Analysis, design and development of advanced planning systems

Melina Vidoni*, Jorge Marcelo Montagna and Aldo Vecchietti

INGAR – Instituto de Desarrollo y Diseño,
CONICET-UTN, Argentina
Email: melinavidoni@santafe-conicet.gov.ar
Email: montana@santafe-conicet.gov.ar
Email: aldovec@santafe-conicet.gov.ar
*Corresponding author

Abstract: Small and medium-sized organisations (SMO) often need to optimise their operations to improve their effectiveness in a competitive world. Advanced planning systems (APS) emerged to provide optimal operations in several areas of an organisation, including production planning and scheduling, logistics, financial, among others. Currently, there is a gap in the literature regarding proposals, from a software engineering perspective, to assist SMOs in the design and development of APS systems. This article proposes an architecture for the APS domain that supports the implementation of specific applications and can be adapted through variation points. The proposal is evaluated with architecture trade-off analysis method (ATAM) involving two different groups of stakeholders. Also, a process is included to show how to use the architecture in the implementation of a specific case. Finally, a case study based on a local industry is developed.

Keywords: advanced planning systems; APS; software architecture; software analysis and design.

Reference to this paper should be made as follows: Vidoni, M., Montagna, J.M. and Vecchietti, A. (2020) 'Analysis, design and development of advanced planning systems', *Int. J. Industrial and Systems Engineering*, Vol. 36, No. 3, pp.361–383.

Biographical notes: Melina Vidoni graduated from the Universidad Tecnológica Nacional (UTN), where she also received her PhD. She is currently a Postdoc Fellow with a full time scholarship under the supervision of Prof. Vecchietti at the National Council for Technical and Scientific Research of Argentina at INGAR CONICET-UTN. She is also a Teaching Assistant at 'Data Management' at the UTN, Santa Fe, for the Information Systems Engineering degree. Her postdoc research work is focused on integrating APS with big data, and working with software engineering concepts in operations research.

Jorge Marcelo Montagna obtained his PhD in Chemical Technology at the Universidad Nacional del Litoral, Santa Fe, Argentina. He is a Principal Researcher of the National Council for Technical and Scientific Research of Argentina (CONICET) at INGAR. Also, he is a Professor of Information Technology Management at the Universidad Tecnológica Nacional. His main expertise area is process system engineering, working on the mathematical models for operations optimisation in production planning, supply chain and scheduling.

Copyright © 2020 Inderscience Enterprises Ltd.

Aldo Vecchietti is a Chemical Engineer from the Universidad Nacional del Litoral, Santa Fe, Argentina. He also obtained his PhD in Chemical Engineering at the same university. He is a Principal Researcher of the National Council for Technical and Scientific Research of Argentina (CONICET) at INGAR (CONICET-Universidad Tecnológica Nacional), where he is also the Institute Head. He is also a Professor in the Information System Engineering Department at Universidad Tecnológica Nacional, Santa Fe, Argentina. His expertise area is process system engineering, working on optimisation mathematical models for planning and scheduling of manufacturing and production companies and its supply chain. He has an extensive experience in consulting works with private production companies.

This paper is a revised and expanded version of a paper entitled 'Towards a reference architecture for advanced planning systems' presented at 18th International Conference of Enterprise Information Systems (ICEIS), Rome, Italy, 25–28 April 2016.

1 Introduction

Nowadays, small and medium-sized organisations (SMO) face a competitive business environment and are induced to optimise their operations with the aim of diminishing costs, growing profits, and increase customer satisfaction (Rodriguez and Vecchietti, 2013; Pattanayak et al., 2019). Advanced planning systems (APS) is a computer system that oversees the optimisation of the company's production operations. APSs are implemented and linked to the existing software environment, which often is an enterprise information systems (EIS).

However, the implementation of APS in SMO is a significant effort for two main reasons. First, the processes of the companies have characteristics that are particular of their business (Schönig et al., 2016). This implies that APS must include a customised optimisation model to fit the organisation's needs (Aslan et al., 2012; Lie et al., 2018). Although there are world class ERP systems and other proprietary software systems that offer packaged APSs (Myers et al., 2011; Stadtler, 2005), they are generic and cannot be adjusted to all possible business processes and its specific details. The second reason, as Lavastre et al. (2012) pointed out, is that acquiring a packaged APS is often out of scope for SMO. This happens not only because of their financial limitations but also because they have a reduced information and communications technology (ICT) sector with lower training when compared to larger organisations (Klára et al., 2011). As a result, many companies use spreadsheets for its APS applications (Cornelis de Man and Strandhagen, 2018), these authors remark about the need to adopt more sophisticated systems to replace the use of spreadsheets. Another option for SMO is to implement their own APS through custom developments (Hahn and Packowski, 2015). Nowadays, some efforts are directed to the development of a cloud-based APS to allow its implementation in SMO (Hsu et al., 2018).

The development of APS is not conventional or straightforward. APSs are linked to models for optimisation, simulation, and other mathematical methods (Vidoni and Vecchietti, 2015), which requires, as part of the development process, the knowledge of experts capable of generating them (Gayialis and Tatsiopoulos, 2004). This situation increases APSs complexity since it involves stakeholders having different views and

goals for the same project. The analysis and design of these systems need more research from a software engineering perspective (Zoryk-Schalla et al., 2004) to facilitate its implementation for SMO. To cover this gap, this article provides a reference architecture (RA), named APS-RA, that acts as a framework to assist developers in APS's system implementations, through a software architecture, specific to their organisation needs. Gupta et al. (2018) highlight the importance of software architecture for enterprise applications, they mention that "the efficiency of architectural and design attributes in enterprise applications is vital for a successful implementation."

APS-RA is positively evaluated by two different groups of stakeholders, using the widely accepted architecture trade-off analysis method (ATAM) (Kazman et al., 2000). The transition from the APS-RA to the specific implementation is also approached by providing a process that is compatible with any development lifecycle.

In summary, the goal of this proposal is to present an instrument to assist a wide range of SMO on the design, development and implementation of APSs, by providing a software architecture tailored to their needs.

This article is organised as follows. Section 2 presents the APS domain, discusses the problem to be solved and the main techniques applied. Then, Section 3 presents the proposed framework, named APS-RA, the design decisions adopted and how it can be adjusted to a specific APS domain; it also presents the evaluation process and the most relevant feedback. Section 4 provides a brief analysis of software development lifecycles (SDLCs) and describes the guidelines to use the RA. After that, in Section 5, those concepts are applied to an industrial case study where the results obtained are discussed. Finally, Section 6 and Section 7 present the discussion and conclusions of this work.

2 The challenge of APS

APSs are a computer support to simplify and automatise the solution of models that optimise a given company operation and present their results in a user-friendly interface. For this purpose, APSs obtain the required data from the current EIS and store back only the accepted results (Kallestrup et al., 2014). APSs can be implemented for a company's operations or its supply chains (Musselman et al., 2003; Stadtler, 2005; Anon., 2018), and they can optimise one or several operations, for different sections, such as production, logistics, stocks, and others (Meyr et al., 2014). Also, this can be done using a wide range of techniques, either by themselves or mixing them (Framinan and Ruiz, 2010). At this point, it is worth clarifying that this article uses the word 'models' as an umbrella term: they can be generated through any technique, such as mathematical programming, genetic algorithms, simulation, and many more.

As was expressed, the goal of this paper is to fill the gap existent in the literature, regarding the design and development of APSs (Zoryk-Schalla et al., 2004; Aslan et al., 2012; Kallestrup et al., 2014). This is done by providing a framework that can assist in the lifecycle of the system to be used by a varied type of organisations: manufacturing, logistic, retail, stores, and others. The pursued benefits are the reduction of the required effort to produce a well-formed and consistent architecture, allowing an early development and shorter lifecycle iterations.

Addressing this challenge implies dealing with two elements: first, to elucidate the domain of APSs, as a group of systems that have many features and goals in common,

but vary in particularities, second, to propose a flexible architectural framework capable to adapt to the needs of different company types.

