Supporting Component-Based Software Evolution

Ross Gardler and Nikolay Mehandjiev

Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, UK
e-mail: {rgardler | ndm} @co.umist.ac.uk

Abstract. An eBusiness is one in which major parts of its core business processes are automated. This dependence on technology enables innovative business models, but also demands that developing supporting software systems involves carefully considering company's future strategy and business models. This paper presents a new approach to semi-automated component-based evolution of eBusiness support systems. The process is automated by linking business strategy with software structure using mappings between business process patterns and software patterns. Identified software patterns are used to guide the procurement of appropriate components using software "test harnesses". A strategic positioning and planning tool, the eBusiness Maturity Model (eBMM) is used at the strategic level, allowing the planning of future extensions of software support to progress up the eBusiness maturity levels. We illustrate the use of this model in an eCommerce start-up and show how the model guides the procurement of software components to support mission-critical parts of the business whilst enabling future growth in the marketplace.

1 Component Based Software Engineering for eBusiness

Software systems to support eBusiness have to be flexible and robust, whilst being developed in "internet time" – weeks rather than months or years. The assembly of systems from pre-existing components, using new standard technologies for inter-component communication, such as XML, and new standard architectures such as ebXML to provide a deployment framework for components, is an attempt to satisfy these requirements.

The emergence of component-based software has seen work progressing in the area of component-based development methodologies (reviewed in [1]), which aim to increase the level of software reuse through the adoption of a new software lifecycle. This new lifecycle includes a modified design phase, containing new activities such as the selection and creation of architectures as well as the selection and customisation of components. A component-based software lifecycle has considerably longer analysis and design phases [2], whilst the implementation phase focuses on component integration rather than on programming [3].

Pre-emptive implementation of components shortens development times, but this is partially offset by the time spent in procuring new components and integrating them with the rest of the system. One way to increase the efficiency of the developers is to increase the level of re-use abstraction by operating with groups of components rather than with individual components. We do this by considering the parts components

play within software patterns and by semi-automating some of the component procurement and integration activities.

There are a number of approaches to component based system composition [3-6] and languages for expressing the composition of these systems [7, 8] however the focus of these efforts is on the technical issues of building systems from components and the definition of such systems. In this work we also consider how to align these models and systems with the strategic goals of an organisation. We provide a methodology for the selection of suitable software patterns to support an organisations progression along their chosen path, whilst using the flexibility of component based systems to provide the ability to realign implemented systems when the future turns out to be different from that planned.

Our approach supports the semi-automatic procurement and re-use of software patterns, components and groups of components through a technique of mapping business strategy, expressed as business process patterns to software design patterns and software structure. This approach is based on two premises:

1. Software patterns can provide the link from business "best-practice" processes to software structure because they can be viewed from two different perspectives. A high-level view sees them as patterns illustrating a fragment of a business process and the system support needed for effective automation of this fragment. Another, more detailed view sees them as an initial specification for the design of supporting software systems [9, 10]. We call the high-level view *"business pattern"* and the detailed view *"software pattern"*.
2. A measure of the suitability of a software pattern should be based not only on its ability to satisfy requirements now, but also on its ability to adapt to requirements in the future. Maximising the ability of software to change without redesign is a cornerstone of the approach to examining software design patterns advocated by the "gang of four" [9]. In our approach, we extend this pattern-based flexibility to the strategic level of business design through our strategic positioning and planning tool, the *eBusiness Maturity Model (eBMM)*.

The selected software patterns are used to guide the semi-automatic procurement of software components within that pattern based on the current conditions in the environment. The techniques presented in this paper for component selection can be applied to other domains in which such transient component selection would be useful, such as telecommunications. However, this paper is concerned with aligning business strategy to software support systems and so, for the sake of brevity, we do not consider these other applications.

2 Our Approach to Component-Based Evolution of Software

Our approach to developing and evolving eBusiness software is illustrated in Fig. 1. In this approach we use a strategic positioning and planning tool (the eBMM) to determine suitable business patterns for the new business processes to be adopted. The libraries from which these patterns are selected reflect the generic operations of a typical business, and the recurring business patterns in typical organisations. They are developed by specialists in each discipline, and represent what is regarded as best

practice in that field. Each business pattern has, stored within the library, one or more related software patterns that have proven to be useful when implementing systems to support that business process. These software patterns are used to determine the interfaces of constituent components, and their initial requirements. Software and business process engineers will examine these patterns in the context of the strategic goals identified with strategic positioning and planning tool. Having examined the "fit" between the design pattern and the real requirements of the system we can fully specify the requirements of each component. These full requirements guide the building of "test harnesses" used when automatically procuring system components.

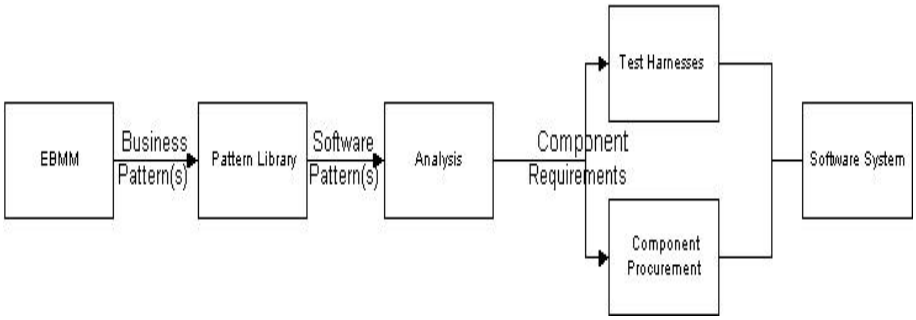


Fig. 1: Developing Component-Based Software for e-business

The whole process begins with the strategic positioning of the business and the identification of priority areas of development for both the business and its supporting software system. This positioning of the organisation is carried out through the application of an eBusiness Maturity Model (discussed in the following section), which provides metrics for measuring the organisations progress towards their desired strategic goals within a specific set of business processes, for example, procurement of raw materials, handling of customer complaints, or quality control. Having identified the strengths and weaknesses of the business it is possible to identify the need for strategic development in particular areas, thereby enabling the organisation to focus resources on the most important aspects of business process and supporting software systems. We consider patterns as being of particular value at this point since they recognise that success is more important than novelty, that is finding a pattern is a matter of discovery and experience, not invention [5]. The eBMM guides the organisation in the discovery of suitable business patterns.

