Solving Acquisition Problems Using Model-Driven Engineering

Frank R. Burton[1,2], Richard F. Paige[2], Louis M. Rose[2], Dimitrios S. Kolovos[2], Simon Poulding[2], and Simon Smith[1]

[1] MooD International, York Science Park, YO10 5ZF, UK
{frank.burton,simon.smith}@moodinternational.com
[2] Department of Computer Science, University of York, YO10 5GH, UK
{paige,louis,dkolovos,smp}@cs.york.ac.uk

Abstract. An acquisition problem involves the identification, procurement and management of resources that allow an organisation to achieve goals. Examples include through-life capability management (in the defense domain), and planning for the *next release* of a software system. The latter is representative of the challenges of acquisition, as solving the problem involves the assessment of the very many ways in which the different requirements of multiple heterogeneous customers may be satisfied. We present a novel approach to modelling acquisition problems, based on the use of Model-Driven Engineering principles and practices. The approach includes domain-specific modelling languages for acquisition problems, and uses model transformation to automatically generate potential solutions to the acquisition problem. We outline a prototype tool, built using the Epsilon model management framework. We illustrate the approach and tool on an example of the next release acquisition problem.

1 Introduction

An *acquisition problem* involves the identification, procurement and management of resources to achieve goals. Consider the following scenario.

> The National Lifeboats Institution is considering its next-generation capability. It has a limited budget, and must satisfy numerous stakeholders such as lifeboat operators, funding bodies, charity workers, and the general public. Its goal is to provide an efficient and effective lifeboat service that protects the public and saves lives. It may acquire improved capability through: better training of current operators, better equipment (e.g., newer lifeboats), better education, etc. It wants to find the most cost-effective capability solution.

This scenario is representative of acquisition problems in numerous domains, including defense, aerospace, logistics, software project management and law. Such problems are complex, and are known to be hard [1,2]. They are challenging for a number of reasons:

- they often involve multiple objectives (e.g., saving lives, minimising cost);
- they are heterogeneous, involving tradeoffs between different types of entities (e.g., better training versus better equipment);

– they are often dynamic: different solutions may be optimal or near-optimal at different times.
– there is rarely a single optimal solution.

Not only is determining optimal or near-optimal solutions to acquisition problems difficult, even understanding the acquisition problem may be challenging, in part because of the uncertainty and imprecision in the problem definition.

We contribute a general approach, based on Model-Driven Engineering (MDE) concepts and technologies, for modelling acquisition problems and calculating solutions that are optimal (Pareto optimality as defined in Section 2.1). The approach consists of a set of domain-specific languages (DSLs) for modelling acquisition goals and scenarios; these scenarios are then manipulated, by a chain of model transformations and model management operations, to produce solutions that are optimal. The advantages of using DSLs and MDE concepts and technologies for modelling and solving acquisition problems are multi-fold:

– Existing techniques used for solving acquisition problems are predominantly domain *dependent*: they rely on domain-specific models and algorithms for expressing how a capability matches a problem. MDE technologies and concepts offer a general, domain *independent* approach to solving such problem while still enabling a domain specific method of expressing the problem itself.
– In particular, the representation of problem concepts, goals and constraints using a DSL-based approach may be more accessible by domain experts.
– Acquisition problems conceptually reduce to manipulating graphs, where nodes typically represent goals and solutions, and edges represent dependencies. MDE technologies allow us to model and attempt to solve acquisition problems in their full generality.
– MDE principles and technologies allow us to abstract away from the complexity of calculating optimal solutions, and to modularise the calculation process. Model transformation, in particular, simplifies mapping (domain) models of problems to optimal solutions.

To illustrate the modelling approach, its novelties and limitations, we apply it to a representative acquisition problem: the software engineering *Next Release Problem (NRP)*. The acquisition objective is to find the 'best' customer requirements to satisfy in a new software release, while staying within budget. This problem, defined in detail in Section 2, is known to be NP-hard [3]. We use it to demonstrate that the MDE-approach can successfully model such complex problems, calculate optimal solutions, and also handle a more general version of the problem that includes dependencies between requirements. We also demonstrate that we can handle continuous variable requirements [2] and provide feedback and explanations of the results, via visualisations.

Beyond offering a solution method for NRP, the presented approach makes multiple generic contributions: the automatic generation of goal models from a partial goal model decomposition with high level descriptions of possible acquirable systems; the automatic evaluation of the generated goal models; calculation of multiple solutions that satisfy multiple stakeholder objectives and that are Pareto optimal, allowing decision-makers to make more informed acquisition; and solution visualisation via MDE tools, particularly EuGENia [4] and GMF [5].

The paper is structured as follows. In Section 2 we present related work and introduce the foundations of NRP, as well as the MDE techniques and technologies we use for our solution. In Section 3 we present our modelling approach, focusing on the DSLs to be used by engineers in representing acquisition goals and scenarios; we also outline the transformations used to produce optimal outputs, and explain how solutions are judged to be optimal. In Section 4 we apply the approach to an instance of NRP. We conclude in Section 5.

2 Background and Related Work

In this section we review the key previous work on acquisition problems, focusing on the Next Release Problem as a representative example. Acquisition problems are, formally, multi-objective optimisation problems. As described in this section we also discuss related work on NRP in more detail, then review the MDE technology that underpins our approach and tools.

