Towards an Agile Lifecycle in Operation Research Projects

Melina Vidoni, Maria Laura Cunico and Aldo Vecchietti
Institute of Design and Development, INGAR CONICET-UTN, Santa Fe, Argentina

Keywords: Operations Research, Decision Support Systems, Systems Engineering.

Abstract: Often, Operation Research (OR) interventions focus more on solving a specific problem than addressing the project as a whole. Even more, developers do not acknowledge OR models as systems that are part of an organisation. The lack of a methodology guiding the project complicates the introduction of changes in the model due to alterations in the requirements. However, these issues have already been acknowledged, addressed and solved in the Software Engineering (SE) discipline. Thus, considering the current contributions from SE to OR projects, and the solutions offered by the first, this article analyses more deeply the similarities existing in the lifecycles of projects aiming to narrow the gap that exists in OR research, due to the lack of project methodologies. A proposal is made regarding the flow of information refinement and lifecycle phases predominant in OR projects; an initial theoretical adaptation of Feature Driven Development showcases their potential and possibilities. After this, current limitations and future works are discussed.

1 INTRODUCTION

The common practice of the developers of optimisation programs -Operation Research (OR) models- is to concentrate the efforts in understanding the problem complexity from imprecise and incomplete requirements, to pose model that can return results that are at least feasible. Therefore, interventions lose the integral vision of the project, and focus only on the mathematical code, leading to modellers not acknowledging models as systems in themselves. Thus, the phases of the projects are not explicitly identified, and the advantages of a methodical and orderly development are abandoned.

This conception of the models in expert mode (Franco and Montibeller, 2010) makes more difficult to face and incorporate any type of changes, prone to occur in the environments in which the organisations develop their activities (Checkland and Poulter, 2010; Franco, 2013; Eden and Ackermann, 2004).

However, similar issues have already been acknowledged and approached in Software Engineering (SE), generating several widely accepted solutions. Those are visible in the evolution of software lifecycles, from Royce's waterfall, going through incremental, spiral, and towards agile and leagile methods (Munassar and Govardhan, 2010).

In particular, *agile methodologies* (Coram and Bohner, 2005) encourage accepting requirements at any stage of the development process, by actively involving customers (Chow and Cao, 2008).

Thus, considering that OR models must be treated as projects, able to adapt to change, and identify and follow phases to obtain a methodical and ordered development method, the SE approach thought agility to this issue becomes highly relevant.

Even more, authors in the OR field recognise the importance of SE practices and its contributions. They state it includes a rich set of techniques, concepts, experiences and methodologies drawn from its relationship with other disciplines (Mingers, 2001), and that it may bring significant and positive development to OR processes (Mingers and White, 2010). Also, Marttunen et al. (2017) point out that future research should consider a more comprehensive view of the project. More practically, SE methodologies have been fruitfully applied in different interventions and projects (Ormerod, 2008). All this, added to the correlation existing between systems and models, suggests that OR projects and models could benefit positively from including these practices and moving into a global project approach.

Therefore, this position proposes to use the experience obtained in SE regarding agility and project management, and apply it to OR projects. Although this is centred on the development of OR-models, it can also be applied to projects that implement both software and optimisation models at the same time, by integrating the latter to the current software lifecycle.

Vidoni, M., Cunico, M. and Vecchietti, A.
Towards an Agile Lifecycle in Operation Research Projects.
DOI: 10.5220/0006760504320440
In *Proceedings of the 20th International Conference on Enterprise Information Systems (ICEIS 2018)*, pages 432-440
ISBN: 978-989-758-298-1

Therefore, it first identifies the phases of OR lifecycles, and their information refinement flow, in the attempt to reduce the current gap in this subject. Feature Driven Development (FDD) is used as an example of adapting these concepts into SE lifecycles, as the first step towards a framework to adapt any agile methodology for OR projects.

2 LIFECYCLE ANALYSIS

There are currently many agile methodologies (AM), each one present benefits and drawbacks (Dybå and Dingsøyr, 2008). Nonetheless, all of their lifecycles share the same basic phases, regardless of their implementation (Leau et al., 2012).

Since this work aims to propose a project methodology to the development and implementation of an OR model, the first step is to adapt AMs to discover, define and establish the phases that are part of any process, and their information. This is an issue not yet covered, even when even it is recognised as a crucial aspect (von Winterfeldt and Fasolo, 2009).