This proposal is based on two different elements: first, it works with RA which is an abstraction from traditional software architectures, that defines terminology, common elements and functionality of the target domain, instead of a particular case (Martínez Fernández et al., 2013). Second, it targets the adaptability to different cases, through a what-if analysis of different situations to be considered, including the application and the type of company. This is achieved by documenting in the RA the variation points, which are templates that consider all possible situations for a given element, define its impact on the architecture and provide guidelines to be adapted to each one (Bachmann and Bass, 2001). In this proposal, the variation points are defined through three items (Clements et al., 2010):

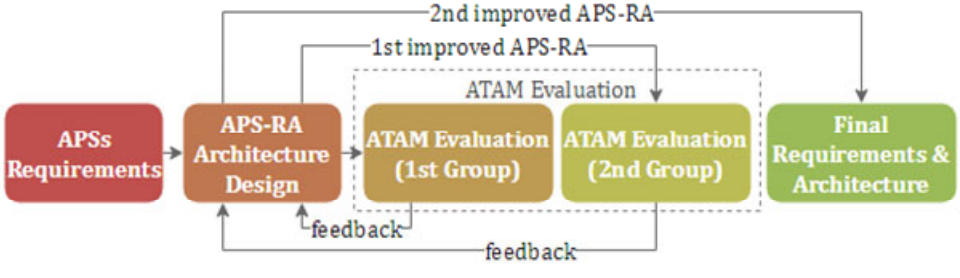
- *Options:* List the alternatives considered at this point. They are not exclusive, a variation point may need to apply one or more options at the same time, depending on the specific development case.
- *Effects:* Define the implications of every option of the variation point, it establishes the requirements needed for each option.
- *Execution:* Are step-by-step instructions, for each option, indicating how to modify the architecture to adapt the affected element to a specific requirement.

Variation points are essential elements of a RA, as they allow its adaptability.

3 APS-RA: reference architecture

The RA proposed in this article is generated through the iterative and incremental process that can be seen in Figure 1.

Figure 1 The iterative process followed to design APS-RA (see online version for colours)



The construction of APS-RA is developed using functional requirements and quality attributes elicited on previous work (Vidoni et al., 2018). Its presentation is organised with the '4 + 1' view model (Kruchten, 1995), as suggested by the standard 42010:2010 (ISO/IEC/IEEE, 2011), where each viewpoint is targeted to a different group of stakeholder.

Then, APS-RA is evaluated to validate and verify if it satisfies the elicited requirements. However, as previously stated, since APSs depend on the models used for the optimisation of operations, the experts that develop said models are also part of an

APS development project (Gayialis and Tatsiopoulos, 2004) and should be included in the evaluation. The selected evaluation process is ATAM (Kazman et al., 2000), and for APS-RA, this process is separated into two stages: the first one using a group of optimisation experts and focused on the APS-model-solver interaction and the second one with software engineers performing a traditional assessment. As a result, each evaluation stage generates a refined version of the framework.

To transmit the magnitude of APS-RA, Table 1 presents the number of diagrams, elements, variation points and stakeholders involved in each view. There are a high number of variation points to include numerous implementation alternatives, which can also be combined, increasing the APS-RA adaptability.

Table 1 The number of diagrams, elements, variation points and stakeholders of PS-RA

View	*Diagrams*	*Elements*	*V. points*	*Stakeholders*
Logical	1	33	12	End users, low-level managers
Development	2	34	7	Software engineers, testers, project team
Process	11	296	11	Software engineers, networks team, mid-level managers
Physical	1	22	16	End users, operators, developers, high-level managers
Scenarios	4	69	4	All stakeholders
Total	19	454	50	

However, although the APS-RA contains all the '4 + 1' views, this article only introduces those most useful to developers. The complete documentation of APS-RA and its requirements is available as a dataset (Vidoni et al., 2018).

3.1 Logical view

The logical view provides a direct match between functional requirements and software elements (Kruchten, 1995), and it is targeted towards those groups of stakeholders that require a complete vision of the APS domain. Figure 3 presents the view using a UML model diagram to show a domain abstraction, by detailing the logical behaviour or aspects (Object Management Group, 2015). In this diagram, the relationships between elements are kept to a minimum, as they are detailed in other views.

This view presents the APS with a three-layered architectural style. The presentation layer is organised with a model-view-controller pattern to offer a graphical user interface, while the data layer handles interactions with data sources; finally, the solving layer contains the business logic of the APS system and interacts with the previous ones. Therefore, the actors interact with different layers, depending on their relationship with the APS: users interacts with the system through the graphical interface on the presentation layer, while external sources of data – such as the data sources, among others – only interoperate through the data layer.

The next subsections present the most relevant packages of the solving layer and introduce the variability of the most significant elements. Some of the most relevant variation points are discussed and presented with flow diagrams. In there, diamonds with a × signify exclusive choices (only one option can be selected), and diamonds with a + denote that more than one choice can be applied at the same time.

3.1.1 Factory Planner

The package Factory Planner is vital for APSs because it embodies the requirements for each operation problem to be solved: planning tasks that can be implemented on specific companies through one or more instances of this package. Therefore, there must be at least one instance of this package. This is a variation point visible through the cardinality present in the diagram (Figure 3) and is presented as a flow in Figure 2.

Figure 2 Number and composition variability for the *Factory Planner* (see online version for colours)

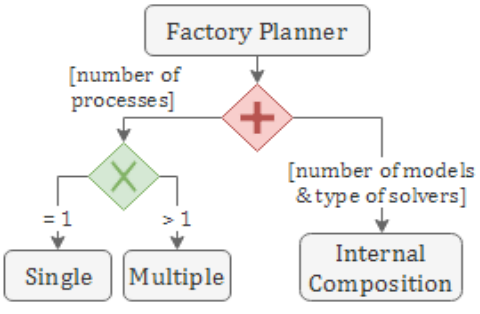
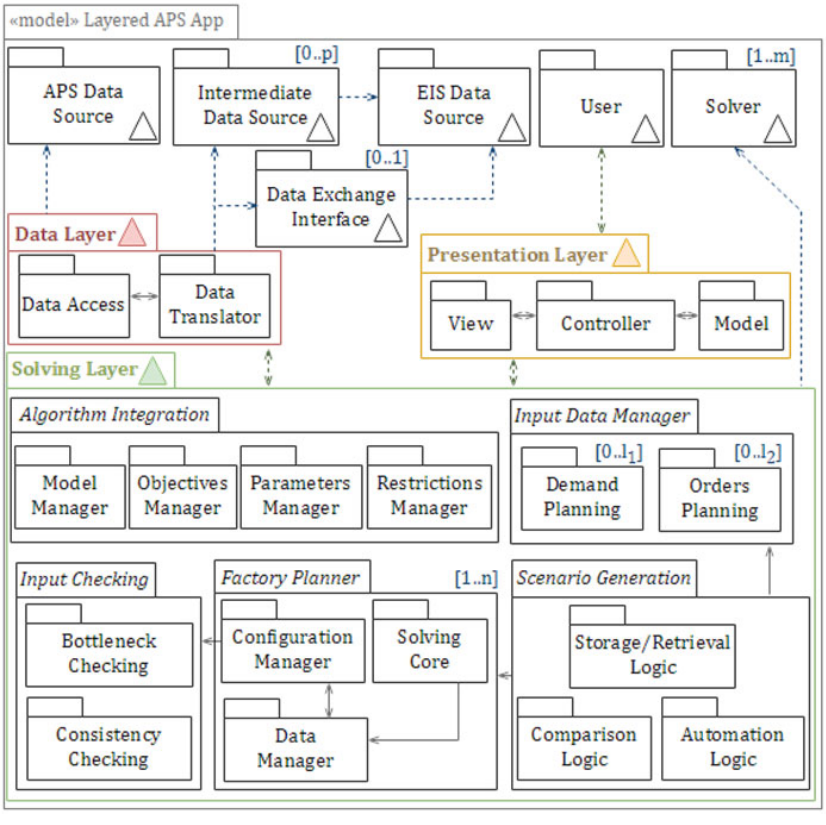