However, design patterns are generic and thus unlikely to fully cover the unique requirements of an organisation. We therefore need to complement the mapping between business process and software structure provided by the patterns with organisation-specific mapping between business needs and software support requirements. These organisation specific mappings enable the identification of differences between the generic patterns and the specific implementations required. This phase is akin to traditional Business Process Reengineering.

After selecting and customising the business patterns that underpin these improvements, we are now in a position to select, from the library of best practice,

suitable software design patterns, in the absence of a suitable pattern a new one must be developed using conventional software engineering techniques. The patterns are evaluated against their ability to guide future enhancements to the supporting software system. That is, the software patterns are selected based on their ability to support the expected future requirements of the system, as indicated in the strategic planning process discussed in the next section.

These enhancements are designed and implemented using a modified version of Extreme Programming [11]. Born of industry rather than academic research, Extreme Programming is a software development methodology that has emerged in response to the need for fast results and maintainable software. It differs from other development methods in a number of important ways, those that make it particularly important to our approach, which focuses on re-use and re-design, are:

- Test harnesses are written for all code units before any implementation code is written.
- Code that satisfies the test harnesses satisfies the requirements, the goal is to make it work first, and to improve it through re-design second
- Design is continual and code is refactored to fit the improved design of today, this refactoring ensures maximum re-use of code.

Iterations between development phases are rapid, focusing on frequent incremental improvements rather than infrequent major alterations. Having identified the major patterns in the software enhancements, we can decompose them into components (possibly themselves defined by further design patterns in a recursive manner). These components are specified by the generic interfaces gleaned from their software design pattern and complemented by a set of requirements unique to our particular domain, identified through more traditional requirements analysis techniques but constrained by both the business and software design patterns selected. It is during this phase of development that the organisation introduces their unique perspective on the design patterns. By providing a proven starting point in the form of proven patterns we aim to allow systems and business analysts to focus on improvement rather than (re)invention. The resultant structured specification of requirements and interfaces to each component are used to pursue the automated process of (a) finding components that will interoperate to fulfil this design and (b) integrating them together.

To ensure a component discovered is suitable for use in our final system we use the pattern and component specifications to build a set of test harnesses for individual components, for sub-systems containing a number of individual components and for the system as a whole. For this purpose we use an open source regression-testing suite, JUNIT [12]. These test harnesses are used to validate the suitability of the components found. As groups of components matching the software patterns are completed the resulting system can be tested for requirements coverage on increasingly higher levels of abstraction.

Before we can proceed with testing components we must first source them. Component procurement contains a stage of automated multi-criteria negotiation of contracts, using the specified system requirements as criteria. Note that the automated procurement process may satisfy the component requirements in any way, that is it may be a single component or a group of components that together fully satisfy the requirements. In this way, components in one pattern can be further decomposed into

patterns and components. In order to ensure full coverage of the requirements each component or set of components are tested using the provided test harnesses.

The writing of test harnesses is considerable quicker than writing the components themselves. They can be used to test a number of components enabling the most suitable to be selected based on criteria such as speed of operation, price, reliability etc. Furthermore, should a component not be available that performs the required action they can be used as a requirements specification for software developers, if the developed software passes the tests then it covers all requirements. Since the test harnesses are themselves program code there is no longer any ambiguity about what outputs are expected given certain inputs.

In the remainder of this paper we examine this process in more detail. We first examine our strategic positioning tool and how it is used to guide selection of suitable business and software patterns. We then illustrate these patterns are turned into an executable system by examining a case study.

3 An Introduction to the EBusiness Maturity Model

Our first step is the strategic positioning of an organisation to help identify which business processes should be targeted for improvement through the introduction of information technology. This stage is guided by the eBusiness Maturity Model, which is based on the principles of the Capability Maturity Model for software Engineering [13] and the eBusiness Guidance Model [14]. These models provide a way of categorising an organisation according to their ability to perform their functions in what is regarded as the most efficient and effective way. This categorisation is against a number of defined levels, termed maturity levels. Entry in to a higher, more mature, level requires improvements in the way the organisation operates. Initially these levels are defined by accepted best practices in the specific industrial sector, however, over time they are customised to suit the needs and objectives of the organisation. In contrast to the CMM, which focuses on software development processes and organisations, eBMM and EBGM focus on the business processes in commercial entities. The difference between the latter two is the size of organisation they target, the EBGM is focussed on large organisations whilst the eBMM has been developed by us to help small and medium companies evolve their processes in an Internet environment.

The eBMM can be used in a “strategic positioning” mode, which identifies the current position of a company against a particular industry sector. Once this is done, the eBMM can be used in a “strategic planning” mode, where the company is identifying priority areas of improvement, recommending changes in business operations so that it may progress up the maturity levels. eBusiness is characterised by a close relationship between the business and the software systems that support it, therefore the recommended changes in business operations will normally be directly translatable into requirements for new software functionality to support the new operations, and therefore the business process patterns will be translatable into software design patterns.

The transition from strategic positioning to strategic planning through the identification of software support requirements in the form of design patterns constitutes the first phase of our approach. This transition is guided by the eBusiness Maturity Model and related business patterns, which are themselves subjected to development and evolution as the company internalises and enhances a generic eBMM based on feedback and experience. Our motivation for the design of the eBusiness Maturity Model is similar to the motivation in examining design patterns expressed by the "gang of four":

"Consider what should be variable in your design. This approach is the opposite of focusing on the cause of redesign. Instead of considering what might force a change to a design, consider what you want to be able to change without redesign." [9]

The following section provides further details of the interplay between eBMM customisation, business patterns and software patterns.