2.1 Multi-objective Optimisation Problems

NRP, like other acquisition problems (e.g., Through-Life Capability Management [1]), is an example of a multi-objective optimisation problem. In contrast to single-objective problems, when multiple competing objectives exist, there is normally not a single 'best' solution. Instead, a solution may be the better at one objective but worse than other solutions at a different objective. A solution, X, is said to be *dominated* by another solution, Y, if Y is strictly better than X on at least one objective and as good as X on the remaining objectives. This leads to the notion of a *Pareto front* consisting of those solutions that are not dominated by any other solution. In other words, a solution on the Pareto front may only be bettered on a particular objective if we are willing to accept a worse score on another objective. These solutions on the Pareto front are considered to have *Pareto optimality*. (Deb in [6] presents a detailed explanation of multi-objective optimisation and the concept of a Pareto front.)

When applying search techniques to multi-objective problems it is necessary to quantify how 'good' all solutions are in terms of Pareto optimality, whether or not they are on the Pareto front. An appropriate method is to assign solutions on the Pareto front a *non-domination rank* of 0, then to temporarily ignore the solutions on this front in order to find a new Pareto front which is assigned rank 1, repeating this process until all solutions are ranked [7].

An example of a Pareto front for two objectives, together with solutions having non-domination ranks of 1 and 2, is shown in Fig. 1. For each solution on the Pareto front, it can be seen that no other solution dominates it, but no one solution is optimal for both objectives. Solutions on the Pareto front are the best representatives of different possible trade-offs beyond the competing objectives, and so when making decisions about a multi-objective optimisation problem, typically only the solutions on the Pareto front need be considered.

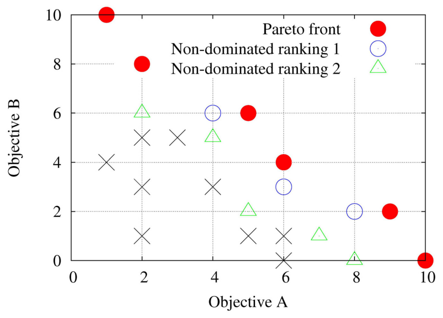


Fig. 1. Example of a Pareto front with the first two non-domination rankings marked. For each objective, a larger value represents a better solution.

2.2 The Next Release Problem

The Next Release Problem involves finding the ‘best’ set of customer requirements that will be satisfied in the next software release while staying within budget. In the NRP there are multiple customers with numerical weights to indicate their importance to the software company. Each customer requests one or more requirements and each requirement has an associated cost; requirements may also have causal dependencies. The original objective of the NRP was to select which customers should have their requirements satisfied. A consequence of framing the NRP with this objective is that solutions can be unbalanced: a small proportion of customers might have all of their requirements satisfied, and the requirements of some customers might be ignored. The approach to NRP presented by Bagnall et al [3] is limited in this manner, as it is very likely for some customers to have no requirements satisfied at all.

Later work [8–11] overcomes this limitation by having customers place weights of importance on each requirement and hence modifying the NRP so that requirements, instead of customers, are selected for inclusion. The objective in the modified NRP is to select which requirements to satisfy, in order to maximize the total customer satisfaction subject to the provided numerical weights. A further variant on the problem [9] considers the decomposition of a system, where a system is made up of components (typically software features) that work together, and focuses on selecting which new components will be added in the next release.

2.3 The Multi-objective NRP

In the multi-objective Next Release Problem (MONRP) [10], the cost becomes an objective rather than a constraint. The formulation of the problem in terms of customers and requirements is equivalent to variants of NRP described above that consider weights on each requirement indicating the importance of that requirement to the customer. There are now two competing objectives: to maximise the total customer satisfaction, and to minimise the cost to the company (the sum of all the costs of the selected requirements). The solution of the multi-objective NRP problem is therefore a Pareto front of points, each representing a different combination of requirements to include in the release.

Zhang [10] presents an approach to solving MONRP by calculating maximum customer satisfaction for all possible available resources, and hence demonstrates the effects of varying the budget on customer satisfaction. Zhang shows that it is possible to determine situations where increasing the budget only slightly can have a large effect on customer satisfaction, or that the budget can be significantly reduced with only minimal effect on customer satisfaction. However, this approach does not consider dependencies between requirements. In follow-up work, Zhang [2] identifies multiple challenges for MONRP, two of which are: providing sensible feedback and explanation of results; and handling continuous variable requirements (i.e. requirements that can be partially fulfilled, such as increasing the responsiveness of a system). Besides handling requirements dependencies, the modelling approach and tool we present in this paper addresses the latter completely, and the former partially (via visualisations). MDE technologies enable both of these contributions, as we discuss in more detail in the following section.

Various optimisation techniques have been applied to the MONRP, including greedy search, hill climbing, simulated annealing, genetic algorithms, NSGA-II, MoCell and Ant-based search [3, 8, 11, 12]. Since NSGA-II [7] has demonstrated acceptable results in a number of empirical studies on MONRP [10, 11, 13], we choose it as the basis of solution calculation.

2.4 MDE and Model Management

Our modelling approach and supporting tools are based on MDE principles [14] and technologies. The tools we present in later sections exploit Eclipse's EMF/Ecore for defining models and DSLs, and use the Epsilon platform [15] for automatically generating solutions to acquisition problems. Epsilon is a framework of task-specific languages for model management, and comprises a core language (EOL) upon which other task-specific languages are defined. These include a model-to-model transformation language (ETL) [16], a model migration language (Flock) [17], a model-to-text language (EGL), and a tool for generating GMF editors (EuGENia) [4].