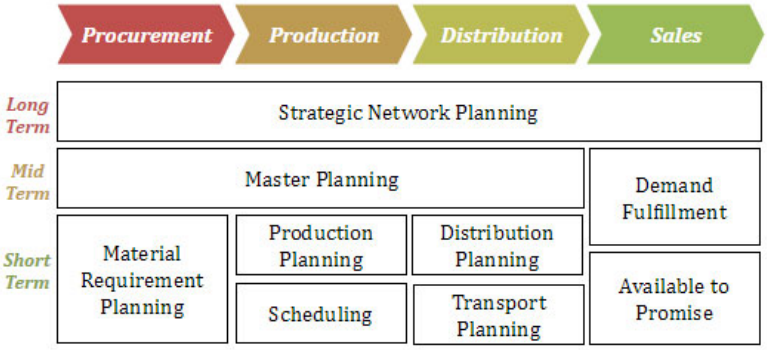
Figure 3 Model diagram for the logical view of the APS-RA (see online version for colours)



For example, a company that requires optimising its logistics operation and its material purchases will have an APS with two instances of Factory Planner, one for each operation.

Consequently, this package is related to the planning tasks of the supply chain matrix proposed by several authors (Stadtler, 2005; Hadaya and Pellerin, 2008; Stadtler et al., 2012; Fleischmann and Meyr, 2003), which can be seen in Figure 4. This matrix classifies and organises the different types of operations that can be optimised, according to the area they belong and the horizon time they target. This is highly relevant, as the matrix is widely accepted and mentioned in the APS literature (Fleischmann and Meyr, 2003; Hadaya and Pellerin, 2008; de Sousa et al., 2014).

Figure 4 Supply chain matrix and types of operation to be optimised (see online version for colours)



Thus, the APS-RA proposal includes the concepts of this matrix through the Factory Planner package and its variation points. The operations shown in the matrix of Figure 4 can be implemented for a company (or its supply chain), by creating one or more instances of Factory Planner.

Therefore, this approach allows focusing on the construction of the software-intensive elements, leaving the type of problem to solve to a concrete implementation. This variation point demonstrates the adaptability of the proposed APS-RA by not limiting the implementation decisions.

Factory Planner is composed of three types of components. A data manager that translates the model from the format used in the APS to the one required by the solver, a configuration manager to create a specific scenario by selecting a model and its input data, and a solving core that controls the solution of the created scenario.

Solving core variability

The solving core is a vital element of the Factory Planner, it is subjected to a great variability, since it can address a plethora of implementation alternatives.

A primary variation point is presented in Figure 5, which related to the type and location of the solver to use. There are two possible situations to evaluate: the number of available solvers, and their location. Regarding the latter, the alternatives are:

a the solver as an external tool, i.e., MATLAB, MS Excel, General Algebraic Modelling System (GAMS), or others – which mostly applies to packaged solving approaches

b the solver developed alongside the model, mostly for non-heuristic approaches.

Figure 5 The variability in the number of *solvers* and their locations (see online version for colours)

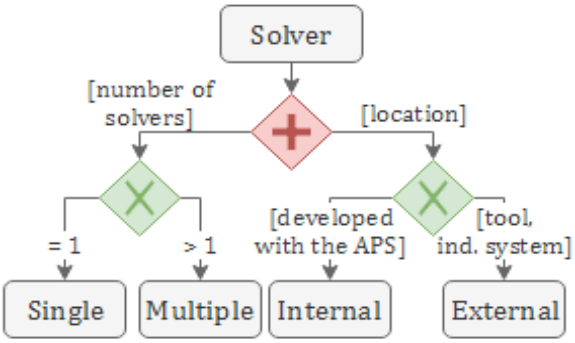
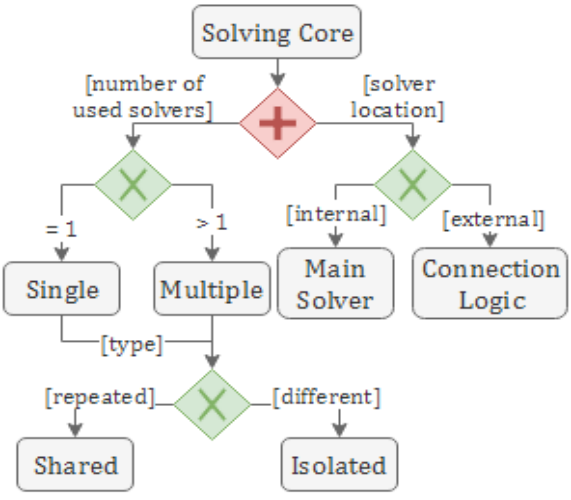


Figure 6 Composition, number and use variability of the *solving core* (see online version for colours)



These alternatives also affect the role of the solving core, generating an additional variation point, visible in Figure 6. There are three points of change. First, if the solver is external, then the core only needs to send and receive messages from it (acting as connection logic), but if the solver is developed with the model, the external actor does not exist, and the core becomes the principal solver. Second, the number of existing solving cores depends on the available solvers. Third, and related to this, a Factory Planner can have multiple models, each of them may use different solvers or share the same one; this sharing may also happen between Factory Planners.

For example, if all the models of the Factory Planner use the same solver, then the solving core and the solver can be implemented once and shared among all models. The opposite case is when each model requires a different solver, having as many solvers as models. Nonetheless, a Factory Planner may have more than one model, and thus, it can also have several solving cores. These possibilities can also be combined, i.e., in the case of multiple solvers, some may be external actors, and others may be internal components only.

This level of detail is considered for each variation point, on the elements of the APS-RA (see Table 1). Hence, this framework architecture provides a level of adaptability that is a novel approach to the APS domain.

3.1.2 Other packages

There are also other important features and functionalities of APS represented in this proposal. However, it is not the intention of this article to act as the architecture specification document; instead, it aims to present the main topics of APS's that are reflected in the RA. Therefore, the remaining packages are summarised in this section, and its complete specification is available as a dataset (Vidoni et al., 2018).

The package Scenario Generation is related to the generation and automation of scenarios for each operation to be optimised. It includes the logic to perform the automation of the scenario (automation logic), the configuration of each one by selecting the input data and the specification of the model parameters (on configuration logic), and the obtaining, translating and saving the data via the storage/retrieval logic. This package is vital to use the Factory Planners efficiently and it does not have many variation points.

An important aspect when performing an operation optimisation is to check input and output data to avoid errors and infeasible solutions. In this sense, the checking package contains the logic to evaluate the consistency of input data to avoid compromising computational running time of an optimisation that would be potentially infeasible. There are two types of evaluation: consistency is related to data checking, while bottleneck evaluates resources and machine overload. Both packages are also affected by variability; for instance, a company may choose not to perform checking, may not have the available data, or the models may not be prepared for such tasks.

Third, algorithm integration allows the replacement and update of operations models and components – such as objectives, restrictions and parameters default values – without requiring a new development iteration of the lifecycle. However, some of these features may be restricted by the solving approach, the available model, or other conditions. For example, models that implement their solver may be forced to be added on coding time, forcing a new iteration on the lifecycle.

Finally, each company may work with a different production strategy – make-to-order, make-to-stock or engineer-to-order, this can vary for each product. This is implemented on input data manager. However, since not all alternatives can be shown in the diagram, the logical view depicts the most complicated case. This happens when the demand planning sub-package is part of the APS, as stated to be the most common occurrence (Kallestrup et al., 2014), and the orders planning acts as a translator for the specific data incoming from the EIS.