4 The EBusiness Maturity Model and the Selection of Software Patterns

The eBusiness Maturity Model is a model created by the strategic managers of an organisation on the basis of a generic model, which is developed for a particular industry sector. Each level of this model is aligned to a number of business patterns and software design patterns that guide the development of the business processes and supporting software systems of an organisation. Some of these patterns will span a number of levels within the maturity model, whilst others may correspond to a single level. This is true of both the business patterns and the software patterns. That is, some patterns will be the same within many organisations regardless of size and maturity, whilst others will need to change as the company grows and matures.

Once the organisations strategy managers have identified their starting and goal positions within the model, they can use the model to define a path from current position through to the desired goal position. This path is initially defined in terms of generic objectives and high level business processes, which can then be further refined, by business managers, with detailed metrics against which organisational performance can be measured. Once a detailed progress plan is defined, it is passed to the software analysts who convert the plan to a roadmap of requirements for the supporting software system. This will enable the selection of software patterns from the library. Patterns are selected based on their applicability to the expected future processes identified by the strategy roadmap. For example, if an organisation's strategic plan includes progress from Level 3 to Level 6 in their supply chain management operations, software analysts can examine typical software patterns for organisations operating at these levels. Patterns that are applicable across multiple levels will be more suitable than those that are constrained to a single level. Thus, strategy managers, business managers and software engineers can focus on what they want to be able to change without redesign [4].

For any organisation, the process of software development is intertwined with a continuous process where the eBusiness Maturity Model is itself refined by the

organisation's managers in response to changing strategic goals or business environment. Fig. 2 illustrates this process. It starts with a generic maturity model and infrastructure that has been selected for its applicability to the organisation in question, for example, a set of best practice models for supply chain management, together with a typical system design for supply chain management software. This generic model and software design is quickly customised to more closely fit the strategic goals of the organisation and the resultant customised model is used to identify areas for potential improvements in existing software solutions. We then enter a new cycle, adapting the eBMM to reflect new strategic goals and environmental conditions, the maturity model is customised, the business is evaluated against the new model and further customisations of the business process and software systems are identified and prioritised. The eBMM ensures all stakeholders in the system are involved in each iteration of the cycle, thus a change in strategy that will cause problems in implementation (of either business or software processes) is identified early and appropriate action can be taken.

To illustrate the use of our approach in practice we will now examine an example development of software systems for an eBusiness organisation.

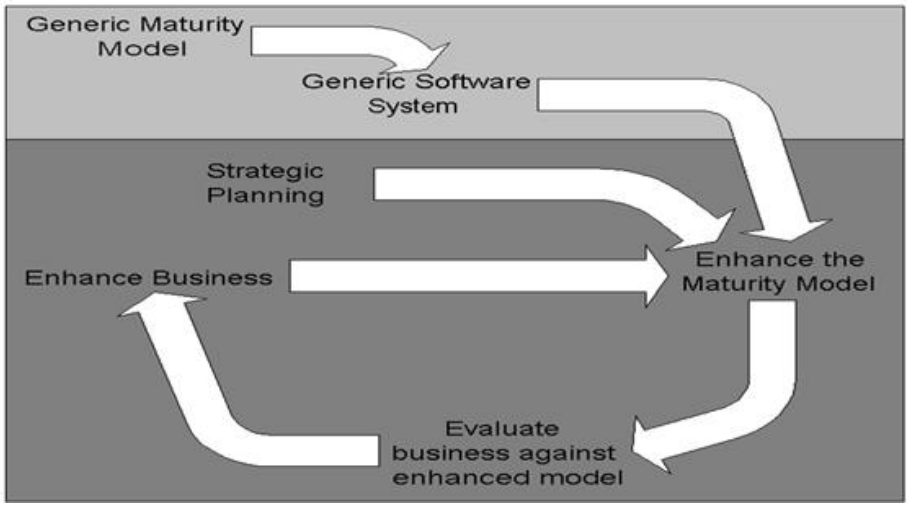


Fig. 2: Applying the EBusiness Maturity Model

5 Developing Software for a Start-up EBusiness

We will examine an eBusiness start-up to illustrate the application of our approach. This hypothetical organisation sells products from a large number of suppliers to the end users of those products. A full market analysis has identified their customer base, product lines, and key suppliers. They have drawn up a business plan, which clearly predicts their future growth for the next three years and provides indications of expected growth beyond this period. Immediately we see one of the main problems with traditional software development techniques. Software is inflexible. It is written

to satisfy a set of requirements as they are understood today. This understanding is lacking in two ways. Firstly, it is impossible to accurately predict the future, for example, some revenue streams may turn out to be less lucrative than predicted, consequently, different streams, not originally planned for may be introduced. Secondly much of the detail in requirement specification is lost in the translation between business leaders and software engineers. Unfortunately, we have not identified a way of increasing the accuracy of future prediction. However, experience and an increased understanding of the domain can minimise misunderstandings between software engineers and business managers.

The use of best practice models facilitates the transfer of knowledge between people and organisations [15]. Therefore, the identification of best practice models, in the form of business patterns, to provide a starting point for business and software development is the first phase of development. These models are then enhanced to match the company's specific goals and objectives. The eBusiness Maturity Model provides a methodology for identifying such best practice models and for the subsequent selection of software design patterns to guide the implementation of software to support the recommended business processes. Later in the life of the business, the eBMM is used to prioritise and specify improvements in supporting software systems.

5.1. Creating a Personalised EBusiness Maturity Model

In the remainder of this example, we will focus on the core functionality of our eBusiness organisation, that is, supply chain management, the procurement of supplies and the subsequent resale to customers. We have chosen this aspect of the business because it is a key area in most businesses, that is, all businesses need to buy and sell things, and because it is an area in which the adoption of technology can provide many benefits. It should be noted that the organisation would not normally select just one aspect of the business. They would develop a Maturity Models that reflect the strategic goals of the organisation in all areas of business.

