Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 11 – 18

Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

System of Systems Acquisition Trade-offs

Frank R. Burton[a,b], Richard F. Paige[a], Simon Poulding[a], Simon Smith[b]

[a]*Department of Computer Science, University of York, YO10 5GH, UK*
[b]*MooD International, York Science Park, YO10 5ZF, UK*

Abstract

During System of Systems acquisition, organisations aim to satisfy their goals through the procurement of systems, people, training and processes. The most successful organisations undertake these acquisitions as efficiently as possible. For large organisations, a set of goals may be satisfied – to varying degrees – by any one of a large number of potential resource combinations, and so it is necessary to consider the trade-offs exhibited by each combination to find an effective System of Systems architecture.

In this paper, we present an approach, supported by tools, that combines techniques from the fields of goal-modelling and multi-objective optimisation to effectively explore these high-level organisational trade-offs. The decision maker can state the organisational problems and provide high-level descriptions of the existing and acquirable systems using a custom domain specific language. The tool then explores the space of potential resource combinations using a multi-objective genetic algorithm and provides the decision maker with a Pareto front of acquisition plans that represent the best trade-offs between the various goals and costs.

Furthermore, the technique and tool enables acquisitions to be scheduled to take an account of such factors as maintenance costs, existing system retirement dates, upfront costs for bringing new systems into service, the budgetary constraints of the organisation and how this affects the organisation's ability to satisfy its goals through the acquisition. The approach is evaluated on a realistic case study in which new systems are acquired to support a military scenario whilst considering the existing in-place systems and through-life implications.

Keywords: System of Systems, Acquisition, Trade-offs, Goal Modelling, Metaheuristic Search

1. Introduction

A System of Systems (SoS) is composed of many independently functioning, and often heterogeneous, systems. Such a SoS typically exhibits emergent capabilities not possessed by any of its individual systems. SoS tend to form naturally in many large organisations as the organisations evolve and their needs change[1,2].

There is a desire to manage the acquisition of SoS. One method used for this purpose is Capability-Based Planning, in which current capabilities – abilities that a SoS can provide to users – are assessed and then the required capabilities are determined[3]. The gap between the current and desired capabilities is mapped (in some way) to the procurement of new systems that fill the gap.

The capability-based planning approach used by the UK Ministry of Defence (MoD) is called Through-Life Capability Management (TLCM). The programs addressed by TLCM are representative of the problems of SoS acquisition. Firstly, the procured systems are typically heterogeneous, and this is illustrated by MoD's categorisation of systems according to Defence Lines of Development (DLoD): Training, Equipment, Personnel, Information, Doctrine and Concepts, Organisation, Infrastructure and Logistics. Secondly, the relationship between capabilities and the heterogeneous systems that compose them is known to be many-to-many[4], and so entirely different combinations of systems can result in the *same* military capability. For example, a capability to deliver ordnance at range can be produced by an artillery system with supporting personnel, training and ammunition; or by a jet aircraft with a trained pilot and an operational airfield.

In this paper, we present a technique and tool called CATMOS: Capability Acquisition Technique with Multi-Objective Search that combines approaches from the fields of goal-modelling and multi-objective optimisation to effectively explore high-level acquisition trade-offs. The technique presented in this paper extends our previous work[5] by supporting through-life planning, partial component satisfaction, complex aggregations in the goal-tree, upgrading systems, accumulation of capabilities, and by using a more effective metaheuristic search algorithm.

In section 2, we discuss existing approaches to capability-based planning, and provide an overview of the goal-modelling and multi-objective optimisation approaches that CATMOS uses. We describe the CATMOS technique and tool in section 3 with a focus on features that extend our previous work. We apply CATMOS to a realistic case study in section 4 in order to demonstrate these features.

