Lean Software Development Domain

Marcelo Udo[1] and Tiago Stegun Vaquero[2] and José Reinaldo Silva[2] and Flavio Tonidandel[1]

[1]Centro Universitário da FEI
IAAA – Artificial Intelligence Applied in Automation Lab – São Bernardo do Campo - Brazil
[2]Escola Politécnica – Universidade de São Paulo
Design Lab, Mechatronics and Mechanical Systems Department – São Paulo - Brazil
m_udo@fei.com.br, tiago.vaquero@poli.usp.br, reinaldo@usp.br, flaviot@fei.edu.br

Abstract

The AI Planning field has pursued the goal of applying all developments already achieved in order to conquest real and successful cases. Considering this objective, interesting and challenging real problems are found in the software development field. This paper was elaborated based on complex problems encountered in the software development area regarding planning, scheduling and domain knowledge, where the main goal was to demonstrate an implementation of AI Planning and Knowledge Engineering (KE) for Planning & Scheduling for solving such real problems. This text details how we acquired domain knowledge by using itSIMPLE (through UML), a KE tool, and a planner in order to keep planning and scheduling safe and sound for a Lean Software Development domain. After acquiring domain knowledge from the Lean Software Development domain, we were able to (re-)plan software development activities by using itSIMPLE in conjunction with Metric-FF to have, when necessary, a new plan to give feedback about installed capacity of the domain and resources. By doing so, we reduce some old problems commonly found in software development processes. We also present some benefits achieved so far by using both KE and AI Planning technologies.

Introduction

The professionals involved with planning and management activities in the software development field have showed several problems regarding delivering software systems on time, on budget, and on quality.

However, there is another calamitous problem in this field: lack of standardization of activities, work products, product components, and products (it is important to mention that work products and final products are different concepts; considering work products the customers do not receive them and they are actually used just as outputs of development processes activities (Kulpa and Johnson 2003)). In this way, approaches such as Lean Software Development (Poppendieck and Poppendieck 2003) have been receiving great attention because of its level of standardization, which is sometimes considered better than other approaches of software development. The Lean Software Development domain can be enriched with better results when combined with Knowledge Engineering (KE) processes. Since the Lean Software Development involves a lot of planning and scheduling activities, mainly in management tasks, it is an interesting real domain to be modeled in an automated planning & scheduling approach in order to observe the benefits of applying such techniques for software development processes.

By dealing with real problem, KE processes and tools are essentials for acquiring domain knowledge. According to Dana Nau (2007), acquiring domain knowledge is one of the most important issues of automated planning research. In fact, planners could be more useful and reliable for real-world problems with new methods, tools, and practices for acquiring domain knowledge. For this reason, some researchers identified necessary improvements for KE applied to Automated Planning (McCluskey 2000).

The KE tool we used in the current work was itSIMPLE (*Integrated Tools Software Interface for Modeling PLanning Environments*) (Vaquero, Tonidandel, and Silva 2005), first demonstrated during ICKEPS'05, which was an initiative towards accelerating Knowledge Engineering research. The continuous use and improvements of this KE tool have generated works to demonstrate the importance of Requirements and Knowledge Engineering in the AI Planning & Scheduling area and also the fundamental role of a structured design life cycle in real applications (Gomes et al. 2007; Vaquero et al. 2006; Vaquero et al. 2007; Vaquero, Tonidandel, and Silva 2005).

In this paper we aim to overcome some problems encountered in management, planning and scheduling activities during software development process while emphasizing the important role of knowledge about actions necessary to produce software in different platforms and technologies. In this work we show how we are so far treating some of these problems by using AI Planning and Knowledge Engineering for Planning & Scheduling (KEPS) with a Lean Software Development approach. We also present some tests and some issues, which brought us new understandings about the use of a KE tool together with the planner Metric-FF (Hoffmann 2003) for simulations and re-planning applied for the Lean Software Development.