Having identified supply chain management as one of the key functional areas of the business, the strategic managers set about examining what the latest state-of-the-art business processes are. They consider many current management views on the importance of supply chain management and find themselves focusing on the four stages of purchasing development [16], and the Strategic Transition Model [17]. Having understood this work, they select a number of "close fit" generic process and maturity models for their organisation. These models have a number of associated best practice models, such as those found in the MIT Process Handbook [18]. These models are then customised using information in their business plan (market segmentation, customer and supplier profiles etc.) resulting in version one of the companies personalised eBusiness Maturity Model.

The resultant model provides six levels of maturity ranging from Archaic, in which procurement is unplanned and no technological support is provided through to progressive in which every stage of the supply chain is supported by software systems. We shall not discuss the full set of criteria that separate each level from the next, instead we shall focus on a few key criteria separating Levels 3, 4 and 5. These

criteria are show in Table 1. In this table we can see that at Level 3, computer support systems are focused internally, at Level 4 the organisation is beginning to open its systems to the outside, whilst at Level 5 the organisations internal systems are fully integrated with those of their suppliers.

Table 1. EBusiness Maturity Model Levels (Procurement Best Practice)

Level	Key points
3: Latent	▪ Computerised Stock control ▪ Searchable supplier catalogues (manually updated)
4: Emerging	▪ Online ordering ▪ Automatic updating of suppliers catalogues
5: Open	▪ Full integration with suppliers systems allowing for real time updating of supplier catalogues ▪ Some goods are delivered directly by the suppliers and not held in stock

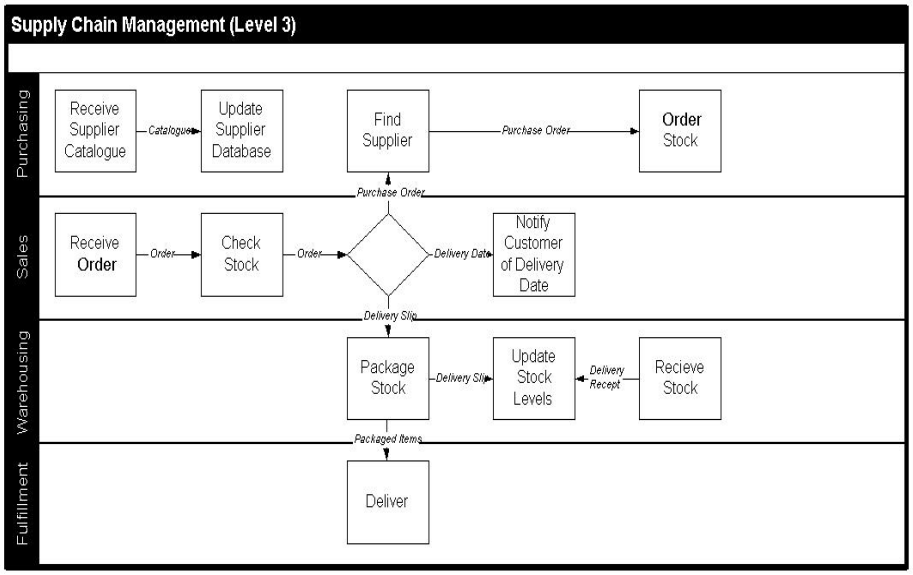


Fig. 3. Simplified (Level Three) Supply Chain Management Business Process

We must now select a suitable business pattern that will allow progression from Level 3 to 5 with the minimum of upheaval. This can be selected from a repository of best practice models [19], where they are aligned with the selected business process. The patterns are customised where necessary using goal-based process analysis [20]. When a suitable best practice model is not found, new ones are created [21].

The simplified business processes for Level 3, 4 and 5 organisations are shown on Fig. 3 to Fig.5. It can be seen from these figures that much of the business process remains unchanged between levels, and that we have started to modularise the

business in order to concentrate on individual parts of the process pattern [22]. In Fig. 3 we can see that there are four departments involved with processing each order and that there is no integration between the suppliers catalogue and the organisations internal catalogue. All updates are carried out manually, and therefore any searches on the catalogues may not reflect the most current data.

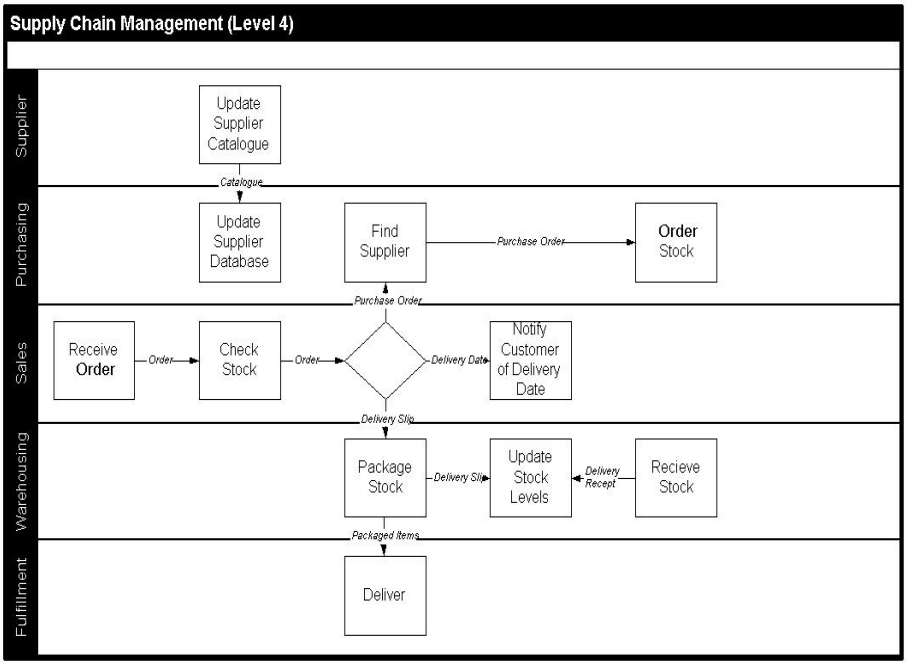


Fig. 4. Simplified (Level Four) Supply Chain Management Business Process

In Fig. 4 we can see our systems must integrate with those of our suppliers. This has implications for how we store the catalogue information internally, but it does not directly affect any other business process. You will recall (Table 1) that at this level we can also receive orders online. Again, this does not affect the internal operation of the system since these will appear as any other order in the sales department.

