Towards a Software Product Line for Automated Transportation Systems

Danny Weyns, Alexander Helleboogh, Tom Holvoet
DistriNet Labs
Katholieke Universiteit Leuven, Belgium
Contact: Danny.Weyns@cs.kuleuven.be

Kurt Schelfthout, Wim Van Betsbrugge
Egemin International nv.
http://www.egemin.com/

Abstract

Since the seventies, Egemin has successfully been providing full life cycle support for automated transportation systems (ATS) used to automate internal logistics in production and warehousing environments. With increasing customer demands a number of considerable problems related to the company's engineering practice came to the surface. Examples are duplication of solutions and erosion of the software and its architecture. With the introduction of a software product line (SPL) Egemin aims to shift the focus of its activities from developing individual products (ATS for clients) towards the development of a repository of core assets for a family of ATS supported by an explicitly defined production process. The objective is to establish planned reuse aiming to improve the quality of the software and ultimately to increase productivity. A particular challenge with the introduction of the SPL is support for runtime variability that allows an ATS to dynamically adapt its behavior to changes in the operating environment. In this paper, we outline how Egemin is introducing a dynamic SPL for ATS. We give particular attention to support for runtime variability.

1 Introduction

Egemin is a leading company that provides among other solutions, full life cycle support for automated transportation systems (ATS). An ATS uses multiple automated guided vehicles (AGV) to transport loads in an industrial environment, see Fig. 1. Such systems can be used for distributing manufactured products to storage locations or as an interprocess system between various production machines.

The current practice in Egemin is to develop individual products. For each new ATS, software from previous projects is reused opportunistically. There is no explicitly defined process to deal with product development. This approach has been giving rise to a number of considerable problems, such as duplication and inconsistencies of solutions. A particular onerous problem is the erosion of the architecture and software components. With the introduction of a software product line (SPL), Egemin aims to shift the focus of its activities towards the development of a common core asset repository for a family of ATS and the definition of an explicit production process. This way, Egemin aims to establish planned reuse to improve the quality of the software, to better streamline the development activities, and ultimately to increase productivity.



Figure 1. AGV at work in DaimlerChrysler

In this paper, we give a brief overview of the current practice of Egemin with developing ATS and we study how a SPL can be introduced to improve the company's engineering practice. This is the first step in the process of fielding a SPL for ATS in the company. In this effort, Egemin will collaborate with DistriNet Labs. A particularly challenging issue of the SPL will be the support for runtime variability allowing an ATS to dynamically rebind variation points to adapt its behavior with changes in the operating environment. From a research perspective, the introduction of the SPL in Egemin provides an excellent setting to study the fielding of a dynamic SPL in practice.

Overview. In the next section, we give an overview of Egemin's current approach for developing ATS and we point to a number of problems. In section 3, we outline how a SPL for ATS can be introduced by Egemin. We give particular at-

tention to support for runtime variability. Finally we draw conclusions and look at future steps in section 4.

2 Current Practice: Product Development

We start this section with a brief introduction of AGV transportation systems. Then we explain the structure of the ATS software at Egemin and we explain a number of problems with the current approach.

2.1 AGV Transportation Systems

An AGV transportation system consists of a number of AGVs. AGVs are fully automated, custom-made vehicles that are instructed by control software to perform transport tasks. AGVs are provided with low-level control software connected to sensors and actuators to move safely through the warehouse. While moving, the vehicles follow specific paths in the warehouse by means of a navigation system which uses stationary beacons in the work area, typically laser reflectors on walls or magnet strips in the floor. AGVs are equipped with infrastructure for wireless communication.

AGV transportation systems have to deal with dynamic and changing operating conditions. The stream of transports that enter a transportation system is typically irregular and unpredictable. The system may execute in different execution modes depending on production conditions or the occurrence of particular events in the environment. AGVs can leave and re-enter the system for maintenance, production machines may have variable waiting times, etc. All kinds of disturbances can occur, particular areas in the warehouse may temporarily be closed for maintenance services, supply of goods can be delayed, loads can block paths, AGVs can fail, etc. Despite these challenging operating conditions, the system is expected to operate robustly and efficiently.