2.1 Information Refinement

Every OR project implies a transformation of information (Ormerod, 2008): it starts with an idea and aims to obtain specifics answers -an optimisation model, an integrated DSS, and so on- (von Winterfeldt and Fasolo, 2009). For this, there must be a refinement and evolution of information, with intermediate states. This becomes the goal of a given phase during the development, contributing to the validation and verification of the obtained model.

Figure 1 summarises the flow affecting this proposed refinement of OR information. In this figure are detailed the relationships upon which the validation and verification are performed. The states are:

- Ideas: A proposal for the project from the clients' standpoint, generally written by them with their vocabulary and format. Often, they do not accurately reflect what clients truly need.
- Requirements: Refines the 'Ideas' to establish definitions that act as a common ground between the stakeholders of the project. It should contain a glossary of terms, the project goals, estimated costs and time frames, a list of people involved, descriptions of involved processes, expected results, details on how the model will fit in the organisation, and so on. This information requires an agreement with all stakeholders.
- Formal Specification: Elaborated over the 'Requirements' to explicitly detail them in a structured manner that provides specific information to model the target aspect of the addressed situation. First, this requires splitting the working structure into areas or sectors, consisting of their particular goals, inputs and outputs -tangible and intangible-, procedures, and requisites; it should include relevant information for each of them. Second, a compilation of required and generated data, involving the one used as input in the model and provided by the organisation, and a list of expected results, their presentation and format. Third, a prioritisation of the requirements and the links and dependencies between them, as well as the qualities expected for the model.
- Mathematical Design: Equivalent to Software Architecture (SA), it is the main step before coding the model. It consists of diagrams and documentation that allow organising the model and establishing the design decisions adopted, working upon the obtained 'Formal Specification'. It is aimed to modellers. This design should work with the SA concept of points-of-view: the artefacts should be targeted to different stakeholders, they should see the same model from dissimilar perspectives with complementary specifications.
- Mathematical Model: The programming code for the model, written in a specific language and following the design established in the previous refinement. It should be verified against the 'Mathematical Design' to ensure that what is built is the correct product.
- Answers: Obtained from the 'Mathematical Model', and additional reports that clarify and organise the information to be presented to the client. It should answer what is established in the 'Requirements'.

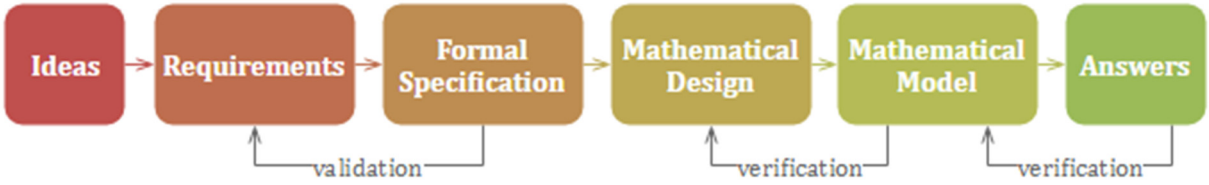

Figure 1: Refinement of information during an OR project.

It is worth noting that this refinement is not sequential, but a *progressive elaboration* as defined in Project Management: continuously improving and detailing it, as more specific and accurate information becomes available as the project progresses (Project Management Institute, 2017).

Thus, while it is possible to go back several states to add more detail, it is not allowed to 'skip' more refined states when moving forward. For example, it is possible to go back from 'Mathematical Model' to 'Formal Specification', but after the changes are done, the progression must improve the 'Mathematical Design' before addressing 'Mathematical Model' once again.

2.1.1 Information Representation

As in SE, each state of information refinement is materialised as *artefacts*: tangible by-products produced during the lifecycle that describe a given aspect of the system, which can be represented using different notations (IEEE Computer Society, 2014). This aspect is significant since documents the process and its results, both final and intermediate while acting as a measurement of the project state.

This article proposes the adaptation of existing SE documentation techniques because they are widely accepted and known among practitioners, and their benefits thoroughly analysed (Chaudron, Heijstek and Nugroho, 2012; Nugroho and Chaudron, 2008). Consequently, reusing and adapting them to OR projects allow establishing a common ground for communication between stakeholders, ensuring best practices by working with approaches already proved and accepted.