3.2 Development view

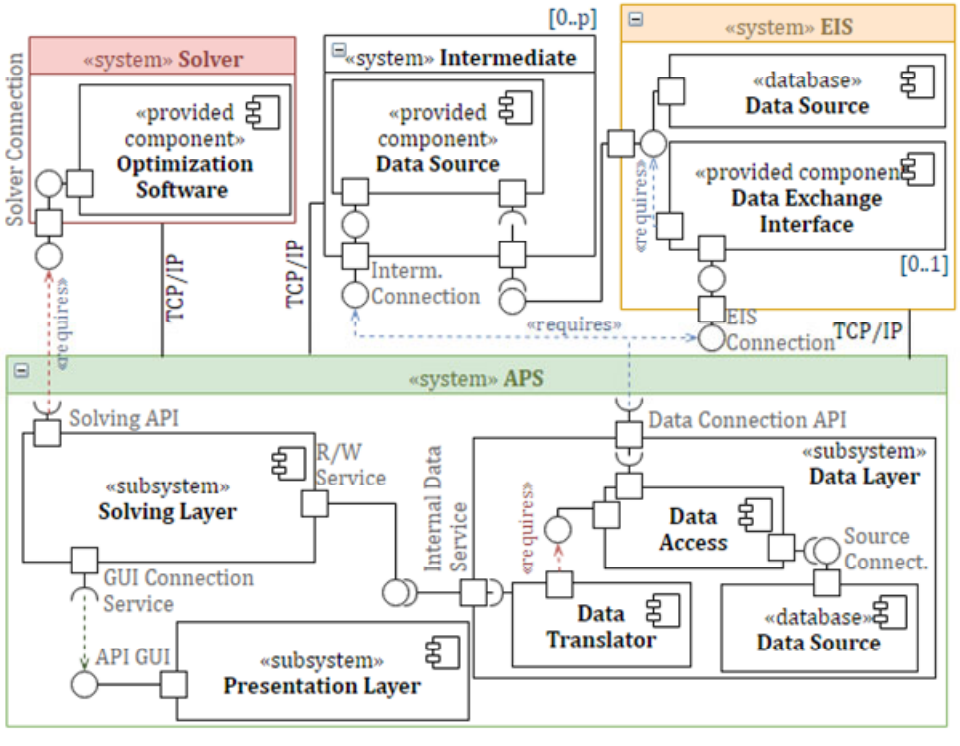
The goal of the development view is to showcase modules in a development environment, disclosing precedence relations among components (Kruchten, 1995). Thus, it is targeted mostly towards developers and is separated into two component diagrams.

The development view complements the logical view by detailing the relationship between elements, which simplifies decisions to move from design to development. Examples of this are the precedence relations between elements, prioritisation of components, development order and so on. Therefore, individual subsystem or components are not described again, as they are the same from the logical view, but presents a different perspective directed to developers.

3.2.1 General diagram

Figure 7 can be considered a context diagram, in which the APS is shown to interact with non-human actors and includes simplified relations among internal components. The precedence relationships are visible through the provided and required interfaces (notations and respectively) that represent requests and responses (Ivers et al., 2004) among the components.

Figure 7 First component diagram for the development view of APS-RA (see online version for colours)



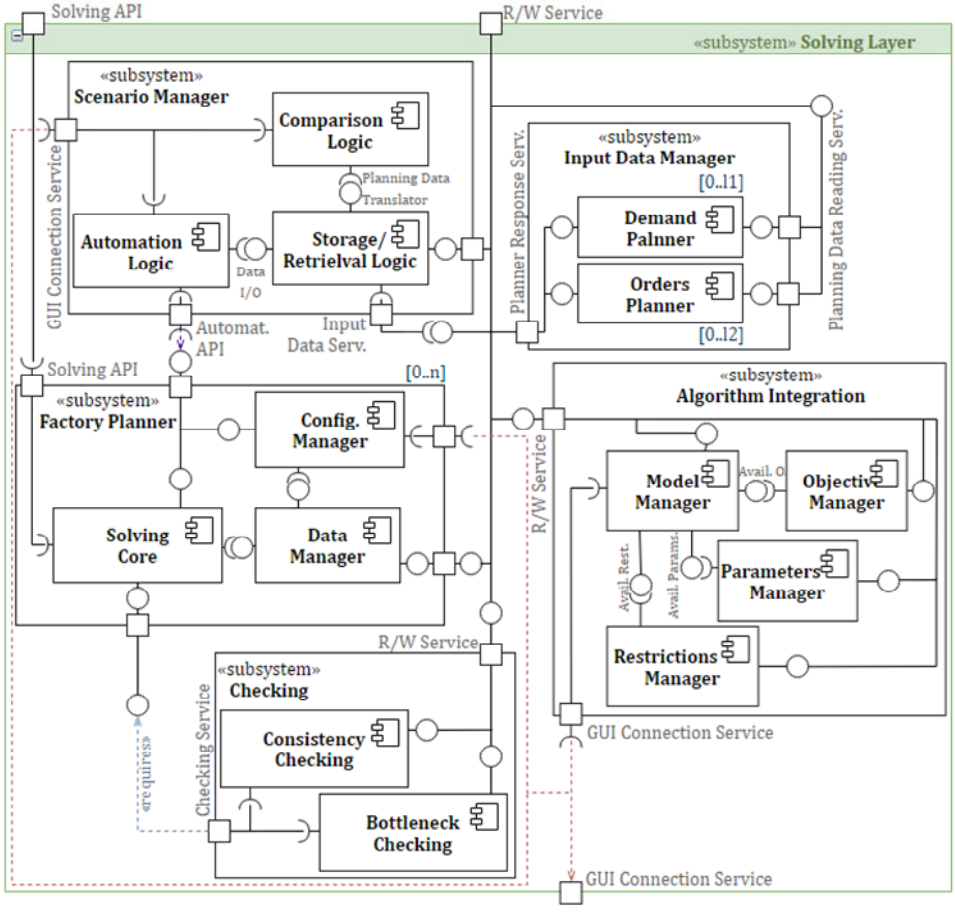
In Figure 7, it can be seen that to define the data access implementation, the data sources details must be specified first; likewise, the implementation of the solving core

connection with the solver depends on the latter. Using a particular example, in a company that uses two different database management systems (one for the EIS and another one for the APS), the packages data access and data translator will have two instances each, both of them connected with the corresponding pair.

3.2.2 Solving layer diagram

Figure 8 expands the solving layer, complementing the representation provided by the logical view, specifying more complex relationships. Thus, the connection to other layers (seen in Figure 7) is detailed, showing precisely which components should perform each specific link.

Figure 8 Second component diagram for the development view of the APS-RA (see online version for colours)



The relationship between the solver and the solving core is also an essential point. This view sheds more information regarding it, clarifying that the solving core component design depends on the implementation of the solver. This describes that the decision to use a given solving approach for a particular model impacts not only in the way that the

problem is translated to an optimisation model, but also the logic of the software component, and how to provide a seamless and transparent workflow to the user. Related to this, the sharing of solving cores between different Factory Planners addressed in Section 3.1.1 does not affect the relationships between those components and the other elements of the package. However, it does change the number of instances of the relations between them.

3.3 Architecture evaluation

This evaluation is performed by applying ATAM (Kazman et al., 2000). The method was selected since is a highly accepted (Dobrica and Niemela, 2002). It is not the goal of this article to describe its steps or its artefacts. Therefore, this section focuses on discussing the results and the feedback obtained through the evaluation. The generated artefacts are available as part of the dataset (Vidoni et al., 2018).

Also, an essential aspect of RAs is that they do not have individually identified stakeholders, but only target groups. As a result, both evaluation stages are performed by representatives of different disciplines, invited to take part in the project.

3.3.1 Stage 1

This first evaluation is done with seven participants, all known researchers with expertise in both optimisation and academy-industry collaborations in the area. Therefore, they work only with specific views of the APS-RA (logical, process and scenarios), and avoid specific software engineering steps, such as identifying architectural patterns and styles.