The tool that supports our acquisition modelling approach is implemented as a set of DSLs, general model management code and model-to-model (M2M) transformations. Graphical editor support for the tool is provided via EuGENiA and GMF [5].

3 Modelling Approach

As part of a collaboration between MooD International and the University of York, we have developed a general-purpose modelling approach and tool designed to support acquisition problems. The approach is based on a set of DSLs that are used to model an acquisition problem as well as mechanisms and resources that can contribute towards fulfilling the problem's goals. We first describe the modelling approach, and then explain how the tool calculates solutions, using MDE techniques.

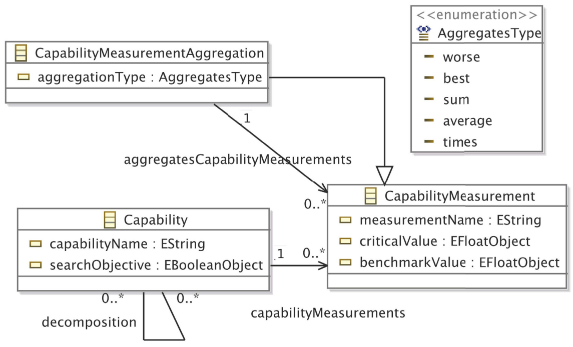
3.1 Concept and DSLs

The modelling approach is based on notions of goal modelling [18]. Goal models are normally applied to acquisition problems by first defining the top-level goals of what is required; these top-level goals are then decomposed into sub-goals until it is possible to identify a system or process that enables the goal to be fulfilled. In some cases, it is possible to determine whether a system or process *satisfies* a goal; in others, a looser notion, *satisficing*, is used to indicate that a system or process goes some way to meeting a goal.

Most approaches to calculating solutions to goal models terminate after identifying a complete goal model, i.e., one in which all goals are fulfilled. However, this is insufficient for solving acquisition problems in their full generality. For example, consider Through-Life Capability Management (TLCM) [1]: there are multiple ways for the same goals to be met by completely different supporting systems. Moreover, different acquisition strategies meet the goals to different degrees and have different costs.

Our modelling approach separates the modelling of the goals from the modelling of the systems and processes and how they interact. By underpinning the goal model and the system and process models (component models) with domain specific modelling languages, we can use MDE techniques to freely manipulate them. The use of MDE allows us to manipulate these models to automatically compose completed goals models. Additionally, with the aid of a multi-objective search technique, we can search for a Pareto front of completed goal models representing the various acquisition trade-offs to aid the acquisition decision makers. Goal models are a generic acquisition technique. This means that we can perform trade-off analysis on a very general class of acquisition problem. Being able to manipulate goal models using MDE techniques is the main enabling contribution that enables this. The application of our approach to NRP is purely illustrative.

The modelling of the high level goals and how they decompose is contained in the *Scenario Model*, which is captured in the Scenario DSL, illustrated in Fig. 2.



<<enumeration>>
AggregatesType
worse
best
sum
average
times
CapabilityMeasurementAggregation
aggregationType : AggregatesType
1
aggregatesCapabilityMeasurements
0..*
CapabilityMeasurement
measurementName : EString
criticalValue : EFloatObject
benchmarkValue : EFloatObject
Capability
capabilityName : EString
searchObjective : EBooleanObject
1
0..*
0..*
0..*
decomposition
capabilityMeasurements

Fig. 2. Scenario Metamodel in Ecore

The main element in the Scenario DSL is the concept of a *Capability*, which is a TLCM term for a Requirement or Goal [1]. Capabilities can be decomposed into sub-capabilities and can have associated quantitative measures. Capability measurements are either fulfilled directly by a *CapabilityProvision* (in a Component Model, see Fig. 3), or indirectly through the *CapabilityMeasurementAggregation* model element. This states how the measure is derived from other capability measurements. Capabilities can either be not fulfilled (typically by the critical value for the measurement not being reached), partially fulfilled (by the critical measurement being met), or completely fulfilled by the benchmark value being met.

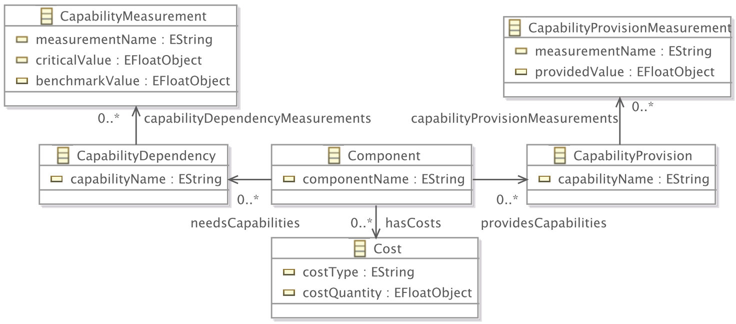

CapabilityMeasurement
measurementName : EString
criticalValue : EFloatObject
benchmarkValue : EFloatObject
0..*
capabilityDependencyMeasurements
CapabilityProvisionMeasurement
measurementName : EString
providedValue : EFloatObject
capabilityProvisionMeasurements
0..*
CapabilityDependency
capabilityName : EString
Component
componentName : EString
CapabilityProvision
capabilityName : EString
0..*
needsCapabilities
0..*
hasCosts
0..*
providesCapabilities
Cost
costType : EString
costQuantity : EFloatObject

Fig. 3. Simplified Component Metamodel in Ecore

The modelling of the systems/people/processes that can satisfy the goals in the *Scenario Model* are captured in *Component Models*; the abstract syntax is illustrated in Figure 3. The main element in the DSL is *Component*, which represents some system or process that can be acquired. Each component has a name, can provide multiple capabilities, can itself depend on multiple different capabilities, and can have multiple costs (this last element is essential, because some components will cost more in different contexts, e.g., a component used in a safety critical project may require more review by experts, and hence will be more expensive). There is an essential notion of coherency and completeness with components: if a dependency associated with a component is unsatisfied, that component cannot satisfy any other component with a provision.