At Level 5 (Fig. 5) we see much more integration with the suppliers systems, some of which affect the functionality of existing business processes and consequently, the software supporting those processes.

If we suppose the organisation decides to enter the market at Level 3 but has a stated strategic goal to progress to Level 5, it is important that design decisions made when implementing Level 3 systems facilitate the progression to Level 5 without major system rebuilds. This has major implications for the software patterns chosen. We shall discuss this in a later section, but first we must consider how to measure progress through the maturity levels.

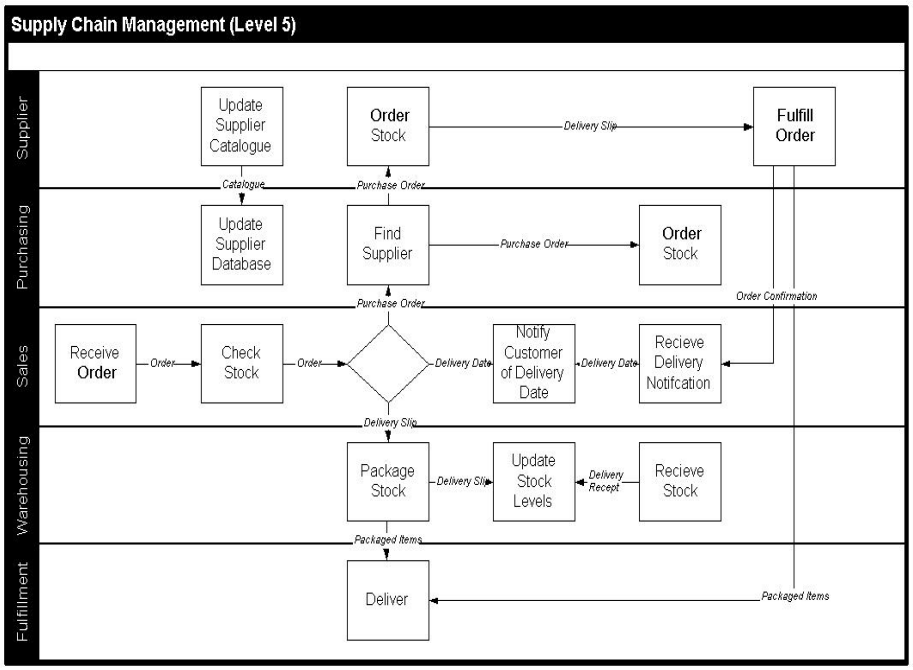


Fig. 5. Simplified (Level Five) Supply Chain Management Business Process

Table 2 Example Metrics for Supply Chain Management EBMM

Level	Requirements	Metrics
3: Latent	▪ Computerised Stock control ▪ Searchable supplier catalogues (manually updated)	▪ All stock level changes recorded and immediately visible on the system ▪ All supplier catalogues searchable by supplier name, product name, our product ID or supplier product ID ▪ Management of preferred supplier status
4: Emerging	▪ Online ordering ▪ Automatic updating of suppliers catalogues	▪ Orders placed via email & web interface as well as telephone, fax and post ▪ All preferred suppliers and 60% of other suppliers catalogues automatically added to our database without human intervention

5.2. Specifying Metrics Within the eBusiness Maturity Model

The next step is to identify metrics against each of the levels of the eBusiness Maturity Model, to enable strategic positioning of the organisation, and to assist in strategic planning for progression through the levels of maturity. Guidance for these

metrics is provided in the generic maturity model. However, it needs to be customised for the particular organisation. In this section we shall focus on the progression from Level 3 to Level 4 in our example above. That is, we shall focus on the integration of supplier catalogues with our system, for simplicity we shall not consider progression to other levels or the many other metrics that are applicable in the real world at each of these levels.

Management will identify suitable metrics for measuring performance against the levels of the maturity model. Table 2 shows the metrics selected for our example.

5.3. Identifying Software Requirements and Design Patterns

Having identified the metrics by which we can measure entry into each level of the maturity model, together with the strategic goals we are trying to achieve and the business processes to be followed, it becomes possible to allow software engineers to examine the strategy model and identify suitable software patterns. In doing so the software engineers will be able to estimate both the cost and the risk involved with entry to each of the levels and thus facilitate a strategic decision as to which level the company can safely move to, or, in the case of a start-up company, at which level it can start operating. For example, Fig. 6 and Fig. 7 below show UML Activity Diagrams that illustrate possible software patterns for the processing of supplier catalogues at each of Level 3 and Level 4 of our model. It can be seen from these diagrams that the Level 4 pattern is, very similar to the Level 3 process. Indeed, if we collapse each of the Level 3 steps from "retrieve product details" to "enter new/changed/details" into a single process called "Translate Format", we have an identical system. Therefore, selecting this software pattern (as opposed to others that may be available) should present minimal risks to future development. Furthermore, because the interfaces to the processes are the same we can run both of these patterns in parallel at Level 4, which only specifies 60% of non-key supplier catalogues need be integrated thereby requiring the Level 3 process to remain in place for the remaining 40%.