After completing the process, participants agree that APS-RA structure is flexible enough to be used on different implementations, with a wide range of combinations of target operations, models and solvers. Two reasons sustain this conclusion:

- The APS-RA definition does not restrict the technique to be used, which can be concluded by analysing the adaptability of the APS-RA elements, and the functionalities assigned to them.
- The Factory Planner element and its variability enable the adaptation of APS-RA to a wide range of processes. By adding instances of this element, the architecture can include different areas requiring optimisation, in both a single organisation or in a supply chain.

The evaluators requested minor changes to the documentation definition, to improve its readability. Likewise, they suggest three features that are not considered in the initial version:

a Constraints grouping: The model's restrictions must be grouped by topic when the user is selecting or updating them.

b Models versioning: A new model version must be generated, instead of deleting or replacing it.

c Traceability: Applying traceability to the solutions by storing which model version is used to obtain a given solution.

3.3.2 Stage 2

This is done with a group of ten advanced system engineering students and two architecture professionals. They work with the complete documentation of the first improved APS-RA, generated after applying the feedback obtained from the previous evaluation.

To assess the APS-RA from a wide range of perspectives, participants are divided into traditional software development roles, generating teams, and evaluate the architecture from that standpoint. These are: 'data management', 'resources management', 'software development', 'systems integration' and 'privacy and security'. This approach has two main advantages: tackling critical areas of any software-intensive system, and ensuring that all aspects are equally evaluated. However, the results of each team are peer-reviewed continuously by the other teams, to agree on necessary concessions and guarantee they all work towards the same goal.

Participants agree that the APS-RA successfully reflects its requirements, and using its variation points, can act as a framework for the APS domain. Numerous reasons sustain this valuation:

a Variation points consider enough possibilities, and by combining them, it is feasible to adapt APS-RA to a plethora of requirements. Several tactics, risks and sensibility points are also linked to the variability: this means that the evaluation results provide additional insight into the consequences of each possible choice for the existent variation points.

b The structured documentation follows a standardised and widespread approach, which corresponds to the 'views and beyond' style (Clements et al., 2010). The documents contain a precise definition of the APS-RA, sufficient to apply in practice. Also, it includes enough information for developers to be used in a specific implementation.

c Some strategic architectural patterns, such as layers (Figure 2) disclose a logical distribution of elements and can be adapted to each implementation, e.g., by selecting different programming languages and frameworks. For instance, a client-server application can be generated by unifying the content of two layers (i.e., presentation layer and solving layer), while a web application would directly translate the logical layers into physical code-specific tiers.

They propose minor modifications, such as rewriting sentences to increase the documentation readability and suggest distributing the list of risks, sensibilities and trade-offs resulting from this evaluation, alongside with the documentation. The objective is to make this information available to improve decision making when designing a specific system based on the APS-RA.

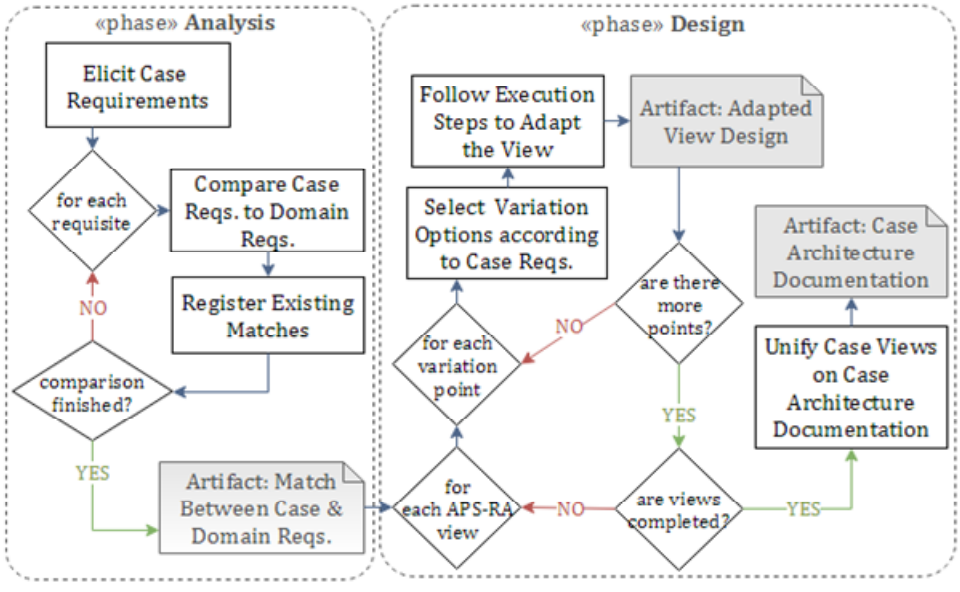
After this process, this feedback is applied to the first improved APS-RA, obtaining the second improved version, which is the one discussed in this article.

4 Application process

The use of APS-RA needs to be integrated into the SDLC of the organisation that implements a specific APS. This is done with a generic process that fits the majority of

SDLCs and is centred on the shared phases of analysis, design and development, instead on a particular SDLC techniques. This is due to most lifecycles share the same overall stages (Soepardi et al., 2018). Figure 9 is a flow diagram that describes this process, where the diamonds notation represents loops or decisions, and the paper notation describes the artefacts.

Figure 9 Steps to use the APS-RA during analysis and design phases (see online version for colours)



In the diagram, the 'case requirements' includes those elicited for the particular implementation or case study, while 'domain requirements' are those used to develop the APS-RA, and available on its documentation. Also, this proposal suggests the generation of three additional artefacts:

- Matching between case and domain requirements: A structured documentation of the correspondence between the case and domain requirements. This can be done using different formats, such as a spreadsheet, a list, or any other. However, this artefact must allow:
  - a to summarise the selected options on each variation point and link them to the requirements
  - b to use these choices to obtain the execution steps from the APS-RA documentation.
- Adapted view design: This is not an additional artefact. It consists of diagrams that are part of the case design and architecture. Each view is generated iteratively and incrementally by following the execution steps of the selected options of a variation point and becomes the general model of the case under development. Also, since points are complementary among views, the project team can check the options across views to ensure consistency.

- Case architecture documentation: This represents an architecture draft, as such, it is not final and can be modified, updated and extended during the lifecycle. For instance, in object-oriented development, the logical view may contain the package diagram generated with the APS-RA, later can be appended with relevant class diagrams.

As a result, a project implemented from the APS-RA that follows this template can get the benefit from a shortened design phase. Instead of creating a specific case architecture from scratch, taking decisions and requiring analysis of possible risks, the project team can make adaptations from the APS-RA by following the template introduced in this section, and the execution steps of the selected options for each variation point. Consequently, the development phase can begin early but, at the same time, work with the knowledge included in a sound, consistent and well-formed architecture.

5 Case study development

The target manufacturing company – referred to by the pseudonym LC – produces corrugated board boxes, customised to each customer's needs. Thus, they work only in a make-to-order strategy. They have two different operations to optimise: box cutting (creating the cardboard from papers in the stock and cutting the box pattern) and box printing (adding labels and images).

This is an academy-industry collaboration between LC and the Academia (a local university research team), focused on updating an existing version of a system previously developed in-house. Hence, the project team is composed of LC's developers and both optimisation experts and developers from the Academia. The new version of the system is called LGO and has two crucial aspects. First, both the EIS and APS are developed together, with the latter being an additional module to the EIS named planning module. Second, it contains two operations to be optimised, box cutting and box printing, each one with three models: production planning, scheduling and rescheduling. The mathematical models are developed with GAMS (GAMS Development Corp., 2011).