The selected artefacts and a brief description of its use can be seen in Table 1, along with the definition of each state of information refinement. This is not an exhaustive list, as other artefacts can be developed for a given project; thus, there is no obligation to generate all of them either.

2.1.2 Additional Considerations

Though the current academic literature provides some contributions to the documentation on OR projects, they are mostly suited to the initial states such as 'Requirements', as they focus on discovering the stakeholders' requisites (Franco and Montibeller, 2010; Marttunen, Lienert and Belton, 2017; Checkland and Poulter, 2010). Therefore, they can also be used in this state.

It is worth noting that there are no artefacts on 'Ideas', as the client mostly provides them; i.e. it can be informal notes from a meeting, formal documents, a Request for Proposal, and many more.

Table 1: Proposed artefacts for each information refinement state.

Refinement	Artifacts
Requirements	User Stories: informal, natural language description of one or more features of a system, written from the perspective of a user (da Silva et al., 2011). System Context Diagram: external factors, events, elements their requests/responses, and their interaction with the target model, in a given environment (Kossiakoff et al., 2016). BPMN Diagram: graphical notation for specifying processes, based on flowcharts, and readable to technical and business users (Object Management Group, 2011).
Formal Specification	Data Template: a tabular organisation of data, describing for each data set (input or output) its meaning, units of measures, type (integer, real number), ranges and precedence, and other relevant details. Areas Template: all areas should be described in the same terms, such as individual goals, required inputs and generated outputs -stating if, e.g., a given area does not require any input-. For ease of comparison, it is recommended a tabular format. Features Lists: they are valuable functions that have business value in the model, defined for all domain areas and grouped in features sets (Anwer et al., 2017). They allow discerning precedencies and dependencies between features, to establish a development priority.
Mathematical Design	UML Diagrams: *Package Diagrams*, show dependencies between directories that group and organise the model elements. *State Diagrams* are directed graphs showing existing transitions and conditions so that an element can change its state. *Activity Diagrams* are behaviour diagrams which shoes flow of control or objects with emphasis on its sequence and conditions (Object Management Group, 2015). Included Files: a colour-coded table, detailing for each package which files are deleted, included or modified at each iteration of the process.
Mathematical Model	Mathematical model and code files: the project directory including the files with the model coded in the selected mathematical programming language.
Answers	Raw Results: obtained from the model in a given file type, with a particular structure. Results Report: including charts derived from the raw results, related analysis, and others.

Thus, it is essential to define which stakeholders participate in the project. This includes the developers -both for software and model, if applicable-, and those coming from the client side. For the latter, it is vital to determine their interest and influence for the project (Ackermann & Eden, 2011), if they are managers or technical staff. This is important to know to negotiate about the project scope, dates and financials, or with whom to validate the documents or asking for feedback and details of the process.

Regarding the process of transformations as a whole, very often modellers and developers work only with the 'Ideas' and attempt to directly transform them into 'Mathematical Models', skipping the intermediate states. As in SE, most modellers prefer a rapid solution in the present instead of a better approach that can take longer but improve the final results (Allman, 2012). However, this leads to each stakeholder working on their vision or perspective of the project, without forming a consensus with the client. As a result, the 'Mathematical Model' requires a numerous amount of rework to answer the client needs.

However, it is important to highlight the fact that the detail of each refinement should enhance the understanding of the project and contribute to its validation and verification. Thus, it is imperative not to incur in unnecessary details that can hinder or unnecessarily delay the production of mathematical code. On the contrary, that refinement should assist in the development, providing clarity and a solid base to work upon.

Finally, there is a parallelism between the 'Mathematical Design' and the concept of Software Architecture (SA). The latter is defined as *"[...] fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [...]"* (ISO/IEC/IEEE, 2011). This standardised definition of SE also defines the goal proposed for the 'Mathematical Design': provide a solid base to code, materialising design decisions, choices, techniques, and requirements to be addressed, before moving into the actual development.