2.2 Structure of the ATS Software

The ATS software that instructs AGVs to perform transport tasks contains two subsystems:

- *E'wms*® (Egemin Warehouse Management System) is the software responsible for managing the transport tasks. Typically, E'wms is connected to an Enterprise Resource Planning (ERP) system from which it receives new orders. E'wms translates the orders into specific transport tasks that are passed on to E'tricc which instructs the AGVs to execute the tasks.

- *E'tricc*® (Egemin Transport Intelligent Control Centre) is the software responsible for executing transport tasks by controlling the AGVs. E'tricc has four main functionalities. (1) Transport assignment: transports from E'wms have to be assigned to appropriate AGVs in an efficient and timely manner. (2) Routing: for the movement of AGVs, efficient routes through the warehouse must be determined. Although the road network is predefined, the best route for an AGV is in general dynamic and depends on the current conditions in the system. Routing of vehicles must be free of deadlocks. (3) Collision avoidance: obviously, while moving AGVs may not collide with each other. Collisions do not exclusively occur at intersections of paths; AGVs also need to avoid collisions while passing each other on closely located parallel paths. (4) Battery charging: when an AGVs battery runs out of energy it must drive to a charging station.

An ATS project typically involves the deployment and integration of both subsystems at a particular site. The know-how to develop, customize and maintain E'wms and E'tricc is considered an important part of the domain expertise of Egemin.

2.3 Problems

Historically, E'WMS and E'tricc are conceived as two separate software systems that are being developed, installed an maintained by two relatively independent teams. The high degree of autonomy of the teams has worked well in many projects. However, with increasing customer demands a number of difficulties came to the surface:

1. Similar services are supported in different manners in both systems. This gives rise to duplication of functionality and leads to inconveniences and inconsistencies when deploying both systems together. Examples are services with respect to security, logging, persistence, visualization and configuration.

2. Client-specific customizations are needed to adapt an ATS to the specific needs of a particular customer. The lack of a explicit process to make such customizations leaves the door wide open to opportunistic reuse and ad-hoc adjustments causing erosion of the software and its architecture.

Fig. 2 gives a schematic overview of Egemin's current approach to produce products. To tackle the difficulties explained above, Egemin engineers asked DistiNet researchers the following questions:

- How to avoid duplication of common services in both systems in the best possible way?

- How to support client-specific customizations in a disciplined and well-structured manner?

- How to push back the erosion of the software architecture and components?

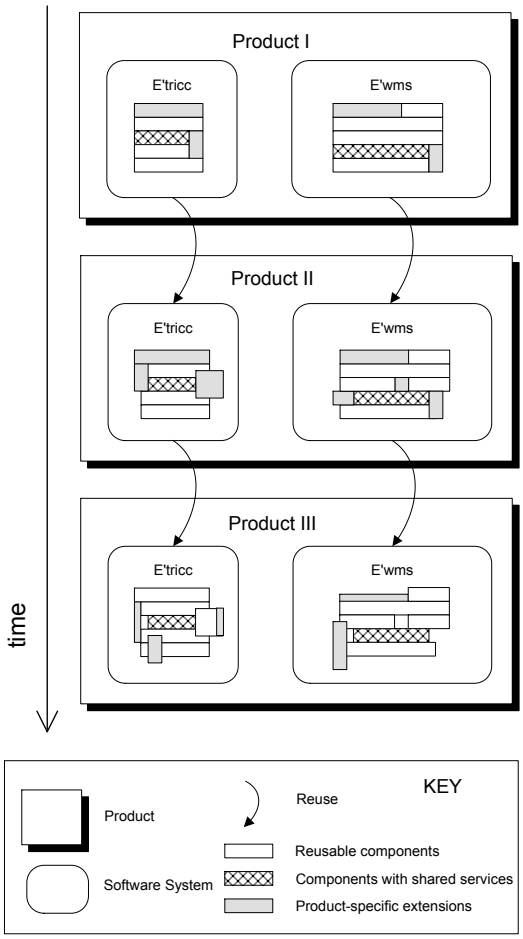

Figure 2. Architecture erosion during development of subsequent products

An integrated approach to solve these questions provides an important opportunity for Egemin to improve the quality of their software and ultimately increase productivity.