This paper is organized as follows. In the first section we clarify some of the main difficulties raised in the Software Development and the Chaos Report. In the next section we briefly discuss the Lean Software Development Domain. Then, we demonstrate the KE process of Acquiring Domain Knowledge and some evidences and issues rose during testing. Finally, we conclude and give some future works.

Software Development and the Chaos Report

According to the Chaos Report developed in 1994 by Standish Group (1995), the majority of software projects fail in terms of budget, time, specification, and so forth. To have a clear understanding, just 35% of software projects are delivered on time and on budget.

The Chaos Report raised some problems with planning and scheduling in this area and also the fact that managers do not have appropriate knowledge of the software development activities. Oftentimes there are no standardized tasks and work products to define how the domain knowledge works. One of the tools very typical to reduce problems in planning and scheduling is the Microsoft Project, but it requires a professional called Project Manager to solve the conflicts above at any time using this tool.

In fact, any company that produces software systems needs information about its installed capacity which is given based on current planning and scheduling and it also needs to have a development methodology. Nowadays, one of the methodologies that has been successful applied is the Lean Software Development – An agile methodology (Poppendieck and Poppendieck 2003). This approach is described in the following section.

Lean Software Development Domain

In the late 1940s, a small company called Toyota changed the way of producing cars and the way managers believed production should work. Since then, many companies have changed their production management by the Lean Thinking (Ohno 1988).

One of the domains affected by the Lean Thinking was the Software Development, which generated the term Lean Software Development, according to the works of Mary and Tom Poppendieck (2003).

The Lean Software Development approach guides companies to standardize methods, activities, and work products by following some main principles: eliminate wastes; amplify learning; decide as late as possible; deliver as fast as possible; empower the team; and built integrity in (Poppendieck and Poppendieck 2003).

One of the main and interesting features of the Lean Software Development approach is the treatment of all work products as they were pieces that compound functionalities of a software release, such as specification, data model, test plan, user guide, and others. This feature is the main focus of this paper where Knowledge Engineering for Planning & Scheduling was used to support and help software development managers and teams to coordinate their actions while performing and packaging the work products (pieces) properly.

We tested our assumption that Knowledge Engineering for Planning & Scheduling together with Automated Planning technology can be seen as a great solution for the learning time and the production efficacy of Lean Software Development domain because it could diminish the human factor of trying to understand domain knowledge in detail and let it stored into a knowledge base created by a KE tool. In addition, it would let planners do what is necessary when (re-)planning the software production.

Acquiring Domain Knowledge

As mentioned before, in this work we used a Lean Software Development domain because all tasks, activities and work products are standardized and this contributed during KE process in terms of time. In this section we focus on the part of the domain that deals with the production line of a Lean Software Development process.

The most important here is to treat software as a group of small components. Therefore, in the production line we have software *releases* that own several pieces such as *specification, test plan*, *unit testing*, *user guide*, and so on (see partial UML class diagram in Figure 1). In the following topics we present the main elements of the domain model.

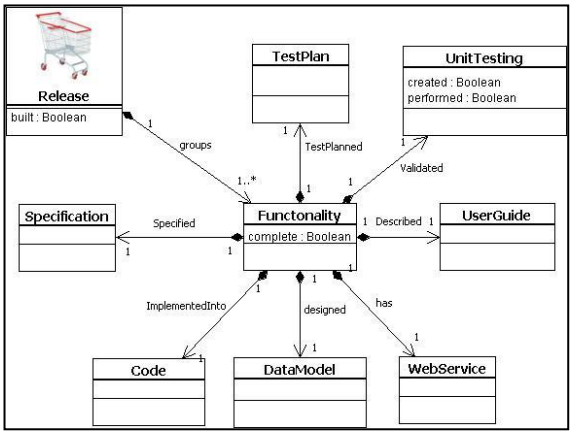

Fig. 1. Software Release and its components.

Acquiring Knowledge Objects

In the current topic we present a list of knowledge objects that compound a software release. According to Nick Milton (2007), Knowledge Objects are the elements that make-up a knowledge base, such as classes or types, relations or predicates, attributes, actions, and values.