Now that software engineers have a set of requirements, metrics against which to measure the coverage of these requirements and suitable software patterns, they can start to build test harnesses against which system configurations can be evaluated. These test harnesses will ensure that each of the components in the software pattern behave as expected. Writing test harnesses could be the subject of a paper in its own right, we will avoid detail in this area for the sake of brevity, but encourage the reader to examine suitable literature on the subject [23, 24]. Armed with these test harnesses we are now ready to attempt to automatically configure the system as defined.

5.4. Automated component procurement and system building

The identification of suitable software patterns and the building of test harnesses provide the requirements for the component procurement stage. Since many functions of business are common it is likely that many of these requirements can be fulfilled by a number of existing components.

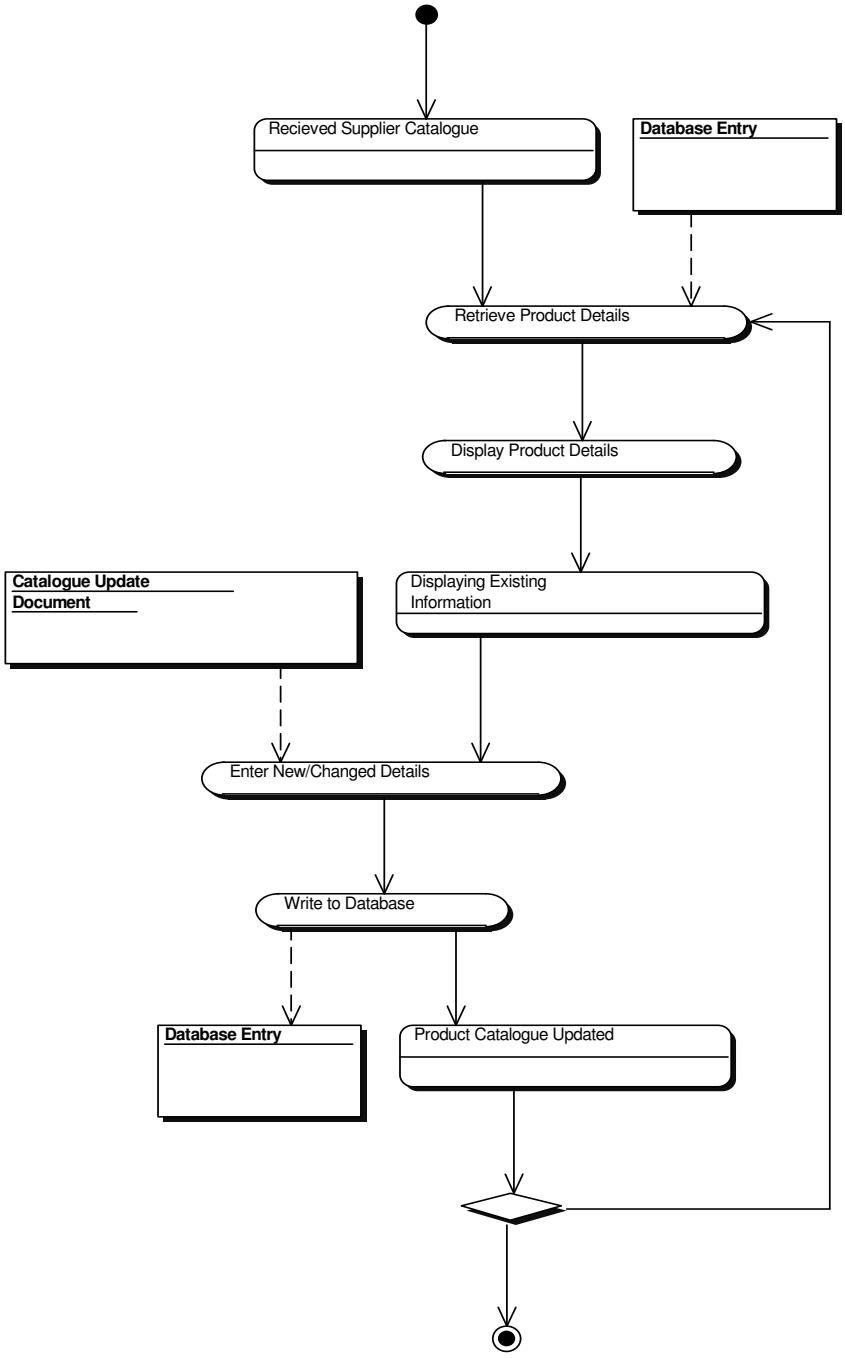


Fig. 6 Level 3 Supplier Catalogue Received Process

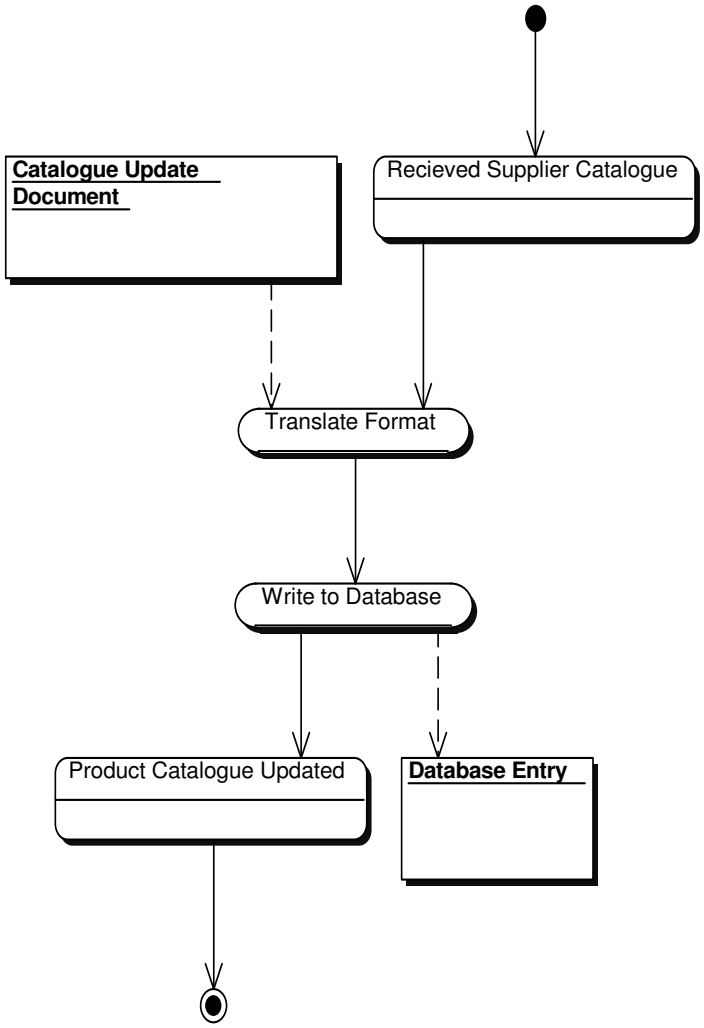


Fig. 7 Level 4 Supplier Catalogue Received Processes

Software engineers normally select these components based on current requirements, they then ”glue” them together to form complete software systems. However, the presence of a “requirements roadmap”, where current requirements are complemented by a set of likely future requirements, makes it possible to automate the selection of a suitable component. Where two components cover today’s requirements, we can asses to the likely fitness of each component for our future needs. This allows us to select the best fit for both today’s system and for the next generation. For example, at Level 4 in our example company we need to integrate suppliers’ catalogues with our system whilst at Level 5 we must directly place orders on suppliers systems. There may be

two components suitable for progression to Level 4. One of these components may provide translation from the suppliers' catalogue format to our internal format, whilst the second is also capable of translating internal purchase orders into a format understandable by the supplier. When building the system for Level 4, the extra functionality of the second component is not required, however, it will be required to progress to Level 5. In this way we can begin to consider future requirements when selecting components for use. The parameters and boundaries for component selection are provided by the results of the strategic positioning and planning using eBMM, combined with the ways in which component interfaces and negotiable parameters are defined at the supplier's side [25, 26]. These parameters will include such things as processing speed, price, service guarantees, and security level.