For this project, the selected lifecycle is feature driven development (FDD), which is an agile methodology (Conboy, 2009), already adopted at LC. Therefore, the application process is merged with FDD, to design the planning module. The steps of Figure 9 are applied in the following manner:

a Since the planning module (the APS) is considered part of the whole EIS system and developed as a module or subsystem, the 'case requirements' regarding its functionalities are elicited all together at the beginning of the project. Then, they are converted to the list of features used to plan the iterations.

b At the start of the planning module iteration, the 'case requirements' – now translated into features – are compared to the APS-RA 'domain requirements'

   This generates, for each view, a table disclosing each element with its selected options and the reasoning about the requirements leading to it. Table 2 is one of the generated comparisons, in particular for the logical view of LGO planning module.

c After obtaining all of the tables during the iteration's design phase, the project team uses them to create the main diagrams of each view. These diagrams are later complemented with additional ones, such as class diagrams, state diagrams, and so on.

Regarding this, and comparing the specific architecture with the APS-RA, some of the resulting diagrams are presented and discussed.

Table 2 Selected options for the variation points of APS-RA for the logical view of LGO planning module

LGO planning module		
Element	*Option*	*Reasoning*
Intermediate	Null cardinality	It is not used on LGO.
DEI	Null cardinality	As LGO has EIS and APS unified, no interface is required.
Solver	Only one external actor	Both operations are composed of three models each, which work sequentially (planning, scheduling and rescheduling). Since all existent models work with the same version of GAMS, there is only one external *solver* actor with a corresponding *solving core* component, shared across all instances of *Factory Planner*.
Presentation	MVC	The MVC pattern is used and supported by Spring MVC. Therefore, the notional relations shown on the APS-RA logical view (Figure 1) are replaced by Spring relations.
Consistency checking	Software checking	Checking is done by software, by obtaining data for each scenario. This includes input data obtained from the database and user-input parameters.
Bottleneck checking	None	There is no bottleneck checking due to the lack of required information on the database to perform it.
Factory Planner	More than one	There are two operations to be optimised and therefore two *Factory Planner* instances: box cutting and box printing.
Solving core	Internal component	Since both *Factory Planners* work with the same solver, the *solving core* has only one instance, which is internal and shared among both *Factory Planners*.
Demand planning	Null	LC does not use sales forecasts and only works as made-to-order.
Orders planning	Internal EIS module	The package organises and translates data obtained from the EIS, acting only as a translator.

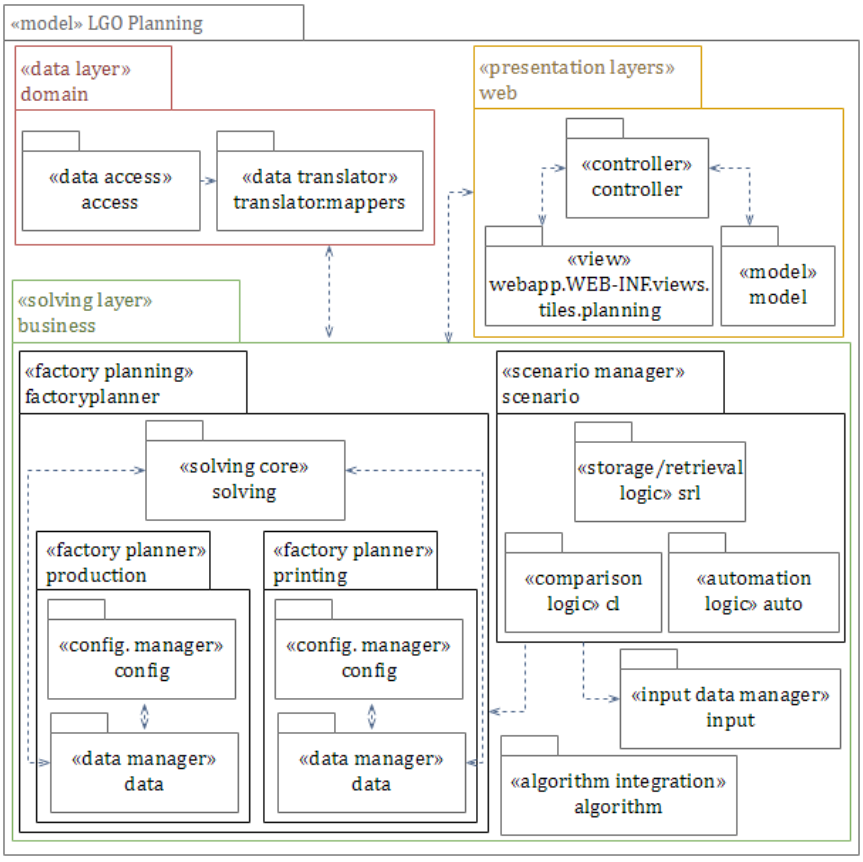
5.1 Logical and development views

First, Figure 10 presents the logical view of LGO planning module, using a UML package diagram with the aim of later adding a class diagram. The APS-RA names are included in the packages as stereotypes, to add a visual matching. It is worth noting some differences with the APS-RA logical view, as the specific package diagram does not have actors, and does not need cardinalities.

However, some of the choices are a consequence of the technology selected for the implementation (such as the names of the packages), while others depend on business restrictions.

Regarding the latter, several examples can be highlighted. Since both Factory Planner instances use the same external solver actor, there is only one solving core that works for both of them. Another aspect is the simplification of the input data manager, given the fact that only the orders planning is being used at LC Company. Also, the decision in LGO is to perform the data checking by software instead of the model, allowing the latter to focus on the operation optimisation leaving the software its fundamental task; this change not only reduces waiting time but improves performance and modularity.

Figure 10 Logical view of LGO planning module (see online version for colours)



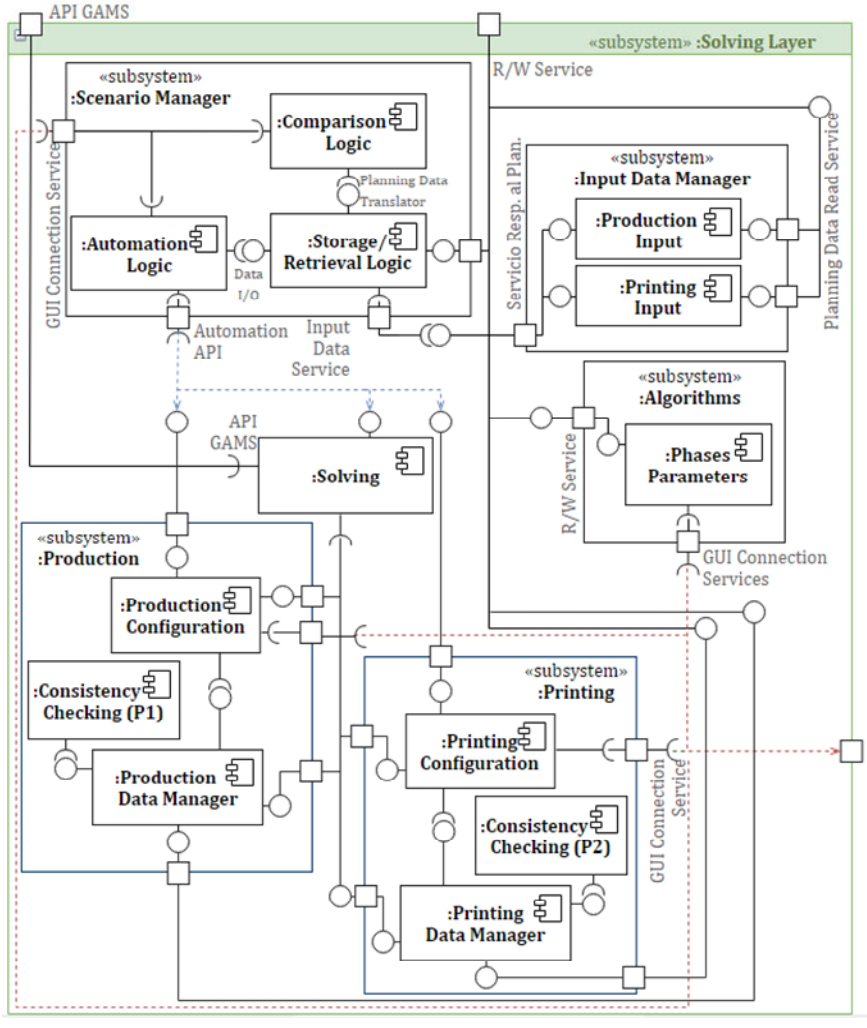
This initial logical view is seamlessly generated from the APS-RA, assisting on kick-starting the architecture design.