The similarity between 'Mathematical Design' and SA and the inclusion of different stakeholders in the project reinforces the need of having documentation that can be understood by all of them. Thus, the 'Mathematical Design' should also be presented with *viewpoints*, as defined in SA: different artefacts that express the design -or a part of it- from the perspective of a given group of stakeholders (ISO/IEC/IEEE, 2011). Viewpoints are complementary, as they show diverse aspects of the same model from different standpoints, contributing to achieve consensus and obtain a solid foundation.

2.2 Phases

The information refinement is organized in *phases* that define different periods of the lifecycle; each one has a defined goal concerning the flow of information to be used, and the refinement it produces. The output of each phase generates the artefacts used as input for the next one: the information evolves with the project.

As a result, phases are related to short-term goals of the project *per se*, and not to the situation it aims to solve. Figure 2 presents the proposed phases, and they are individually described in the following subsections. This proposal uses names established in SE standards, to simplify the adaptation of existing methodologies while using a nomenclature that is known to both practitioners and academics.

It is worth highlighting that, to define specific practices and technical recommendations, the first step implies identifying and establishing the process of an OR project as a whole, and define its intermediate goals and the type of information used. Thus, this is the main objective of this position paper.

2.2.1 Analysis

'Analysis' is defined as *"[...] the process of analysing requirements to: (1) detect and resolve conflicts between requirements, (2) discover the bounds of the system and how it must interact with its organizational and operational environment, and*

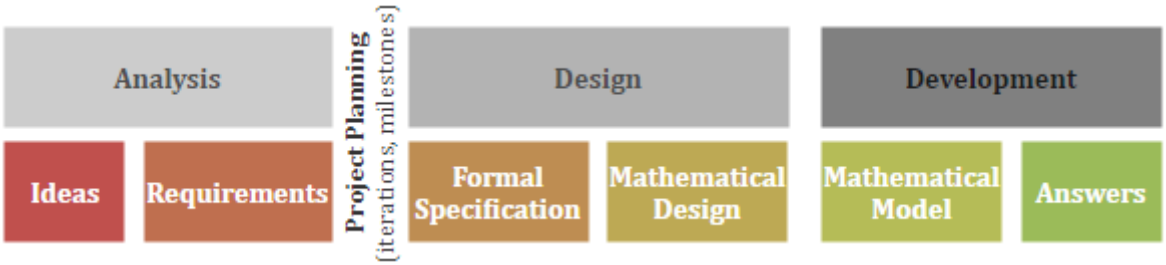

Figure 2: Phases and Information Refinement characterizing OR projects lifecycles.

(3) elaborate system requirements […]" (BKCASE Editorial Board, 2017).

Therefore, it focuses on defining who is part of the project (both clients and developers/ modellers), what the project is about, and what the clients truly need. For this, it is essential to understand the value of the project as a whole, integrate the participants' knowledge and favour the synergy of collective work. Since the current *Soft OR* methodologies are mostly focused in understanding requirements and integrating stakeholders in the process, they can be applied during this phase (Checkland and Poulter, 2010; Eden and Ackermann, 2004; Franco and Montibeller, 2010).

The knowledge obtained in this stage should give a basic understanding of project milestones, allowing the team to have an estimation of the time required to perform the 'Design' phase.

2.2.2 Design

The IEEE defines it as *"[...] the period in the software lifecycle during which definitions for architecture, software components, interfaces and data are created, documented and verified to satisfy requirements [...]"* (ISO/IEC/IEEE, 2010).Thus, 'Design' aims to structure the requisites as a formal modelling that does not require mathematical code. This is used not only as documentation, composed of the 'Formal Specification' and the 'Mathematical Design', but also as a base upon which to prioritise requisites, define who will be in charge of each step, and how the requisites will translate to the model.

While the 'Analysis' is characterized by a relationship between clients and modellers/developers, the 'Design' requires that the second group translate the obtained information into refinements that will provide a structure to model the targeted situation, without applying the mathematical code. As a result, any misunderstandings, lack of detail and missing agreements dragged on from the 'Analysis' phase will negatively affect the 'Design', and cause defective refinements of information.

SE has long agreed that due to the increasing complexity of building systems (Shaw and Clements, 2006) the Software Architecture is essential to obtain a solid and successful design and subsequent development (Frakes and Kang, 2005). Therefore, the parallelism between the 'Mathematical Design' and SA makes this phase as highly relevant to any OR project; hence, it should be granted more importance. However, and within the authors' knowledge, the proposals to formalise this phase are scarce. Although there are applicable artefacts of great practicality, their poor consideration and use in OR generates a gap in research and interventions.

