkIn operation counts the scanned/keyed-in items in its respective PRODUCT line. The till remains closed. The checkOut operation accepts customer gender and age details as entered by the clerk (perhaps, a higher level specification should leave these details as an abstraction). This operation is the only way in which the till can be opened, which means that the requirement for entering these details before checkout is enforced. The checkOut operation also outputs the total cost of items scanned. Note the role of the till state in controlling the allowable sequence of behaviours, which we could express as a regular expression as (checkIn*; checkOut; pay)*. The use of a control state to constrain allowable behaviours is typical in state-based specification techniques.

The business intention Perform Payment Transaction and associated strategies S1.1.3.7, S1.1.3.8, S1.1.3.9 and S1.1.3.11 are represented in the pay operation. It takes cash as an input (from the customer), if the till is opened. Then, the operation pay resets the till and items to the initial state, ready for the next customer. It also outputs the appropriate change and a receipt for the current customer. The PointOfSale machine addresses business goals Enter Consumer Profile Data and Perform Payment Transaction and all relevant domain entities, cash drawer and product.

The Consumer is not a part of the specification of the software because it has no direct interaction with the application. The Clerk is labelled Biddable because its job is to operate the software in accordance with its job description and the PoS application process.

The strategies By sighting customer, S1.1.3.3 and By customer departure, S1.1.3.12 are not modelled in the B-method specification because they are not directly related to the software. The strategy Transmit data to store computer, S1.1.3.10 is not currently recorded in the PointOfSale machine. In an extended specification, incorporating the Store machine, these customer details would cause an update to that machine, but not the PointOfSale machine. Thus, we have managed to separate the Point of Sale
systems from the interacting systems of SEJ. The checkOut operation inputs gender and age, but does not update any customer details within the PointOfSale machine.

5. Discussion

In this paper, our main objective is to align Map requirements modelling approach to B-method software development approach. For this purpose, we have used a detailed level example of a contextualised Map MD and specified in B-method. The reason for choosing this example was to provide a concrete analysis of the requirements and their satisfaction in B-specification (section 4). We animated the B-machine by using B-toolkit and invoked three operations. The operations were displayed on screen with preconditions and guards simplification. We input data for three operations and outputs were displayed satisfying the Map requirements.

The alignment of strategy-driven requirements modelling technique Map with the formal methods B-method has a number of advantages over the approach presented in [9]. Snook et al. [9] presented an approach that uses UML-based process for obtaining a generic model of requirements. This model gives little indication about identifying organizational intent and its refinement. They also realize the importance of the reuse of generic sets of requirements driven by requirements evolution [9], but do not give indication of any technique to be used for requirements evolution. Contrary, strategy-driven Map allows refinement of the Maps as shown in Figure 4. In addition, Map also offers development of As-Is and To-Be models to address the requirements evolution. Map also facilitates the gap analysis technique to address new changes in the system [11][12].

Specifically, we have presented an approach in which a goal/strategy modelling approach Map is aligned with B-method software development approach. It allowed translating Map processes, domain entities and their shared phenomena in B-specification and subsequently refinement and implementation models. General insight of our approach is that the requirements engineering research is dominant by goal modelling approaches. Our alignment approach can encourage other goal modelling techniques to be aligned with B-method or other formal methods. So that formal software development approaches can be developed and evaluated against each other. Therefore our approach set a trend of translating goal-oriented requirements modelling approaches into formal methods. However, we encountered anomalies during the alignment approach. The Map MA presents SEJ’s strategic intent in an enterprise domain but the refined Map MD shows a limited domain of Point of Sale (PoS) Systems. It appears that the refinement process has narrowed down the focus as we reach to a detailed level Map MD. To specify Map requirements in B-method we need a detailed level Map which we achieve through refinement. In the refinement process presented in Figure 4, it is clear that we have refined only once facet of each higher level Map. To achieve a