The type *Release* (illustrated in Figure 1) owns a relationship "whole-parts" with *Functionality* and this means how many functionalities a software release has to

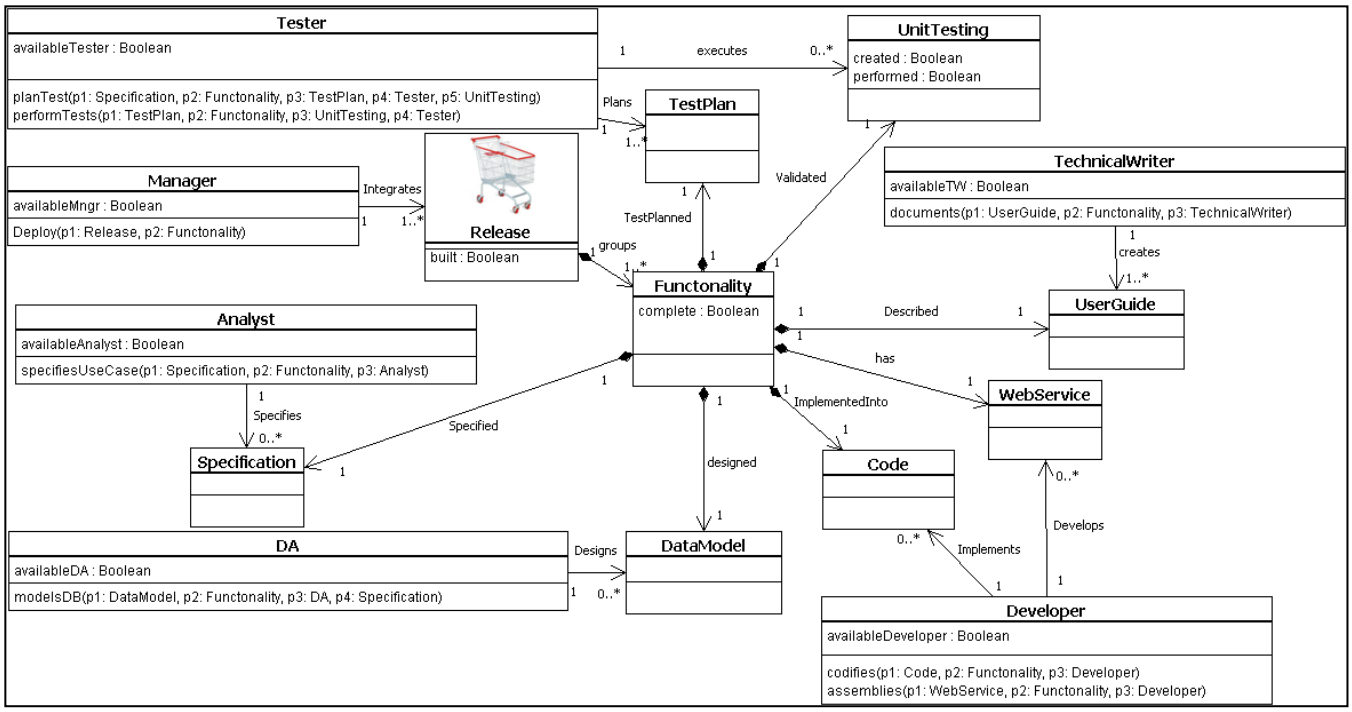

Fig. 2. Class diagram of the Lean Software Development Domain.

have. This characteristic is represented by the relationship *groups*, also represented as predicate *(groups ?rel – Release ?fun – Functionality)* in PDDL (Planning Domain Definition Language) (Fox and Long 2003). When the software releases are complete, the attribute *built* shows whether it is built or not, also represented as predicate (*built ?rel - Release)*. According to the class diagram in Figure 1, it is possible to observe that there are many "whole-parts" relationships among the type *Functionality*.