Finally, the information refinements artefacts of this phase assist in the integration of the obtained models to the organisation processes and decisions, as well as to the existing DSS. By applying the SA concept of viewpoints, its use becomes natural for other groups in the organisation -e.g. software development, project management, decision makers- favouring the integration of the model.

Also, strong and well-formed documentation provides the base for reusing code in future projects, reducing costs, and ensuring the use of improved solutions; code and design reuse are essential for SE (Frakes and Kang, 2005). It contributes to the agility of projects: instead of creating a new solution, adopting a new one may be more efficiently (Dingsøyr et al., 2012). However, this is only possible if said solutions are correctly documented.

2.2.3 Development

This phase is defined as *"[...] the specification, construction, testing and delivery of a new application or of a discrete addition to an existing application. [...]* (ISO/IEC/IEEE, 2010). As such, the goal of the 'Development' is to code the model in a given mathematical language, following the specification created during the 'Design' phase: it is the most traditional and core activity of any OR project. It also includes the generation of answers, its testing regarding the inputs, and its validation to what is specified in the 'Requirements' and goals of the project.

It is important to highlight that most OR projects focus excessively on the 'Development', often ignoring or reducing the other phases even though the 'Mathematical Code' is a result of the previous refinements. Though some interventions using *Soft OR* perform the 'Analysis', the majority ignores 'Design'. This causes several issues during the coding, leading towards models that are not adaptable, well-documented or based upon a thought-out structure that simplifies testing and allows reuse.

3 EXAMPLE: ADAPTING AN AGILE LIFECYCLE

This section uses an existing agile methodology to exemplify how the presented concepts can be used to adapt and adopt a SE lifecycle to OR projects.

In particular, this example is done with Feature Driven Development (FDD). This is a model-driven

short-iteration process that consists of five activities. Those are: a high-level walk-through of the scope of the system and its context (*Overall Model*), a decomposition of the domain into subject areas containing business activities (*Feature List*), the planning of the project based on said features (*Plan by Feature*), the design package of the selected features per iteration (*Design by Feature*) and the development and testing of the programming code based on the previous design (*Build by Feature*) (Anwer et al., 2017).

This AM is selected due to being one of the most currently used and accepted, which is adaptable to a wide range of projects and teams sizes (Dybå and Dingsøyr, 2008). Also, FDD does not require extensive knowledge about the method itself, unlike other methodologies such as Scrum. Therefore, this versatility allows adapting FDD to many types of OR problems, from reduced and concrete cases to wide, all-encompassing optimisations.

As with most AM, FDD iterates over the phases of 'Design' and 'Development', producing incremental releases of the system under development, which are valuable for the client, contributing to an early return of the investment (Chow and Cao, 2008).

FDD can be adapted to OR projects by using the phases, information refinement and artefacts previously proposed. The goal of this is exposing the possibility of extending SE methodologies and concepts into OR projects; however, it is only a recommendation, and this adaptation can also be achieved in different ways, and with different AMs.

Figure 3 showcases the original process and its activities (top) compared to the proposed adaptation (bottom).Since the three phases ('Analysis', 'Design' and 'Development') are organised with the same goals of SE lifecycles, the parallelism between original and adapted FDD is direct.

However, it is worth noting that the 'Planning Project' phase can be performed upon different artefacts, depending on the selected AM: a) small, low-risk projects and teams will be prone to use 'Requirements' -e.g. Extreme Programming or Crystal Clear-, while b) mid/large projects with complications such as mid/high risk or geographic distribution, would prefer to use a more elaborated documentation like 'Formal Specification' -e.g. FDD or Scrum. Though practical testing centred in OR is required for validation, existing SE projects validate that cases a) produce less documentation, while case b) focuses more on the design (Dingsøyr et al., 2012), changing what the AM needs to establish the project's iterations, content and milestones.