3 Towards a Software Product Line

The introduction of a SPL for ATS is just the approach for Egemin to tackle the problems mentioned above. In a SPL, products (ATS for clients) are developed from a common set of assets in a prescribed way. This contrasts with systems being developed separately. Central to a SPL is planned reuse of the core assets. However, the shift towards a SPL is not trivial. It not only implies technical change, but also organizational change. In this section, we outline an approach for introducing a SPL for Egemin's ATS. Fist we give an overview of a number of important core assets. Then we zoom in the challenges with respect to support for runtime variability.

3.1 Core Assets and SPL Architecture

With the introduction of a SPL, the core of activities of Egemin becomes the development, maintenance and evolution of a common core asset repository for a family of ATS and the definition of an explicit process to derive concrete products from the core assets. Fig. 3 gives an overview of product development with core assets.

One of the first steps to field a SPL for ATS will be the identification and documentation of the scope of the SPL. The scope describes which products will belong to the product line. Currently, the kind of ATS supported by Egemin is not explicitly documented. Example approaches to scope a SPL are proposed in [9, 5]. The scope needs to be further refined by specifying common requirements that are shared among the products. Egemin applies use cases for the specification of functional requirements and utility trees with quality scenarios [2] for the quality requirements. Several of the requirements have to provide variability to specify requirements for concrete ATS. An example of a quality

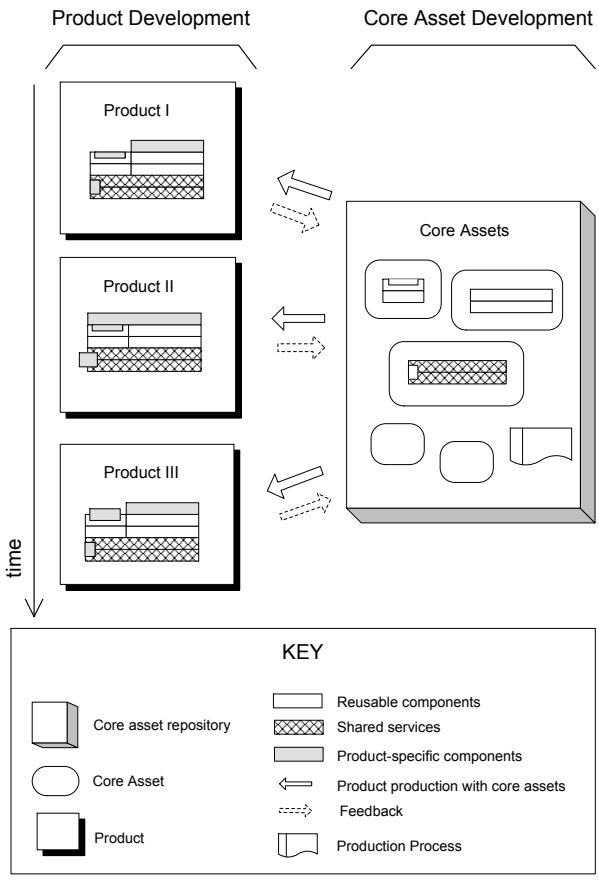

Figure 3. Product development with core assets

scenario without variability:

> **S1:** *An AGV gets broken and blocks a path under normal system operation. Other AGVs have to record this, choose an alternative route if available and continue their work.*

An example of a quality scenario with variability:

> **S2:** *When* `condition`*, the ATS dynamically switches operation from normal mode to* `specific mode`*. Under* `specific mode`*, the system guarantees* `specific performance`*.*

This quality scenario stipulates that the system should dynamically change its operation mode when the operation conditions change. `Condition`, `specific mode` and `specific performance` are variation points in the scenario. We give a concrete example of a quality scenario in section 3.2.1.