There is a kind of knowledge object in the Lean Software Development domain that has the responsibility to change the states in the domain and they are also called Agents. The agents of the current domain model are:

- *TechnicalWriter*: is the agent responsible for documenting technically software functionalities;
- *Tester*: is the agent responsible for planning test coverture and for performing test execution;
- *Developer*: is the agent responsible for codifying knowledge in some computer language;
- *Manager*: is the agent responsible for managing lean software production;
- *Analyst*: is the agent responsible for eliciting, analyzing, and modeling domain knowledge;
- *DA* (Data Analyst): is the agent responsible for modeling data for databases.

These agents are responsible for the workflow of the production line in a Lean Software Development domain. The most important predicates or relationships of those agents listed above are the ones that defines the availability of an agent to start a given task or action, for instance, *(availableDeveloper ?dev – Developer)*. This predicate holds the availability of *Developers* for working on an order.

Acquiring Actions for the domain

Now we present a list of actions (in PDDL format) performed by agents in the Lean Software Development Domain (see also the complete UML class diagram for this domain in Figure 2):

- *(:action makeDocuments)* – is a *technical writer*'s action dedicated to document user guides for software releases;
- *(:action planTest)* – is a *tester*'s action dedicated to plan tests that cover all functionalities that compound software releases;
- *(:action performTests)* – is another *tester*'s action dedicated to perform tests once planned;
- *(:action codifies)* – is a *developer*'s action dedicated to software code;
- *(:action assemblies)* – is another *developer*'s action for assembling web services;
- *(:action deploy*) – is a *manager*'s action for deploying software releases to customer systems environments;
- *(:action specifiesUseCase)* – is an *analyst*'s action dedicated to specification. In our case, we use Use Cases documents;
- *(:action modelsDB)* – is a *DA*'s action for modeling databases.

Knowledge about Agents using Lean Principle

As a principle, Lean Software Development domain will split any project into software releases. So, the production line agents will have a good view of what constitutes the software project.

We describe here the main activities performed by the agents of Lean Software Development domain and its implications with other knowledge objects.

Production Manager compounds software releases with functionalities. The main responsibility of manager of the production line is to integrate software releases grouping all *functionalities* necessary to a given *release*.

Production Manager compounds functionalities with work products. Other responsibility of manager of the production line is to constitute *functionalities* with *specifications*, *unit testing scripts*, *test plans*, *codes*, *web services*, and so forth.

Each Agent of the production line creates work products that compounds functionalities. At first, the responsibility of *analyst* is to specify all documents necessary for the *functionalities*. Afterwards, there are two agents that can go on: *Tester* and *Data Analyst* (*DA*). *Tester* will elaborate a test plan and prepare unit testing scripts. *Data Analyst* will design a data model. Both will do them incrementally.

Planning Problem Example

In this topic we present the initial and goal states of a common planning problem for the Lean Software Development domain.

Initial State: there are all agents available and software releases to be built with their functionalities.

Goal State: all software releases are completed with their functionalities. Each completed functionality will be together with its work products such as *specification, test plan, unit testing, code, web service, user guide*, and so forth.

Supposing a simple and illustrative planning problem scenario where a software release must be develop by a software team that has the following members: *Analyst* (*CARLOS*), *Data Analyst* (*ALBERTO*), *Testers* (*RAFAEL* and *ALINE*), *Technical writer* (*CELIO*), and a *Developer* (*DANIEL*). These members are the agents of this domain scenario and they will probably perform actions such as: specifies Use Case, models DB, testers plan Tests, codifies program, testers perform Tests, technical writers make documents, managers deploy releases, and so forth. In this example there are also domain objects, such as functionality (*F1*), specification (*SPEC1)*, data model (*DM1*), test plan (*TP1*), unit tests (*UT1*), program (*PROGRAM1*), user guide (*UG1*) that must be arranged and coordinated to produce the software release *R1*. At the end, customers will receive the software system with the aimed release *R1* with a set of functionalities required (in this case only *F1*).