Regarding specific considerations for the adapted FDD, the 'Overall Model' is originally started by the artefacts of 'Requirements'. Then, after 'Build Feature List', it can be complemented with, for example, a Package Diagram from the 'Mathematical Design'. As a consequence, the latter is then refined by iteration, and its artefacts are fed to the 'Overall Model', incrementing and completing it. For instance, the artefact *files table* can use a colour code to denote added, modified or deleted files -e.g. green, blue and red-, showcasing at each iteration how each package progresses. This is similar behaviour to what happens between package and class diagrams in SE (Chaudron, Heijstek and Nugroho, 2012).

After that, on 'Build Feature List', the features of the 'Formal Specification' can be grouped by areas, and additional artefacts can be produced highlighting the relationship between data and features, to establish similarities that can assist in the prioritisation and assignment of features.

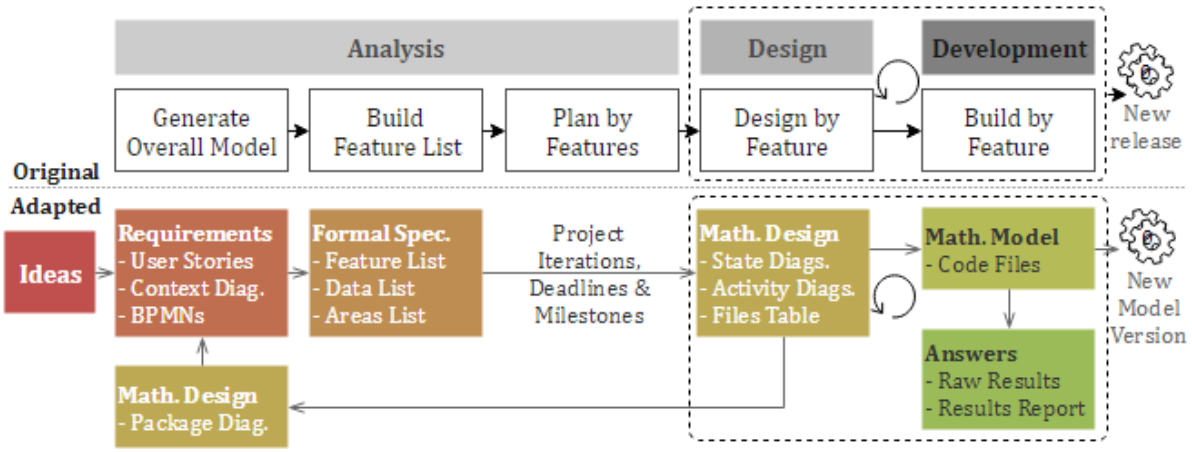

Figure 3: Original FDD (top) vs. adapted FDD (bottom).

Since FDD iterates over 'Design' and 'Development', the artefacts of this phase are incrementally built upon the previous release, adding functionalities from the 'Feature List' and following the order established in 'Plan by Features'. As a consequence, the use of project management tools such as baselines and version control is critical to managing the artefacts (Project Management Institute, 2017).

Thus, any new requisite ('Idea') happening during the project, should be translated into the 'Formal Specification'. From there, the project needs to reorganise its iterations to include the new features, if it corresponds, and update 'Mathematical Design', 'Mathematical Model' and 'Answers'.

Finally, it is not the goal of this position paper to adapt each FDD-exclusive artefact to OR-projects but to establish that, from a theoretical standpoint, the proposed phases, information refinements and artefacts exploit similarities with SE while tailoring to OR-specific needs, allowing porting agile lifecycles. As there are many AMs, this example case demonstrates that this is possible, enabling several avenues for future works.

4 DISCUSSION

The goal of the proposed phases, information refinements and artefacts is to provide the conceptual base to understand the process from a project management approach. Therefore, their real potential is allowing generating a framework to adapt existing AMs into OR projects, to fully apply agile concepts in interventions of this discipline, enhancing and exploiting its relationship with SE. It opens the possibility of using a wide range of methodologies already available, accepted and tested, without the creating new and exclusive approaches. The FDD example is an initial test, backing these concepts. Still, as this is a position, this remains the main avenue for future works and new developments.

However, the proposed extension of the SE contributions in OR implies a change of paradigm and working practices; this implies exploiting the advantages of generating and using methodological practices that address a project as a whole, instead of only addressing the mathematical modelling and code. As such, any change is faced with the resistance to alter current practices and move towards something new, affecting the perception of the proposed phases and information refinement.