One of the most important core assets that has to be defined is the SPL software architecture. To tackle the problems related to the erosion of the software architecture of E'wms and E'tricc, the ATS software architecture is currently under revision. The common functionality is separated from E'wms and E'tricc core and accommodated in a subsystem called E'pia® (Egemin Platform for Integrated Automation). E'pia will provide general support for system configuration, communication, persistency, security, logging, globalization, visualization and diagnosis. E'pia also handles the interfacing with the low-level control software of AGVs. Fig. 4 shows an overview of the subsystems of the SPL architecture for ATS.

A dedicated team is responsible for the development, maintenance, and evolution of each of the subsystems. Products are derived from the subsystems by instantiating the variation points of the subsystems according to the client-specific requirements. Fig. 5 gives a schematic overview of product development. Each product team includes a number of members of the E'wms and E'tricc core asset teams which bring specific know-how into the product team. This structure also supports first-line feedback from the product development teams to the core asset teams.

Besides scope, requirements, software architecture, and components, various other core assets have to be defined or

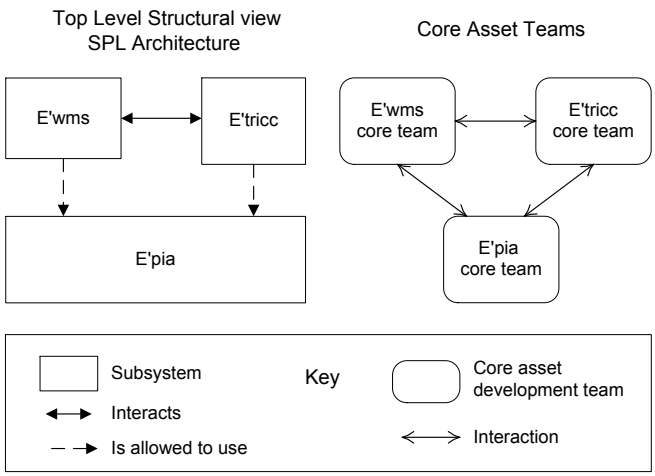

Figure 4. SPL architecture and corresponding core asset teams

mined. Examples are domain models, test scenarios, planning and budget models, etc.

3.2 Challenges in Runtime Variability

Disciplined approaches for supporting variability in SPL have gained increasing attention in research. [10] gives an overview of the kind of variabilities that can be supported in SPL architectures. One interesting approach is proposed in [4] where the authors use change sets and relations to model variability of SPL architecture. However, the definition of rigorous specification languages for variability of SPL architecture applicable to realistic systems requires further research [8]. Variability of SPL architecture is typically considered from a static perspective, i.e. variation points are bind at design time or load time. Due to the dynamic operating conditions of ATS, variability shifts toward a dynamic perspective, where systems modify their structure/behavior during execution. Such modifications can be achieved by dynamically rebinding variation points at runtime.

In this section, we list a number of concrete scenarios of runtime variability in ATS. We elaborate on one particular example showing how this scenario can be supported in the SPL architecture. Finally, we list a number of challenges regarding runtime variability that have to be tackled during the development of the SPL for ATS.

3.2.1 Scenarios for Runtime Variability in ATS

Dynamically Switching Operation Mode. Particular events in the environment may happen that change the operation conditions of the ATS drastically. A typical example is the arrival of a truck with pallets of raw material that have to be unloaded and stored in the warehouse. Such an event should trigger the system to switch operation mode; this can be expressed as an instantiation of the variation points of quality scenario S2 explained in the previous section:

> *When* `a truck with pallets arrives`, *the ATS dynamically switches operation from normal mode to* `unloading and storing pallets`. *Under* `unloading and storing pallets`, *the system guarantees* `that per AGV ready for use 10 pallets are stored per hour`.

In section 3.2.2, we explain how dynamically rebinding of a variation point provides an architectural solution to switch the operation mode of the ATS.

Support for Openness. Openness of an ATS refers to the system's ability to deal autonomously with AGVs leaving and (re-)entering the system at runtime. Examples are an AGV that temporarily leaves the system for maintenance, and an AGV that re-enters the system after its battery is recharged. Openness is also important in the start-up phase when the system is gradually deployed and tested.