The following list of actions represents the plan generated by an automated planner for producing the release *R1* in the described scenario. However, when considering a real scenario the planner must reason about producing several releases with sets of functionalities using the resources adequately in the Lean Software Development domain:

0.00: (SPECIFIESUSECASE SPEC1 F1 CARLOS) [10.00]
10.01: (MODELSDB DM1 F1 ALBERTO SPEC1) [8.00]
18.02: (PLANTESTS SPEC1 F1 TP1 RAFAEL UT1) [4.00]
22.03: (CODIFIES PROGRAM1 F1 DANIEL) [10.00]
32.04: (PERFORMTESTS TP1 F1 UT1 ALINE) [10.00]
42.05: (MAKEDOCUMENTS UG1 F1 CELIO) [8.00]
50.06: (DEPLOY R1 F1) [4.00]

When manager has the solution plan for the production line, he prints cards, according to that plan, which are work orders to the software team. Each card is put on a board called Kanban (Ohno 1988; Poppendieck and Poppendieck 2003), which is a Lean Visual Tool for monitoring and controlling activities. When there is a strong deviation in the plan detected in the Kanban, it is necessary to re-plan the software releases. Since re-planning (from scratch) is the approach used in this work, it is necessary to define current state and goals to generate a new plan that considered the deviation (other approaches besides re-planning, such as plan repair and reuse, will be researched in future works). For the definition of the current states and the goal states we used itSIMPLE (Vaquero et al. 2007) together with Metric-FF (Hoffmann 2003) for generating plan (either for planning or re-planning), and also the Lean Visual Tool for monitoring activities status. In addition, managers can visualize and track (before the execution) the use of resources through itSIMPLE (Vaquero et al. 2007).

This approach helps managers to visualize problems with the software releases or project which will not be delivered on time. However, they will have time to apply corrective actions to the plan for maintaining commitments with customers.

We used this approach to contrast with the traditional tool of software development management and planning activities: Microsoft Project. Another commercial tool used for project management that is important to mention is the Primavera Systems. However, this tool would be a waste according to the lean thinking, because it would increase the complexity of management (Jeong 2003). Thus, we will show in the next section the approach discussed for testing it against some issues.

Testing some Issues

During the whole KE process for the Lean Software Development domain we captured some issues about the use of KE for Planning & Scheduling and Automated Planning. The first issue was that Manager takes too long

to control a project schedule when it is in a low level of detail. It becomes a paradox because in order to Manager have as much control as he can, he must control activities in a low level of detail. The second issue we elicited was that Manager looses the control of project schedule when he needs to simulate new project schedules and they are in a low level of detail. It also becomes a paradox because all the time salesmen needs feedback about future installed capacity of software production and new simulations need to be performed. The third issue is the fundamental idea of Lean Software Development by which software production companies should work like industrial companies.

For testing all issues listed above as we were modeling the Lean Software Development domain, the following scenario was elaborated and repeated for each set of software releases to compare results between two different ways of planning and managing:

- Three sets of samples of software releases (20, 40 and 80 releases);
- Each set of sample was tested using both itSIMPLE & Metric-FF and MS Project 2003;
- When *Manager* used itSIMPLE & Metric-FF (i.e., using AI Planning), he should simply update the current situation and/or the desired situation of production line and its resources. Then, he should simulate the new plan for production line.
- When *Manager* used MS Project 2003, he should update each task affected by the current situation manually. He should visualize each resource affected and solve conflicts with resources overload. This *Manager* should take care of not skipping any action necessary in the schedule;
- The size of team was the same for each test;
- Actions and Time consuming were analyzed given the use or not of AI Planning.

After testing all samples, when *Manager* used MS Project he had to control all resources and tasks so that he did not have resources overload. For this reason, he needed much more time to solve those conflicts using the tool. On the other hand, when he used itSIMPLE (Vaquero et al. 2007) with Metric-FF (Hoffmann 2003), he first defined the initial state and the goal state of software production and then generating the plan. For each change in the production he re-planned by putting current state in the place of the initial state maintaining the goal state. This re-planning activity of the *Manager* using itSIMPLE & Metric-FF was agile and all actions could be transformed into work orders for the software team.