2. Background and Related Work

2.1. Capability Management Approaches

TRAiDE[6] and CapDEM[7] are existing approaches that support capability-based planning by bridging the gap between the desired capabilities and potential compositions of systems. TRAiDE[6] is a workshop-based process that takes place over several months during which stakeholders discuss how to obtain the wanted capabilities. CapDEM is similar in that it aims to create a shared networking decision making environment for the stakeholders[7]. Both techniques involve the aggregation of a large number of data sources and the presentation of this information to the stakeholders. The CATMOS technique presented in this paper is designed to complement these techniques and to assist the stakeholders in understanding the large amount of information provided to them by exploring the trade-offs between potential solutions to the problem.

2.2. Goal Modelling

Goal modeling is an early requirements engineering technique[8,9] where requirements are modeled as goals. These goals can then be repeatedly decomposed into sub-goals until they are sufficiently low-level to be met directly by supporting systems, people or processes. Two of the main techniques are KAOS[8] and i*[9]. Capability-based planning has similarities with the KAOS approach, which considers goals to belong to the system as a whole rather than to individual internal agents. A capability is the same conceptual abstraction as a goal in KAOS, and both concepts can be repeatedly decomposed and can be eventually satisfied by systems and people.

The CATMOS technique uses a goal model as its starting point, but whereas goal-modelling techniques typically derive a single acquisition solution that satisfies the goal model, CATMOS analyses many different acquisition solutions in order to explore trade-offs between the competing goals as well as other objectives such as cost.

2.3. Multi-objective problems

Multi-objective problems have more than one objective to be met, and these objectives are often in competition with one another. In this situation, there is typically no single solution that is optimal for all the objectives: one solution can be better at one objective and another solution better in a different objective. Capability-based planning is an example of such a multi-objective problem: each capability may be considered to be an objective, and each feasible combination of systems is a solution.

A sophisticated approach to multi-objective problems is Pareto optimality, based on the concept of domination[10]. A solution, X, is said to be dominated by a solution Y if Y is better than the X for one or more objectives, and at least as good as X on the remaining objectives. The set of solutions that are not dominated by any other solution is called the Pareto front: these are the best possible solutions over the range of possible trade-offs between the objectives. The trade-offs illustrated by the Pareto front enable a more informed decision by the end-user: typically the user selects the solution on the Pareto front that represents the most desirable trade-off.

The CATMOS technique treats TLCM as a multi-objective problem, and applies multi-objective optimisation, in the form of a genetic algorithm, to identify the Pareto front and thus the optimal trade-offs between the competing capability objectives.

2.4. Domain Specific Modelling Languages

The implementation of CATMOS uses a Domain Specific Modelling Language (DSML) to describe both the capability requirements and potential systems that could be acquired to satisfy these requirements. A DSML is a dedicated modelling language defined for a specific purpose, and a metamodel defines the concepts described by the DSML and the relationships between these concepts. Not only is a DSML a convenient representation for both the problem and solution in CATMOS, it also permits textual and graphical user interfaces for the tool to be generated semi-automatically from the metamodel using XText[11], and the combination of EuGENiA[12] and GMF[13], respectively.

3. CATMOS Technique and Tool

3.1. Process Overview

The input to the CATMOS technique is a description of the problem, i.e. the capability goals and the acquirable systems that could satisfy those capabilities, in the form of a DSML. From this description, the tool derives a set of *different* solutions to the problem in form of satisfied goal models. This set of solutions forms the initial population of the genetic algorithm that combines and modifies the solutions in order to explore the solution space and identify a Pareto front of solutions that represent the best trade-offs between the capability objectives. The output from the tool is the Pareto front of solutions, and in the case of through-life capability management, a schedule of acquisitions.

The following sub-sections describe each part of this process in more detail.

3.2. DSML Problem Representation

Figure 1 shows the simplified metamodel of the CATMOS DSML that describes the problem in two parts: a goal tree and a set of components.