In earlier work, DistriNet Labs and Egemin developed an innovative multiagent-based ATS to support openness [14, 12]. Although the results of the prototype where highly promising, the study learned us that switching from a centralized toward a decentralized agent-based architecture is a big step with far-reaching effects for the company, for the software as well as for the developing organization. The experience indicates that the introduction of an agent-based approach should be done in a controlled way, step-by-step [13].

Rearranging Stock. Typically, E'wms derives transport tasks from the orders it receives from an ERP system it is

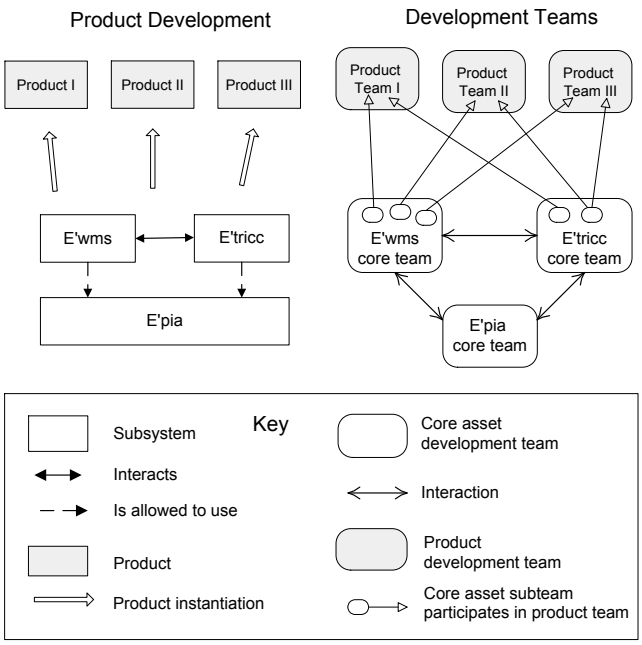

Figure 5. Product development and corresponding development teams

connected to. However, the way goods are distributed in a stock has an impact on the time needed for their retrieval. During off-peak hours, E'wms will revise the distribution of goods such that retrieval times are reduced. To reallocate the goods in the stock, E'wms injects dedicated transport tasks, which are typically executed by a subset of AGVs.

Manual Control of AGV. In an ATS, human intervention is sometimes necessary, e.g. for maintenance or in case a fallen load blocks a path. Therefore, an ATS is equipped a number of manual control mechanisms that override its normal operation. An example of such a control mechanism is a custom transport task for a particular AGV. A human can define such a task to direct a particular AGV to a specific location in the warehouse. Another example is a remote control that allows a human to directly control the wheels and the lift of an AGV. This can be used to move an AGV freely, without being restricted by the paths in the warehouse, etc.

3.2.2 Dynamically Switching Operation Mode

We now illustrate support for runtime variability with the concrete example explained above in which the ATS dynamically detects the need to switch operation from normal mode to specific mode when a truck with pallets arrives at the warehouse. We explain how dynamically rebinding the transport assignment approach provides an architectural solution to deal with this particular quality requirement. Fig. 6 shows the main components of the E'tricc subsystem.

Client I/O provides the interface to client-specific plugins such as modules for visualization, diagnosis, statistics, and manual assignment of tasks to AGVs. E'wms I/O interfaces with the warehouse management system. Via this interface, E'tricc receives transport orders and informs clients about the status of the transports. Common Services I/O interfaces with E'pia. Transport management takes care for allocating transport tasks to AGVs. Traffic control takes care for routing, collision and deadlock avoidance, battery management etc. Visualization provide the basic functionalities for user interfacing. Finally, AGV1, AGV2, ... AGVn are components that represent the different AGVs in the system.