An end-user models their goals (using the Scenario DSL) and the components (using the Component DSL). We then use MDE techniques to implement search-based algorithms to calculate optimal solutions; this is described in the next section.

3.2 Calculating Solutions to the Acquisition Problem

After specifying a set of goals and available components that can contribute to solutions, we want to calculate solutions to the acquisition problem. This is a search problem. However, it is a non-trivial search problem for reasons discussed earlier: components can contribute to multiple goals; a goal may have multiple possible components that can fulfil it; and, ultimately, the problem may have an arbitrary number of solutions. As such, we need to define a notion of what constitutes an *optimal* solution for such multi-objective problems. Our modelling approach thus calculates a Pareto front. The objectives for the Pareto front are set within the Scenario Model.

The goal model is constructed via a chain of M2M transformations that connect *Capability* model elements from the Scenario DSL with matching *CapabilityProvision* model elements from the Component DSL, thus creating a relationship between them in the Satisfied Scenario DSL. The *CapabilityDependency* in the Component DSL will also be matched with a *CapabilityProvision* in another Component. During this process, when a *Capability* or *CapabilityDependency* model element is connected to a *CapabilityProvision* model element, the provided values from the *CapabilityProvision-Measurement* model elements attached to the *CapabilityProvision* are compared with the critical and benchmark values from the *CapabilityMeasurements* model elements attached to the *Capability* or *CapabilityDependency* model element to determine how well the *Capability* or *CapabilityDependency* is satisfied. There is normally more than one *Component* which can satisfy a *Capability* and in some cases it may be preferable for a *Capability* to remain unfulfilled (e.g. to reduce costs). Consequently, the transformations are *stochastic*.

The overall approach is illustrated in Fig. 4. An end-user provides a single scenario model, multiple component models and a user interface model. The user interface model merely provides the settings for the search algorithm. The scenario model and component models are transformed via a M2M transformation into an intermediate DSL which express a correspondence [19] (Correspondence model). The correspondence model only holds information on *how* the models involved in the search can be *structurally connected* to each other. The next M2M transformation is stochastic and produces a completed correspondence model that captures *exactly one way in which* the models can be connected. This M2M transformation is executed multiple times to generate an initial population for the search method (described below). The completed correspondence model does not capture details such as measurements, costs, etc. A final M2M transformation is used to produce satisfied scenario models, which are the completed goal models that can be evaluated against their objectives. The transformation takes in the structural information from the completed correspondence model and the details from the original scenario and component models. The satisfied scenario metamodel is a composition of the scenario and component meta models with additional relationship for connecting them together.

To derive a Pareto front of solutions, we utilise a search method based on the multi-objective optimisation algorithm NSGA-II [7]. The starting point is the initial population, termed the ‘parent’ population, of different completed correspondence models produced by the stochastic process described above, and the search proceeds as a sequence of iterations, typically called ‘generations’, as follows. An ‘offspring’ population is created by applying a single-point crossover operator that swaps parts of two randomly-chosen models from the parent population to produce a new completed correspondence model. Each new model generated in this way is evaluated against the objectives using a M2M transformation to create a corresponding satisfied scenario model. The parent and offspring populations are then combined and the solutions ordered by non-domination rank. (The concepts of domination and non-domination rank are explained in Section 2.1) The better half of the combined population – the solutions with lowest non-domination rank – are chosen to form a new parent population. The algorithm continues in this way until a specified number of generations is reached. The output is a Pareto front of satisfied scenario models representing the different possible trade-offs between the objectives.

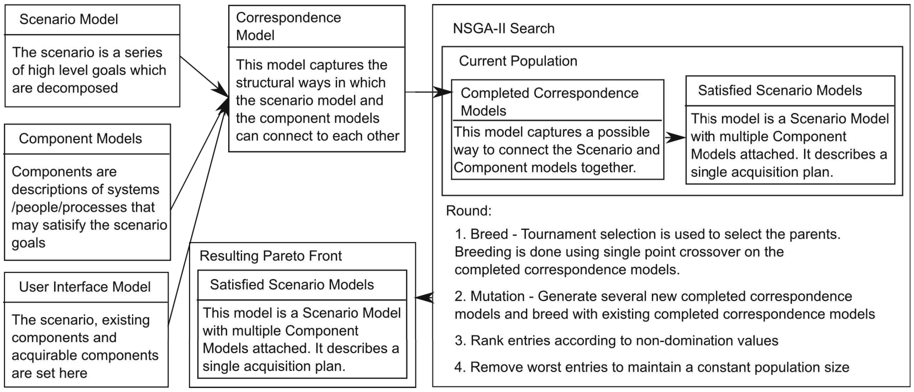

Scenario Model
The scenario is a series of high level goals which are decomposed
Component Models
Components are descriptions of systems /people/processes that may satisify the scenario goals
User Interface Model
The scenario, existing components and acquirable components are set here
Correspondence Model
This model captures the structural ways in which the scenario model and the component models can connect to each other
Resulting Pareto Front
Satisfied Scenario Models
This model is a Scenario Model with multiple Component Models attached. It describes a single acquisition plan.
NSGA-II Search
Current Population
Completed Correspondence Models
This model captures a possible way to connect the Scenario and Component models together.
Satisfied Scenario Models
This model is a Scenario Model with multiple Component Models attached. It describes a single acquisition plan.
Round:
1. Breed - Tournament selection is used to select the parents. Breeding is done using single point crossover on the completed correspondence models.
2. Mutation - Generate several new completed correspondence models and breed with existing completed correspondence models
3. Rank entries according to non-domination values
4. Remove worst entries to maintain a constant population size

Fig. 4. Solution architecture

The approach offers a number of novelties, including the following.