The goal tree defines the multiple capabilities (i.e. acquisition objectives) to be met. It is essentially a goal model decomposed using a technique such a KAOS, but for which the satisfaction of these goals has *not* yet been

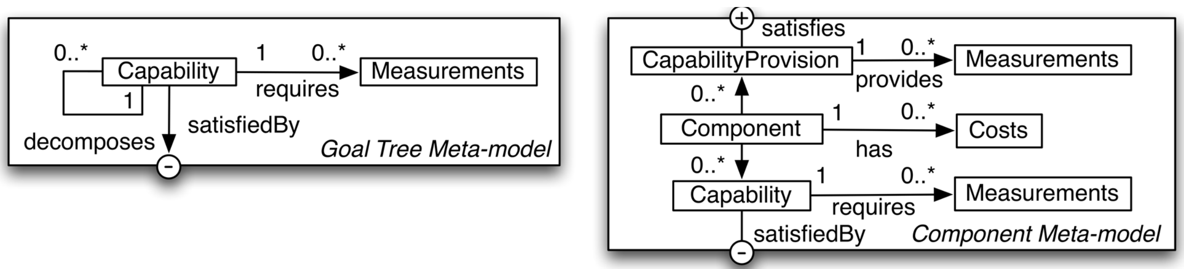

Fig 1. CATMOS Simplified Metamodel

determined. (We use the term *goal tree* to distinguish this from a *goal model* in which systems that satisfy that capabilities have been identified.)

The goal tree is described in terms *Capabilities* – such as 'Long Range Bombardment' – and associated *Measurements*. *Measurements* are real-world scalar or categorical values that potential solutions can be assessed against. For example, a capability of 'Long Range Bombardment' may have associated target measurements for the range and explosive force. *Measurements* specify both critical and benchmark values that indicate thresholds for minimal and maximal satisfaction of the *Capability*, respectively. One or more of the capabilities in the goal tree are marked as the high-level objectives to be optimised by CATMOS.

Capabilities can also be decomposed into smaller sub-capabilities. These decompositions can be either functional or area based. A functional decomposition splits a capability into smaller capabilities that, when put together, satisfy the capability. An area decomposition is used when the same capability needs to be carried out in highly diverse environments – for example, at sea and also at high attitude – but systems that meet the capability in one environment will typically not work in the other.

The set of components define the possible acquisitions that could be used to satisfy the capability objectives. *Components* represent acquirable systems, people, and processes. Each *Component* has a number of *CapabilityProvisions* that state the *Capabilities* that the *Component* is able to satisfy. The *CapabilityProvisions* have *Measurements*, which are the actual real world measurements associated with that *Component*.

A particular feature of CATMOS is that it permits the modeling of *Components* which themselves depend on other *Components* to operate. This is represented by allowing *Components* to optionally require *Capabilities* that must be satisfied before the *Component* can be used.

3.3. Deriving Satisfied Goal Models

The next step in the CATMOS process is to automatically derive a set of different *satisfied* goal models, i.e. combinations of components that satisfy the capabilities defined in the goal tree. This set will be the starting point for the multi-objective optimisation in the next step.

This is achieved by attempting to satisfy all the unsatisfied *Capabilities* by including *Components* with the relevant *CapabilityProvisions* and connecting the *CapabilityProvision* and the *Capability* together. A *Capability* is unsatisfied if it is not being satisfied by a *CapabilityProvision* and it does not decompose into other *Capabilities*. Including *Components* can add more unsatisfied *Capabilities* to the goal-model from their optional dependencies. Once all the *Capabilities* are satisfied the goal model is complete.

However for any one unsatisfied *Capability,* there may be more than one choice of *CapabilityProvision* that satisfies it, and so a *CapabilityProvision* is selected at random. This can lead to dead-ends in the derivation if a particular sequence of random choices reaches a point at which there are no free *CapabilityProvisions* that can satisfy a *Capability*. This problem is avoided in CATMOS by using a backtracking parser that can undo the derivation; SWI-Prolog[14] is used for this purpose.

Since the derivation of goal models is stochastic, a set of *different* goal models is constructed by simply repeating the derivation process multiple times.

3.4. Optimising the Goal Models

For any reasonably complex problem, the satisfied goal models produced by random derivation described above are unlikely to be optimal or even close to the Pareto front. Moreover, the small set of goal models will not constitute an exhaustive exploration of the space of possible solutions: such an exhaustive exploration will typically be computationally infeasible. Therefore, we apply metaheuristic search to optimise the solutions by performing a guided (but not exhaustive) exploration of the solution space, using the randomly derived goal models as a starting point.