5.5. An Example System Build

As an illustration of component procurement and system build, consider a software component to handle the integration of an internal catalogue system with those of suppliers (a requirement of Level 4). There may be a number of components available to perform this action. How do we choose between these components? The process that the system builder follows is illustrated in Fig. 8 below.

Firstly, we use the customised eBMM to communicate the strategic goals of the organisation and the tactical plan of implementation to the software developers (Table 1). The tactical managers have identified the need to integrate the catalogues of 100% of preferred suppliers and 60% of other suppliers. Furthermore, it has been specified that we must integrate with 70% of non-preferred suppliers at Level 5. The developers also note that at Level 5 the system must be capable of placing orders directly with suppliers. This implies a two-way communications between organisations. The software developers first define a set of software design patterns that will facilitate the building of a software system to support these needs. These patterns define what components are needed, together with their inputs and expected outputs. Thus developers are able to build a set of unit and functional tests to ensure that components and groups of components perform as required [27].

Table 3. Component Facilities

Component	Preferred supplier formats supported	Other supplier formats supported	Additional Supplier formats supported	Two way translation	Speed rating
A	100%	92%	23	No	97
B	62%	75%	12	Yes	95
C	54%	61%	7	Yes	89
B + C	100%	95%	9	Yes	92

Using these components definitions the system builder can automatically query a directory lookup service, such as UDDI in order to discover a number of components that claim to be suitable. Let us suppose that there are three such components. The table above shows some of the facilities of these three components. These figures are

obtained from the component library and/or by testing the components in test harness created by the software engineers for this purpose. It can be seen from this table that there are only really two choices for this component. Either we use component A or we use a combination of component B and C. The selection of component A will allow us to build a Level 4 system immediately, however, the combination of B+C will require some intervention by the developers. Therefore, the system builder will select component A and notify the developers of the need to provide more details about how to reach Level 5.