- *Automatic construction and evaluation of goal models.* The tool uses model management to take a partially decomposed goal model along with components models (representing systems/processes/people) and uses them to automatically generate multiple completed goal models, which represent different viable acquisition plans. The completed goal model is automatically evaluated by using the measures contained in the goal model and the component models used in its construction.
- *Trade-off support between stakeholder goals.* The approach has the ability to find the Pareto front between different stakeholder goals and the different involved costs when generating solutions to the acquisition problem of interest. This enables decision makers to consider trade-offs in the acquisition plans which is useful in cases where there are limited resources and multiple ways to achieve the same goals such as in Through Life Capability Management.
- *Solution visualisation.* The tool generates solutions to acquisition problems using the satisfied scenario DSL. Since a DSL is used, the model, containing the acquisition plan, can be visualised using MDE tools such as EuGENia [4] and GMF [5]. The solution visualisation is illustrated briefly in the next section.

4 Application to the Next Release Problem

We illustrate the modelling approach on instances of the Multi-Objective Next Release Problem (MONRP). First, by using a small representative example, we will demonstrate the modelling approach (and supporting tools, implemented using the Epsilon framework), the calculation of solutions, show how the approach can support problems that are not normally supported by NRP tools. Second, to explore scalability, we apply the approach to a much larger randomly generated dataset for a different instance of the MONRP.

4.1 Stock Control System Example

Our first example is based on the following scenario. A small shop is considering upgrading their existing stock control system. There are two main stakeholders: the shop manager (paying for the upgrade) and shop clerk (who uses the system). The shop manager has three requirements for the upgraded system: monthly reports on the shop's stock flow, email notifications for when stock needs to be reordered, and an automatic ordering system for new stock. The shop clerk also has three requirements: means to track stock (other than manual stock tracking), a better user interface for the system, and a requirement shared with the manager: that of automated stock ordering.

Producing email notifications and automatically ordering stock both depend on common code for determining when stock of an item is likely to run out. There are two different potential improved stock control systems: a cheaper barcode system and a more expensive RFID tag system which is easier to use and better at tracking stock. Such scenarios, where two different component satisfy the same requirement to different degrees, are not supported by existing NRP methods. Additionally, the manager's requirements are deemed more important than the clerk's. Moreover, the manager does not have enough money available to pay for all the upgrades.

4.2 Scenario Model

The first step is to model the acquisition problem in the Scenario DSL (Fig. 5). The Scenario DSL captures both the structure of the MONRP itself, the details of the two customers (manager, clerk) and their requirements. The objective is to calculate the weighted sum of the customer satisfactions. The capability *Next Release Problem* is the search objective and its measure is the sum of the shop manager's and shop clerk's satisfaction, which are weighted. The shop manager and shop clerk are modelled as capabilities which decompose into their requirements, also modelled as capabilities. A requirement whose satisfaction can be measured by a real value (as opposed to a requirement that is either fully met or not met at all) is called a continuous variable requirement. Continuous variable requirements are supported by the approach by giving each requirement a measure with a critical value of 0 and a benchmark value of 1. A

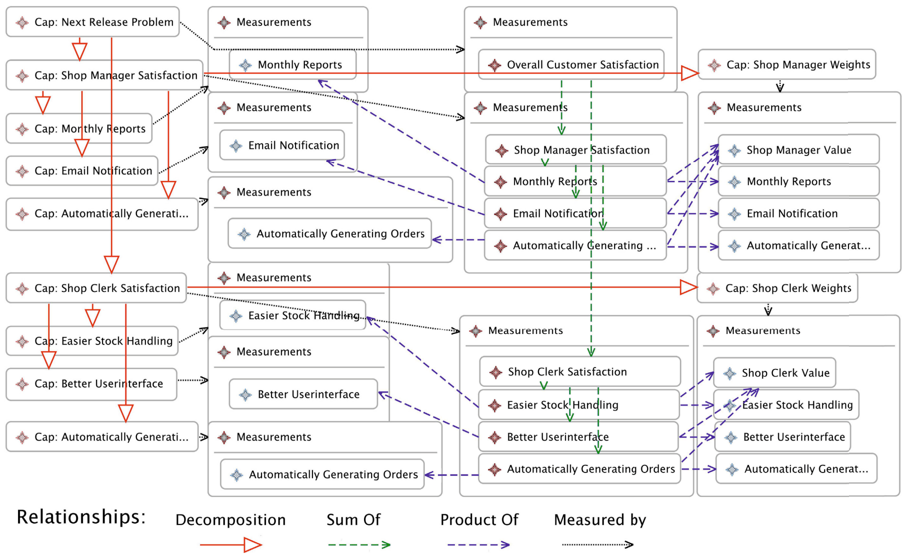

Cap: Next Release Problem
Cap: Shop Manager Satisfaction
Cap: Monthly Reports
Cap: Email Notification
Cap: Automatically Generati...
Cap: Shop Clerk Satisfaction
Cap: Easier Stock Handling
Cap: Better Userinterface
Cap: Automatically Generati...
Measurements
Monthly Reports
Email Notification
Automatically Generating Orders
Easier Stock Handling
Better Userinterface
Overall Customer Satisfaction
Shop Manager Satisfaction
Automatically Generating ...
Shop Clerk Satisfaction
Cap: Shop Manager Weights
Shop Manager Value
Automatically Generat...
Cap: Shop Clerk Weights
Shop Clerk Value
Relationships:
Decomposition
Sum Of
Product Of
Measured by