The NSGA-II[15] algorithm, a state-of-the-art genetic algorithm for multi-objective problems, is used for this purpose. Genetic algorithms are inspired by Darwinian evolution: over a series of generations, 'parent' solutions are bred and mutated to form new 'child' solutions and the best (or 'fittest') of these solutions survive to the next generation.

To facilitate the use of NSGA-II, each goal model derived in the previous step is converted to a more amenable representation in the form of a sequence of tuples, <sourceComponent, capability, targetComponent>. Each tuple represents a connection between a *Component* through its *CapabilityProvision* to a *Capability* on another *Component* or to a *Capability* on the goal-tree. The set of randomly derived goal models in this new representation forms the initial population of parents in the genetic algorithm.

Two genetic operators create new 'child' goal models during each generation. Firstly, a crossover operator combines two parent goal models by randomly choosing a boundary between the tuples and taking tuples from one parent on one side of the boundary and tuples from the other parent on the other side of the boundary to form a child goal model. If the two parts of the new child model – one from each parent – do not connect together, it is repaired by randomly connecting any unsatisfied *Capabilities* and unused *CapabilityProvisions*. Secondly, a mutation operator deletes tuples at random in order to encourage the creation of smaller (and therefore less costly) acquisition plans.

The 'fitness' of new child goal models is evaluated by considering to what extent the critical and benchmark *Measurements* of each capability objective are met in the model. This requires the status of every *Component* and *Capability* in the tree to be analysed. *Components* only provide their *CapabilityProvisions* once their dependencies have being satisfied; if so, the measurements of each *CapabilityProvision* of the *Component* are passed up to the connected *Capability*. The status of a *Capability* is evaluated as the average level of fulfillment of its measurements. The status of each high-level capability objective is used by the genetic algorithm to determine the domination of the new child goal model. The algorithm retains the least dominated solutions in each generation, and over time this selection pushes the population of goal models towards Pareto optimality.

3.5. Advanced Features

When representing, deriving, and optimising goal models, CATMOS employs a number of advanced features that permit more realistic representations of the problems and potential solutions.

Partial Dependency Satisfaction: *Components* are permitted to operate when their required *Capabilities* are only partially specified. An embedded programming language called Lua[16] is used to implement this feature: all *Measurements* can have a Lua script attached to them that alters that *Measurement's* provided value based on the level of satisfaction of the *Components'* required *Capabilities*.

Complex Goal-tree Decomposition: Similarly, *Measurements* on *Capabilities* that decompose into smaller *Capabilities* can have Lua scripts attached to them allowing the decision maker to specify how to aggregate the satisfaction of the children *Capabilities* into the satisfaction of the parent *Capability*.

System of Systems Properties: This feature enables the capture of global properties across the entire SoS. For example, a property to be optimised may be the weight of an aircraft, which is calculated by summing the weights of all the constituent systems. The evaluation of such properties is achieved by allowing Capabilities to be marked as standalone: this indicates that they are not satisfied by *CapabilityProvisions* but instead Lua scripts are specified to calculate the global property.

Capability Upgrades: The acquisition of a new system can improve the abilities of an existing system. A *Component* can have *CapabilityUpgrade* which specifies that it adds to, modifies, or deletes the capabilities of another target *Component*.

Capability Accumulation: A common occurrence in SoS acquisition is that rather than acquiring just one instance of a system, acquiring multiple instances of the system produces more capability. For example, twenty fire trucks can put out much larger fires than a single fire truck. This feature is supported by selecting a *Measurement* to be accumulated on a *Capability* on multiple satisfactions.

3.6. Through-Life Scheduling

The CATMOS tool will normally produce a Pareto front of the desired capability objectives against the costs without consideration of the timing of acquisitions. An alternative option is to specify through-life planning information such as the organisation's budget, the in-service and acquisition dates for existing and acquirable *Components* and stating on *Capabilities* when they should start and stop.