We present in the table below some comparisons between the use of automated planning and the current way of managing software activities. These tests were performed from January to April of 2008 (4 months). We used a real company order to perform these tests.

In the first test, described in Table 1, the production line had a sample of 20 *releases* for a software team of 50 members (*Agents* in the domain) and for this situation the respective number of Actions and Time consuming with Planning using MS Project 2003 are showed in the table. The second test was equal to the first one, except that *Manager* used itSIMPLE & Metric-FF. The other tests were simply an increasing in the sample of software *releases.*

Releases	Team	Actions	Time of Planning & Scheduling	Using AI Planning
20	50	147	28 hours	No
20	**50**	**140**	**9.7 hours**	**Yes**
40	50	299	39 hours	No
40	**50**	**283**	**11 hours**	**Yes**
80	50	590	67 hours	No
80	**50**	**567**	**17 hours**	**Yes**

Table.1. (Re-)planning activities: AI Planning x Manual Planning

The more software releases increases, the more time will be necessary for the activities of planning and scheduling. In this way, as software releases increases, the use of MS Project becomes very time consuming instead of what took place with itSIMPLE & Metric-FF (Figure 3). Curiously, when the *Manager* used itSIMPLE & Metric-FF to make production simulations, it became an interesting support tool like other ones used in Manufacturing; for example, Preactor software tools solutions (Liddell, 2008). For a clear understanding about the tests, Figure 3 illustrates the differences cited above.

Fig. 3. MS Project 2003 x itSIMPLE & Metric-FF.

By analyzing the results in Figure 3, we can describe the following. When *Manager* used MS Project 2003 to plan and re-plan, taking in consideration resources overload, for 20 software releases he needed 28 hours whereas by using itSIMPLE & Metric-FF he needed only 9.7 hours. For 40 software releases, by using MS Project 2003 he took 39 hours whereas by using itSIMPLE & Metric-FF he used 11 hours. We tested up to 80 software releases for MS Project 2003 with 67 hours and for itSIMPLE & Metric-FF with 17 hours. The curve regarding itSIMPLE & Metric-FF represents a great result in terms of time used for planning, re-planning, and scheduling in software management

activities. In addition, the manager can avoid mistakes derived from manual planning & scheduling process.

Thus, we had some observations regarding the issues rose before the tests:

- First issue: *Manager* takes too long to control a project schedule when it is in a low level of detail. That is true and principally when using MS Project;
- Second issue: *Manager* looses the control of project schedule when he needs to simulate new project schedules and they are in a low level of detail. That is also true nowadays in the area, but applying AI Planning together with KE, the results can be different;
- Third issue: software production companies should work like industrial companies. That is true from our point of view because we observe that a shop floor tool for a software production company can have AI Planning Technology.

As mentioned before, nowadays many managers in the software production count on software tools like Microsoft Project to perform activities like those we treated in this paper and the main problem of them is the lack of automatic scheduling and the dependence of personal knowledge to correct plan manually. We could use tools like Preactor solutions (Liddell 2008) but we would lose explicit knowledge acquired in the Knowledge Engineering process.

Applying the KE process, we were able to create a knowledge base with all knowledge necessary for itSIMPLE tool together with a planner Metric-FF to generate a plan which is translated into work orders for the lean software team.

The other important gain achieved with KE for Planning & Scheduling is that we could hire a new *Manager* without demanding that he understood as soon as possible all standards the Lean Software Development domain owns at once. In addition, itSIMPLE maintains the domain modeled in UML which is a language familiar to the software team members (Berardi et al. 2003).