Fig. 5. Shop Case Study Scenario Model

slight modelling quirk is that because the Scenario DSL cannot hold numerical values, the weights have to be modelled in the Component DSL and therefore there are capability decompositions for the shop manager and the shop clerk which take into account the numerical weights for each [1]. The two weight components are set in the User Interface model as pre-existing components and so are always included in the generated goal models. The customer satisfactions have attached measures that are numerical and match the MONRP definition by using the multiplication and sum aggregations relationships.

4.3 Component Models

Examples of the component models for this acquisition problem are shown in Fig. 6. These examples demonstrate, for instance, that the *Barcode Scanning System* is dependent on the *Stock Management System*, will cost £120 to implement, and will provide *Easier Stock Handling*.

Based on these component models, our approach will generate a Pareto front of goal models using the Satisfied Scenario DSL. The search space in this illustrative problem is reasonability small so the tool quickly finds the optimal Pareto front of solutions. The Pareto front has been extracted from the result set using a model-to-text transformation in Epsilon; it is presented in Fig. 8. In this example, suppose that the shop manager has a budget of £250 and initially selects the solution that costs £230. This solution is

[1] A dedicated model element for this situation as now been introduced in the tool.

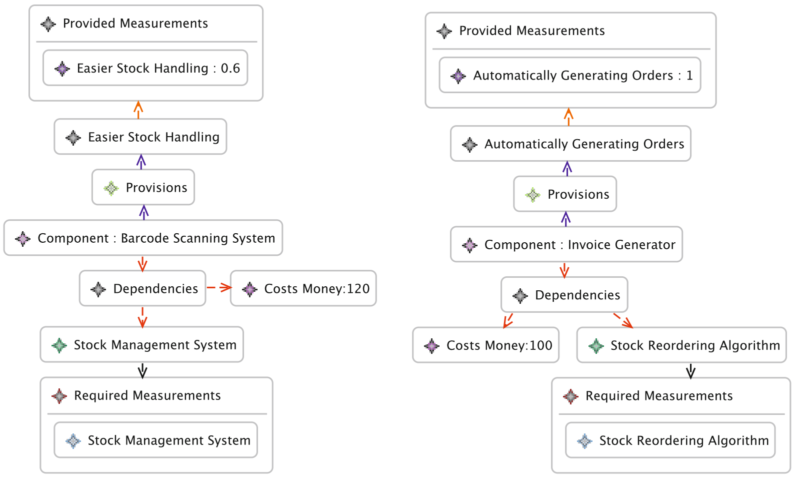


Fig. 6. Shop Case Study Example Component Models

visualised in Fig. 7, which demonstrates how the components are related to the capabilities in order to solve the problem. In the figure, *Capabilities* begin with "Cap:" and *Components* begin with "Comp:".

The shop clerk, seeing that the solution in Fig. 7 only satisfies one of his requirements, is unhappy. To resolve this, the search objectives in the Scenario Model are changed from the Next Release Problem to the two customer satisfactions. This generates a 3-dimensional Pareto front between the shop manager, shop clerk and the cost (Fig. 4.4). After some discussion between the shop manager and the shop clerk, they decide to choose a balanced solution, which gives them both a satisfaction of 0.7 for the cost of £275. The shop clerk agreed to have the extra £25 deducted from his pay.

This illustrates an advantage of an explicitly multi-objective approach to the problem. By deriving a set of potential solutions on the Pareto front, the stakeholders are presented with detailed information on which to base a discussion of trade-offs among the competing objectives.

4.4 Scalability Example

The previous example is small and illustrative; to better assess scalability, we used our approach and tool on a much larger, randomly generated MONRP problem with 5 customers, 50 requirements and 50 components. The customer and requirement weights are all randomly generated and normalised.

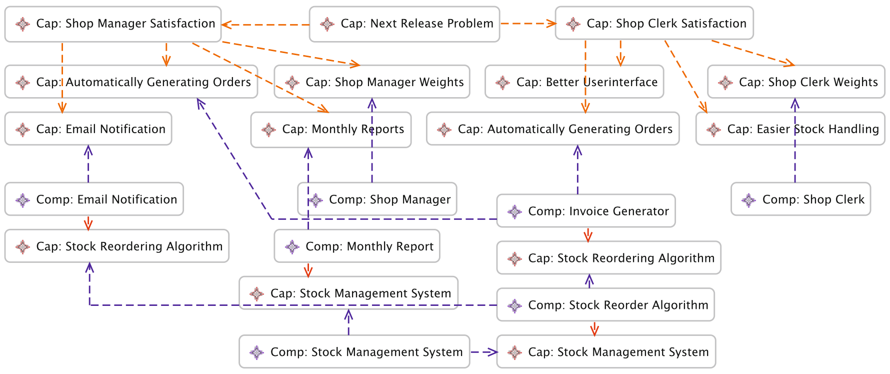
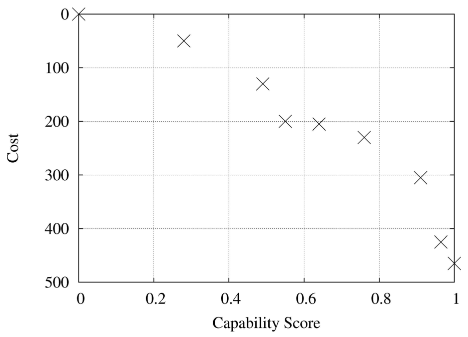