In this case, a custom algorithm is run to schedule the goal model within the budgetary limitations before its fitness is evaluated during multi-objective optimisation. This custom algorithm is designed for performance rather than completeness since it must be run many times during the optimisation process. It performs a single pass in chronological order of the capabilities and finds, for each capability, the component that satisfies it for the longest time and schedules both it and its dependencies to meet the capability when it is first required. If the capability is still not fully satisfied for the whole time period then the next best satisfying component is scheduled in a similar manner and so on.

The scheduled goal model is evaluated each time a *Component* enters or leaves service and when *Capabilities* start and stop being required in order to determine the average satisfaction of the *Capabilities* over time.

4. Case Study

The case study presented here demonstrates the ability of the CATMOS technique and tool to solve a realistic acquisition scenario by optimising a plan for through-life capability across multiple objectives.

4.1. Problem Description

The task is to secure a forward base in hostile territory to cut off enemy supplies from crossing a river. The three high-level objectives are: clearing a route to the site of the new forward base, holding the forward base, and preventing the enemy from transporting supplies across the river. These objectives have been further decomposed and this decomposition is shown in figure 3.

There is an existing system of systems already in place for this task that consists of 13 system types, including Vector Vehicle Fleets, Troop Regiments, L118 Light Gun & the Global Hawk. There is also scope to acquire an additional 3 types of systems over the next couple of years.

The existing and acquirable systems are defined using the domain specific modelling language described in section 3.2. An illustrative selection of these definitions is shown in figure 4. For example, the Leyland DROPS Fleet has being defined to provide the 'Supply Goods' capability, which has a single measurement of the weight of goods it can transport. Every fleet of Leyland DROPS costs £3 million to acquire.

An example of a capability definition – a decomposed capability in the goal tree that must be satisfied – is 'Supply Goods'. The capability has a start and end date which are set to beginning and the end of the scenario respectively and a single measurement 'Goods Tons' that is set to be an accumulation. So for every 'Leyland

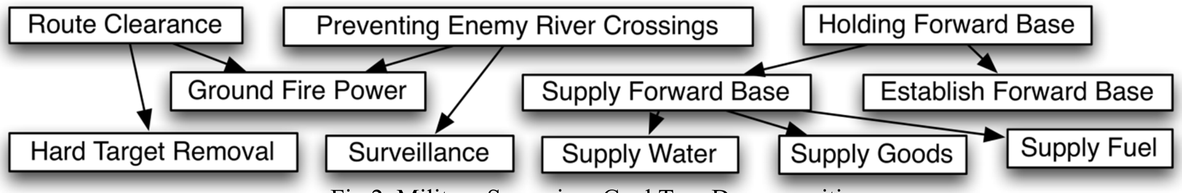

Fig 2. Military Scenario – Goal Tree Decomposition

```
Component "Leyland DROPS Fleet" { //10 Leyland DROPS in a Fleet
          CapabilityProvision "Supply Goods" { reuse 1 Measurement "Goods Tons" { providedValue 40.0 } }
          Cost Money 3.0 } //3 Million Pounds
Component "SAS Training" { CapabilityUpgrade "Better Training" {
          targetComponent "Troop Regiment" CapabilityChange "mod" { //Modify Existing Capabilities
          CapabilityProvision "Ground Fire Power" { reuse 1 Measurement "Ground Fire Power" { providedValue 1500.0 } }
          CapabilityProvision "Surveillance Static Targets" { reuse 1 Measurement "Surveillance Static Targets" { providedValue 5.0 }}
          CapabilityProvision "Surveillance Moving Targets" { reuse 1 Measurement "Surveillance Moving Targets" { providedValue 5.0 }}
          }} Cost Money 8.0} //£8 Million
Capability "Supply Goods" { accumulation "Goods Tons" startDate "01/01/2014" endDate "31/12/2016"
          Measurement "Goods Tons" {  criticalValue 0.0 benchmarkValue 120.0 } }
Component "L118 Light Gun Fleet" { //10 Artillery Pieces
          CapabilityProvision "Hard Target Removal" { reuse 1 Measurement "Hard Target Removal" { script "output = LightGun()" } }
          Capability "Light Gun Tow" {} Capability "L118 Service Contract" {} Cost Money 5.0 } //£500,000 each
ExistingComponent "L118 Existing Service Contract" 1 startDate "01/01/2014" endDate "01/01/2015"
```

Fig 3. Textural Extracts From the Domain Specific Language

DROPS Fleet' that is acquired, the satisfaction of the capability will go up.