In particular, this proposal targets the concept of quality of the intervention as a whole, as part of a system or organisation, instead of focusing on the isolated development of models. Being OR an area focused on the extension/transfer of knowledge is essential that specialists can efficiently identify and coordinate the different stages of the process. As a consequence, this implies that the modellers are required to have project management skills.

Creating a 'Formal Specification' or a 'Mathematical Design' previous to coding is not a current widespread practice. It is possible that many modellers would prefer skipping the 'Design' to move faster towards 'Development', even though a better approach that can take longer but improve the final results (Allman, 2012). Thus, the proposed documentation does not invalidate existing methods and artefacts used in elicitation: it reinforces them by locating them in a comprehensive lifecycle. This favours the balance between phases, improving communication, and creating documentation that enables the reuse of code.

Several limitations and future works arise:

- As in SE, projects need to balance the generation of design with the implementation of code is key (Ruparelia, 2010). As stated by AMs (Dybå and Dingsøyr, 2008), this does not imply the lack of documentation; on the contrary, good documentation practices can save time in the future, revaluing a strategy almost neglected in OR as it is code reuse.
- Because this is a position paper, it lacks an effective application of this new perspective: a practical case study. It is important to generate a base of interventions that feel precedents in the area and contribute to evaluating the success of the proposal. As Ormerod (2008) stated: *"[...] However, success in one intervention is insufficient to validate a perspective such as the TCP. Validation requires living with the perspective over many interventions [...]"*.
- Though the adaptation of FDD to OR is theoretically grounded and consistent with existing SE practices and application reports, a more practical study case is required.

5 CONCLUSIONS

In Operations Research (OR) practice, it is common to find a limited vision of interventions. In these cases, practitioners do not consider the project as a whole, causing a conceptual void regarding any systematisation, and losing the advantages of a progressive and ordered process. As Systems Engineering (SE) already recognised and assessed similar issues, along with the similarities existing between

both disciplines, this position paper aims to make an initial contribution towards adapting the concepts of agility and project management, successfully tested and accepted in SE.

For this purpose, it explicitly defines the phases grouping the information evolution through an OR project, and the artefacts that materialise them. Feature Driven Development is used as an example of adopting an agile methodology from SE to OR.

However, the overall goal is to establish the conceptual ground required to contributing new and better practices for the management and execution of more efficient and orderly interventions.

REFERENCES

Allman, E. (2012) 'Managing Technical Debt', *Communications of the ACM*, vol. 5, no. 5, pp. 50-55, Available: ISSN:1557-7317.

Anwer, F., Aftab, S., Waheed, U. and Muhammad, S.S. (2017) 'Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey', *International Journal of Multidisciplinary Sciences and Engineering*, vol. 8, no. 2, pp. 1-10, Available: ISSN: 2045-7057.

BKCASE Editorial Board (2017) *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, 18th edition, Stevens Institute of Technology Systems Engineering Research Center, International Council on Systems Engineering, Institute of Electrical and Electronics Engineers Computer Society.

Chaudron, M. R. V., Heijstek, W. and Nugroho, A. (2012) 'How effective is UML modeling ?', *Software & Systems Modeling*, vol. 11, no. 4, pp. 571–580, Available: ISSN: 1619-1374.

Checkland, P. and Poulter, J. (2010) 'Soft Systems Methodology', in Reynolds, M. and Holwell, S. (ed.) *Systems Approaches to Managing Change: A Practical Guide*, London, UK: Springer London.

Chow, T. and Cao, D.-B. (2008) 'A Survey Study of Critical Success Factors in Agile Software Projects', *Journal of Systems and Software*, vol. 81, no. 6, pp. 961-971, Available: ISSN: 0164-1212.

Coram, M. and Bohner, S. (2005) 'The Impact of Agile Methods on Software Project Management', 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS), Greenbelt, USA, 363-370.

da Silva, T. G., Martin, A., Maurer, F. and Silveira, M. (2011) 'User-Centered Design and Agile Methods: A Systematic Review', Agile Conference (AGILE), Salt Lake City, USA, 77-86.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2012) 'A decade of agile methodologies: Towards explaining agile software development', *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213-1221, Available: ISSN: 0164-1212.