Conclusion

In this paper we presented the Lean Software Development domain, which is an important approach in the software engineering area taking into account software production issues, where we applied techniques of Knowledge Engineering for Planning & Scheduling and Automated Planning (McCluskey 2000). In this application we illustrated that it is possible to achieve results that are not achievable by using simply a schedule tool like Microsoft Project. In addition, it is feasible to maintain the information of resources and activities status about your organization updated in a real time monitoring and execution by using a simple tool like Kanban – a Lean Visual Tool.

Of course, as time goes on, managers will diminish the learning time for acquiring knowledge about Lean Software Development domain. It succeeds because by using a tool like itSIMPLE together with a planner for planning the Master Plan Schedule, managers will feel more confident to be in charge of a production line as soon as possible. In fact, managers could be able to give information about installed capacity of each production unit regarding resources and customer orders. The mainly tasks that managers have to do are to take care of master plan schedule, the events in the production, and the plan generation.

These results presented in this paper motivated innovations in the itSIMPLE for dealing with exogenous events and simulations for analyzing different courses of actions.

References

Berardi, D.; Cal, A.; Calvanese, D.; and Giacomo, G. 2003. Reasoning on UML class diagrams. Technical Report 11-03. Available at http://citeseer.ist.psu.edu/article/berardi03reasoning.html.

Fox, M.; Long, D. 2003. PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research 20:61-124.

Gomes, M. L.; Udo, M.; Vaquero, T. S.; Silva, J. R.; and Tonidandel, F. 2007. Obtaining States Invariants From Class Diagram in UML.P. In: VIII SBAI - Simpósio Brasileiro de Automação Inteligente, Florianópolis, Brazil.

Hoffmann, Jörg. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. J. Artif. Intell. Res. (JAIR) 20: 291-341.

Jeong, H. 2003. Distributed Planning and Coordination to Support Lean Construction. PhD thesis 2003, University of California, Berkeley.

Kulpa, M. K.; Johnson K. A. 2003. Interpreting the CMMI: A Process Improvement Approach, Auerbach Publications.

Liddell, Mike. 2008. Batch Scheduling in a Lean Manufacturing World at http://www.preactor.com/whitepapers.aspx.

McCluskey T. L. 2000. The Knowledge Engineering for Planning Roadmap in the PLANET final report to the EC, November 2000.

Milton, N. R. 2007. Knowledge Acquisition in Practice: A Step-by-step Guide. London: Springer.

Nau, D. 2007. Current trends in automated planning. *AI Magazine* 28(4):43–58, 2007.

Ohno, Taiichi. 1988. Toyota Production System, English, Productivity, Inc. 1988, published in Japanese in 1978.

Poppendieck, M.; Poppendieck, T. 2003. Lean Software Development, Addison Wesley.

Standish Group 1995. *The CHAOS Report (1994)*. Report of the Standish Group. Available at

http://www.standishgroup.com/sample_research/chaos_1994_1.php .

Vaquero, T. S.; Romero, V. M. C.; Sette, F. ; Tonidandel, F.; and Silva, J. R. 2007. *itSIMPLE2.0 : An Integrated Tool for Designing Planning Domains*. In: Proceedings of 17th International Conference on Automated Planning and Scheduling (ICAPS), Providence, Rhode Island.

Vaquero, T. S.; Tonidandel, F.; Barros, L. N.; and Silva, J. R. 2006. On the Use of UML.P for Modeling a Real Application to the Planning Problem. In: Proceeding of 16th International Conference on Automated Planning and Scheduling (ICAPS). Cumbria, UK.

Vaquero, T. S.; Tonidandel, F.; Barros, L. N.; and Silva, J. R. 2007. Modeling a Real Application as a Planning Problem by using UML.P. In: VIII SBAI - Simpósio Brasileiro de Automação Inteligente, 2007, Florianópolis, Brazil.

Vaquero, T. S.; Tonidandel, F.; and Silva, J. R. 2005. The itSIMPLE tool for Modeling Planning Domains. ICAPS 2005, The First International Competition on Knowledge Engineering for Planning & Scheduling ICKEPS, Monterey, California, USA.