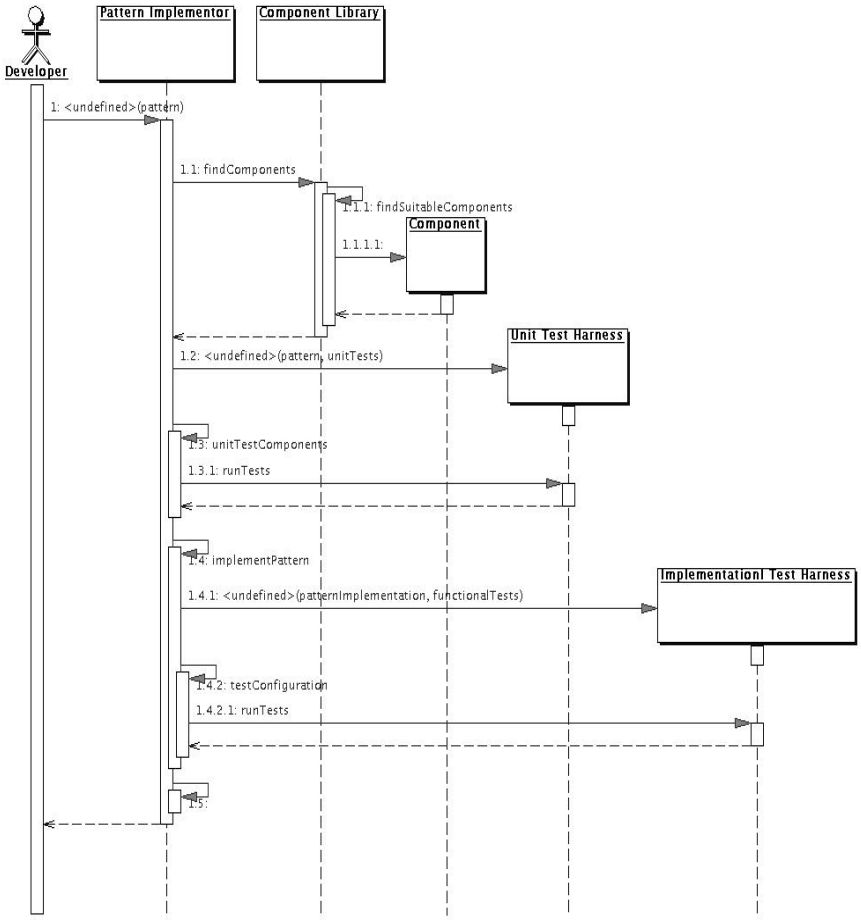


Fig. 8. System Build Sequence Diagram

6 Conclusions

To ensure continued alignment of software systems with the strategic changes in business operations we must enable to software to evolve rapidly. By utilising best practice business patterns and aligning them to best practice software patterns we can automate portions of the software build process. Thus we free developers to concentrate on solving problems that have never been solved before.

We propose the use of generic Ebusiness Maturity Models to communicate these best practices between organisations and individuals. The eBMM will be modified during strategic planning sessions to take into account factors specific to the company and its changing local environment. The EBMM indicates a set of tactical level metrics against which progress towards the strategic goals can be measured.

The identification of these metrics and strategic goals enable us to build a set of test harness that will verify the success or failure of a system build at various levels of abstraction. Since the system is capable of deciding for itself whether it meets its requirements we are therefore able to automate component procurement and system building. We see such an automated procurement as a core enabling factor for speeding up component-based software development, allowing the rapid development of flexible and scaleable systems to support the rapid growth of eBusiness organisations. Our approach ensures that automatically configured systems evolve with and match requirements as the organisation matures and grows.

References

1. Brown, A.W. and K.C. Wallnau, *The Current State of Component Based Software Engineering.* IEEE Software, 1998. **15**(5): p. 37--46.
2. Councill, B. and G.T. Heineman, *Definition of a Software Component and its Elements*, in *Component-Based Software Engineering*, G.T. Heineman and W.T. Councill, Editors. 2001, Addison-Wesley. p. 5--20.
3. Griss, M.L. and G. Pour, *Accelerating Development with Agent Components.* Computer, 2001(May 2001): p. 37--43.
4. Dellarocas, C. *The SYNTHESIS Environment for COmponent-Based Software Development.* in *8th International Workshop on Software Technology and Engineering Practice*. 1997. London: IEEE CS Press.
5. Schmidt, D.C., R.E. Johnson, and M. Fayad, *Software Patterns.* Communications of the ACM, 1996. **39**(10).
6. Heineman, G.T. and W.T. Councill, *Component-Based Software Engineering*. 2001: Addison Wesley. 818.
7. Nagaraj, N.S., et al., *Towards a Flexible Architecture*. 2002, INFOSYS.
8. Dellarocas, C. *Towards A Design Handbook for Integrating Software Components*. in *5th International Symposium on Assessment of Software Tools (SAST'97)*. 1997. Pittsburgh, P.A.
9. Gamma, E., et al., *Design Patterns: Elements of Resusable Object-Oriented Software*. 1995: Addison-Wesley.
10. Riehle, D. *Composite Design Patterns*. in *Object-Oriented Programming Systems, Languages and Applications (OOPSLA '97)*. 1997: ACM Press.
11. Beck, K., *eXtreme Programming Explained*. 1999: Addison Wesley. 190.

12. Hightower, R. and N. Lesiecki, *Unit Testing with JUnit*, in *Java Tools for Extreme Programming*, R.M. Elliot, Editor. 2002, John Wiley & Sons.
13. Bamberger, J., *Essence of the Capability Maturity Model.* IEEE Computer, 1997. **30**(6): p. 112--114.
14. Grant, S. *E-commerce for small businesses*. in *Innovation Through Electronic Commerce*. 1999. Manchester, UK.
15. Malone, T.W., et al., *Tools for inventing organisations: Toward a handbook of organizational processes.* Management Sciences, 1999. **45**(3): p. 425--443.
16. Reck, R.F. and B.G. Long, *Purchasing: A Competitive Weapon.* Purchasing and Materials management, 1988. **24**(3).
17. Cousins, P. and D. Marshall, *The Management of Supply as a Strategic Process*, in *Value Stream Management: Strategy and Excellence in the Supply Chain*. 2000, Pearsons Education Limited: Harlow, UK. p. 189--202.
18. MIT, *MIT Process Handbook*: MIT.
19. Russel, C.M., P.B. Barnsley, and M.R. holladay, *Taking Care Of Supply.* BT Technology Journal, 1999. **17**(4): p. 36--45.
20. Lee, J. *Goal-based process analysis: a method for systematic process redesign*. in *Organizational Computing Systems*. 1993. Milpitas, CA, USA.
21. Ould, M., *Business Processes: Modelling and Analysis for reengineering and improvement*. 1995, West Sussex, England: John Wiley and Sons.
22. Eriksson, H.-E. and M. Penker, *Business Modelling with UML: Business Patterns at Work*. 2000: John Wiley & Sons, Inc. 459.
23. Daley, N., D. Hoffman, and P. Strooper, *Unit operations for automated class testing*. 2000, University of Queensland: Queensland.
24. Harrold, M.J. *Testing: A Roadmap*. in *22nd International Conference on Software Engineering*. 2000.
25. Creech, M.L., D.F. Freeze , and M.L. Griss. *Using hypertext in selecting reusable software components*. in *Third annual ACM conference on Hypertext*. 1991. San Antonio, TX USA.
26. Jarzabek, S. *From reuse library experiences to application generation architectures*. in *17th international conference on software engineering on Symposium on software reusability*. 1995. Seattle, WA USA.
27. Hightower, R. and N. Lesiecki, *Java Tools for Extreme Programming*. 2002, New York: Wiley Computer Publishing. 516.