The standard mechanism for transport assignment applied by Egemin is based on predefined schedules. Schedules are rules associated with AGVs and particular locations in the layout, e.g. "if an AGV has dropped a load on location x, than that AGV has to move to node y to wait for a transport assignment." A plan can be changed, however only under exceptional conditions. E.g., when an AGV becomes defective on the way to a load, the transport can be re-assigned to another AGV. Schedule-based transport assignment allows for advanced client-specific optimizations. However, when the conditions in the environment radically change—e.g. when a truck arrives with pallets that have to be unloaded and stored in the warehouse—the mechanism that is specifically tailored to normal operation mode will perform poorly. To handle this specific execution mode, the transport management component has to switch dynamically the transport assignment approach, see Fig. 6. One

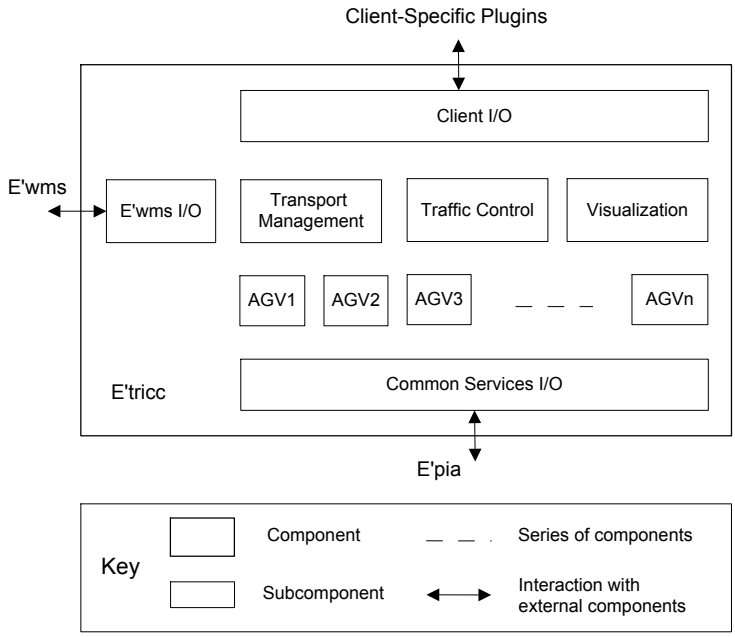

Figure 6. Main components of E'tricc

possibility is to switch to a market-based mechanism. To deal with ongoing dynamics in ATS, we have developed such a mechanism called the Dynamic Contract Net Protocol (DynCNET [11]). The DynCNET protocol consists of four steps: (1) a new transport task is announced; (2) the idle AGV components make bids to perform the task (e.g. a bid can be based on several factors such as the expected time it will take to execute the task and the priority of the transport); (3) the AGV with the best offer is assigned the provisional winner; and finally, (4) the AGV picks the load and starts executing the transport. Between the step (3) and (4) the task can be switched to a more suitable AGV (e.g. an AGV that enters the system after it has charged its battery). As soon as the pallets are stored in the warehouse, the transport management component should restore the default transport assignment approach.

3.2.3 Challenges

Support for runtime variability puts forward various challenges. We list a number of issues that have to be tackled:

- A rigorous specification of requirements for runtime variability is required.
- The requirements have to mapped to design solutions. Achieving an elegant architectural solution is not always evident. For example: switching the transport assignment approach can be relatively easy achieved by swapping the component for transport assignment. However, switching an AGV to manual control involves adaptations to various components such as transport assignment and collision avoidance. The underlying issue is that the combination of some variability scenarios imposes a trade-off: an architecture either facilitates one variability scenario while hindering the other or vice versa.
- Dynamic rebinding of variability points poses a number of technical challenges, including (1) detecting the condition to start rebinding, (2) rebinding the modules at runtime in a consistent way (e.g. state transfer being a non-trivial issue), and (3) restoring the standard approach when the condition for rebinding no longer holds.
- In some cases runtime variability is of a binary nature (i.e. variation points are connected or not); in other cases runtime variability is a dynamic process. An example of the latter is the rearranging of a stock in which the number of dedicated AGVs is a function of the number of regular transports that enter the system.

4 Conclusion and Next Steps

In this paper, we discussed a number of problems that Egemin faces with its current approach for developing ATS. To tackle these problems, we motivated the introduction of a dynamic SPL for ATS. As a first step to field such a SPL,

we gave an outline of the software structure and its associated development teams. We illustrated support for runtime variability, a crucial issue of the SPL for ATS.