### Table I. Requirements satisfied by each operation by using domain entities.

State of the variables after the execution of each operation

<table>
<thead>
<tr>
<th>B-Method Operations</th>
<th>State Variable</th>
<th>Map Requirements</th>
<th>Domain Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op checkIn</td>
<td>Till = Closed</td>
<td>&lt;Start, By keying barcode on PoS, Enter consumer profile data on PoS&gt;</td>
<td>Clerk, Product, Consumer</td>
</tr>
<tr>
<td></td>
<td>Items = Counts</td>
<td>&lt;Start, By scanning product, Enter consumer profile data on PoS&gt;</td>
<td>Clerk, Product, Consumer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Start, By sighting consumer, Enter consumer profile data on PoS&gt;</td>
<td>Clerk, Product, Consumer</td>
</tr>
<tr>
<td>Op checkOut</td>
<td>Till = Open</td>
<td>&lt;Enter consumer profile data on PoS, Entering age strategy, Enter consumer profile data on PoS&gt;</td>
<td>Clerk, Consumer, Cash drawer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Enter consumer profile data on PoS, Entering gender strategy, Enter consumer profile data on PoS&gt;</td>
<td>Clerk, Consumer, Cash drawer</td>
</tr>
<tr>
<td>Op pay</td>
<td>Till = Closed</td>
<td>&lt;Enter consumer profile data on PoS, By opening till, Perform payment transaction&gt;</td>
<td>Clerk, Consumer</td>
</tr>
<tr>
<td></td>
<td>Items = 0</td>
<td>&lt;Enter consumer profile data on PoS, By receiving Cash, Perform payment transaction&gt;</td>
<td>Clerk, Consumer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Perform payment transaction, By generating change, Perform payment transaction&gt;</td>
<td>Clerk, Consumer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Perform payment transaction, Generate receipt, Stop&gt;</td>
<td>Clerk, Consumer</td>
</tr>
</tbody>
</table>
complete enterprise domain of SEJ business strategy at
detail level, we should have refined all the other facets
of Map MD as well as the facets of the following
higher-level Maps.

In [8], we have presented a domain identification
approach for Map by using Jackson’s context diagram.
[8] describes a step-by-step approach for eliciting
constraints and references for Map requirements. We
showed all four Maps embedded in Jackson’s context
diagrams. Domain context of the context diagram
represents constraints and references for a Map model, as
shown in Figure 4. Use of Jackson’s context diagrams
was crucial in achieving alignment between Map and
B-method, because B-method relies on context. In par-


cular, B-machine invariants can be obtained from
constraints, references and shared phenomena between
domain entities in Jackson’s context diagrams.

6. Conclusions and Future Work

We have presented a methodology for rigorous
software engineering approach that preserves align-
ment between organizational strategic intent and for-
mal methods that lead to the implementation of a soft-
ware system. We illustrated this methodology by
developing a Map that addresses strategic intent of Seven
Eleven Japan (SEJ). This is an appropriate case to
evaluate our alignment approach because SEJ has net-
work of franchise stores that work in coordination to
achieve organizational business strategy.

We have started from a high-level strategic Map
model. By using the Map refinement technique, we
generated a detailed level Map that is aligned with the
strategic Map model. We extended this alignment by
specifying detailed level Map requirements in a B-
machine. B-method allows refinement and implemen-
tation of the specified B-machine because it comes
with comprehensive tool support (B-toolkit) that guar-
antees that the implemented model will be correct with
respect to the specification. In this way, we have im-
plemented SEJ strategic intent into software generated
by the B-toolkit. In this paper, we have developed a
specification model from Map MD requirements
model. In addition, we have invoked and tested the B-
machine operations which that satisfies the Map re-


requirements.

In our future research, we will refine and imple-
ment the specification model presented in Figure 5 by
using B-toolkit. The implementation generates code,
which will be accurate against the specification of Map
MD requirements, as guaranteed by the B-toolkit. This
implementation will enable us to study to what extent
the generated code satisfies the strategic intent of SEJ.

References

issues in IT management. Chichester; Hoboken, NJ: Wiley,
2003.
[2] M. Porter, "From Competitive Advantage to Corporate
1987.
ology Trilogy: Business Strategy, Technological Deploy-
ment and Organizational Performance," Journal of Strategic
egic Alignment of Organizational IT Requirements Using
Goal Modeling and Problem Diagrams," Journal of Systems
628-649.
wards Evolution of Strategic IT Requirements”, 22nd ACM
120-127.
[8] A Babar, K. Cox, S. Bleistein and J. Verner (2007), Ident-
ifying Domain Context for the Intentional Modelling Tech-
nique Map, 1st International Workshop on Enterprise Infor-
mation Systems Engineering – WEISE 2007, Regensburg,
Germany, 3-7 September 2007 (workshop held in conjunc-
tion with DEXA, proceedings published by IEEE) (ac-
cepted).
[9] Snook, C., Poppleton, M. and Johnson, I. Towards a
Method for Rigorous Development of Generic Requirements
Patterns, Workshop on Rigorous Engineering of Fault Toler-
ants Systems, Newcastle, UK, Springer-Verlag, 2006, pp. 17-
27.
in a Co-evolution Context”, 13th IEEE Int. Conf on Re-
Gaps in Requirements Change”, Requirement Engineering
[13] M. Jackson, Problem Frames: Analyzing and Structur-
ising Software Development Problem, 1st ed: Addison-Wesley
[14] M. Bensou, Seven Eleven Japan: Managing a Net-
worked Organisation, Insead Euro-Asia Centre, case study,
1997.
Japan, in: Information Technology Strategies: How
Leading Firms Use IT to Gain an Advantage, Oxford