Cap: Shop Manager Satisfaction
Cap: Next Release Problem
Cap: Shop Clerk Satisfaction
Cap: Automatically Generating Orders
Cap: Shop Manager Weights
Cap: Better Userinterface
Cap: Shop Clerk Weights
Cap: Email Notification
Cap: Monthly Reports
Cap: Automatically Generating Orders
Cap: Easier Stock Handling
Comp: Email Notification
Comp: Shop Manager
Comp: Invoice Generator
Comp: Shop Clerk
Cap: Stock Reordering Algorithm
Comp: Monthly Report
Cap: Stock Reordering Algorithm
Cap: Stock Management System
Comp: Stock Reorder Algorithm
Comp: Stock Management System
Cap: Stock Management System

Fig. 7. Shop Case Study Example Solution



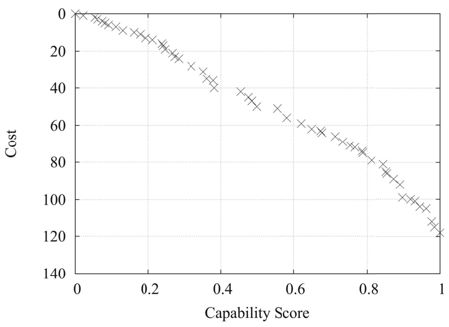


Fig. 8. Shop Case Study Pareto Front **Fig. 9.** Scalability Example Pareto Front

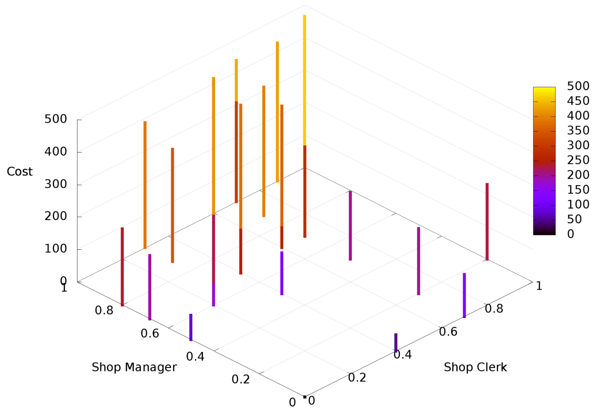


Fig. 10. Shop Case Study 3D Pareto Front

For each requirement a component is generated that provides that requirement. Additionally, there is a 50% chance the component provides another requirement and a 50% chance the component depends on a randomly chosen requirement. The generated Pareto front (shown in Fig. 9) shows that the relationship between customer satisfaction and cost begins near linear, as the software developer can start by selecting software features that are cheap to implement and are highly desirable to their customers. However, as more of the software features are implemented by the software developer, the remaining software features for the software developer to select from contains progressively less desirable and more expensive ones. The result of this is a slight curve in the Pareto front with the last 20% of the customer satisfaction begin twice as expensive to gain as the first 20% of the customer satisfaction.

For the search, a typical population size of 100 and generation count of 100 has being used. The runtime of our prototype on such a problem is approximately 5.5 hours. The randomly generated test data is larger then the largest real world problem in the NRP research field, the Motorola Problem [13]. Our tool is, however, several orders of magnitude slower than dedicated tools for MONRP, which can run in under 1 second [13]. Our focus up until now has not been on performance, but on genericity: our approach supports general acquisition problems and is not specifically tailored for MONRP. Some of the speed reduction we see is however inherent in using general-purpose modelling tools instead of bit strings. A significant reason for lower performance is that the M2M transformation languages are interpreted rather than compiled. In future work, the core M2M transformations may be reimplemented in a compiled language which should significantly reduce the execution time.

4.5 Contributions to the Next Release Problem

Our modelling approach and prototypical tool contribute several specific novelties to the Next Release Problem:

- *Trade-offs in software architecture.* By applying a principled modelling approach, we are able to represent the problem domain concept of customer requirements and the solution domain concept of components separately. In earlier work [9, 10], the customer requirements and components were treated identically. As a result, our approach permits situations in which multiple different components can fulfil the same customer requirement. This corresponds to trade-offs in software architecture and these are not supported by previous work on the NRP.
- *Continuous variable requirements.* A proposed research challenge for the NRP is the handling of continuous variable requirements [2], i.e. requirements whose satisfaction may be partial. For example, a requirement on a web server's response time. It could be a critical that the response time (the continuous variable) is under 300ms and desirable for the response time to be under 100ms. This is easily supported by our tool since it tackles in the problem in a conceptually cleaner way, whereas in previous work [3,8,10,11] on the NRP problem, a requirement can only be satisfied or not satisfied.
- *Visualisation of solutions.* Another research challenge for the NRP is explaining to the stakeholder why solutions are good [2]. By supporting the visualisation of

solutions to show how customer requirements are fulfilled by different components, our tool partially addresses this concern.