Dybå, T. and Dingsøyr, T. (2008) 'Empirical Studies of Agile Software Development: A Systematic Review', *Information and Software Technology*, vol. 50, no. 9-10, pp. 833-859, Available: ISSN: 0950-5849.

Eden, C. and Ackermann, F. (2004) 'Use of "soft OR" models by clients - what do they want from them?', in Pidd, M. (ed.) *Systems Modeling - Theory and Practice*, 1st edition, Chichester, England: John Wiley & Sons, Ltd.

Frakes, W. B. and Kang, K. (2005) 'Software Reuse Research: Status and Future', *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 529-536, Available: ISSN: 1939-3520.

Franco, L. A. (2013) 'Rethinking Soft OR Interventions: Models as Boundary Objects', *European Journal of Operational Research*, vol. 231, no. 3, pp. 720-733, Available: ISSN: 0377-2217.

Franco, L. A. and Montibeller, G. (2010) 'Facilitated modelling in operational research', *European Journal of Operational Research*, vol. 205, no. 3, pp. 489-500, Available: ISSN: 0377-2217.

IEEE Computer Society (2014) *Guide to the Software Engineering Body of Knowledge*, 3rd edition.

ISO/IEC/IEEE (2010) *24765:2010 Systems and software engineering - Vocabulary*, 1st edition, Switzerland: International Standardization Organization.

ISO/IEC/IEEE (2011) *42010:2011 - Systems and software engineering — Architecture description*, 1st edition, Switzerland: International Organization for Standardization.

Kossiakoff, A., Sweet, W. N., Seymour, S.J. and Biemer, S.M. (2016) *Systems Engineering Principles and Practice*, 2nd edition, New Jersey, USA: John Wiley & Sons Inc.

Leau, Y. B., Loo, Y. K., Tham, W. T. and Tan, S. F. (2012) 'Software Development Life Cycle Agile vs Traditional Approaches', International Conference on Information and Network Technology (ICINT), Singapore, 162-167.

Marttunen, M., Lienert, J. and Belton, V. (2017) 'Structuring problems for Multi-Criteria Decision Analysis in practice: A literature review of method combinations', *European Journal of Operational Research*, vol. 263, no. 1, pp. 1–17, Available: ISSN: 0377-2217.

Mingers, J. (2001) 'Combining IS Research Methods: Towards a Pluralist Methodology', *Information System Research*, vol. 240, no. 3, pp. 240–259, Available: eISSN: 1526-5536.

Mingers, J. and White, L. (2010) 'A review of the recent contribution of systems thinking to operational research and management science', *European Journal of Operational Research*, vol. 207, no. 3, pp. 1147-1161, Available: ISSN: 0377-2217.

Munassar, N. M. A. and Govardhan, A. (2010) 'A Comparison Between Five Models Of Software Engineering', *International Journal of Computer Science Issues*, vol. 7, no. 5, pp. 94-101, Available: ISSN: 1694-0814.

Nugroho, A. and Chaudron, M. R. V. (2008) 'A Survey into the Rigor of UML Use and its Perceived Impact

on Quality and Productivity', 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Kaiserslautern, Germany, 90-99.

Object Management Group (2011) *Business Process Model and Notation (BPMN)*, 2nd edition, USA: Object Management Group.

Object Management Group (2015) *Unified Modeling Language TM (OMG UML)*, 25th edition, USA: Object Management Group.

Ormerod, R. J. (2008) 'The transformation competence perspective', *Journal of the Operational Research Society*, vol. 59, no. 11, pp. 1435–1448, Available: ISSN: 1476-9360.

Project Management Institute (2017) *A Guide to the Project Management Body of Knowledge*, 6th edition, Pennsylvania, USA: Independent Publishers Group.

Ruparelia, N.B. (2010) 'Software Development Lifecycle Models', *Software Engineering Notes*, vol. 35, no. 3, pp. 8-13, Available: ISSN: 0163-5948.

Shaw, M. and Clements, P. (2006) 'The Golden Age of Software Architecture', *IEEE Software*, vol. 23, no. 2, pp. 31-39, Available: ISSN: 0740-7459.

von Winterfeldt, D. and Fasolo, B. (2009) 'Structuring decision problems: A case study and reflections for practitioners', *European Journal of Operational Research*, vol. 199, no. 3, pp. 857-866, Available: ISSN: 0377-2217.