Currently, a set of general functional requirements and quality scenarios for the SPL have been defined and based on that the course-grained software architecture has been specified. The next planned steps to field the SPL are:

- Performing a Product Line Technical Probe (PLTP [3]) to characterize Egemin's strengths and weaknesses relative to its product line effort.
- Completing the requirements specification. We plan to develop a disciplined approach for integrating variability in quality scenarios and utility trees.
- Completing the documentation of SPL architecture. We plan to develop a structured approach for documenting runtime variability.
- Architecture evaluation. The crucial role of variability in general, and runtime variability in particular, will require dedicated techniques for evaluating the SPL architecture. Initial experiences with evaluating SPL are described in [7, 1].
- Defining and mining an initial repository of additional core assets in line with the SPL architecture. The initial focus will be on mining software components for the ATS subsystems, the specification and implementation of variability, test scenarios, and the specification of a production plan to derive concrete systems from the core assets.
- Pilot project. We plan to set up a pilot project to apply and evaluate the SPL approach for ATS. The strengths and weaknesses derived from a practical case will provide crucial information for Egemin to go ahead with the SPL.
- Cost-benefit analysis. We plan to develop a cost-benefit analysis method for evaluating the SPL based on the Cost Benefit Analysis Method (CBAM [6]). CBAM supports the analysis of costs, benefits, and planning based on architectural decisions. In the context of SPL, the analysis has to take into account the architectural decisions with respect to variability in requirements and the software architecture.

Acknowledgement

This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund K.U.Leuven. Danny Weyns is supported by the Funding for Scientific Research in Flanders (FWO).

References

[1] N. Boucké, D. Weyns, K. Schelfthout, and T. Holvoet. Applying the ATAM to an Architecture for Decentralized Contol of a AGV Transportation System. In *2nd International Conference on Quality of Software Architecture, Lecture Notes in Computer Science, Vol. 4214*. Springer, 2006.

[2] P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley Publishing Comp., 2002.

[3] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison Wesley Publishing Comp., August 2001.

[4] S. A. Hendrickson and A. van der Hoek. Modeling product line architectures through change sets and relationships. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 189–198, Washington, DC, USA, 2007. IEEE Computer Society.

[5] I. John, J. Knodel, T. Lehner, and D. Muthig. A practical guide to product line scoping. In *Software Product Lines, 10th International Conference, SPLC 2006, Baltimore, Maryland, USA, August 21-24, 2006, Proceedings*. IEEE Computer Society, 2006.

[6] R. Kazman, J. Asund, and M. Klein. Making Architecture Design Decisions: An Economic Approach. *Software Engineering Institute, Carnegie Mellon University*, 2002.

[7] F. Olumofin and V. Misic. Extending the ATAM Architecture Evaluation to Product Line Architectures. In *5th IEEE-IFIP Conference on Software Architecture*, Pittsburgh, USA, 2005.

[8] R. Rabiser, P. Grunbacher, and D. Dhungana. Supporting product derivation by adapting and augmenting variability models. In *SPLC '07: Proceedings of the 11th International Software Product Line Conference*, pages 141–150, Washington, DC, USA, 2007. IEEE Computer Society.

[9] K. Schmid. A comprehensive product line scoping approach and its validation. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 593–603, New York, NY, USA, 2002. ACM.

[10] M. Svahnberg and J. Bosch. Issues concerning variability in software product lines. In *International Workshop on Software Architectures for Product Families*, London, UK, 2000. Springer-Verlag.

[11] D. Weyns, N. Boucke, and T. Holvoet. A Field-Based Versus a Protocol-Based Approach for Adaptive Task Assignment. *Autonomous Agents and Multiagent Systems*, 2008.

[12] D. Weyns and T. Holvoet. Architectural design of a situated multiagent system for controlling automatic guided vehicles. *International Journal on Agent Oriented Software Engineering*, 2(1):90–128, 2008.

[13] D. Weyns, T. Holvoet, K. Schelfthout, and J. Wielemans. Decentralized control of automatic guided vehicles: Applying multi-agent systems in practice. In *Development Track OOPSLA, ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Nashville, USA, 2008.

[14] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In *4th Joint Conference on Autonomous Agents and Multiagent Systems, Industry Track*, Utrecht, The Netherlands, 2005. ACM Press.