In the scenario, it is currently not possible to approve the acquisition of additional troops for political reasons. However, a SAS head training officer suggests an alternative of providing SAS training programmes to regular troops. This is modelled using the 'SAS Training' component and specifying a capability upgrade that targets a 'Troop regiment' and then modifies the values it provides. This is an example of the capability upgrade feature of CATMOS described in section 3.5.

The existing L118 Light Guns that provide 'Hard Target Removal' capability, all require an active service contract to keep them maintained and functioning. However, the existing contract is about to expire and will need to be renewed to keep the L118 Light Guns functioning. This is specified by a 'L118 Service Contract' capability dependency in the definition of the existing L118 Light Guns component, and by setting through-life information on the 'L118 Existing Service Contract' component that states when the existing contract ends. The acquisition budget for the case study has being specified as 4 payments totaling £185 million over the three year period.

4.2. Current Capability Assessment

Using CATMOS to provide an assessment of the current capability (i.e. without making any new acquisitions) shows that the *Route Clearance* capability is only partially satisfied at 71%, the *Holding Forward Base* capability is almost completely satisfied at 93%, and the *Preventing Enemy River Crossings* capability is unsatisfied. For the latter capability, the current SoS can only stop 65% of enemy river crossings but the mission success requirement is that at least 75% of enemy river crossings are stopped.

A review of the generated goal model shows that the *Route Clearance* capability is limited because there is insufficient ability to remove hard targets along the path, insufficient surveillance information on the enemy and not enough ground fire power provided by the troops.

Increasing hard target removal capability can be done in three ways: acquiring additional L118 Light Guns, keeping existing L118 Light Guns in service by renewing the contract, or acquiring new MQ-9 Reapers UAVs. The L118 Light Guns are cheaper for the firepower they provide; however, the MQ-9 Reapers also provide additional surveillance information. Increasing ground firepower cannot be done by recruiting more troops for political reasons, but it could be achieved by the SAS head training officer's plan discussed earlier, or by acquiring more armoured vehicles in form of Mastiffs and Vectors.

For the high-level capability of *Preventing Enemy River Crossings*, the limiting factor is not the ground firepower but instead insufficient surveillance for detecting enemy troop movements. This could be resolved by providing

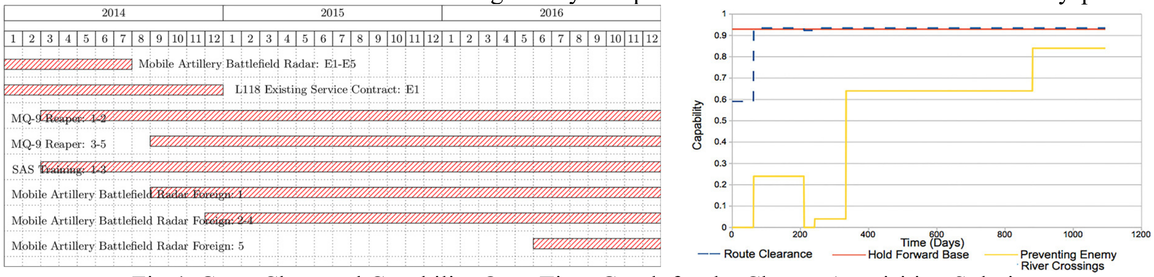

Fig 4. Gantt Chart and Capability Over Time Graph for the Chosen Acquisition Solution

better training to the troops on the ground, or by acquiring UAVs with surveillance abilities such as the MQ-9 Reapers and Global Hawks.