– *Tool Flexibility.* Our tool is generic and allows the end-user to change the problem definition. Here, for example, we have used this flexibility to produce a 3-dimensional Pareto Front of two stakeholder's satisfaction against the cost.

5 Conclusions and Further Work

The paper contributes a general approach, based on Model-Driven Engineering (MDE) concepts and technologies, for modelling acquisition problems and calculating optimal solutions. By doing so, the approach supports engineers in carrying out trade-offs between different acquisition strategies, some of which may never have occurred to the engineers, because they were unable to exhaustively identify all the potential solutions.

The paper discussed the typical challenges present in acquisition problems, using the multi-objective Next Release Problem as an example. In particular, we have demonstrated some of the typical problems associated with calculating solutions and presenting them to the end-user. We have presented our modelling approach, which is based on a number of domain-specific languages and is inspired by goal modelling, and explained how the languages are used to specify acquisition scenarios and potential capabilities, and algorithms for matching solutions against acquisition scenarios, in order to present solutions that are Pareto optimal. A prototype tool was presented, which automatically generates solution models. Both the automatic generation of goal models from the problem description, and the descriptions of the available systems, people and processes that can be acquired, as well as the generation of multiple instead of single goal models to show tradeoffs, are novelties of this work.

We then applied the modelling approach to concrete instances of the Next Release Problem, demonstrating that the approach and supporting tool can be used on both significant examples of the NRP, as well as more general NRP problems than those handled by other approaches (e.g., with dependencies between requirements). Overall, the approach that we have taken contributes a more generic capability for handling NRP problems; the price that we pay comes in the form of performance: because our tools are based on general-purpose MDE tools (i.e., Eclipse EMF, Epsilon) instead of dedicated NRP tools, our approach is less efficient; however, the approach does scale, as our example in the previous section demonstrates.

In future work, the tool will be applied to more general acquisition scenarios and on real world case studies. Since the work is primary motivated by the Through Life Capability Management (TLCM), we will enhance the tool to support multiple releases, similar to earlier work by Greer et al [8]. Additionally, we will manage the additional complexities that arise through TLCM, most notably the larger time scales involved (i.e., acquisition processes that take decades). We also intend to investigate mechanisms for performing sensitivity analysis, to check for robustness of acquisition plans.

Acknowledgements. This work is being sponsored by MooD International and the EPRSC (EP/F501374/1) under the Large Scale Complex IT System research programme.

References

1. McKane, T.: Enabling acquisition change - an examination of the Ministry of Defence's ability to undertake Through Life Capability Management. Technical report (June 2006)
2. Zhang, Y.-Y., Finkelstein, A., Harman, M.: Search Based Requirements Optimisation: Existing Work and Challenges. In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 88–94. Springer, Heidelberg (2008)
3. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The Next Release Problem. Information and Software Technology 43(14), 883–890 (2001)
4. Kolovos, D.S., Rose, L.M., Abid, S.B., Paige, R.F., Polack, F.A.C., Botterweck, G.: Taming EMF and GMF Using Model Transformation. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010. LNCS, vol. 6394, pp. 211–225. Springer, Heidelberg (2010)
5. Eclipse GMF - Graphical Modeling Framework, `http://www.eclipse.org/gmf`
6. Deb, K.: Multi-objective optimization. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, pp. 273–316. Springer, US (2005)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology 46(4), 243–253 (2004)
9. Baker, P., Harman, M., Steinhofel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: 22nd IEEE International Conference on Software Maintenance, ICSM 2006, pp. 176–185 (2006)
10. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 1129–1137 (2007)
11. Durillo, J.J., Zhang, Y.Y., Alba, E., Nebro, A.J.: A study of the multi-objective next release problem. In: 1st International Symposium on Search Based Software Engineering, pp. 49–58 (2009)
12. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Ant colony optimization for the next release problem: A comparative study. In: Second International Symposium on Search Based Software Engineering, pp. 67–76 (2010)
13. Durillo, J.J., Zhang, Y., Alba, E., Harman, M., Nebro, A.J.: A study of the bi-objective next release problem. In: Empirical Software Engineering, pp. 1–32 (2011)
14. Schmidt, D.C.: Guest editor's introduction: Model-driven engineering. Computer 39, 25–31 (2006)
15. Kolovos, D.S.: An Extensible Platform for Specification of Integrated Languages for Model Management. PhD thesis, University of York (2008)
16. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: The Epsilon Transformation Language. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 46–60. Springer, Heidelberg (2008)
17. Rose, L.M., Kolovos, D.S., Paige, R.F., Polack, F.A.C.: Model Migration with Epsilon Flock. In: Tratt, L., Gogolla, M. (eds.) ICMT 2010. LNCS, vol. 6142, pp. 184–198. Springer, Heidelberg (2010)
18. Lamsweerde, A.V., Dardenne, A., Delcourt, B., Dubisy, F.: The KAOS Project: Knowledge acquisition in automated specifications of software. In: Proceeding AAAI Spring Symposium Series, Track: Design of Composite Systems (1991)
19. Bézivin, J., Bouzitouna, S., Del Fabro, M., Gervais, M.P., Jouault, F., Kolovos, D., Kurtev, I., Paige, R.: A canonical scheme for model composition. In: Model Driven Architecture–Foundations and Applications, pp. 346–360 (2006)