Second, Figure 11 presents one of the development view diagrams: the one focused on the solving layer of LGO planning module. As the architectural views are complementary, several of the decisions taken in the logical view also impact this diagram, and other choices are only visible from this perspective.

As an example of the latter, there are two instances of input manager: one for each Factory Planner. However, although the production input obtains sale orders, printing input works with production orders generated through the production optimisation point.

Similarly, the data verification is included directly on each Factory Planner, instead of having it outside, as an independent subsystem.

Figure 11 Solving layer diagram, for the development view of LGO planning module (see online version for colours)



Also, the representation of the solving core as a shared component corresponding with the external solver (GAMS), and with the decisions showcased on the logical view.

The implementation of the requisites related to the automation and scenario allows the specification of four available objectives; the parameters can only be configured using a fixed value. This means that for each setup, each model provides only one solution.

5.2 General assessment

The process applied to this case study presents several advantages compared to a traditional development without using the APS-RA.

First, the project team is assembled with developers of LC and the Academia, optimisation experts, as well as planners and operators, where the latter give insight on the targeted industrial processes. Hence, the original APS-RA views, in particular, scenario and process, are used as part of the elicitation process, allowing external stakeholders to obtain a fast and tangible view of the system and simplify the process to decide if a functionality is required or not for LGO planning module, and how it should work.

Second, APS-RA provides the development team with a common ground of understanding and is used as a glossary to define terms and functionalities to work with. This is particularly useful to reach an understanding of how mathematical programming works and how the resulting models can be integrated. This reduces the time and economic/professional cost required to create a specific architecture. In practice, the planning module development (without the included EIS) is scheduled to last 44 working days, but it is completed in 41 days using two out of five days allocated to the design. This is considered a success for LC Company, where the software development always lasted more time than planned.

Third, the process of matching LGO's requirements to the APS-RA's is straightforward and produces the main diagrams of each view, obtaining an overall model of the architecture. These main diagrams act as a base, and later the project team decides to add only specific diagrams: three class diagrams and a state diagram for the logical view, a deployment diagram for the physical view, and sequence diagrams complementary of the process view.

Fourth and finally, by using the results of the ATAM evaluation to determine the risk points of the architecture design, the project team can handle these issues before they can impact the finished product. For instance, as LGO is a web system, ATAM artefacts dealing with security qualities for a web version of an APS are used by the developers to test LGO planning module under different circumstances, and by the management to take specific decisions regarding possible risks.

6 Discussion

Working with APSs involves several challenges, because the implementation of APSs varies from one another, given the organisation requisites. However, when focusing on the APS domain, instead of addressing each specific APS development, it is possible to see common aspects that characterise this type of systems. APS-RA captures these common characteristics and proposes a set of artefacts and views adapted to the different stakeholders which are involved in an APS implementation.

When analysing the academic literature, it is possible to find some proposals regarding APS architecture (Wang et al., 2004; Wiers, 2002; de Sousa et al., 2014; Framinan and Ruiz, 2010). Most of them are specified for concrete case studies with low possibilities to adapt to other applications, or they are not addressed from a software engineering perspective, because they do not propose any architectural pattern, or lack of a standardised evaluation through an accepted method. APS-RA targets these issues and it is different from existing proposals by using architectural concepts and variation points to provide flexibility. Also, it has been evaluated with ATAM twice, using different sets of participants working with different perspectives to enhance APS-RA capabilities by means of the assessments.

Nevertheless, one feature that characterise the APSs domain is the use of models to optimise or solve specific operations for a company (Harjunkoski et al., 2009; Stadtler, 2005). However, APS-RA does not include any model, instead, the proposal allows the linkage different models that adapts to a plethora of situations. This is highly relevant because those particularities are one of the reasons that lead companies to develop their own APSs instead of using packaged proprietary solutions.

Regarding structural aspects of the APS-RA, some properties must be highlighted:

- Not all requirements need to be implemented: as in the case study, a company may ignore a requisite, and use the variation points to remove unused elements. This may happen because the requirements used for the APS domain are extensive enough to address many situations and need to contemplate possibilities with and without those features.
- Since each viewpoint has multiple variation points, it is not possible to represent all of them in only one diagram. Also, the combination of the options of different variation points generates an exponential amount of possibilities. It is not viable to create one diagram for each possible combination. Therefore, the diagrams show either the most complicated case, or the most used case, according to academic reports. In any case, the reasoning is also detailed on the APS-RA documentation.
- The selection of RAs directly contributes to the APS domain, they bring substantial advantages to the design and development of systems belonging to the domain they are developed for, both economical and regarding to the required professional effort (Galster and Avgeriou, 2011; Martínez Fernández et al., 2013; Angelov et al., 2012). Also, practical application has confirmed these results (Martínez Fernández et al., 2012; Behere et al., 2013).

7 Conclusions

This article proposes a RA for the domain of APS, named APS-RA, which can be used to assist the analysis, design and implementation of this type of systems. The objectives are to reduce the required time to generate an appropriate design and ensure a satisfactory development. APS-RA contributes to close the gap on the APS literature by specifying a RA from a software engineering point of view and provides a framework to assist small and medium organisations in the development of their own specific APSs.

Thus, this proposal works with previously elicited requirements to design the RA. Since the target is the APS domain, assisting as many organisations as possible, a wide range of alternatives is considered on the APS-RA by using variation points. Additionally, this architecture follows practices recommended by international standards, such as the ISO/IEC/IEEE 42010:2011 for software architecture.

The features of APS-RA were evaluated twice, with different groups of stakeholders, using the established method ATAM. This is an important difference respect to other proposals. An implementation process is also provided, detailing how the APS-RA can be used during analysis and design phases on APS software development projects; since the proposed process is not limited to only lifecycle, it can be applied to most current methodologies.

Finally, a case study is successfully implemented, based on a real-life small-sized local industrial company, and demonstrates how the APS-RA is satisfactorily used. This application proved to be decisive, with several benefits worth of being mentioned. First, the use of APS-RA effectively reduced the time required to produce a working application, allowing more time to the development and testing; implying a cost reduction in the project. Second, the project team had a shorter design phase, generating a consistent architecture for LGO. Third, variation points were able to adapt the APS-RA to the requirements of the company and provide enough guidance for the project team to use it successfully. Although APS-RA is being applied to some other local projects it requires more practical testing, as this remains a line of future work.

References

Angelov, S., Grefen, P. and Greefhorst, D. (2012) 'A framework for analysis and design of software reference architectures', *Information and Software Technology*, Vol. 54, No. 4, pp.417–431.

Anon. (2018) 'SCP-matrix based shipyard APS design: application to long-term production plan', *International Journal of Naval Architecture and Ocean Engineering*, Vol. 10, No. 6, pp.741–761.

Aslan, B., Stevenson, M. and Hendry, L.C. (2012) 'Enterprise resource planning systems: an assessment of applicability to make-to-order companies', *Computers in Industry*, Vol. 63, No. 7, pp.692–705.

Bachmann, F. and Bass, L. (2001) 'Managing variability in software architectures', *ACM SIGSOFT Software Engineering Notes*, Vol. 26, No. 3, pp.126–132.

Behere, S., Törngren, M. and Chen, D-J. (2013) 'A reference architecture for cooperative driving', *Journal of Systems Architecture*, Vol. 59, No. 10C, pp.1095–1112.

Clements, P. et al. (2010) *Documenting Software Architectures: Views and Beyond*, 2nd ed., Addison-Wesley Professional, Pittsburg, USA.