4.3. Through-Life Acquisition Plan

The assessment of current capability showed that the three high-level objectives were only partially satisfied, with one – *Preventing Enemy River Crossings* – entirely unsatisfied. A number of potential acquisitions were discussed that could improve capabilities. However, it is a very complicated task for the decision maker to decide which of these acquisitions to make, and when to make them, so as to maximize all three high-level capability objectives while simultaneously accommodating the budgetary constraints and the potential obsolescence of components over time.

To demonstrate how CATMOS can assist the decision maker, the tool was used to explore trade-offs in through-life capability over the next three years of the scenario. CATMOS identified three solutions on the Pareto front that illustrate different trade-offs. The first solution has average (over the three years) capabilities of (91.31%, 92.75%, 24.49%) for *Route Clearance, Holding Forward Base,* and *Preventing Enemy River Crossings* respectively; the second, (91.28%, 92.75%, 43.22%); and the third, (91.29%, 92.75%, 14.00%). Since the main objective is *Preventing Enemy River Crossings,* the decision maker is likely to prefer the second solution.

Having identified a suitable trade-off on the Pareto front, the decision maker can view a corresponding goal model explaining the solution to the problem, a Gantt chart showing the acquisition schedule, and a capability-over-time line graph that identifies any capability gaps. Figure 4 shows the Gantt chart and capability-over-time graph for the second solution derived by CATMOS.

5. Conclusions

In this paper, we have contributed a technique and tool, CATMOS, for supporting decision makers in the management of complex SoS acquisition problems. The technique has been demonstrated using a case study of a realistic military scenario and has been shown to support some of the complex acquisition questions and trade-offs that arise during SoS acquisition.

References

1. W. Owens, The emerging US system-of-systems, Tech. rep., DTIC Document (1996).
2. M. Maier, Architecting principles for systems-of-systems, Systems Engineering 1 (4) (1998) 267–284.
3. P. K. Davis, Analytic Architecture for Capabilities-Based Planning, Mission-System Analysis, and Transformation, 2002.
4. Y. Yue, M. Henshaw, A Holistic View of UK Military Capability Development, Defense & Security Analysis 25 (2009) 53 – 67.
5. F. R. Burton, R. F. Paige, L. M. Rose, D. S. Kolovos, S. Poulding, S. Smith, Solving acquisition problems using model-driven engineering, in: Modelling Foundations and Applications, Springer, 2012, pp. 428–443.
6. G. Symes, A. J. Daw, On the use of information management in the context of Through Life Capability Management (TLCM), Journal of Naval Engineering 45 Book 2.
7. W. Robbins, S. Lam, C. Lalancette, Towards a collaborative engineering environment to support capability engineering, in: Proceedings of INCOSE, 2005, pp. 10–14.
8. A. Lamsweerde, A. Dardenne, B. Delcourt, F. Dubisy, The KAOS Project: Knowledge acquisition in automated specifications of software, proceeding AAAI Spring Symposium series, Track: Design of composite systems (1991).
9. E. S. K. Yu, Towards modeling and reasoning support for early-phase requirements engineering, Requirements Engineering, IEEE International Conference on 0 (1997) 226.
10. K. Deb, Multi-objective optimization, in: E. K. Burke, G. Kendall (Eds.), Search Methodologies, Springer US, 2005, pp. 273–316.
11. S.Efftinge, M. Völter, oAW xText: A framework for textual DSLs, in: Workshop on Modeling Symposium at Eclipse Summit, Vol. 32, 2006.
12. D. Kolovos, L. Rose, S. Abid, R. Paige, F. Polack, G. Botterweck, Taming EMF and GMF using model transformation, Model Driven Engineering Languages and Systems (2010) 211–225.
13. Eclipse graphical modeling project, http://www.eclipse.org/modeling/gmp/.
14. J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, SWI-Prolog, Theory and Practice of Logic Programming 12 (1-2) (2012) 67–96.
15. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
16. R. Ierusalimschy, L. H. De Figueiredo, W. Celes Filho, Lua-an extensible extension language, Softw., Pract. Exper. 26 (6) (1996) 635–652.