Conboy, K. (2009) 'Agility from first principles: reconstructing the concept of agility in information systems development', *Information Systems Research*, Vol. 20, No. 3, pp.329–354.

Cornelis de Man, J. and Strandhagen, J. (2018) 'Spreadsheet application still dominates enterprise resource planning and advanced planning systems', *IFAC-PapersOnLine*, Vol. 51, No. 11, pp.1224–1229.

de Sousa, T. et al. (2014) 'An overview of the advanced planning and scheduling systems', *Independent Journal of Management & Production*, Vol. 5, No. 4, pp.1032–1049.

Dobrica, L. and Niemela, E. (2002) 'A survey on software architecture analysis methods', *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, pp.638–653.

Fleischmann, B. and Meyr, H. (2003) 'Planning hierarchy, modeling and advanced planning systems', *Handbooks in Operations Research and Management Science*, Vol. 11, pp.455–523, Elsevier, DOI: https://doi.org/10.1016/S0927-0507(03)11009-2.

Framinan, J.M. and Ruiz, R. (2010) 'Architecture of manufacturing scheduling systems: literature review and an integrated proposal', *European Journal of Operational Research*, Vol. 205, No. 2, pp.237–246.

Galster, M. and Avgeriou, P. (2011) *Empirically-Grounded Reference Architectures: A Proposal*, pp.153–158, ACM New York, Boulder, USA.

GAMS Development Corp. (2011) *General Algebraic Modeling System (GAMS) [Computer Software]* [online] https://www.gams.com/latest/docs/RN_237.html (accessed June 2018).

Gayialis, S.P. and Tatsiopoulos, I.P. (2004) 'Design of an IT-driven decision support system for vehicle routing and scheduling', *European Journal of Operational Research*, Vol. 152, No. 2, pp.382–398.

Gupta, V., Kapur, P. and Kumar, D. (2018) 'Measuring architecture and design efficiency for enterprise applications', *International Journal of Industrial and Systems Engineering*, Vol. 28, No. 4, pp.494–529.

Hadaya, P. and Pellerin, R. (2008) *Determinants of Advance Planning and Scheduling Systems Adoption*, pp.494–499, IEEE, Sliema, Malta.

Hahn, G.J. and Packowski, J. (2015) 'A perspective on applications of in-memory analytics in supply chain management', *Decision Support Systems*, Vol. 76, pp.45–52, DOI: https://doi.org/10.1016/j.dss.2015.01.003.

Harjunkoski, I., Nyström, R. and Horch, A. (2009) 'Integration of scheduling and control – theory or practice?', *Computers & Chemical Engineering*, Vol. 33, No. 12, pp.1909–1918.

Hsu, T-H., Wang, L-C. and Chu, P-C. (2018) 'Development of a cloud-based advanced planning and scheduling system', *Procedia Manufacturing*, Vol. 17, pp.427–434, DOI: https://doi.org/10.1016/j.promfg.2018.10.066.

ISO/IEC/IEEE (2011) *42010:2011 – Systems and Software Engineering – Architecture Description*, International Standardization Organization, Switzerland.

Ivers, J. et al. (2004) *Documenting Component and Connector Views with UML 2.0*, Software Engineering Institute, Pittsburg, USA.

Kallestrup, K.B., Lynge, L.H., Akkerman, R. and Oddsdottir, T.A. (2014) 'Decision support in hierarchical planning systems: the case of procurement planning in oil refining industries', *Decision Support Systems*, Vol. 68, pp.49–63, DOI: https://doi.org/10.1016/j.dss.2014.09.003.

Kazman, R., Klein, M. and Clements, P. (2000) *ATAM: Method for Architecture Evaluation*, s.n., Pittsburg, USA.

Klára, A., Lubos, P. and Jindrich, T. (2011) 'Long term growth of SME from the view of ICT competencies and web presentations', *Ekonomie a Management*, Vol. 14, No. 4, pp.125–139.

Kruchten, P.B. (1995) 'The 4+1 view model of software architecture', *IEEE Software*, Vol. 12, No. 6, pp.42–50.

Lavastre, O., Gunasekaran, A. and Spalanzani, A. (2012) 'Supply chain risk management in French companies', *Decision Support Systems*, Vol. 52, No. 4, pp.828–838.

Lie, L. et al. (2018) 'Enhancing remanufacturing efficiency in Malaysia through a knowledge support system: a case study of brake callipers', *International Journal of Industrial and Systems Engineering*, Vol. 28, No. 4, pp.451–467.

Martínez Fernández, S. et al. (2013) *A Framework for Software Reference Architecture Analysis and Review*, pp.89–102, s.n., Montevideo, Uruguay.

Martínez Fernández, S., Ayala Martínez, C. and Franch Gutiérrez, J. (2012) *A Reuse-Based Economic Model for Software Reference Architectures*, GESSI – Grup d'Enginyeria del Software i dels Serveis, Barcelona, Spain.

Meyr, H., Wagner, M. and Rohde, J. (2014) 'Structure of advanced planning systems', in Stadtler, H., Kilger, C. and Meyr, H. (Eds.): *Supply Chain Management and Advanced Planning*, pp.99–106, Springer, Berlin, Heidelberg.

Musselman, K., O'Reilly, J. and Duket, S. (2003) *The Role of Simulation in Advanced Planning and Scheduling*, pp.1825–1830, IEEE, San Diego, USA.

Myers, T., MacLean, K. and Westgate, D. (2011) *Oracle Advanced Supply Chain Planning Implementation and User's Guide*, Oracle Corporation, Redwood City, CA, USA.

Object Management Group (2015) *OMG Unified Modeling Language TM (OMG UML)*, 2.5 ed., Object Management Group.

Pattanayak, A., Prakash, A. and Mohanty, R. (2019) 'Managing cost of quality in supply chain: a case study', *International Journal of Industrial and Systems Engineering*, Vol. 31, No. 3, pp.304–323.

Rodriguez, M.A. and Vecchietti, A. (2013) 'Integrated planning and scheduling with due dates in the corrugated board boxes industry', *Industrial & Engineering Chemical Research*, Vol. 52, No. 2, pp.847–860.

Schönig, S., Cabanillas, C., Jablonski, S. and Mendling, J. (2016) 'A framework for efficiently mining the organisational perspective of business processes', *Decision Support Systems*, Vol. 89, pp.87–97, DOI: https://doi.org/10.1016/j.dss.2016.06.012.

Soepardi, A., Pratikto, P., Santoso, P. and Tama, I. (2018) 'An updated literature review of agile manufacturing: classification and trends', *International Journal of Industrial and Systems Engineering*, Vol. 29, No. 1, pp.95–126.

Stadtler, H. (2005) 'Supply chain management and advanced planning – basics, overview and challenges', *European Journal of Operational Research*, Vol. 3, No. 16, pp.575–588.

Stadtler, H. et al. (2012) *Advanced Planning in Supply Chains*, 1st ed., Springer-Verlag, Berlin, Heidelberg, Germany.

Vidoni, M. and Vecchietti, A. (2015) 'A systemic approach to define and characterize advanced planning systems (APS)', *Computers & Industrial Engineering*, Vol. 90, pp.326–338, DOI: https://doi.org/10.1016/j.cie.2015.10.006.

Vidoni, M., Montagna, J. and Vecchietti, A. (2018) *APS-RA: Architecture Documentation & Evaluation*, Mendeley Datasets.

Wang, F., Chua, T., Liu, W. and Yan, W. (2004) *An APS Architecture for Web Services Flased Enterprise Integration*, pp.867–872, IEEE, Singapore.

Wiers, V. (2002) 'A case study on the integration of APS and ERP in a steel processing plant', *Production Planning & Control*, Vol. 13, No. 6, pp.552–560.

Zoryk-Schalla, A.J., Fransoo, J.C. and de Kok, T.G. (2004) 'Modeling the planning process in advanced planning systems', *Information & Management*, Vol. 42, No. 1, pp.75–87.