# Towards Context-Awareness in Model-Based Requirements Engineering

Yaniv Mordecai
Engineering Systems Laboratory,
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA, USA
yanivm@mit.edu
https://orcid.org/0000-0001-5768-1044

Edward F. Crawley
Engineering Systems Laboratory,
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA, USA
crawley@mit.edu
https://orcid.org/0000-0002-3006-1512

***Abstract*—Evolutionary system development and capability deployment are becoming common even in aerospace and defense. As systems evolve, the need to reach significant understanding of the existing architecture is critical for good requirements specification, due to the growing dependency of the requirements on assets in the current architecture. At present, requirement specifications typically do not clearly separate the baseline, or context, from the necessary delta, or prospect. The context informs the prospect and grounds it in the given architecture. The prospect specifies what the requirement owner needs, requires, or expects the system to be or do that is not already in the baseline. We propose a Context-Aware Model-Based Requirements Engineering (CAMBRE) method in which the context of a requirement is modularly composed with its prospect, such that the requirement text is specified in a context-aware manner. We distill those parts of a requirement that are designated for development from those that constitute the background. This approach is acute for complex, evolving, interdependent, or adaptive systems, in which system properties mostly extend or enhance the existing architecture. We implement CAMBRE with Object-Process Methodology (OPM), and demonstrate our approach on the evolutionary extension of a missile defense system with drone interception capabilities to support border protection efforts.**

***Keywords—Model-Based Requirements Engineering; Object Process Methodology; Agile Systems Engineering; Context-Aware Requirements Specification***

## I. INTRODUCTION

Model-Based Requirements Engineering (MBRE) is the use of models to specify requirements [1]. MBRE covers both the documenting of requirements within models, and the modeling of requirements with model-based specification notation. MBRE strives to represent expected or required problem-oriented system behavior, structure, dynamics, interactions, attributes, and non-functional aspects using modeling notation. Specifying requirements differs from specifying system architecture or design as the former focuses on stakeholder needs, intents, and expectations in the context of operational or business problems, whereas the latter focuses on the socio-technical solution that the system delivers.

Traditionally, requirements are recorded as textual statements in text documents or as objects in requirement databases. The SysML Requirements Diagram notation was devised as a way to import textual requirements (with some key attributes) into models [2], and allow for tracing system architecture and design artifacts to those requirements [3]. However, requirement diagrams focus on requirements relationships and contextualization – not on requirement specification. There is no formal guidance for specifying requirements using a specific behavioral or structural diagram.

While stakeholder requirements typically remain textual and sometimes schematic—graphic, yet not formal—system requirements must be formal and consistent, because they drive system and subsystem design and development. However, no matter how much effort and emphasis we allocate to system requirement authoring, they remain text-based, and prone to a variety of errors: misspelling, inconsistent use of reference terms, lack of coherent themes and levels of granularity, solution-bias, contradicting guidance, incomplete specifications, and personal style.

A recently trending approach calls for modeling the requirements [4], i.e., literally building models to reflect what a system is required to be or do, and map the requirement model artifacts to design model artifacts that specify how the system should be built to meet those requirements. This derivation has not yet been standardized in SysML [5].

Apart from the need to make stakeholder and system requirements a bit more formal for communication with stakeholders, there are concrete stakeholder directives that call for exploring the model-based requirements specification approach: a) constant change in stakeholder needs; b) constant change in environmental, situational, and circumstantial contexts; c) gradual transition of operational stakeholders to preferring evolutionary growth, in order to minimize complexity, risk, cost, and schedule and in order to spur adoption and operationalization in a timely manner; and d) highly interdependent and interconnected systems that make requirements depend on each other.

Evolving architectures are becoming commonplace, even in aerospace and defense, due to the above trends, and thanks to the growing versatility and reusability of modern technologies. Highly-interdependent and interconnected systems mandate ongoing, often Sisyphean efforts to mitigate requirement inconsistencies and discrepancies. For example, one engineer had been relying on a graphic processor to perform some geospatial renderings, but found out later in the design process that the graphic processor had previously been scoped out, and he had to find another solution for multiple requirements related to geospatial analysis. In another example, another engineer assumed a central navigation computer would provide an integrated position reading, only to find out later on – too late – that it would not be able to support external consumers, like her device. She had to come up with her own navigation solution, which entailed a significant impact on the cost and schedule of her device.

Building on top of an existing architecture while focusing on the required modifications, or the *prospect* (also known as "deltas") is key to successful delivery. Lack of clarity about required modifications versus their baseline, or *context*, may result in redundant effort, redesign, duplicate functionalities, or focus on secondary aspects. In many organizations, the phrase "reinventing the wheel" is all too common.

978-1-6654-4439-2/21/$31.00 ©2021 IEEE

Requirements modeling *per se* is unable to distinguish the prospective part of the requirement from its context, i.e. what needs to be done versus the background, baseline, or reference. This problem is common in textual requirements specification: we try to keep requirements succinct and concise in order to assure readability, clarity, and consistency. However, this approach only increases confusion and leaves room for subjective interpretation. Even simple requirement statements are difficult to analyze for context and prospect.

The need to clarify the context of a requirement while keeping it compact and solution-neutral as much as possible is quite a conundrum. We suggest a simple, yet robust approach to separate context from prospect, in a way that allows for modeling whatever context is necessary for understanding the requirement, without compromising the clarity, readability, conciseness, and coherence of the requirement proper. We can then generate the text for the requirement along with an introductory part that covers just enough context to understand the prospect. Furthermore, any additional context that comes up as necessary for better understanding of the requirement can be added to the requirement specification model without changing anything in the requirement proper.

The rest of this paper is organized as follows: We review related work on requirements specifications and MBRE in section II. We explain the need for context-prospect separation in section III, describe a framework for context-aware model-based requirements specification (CAMBRE) in section IV, and show how CAMBRE can be implemented with Object-Process Methodology (OPM), a leading model-based systems engineering framework, in section V. We provide an example for enhancing a ballistic missile interception system to cope with hostile drones in section VI, and conclude our study in section VII.

## II. Related Work

SysML's Requirements Diagram ('*requiagram'*) notation, which was introduced as a SysML-unique extension to UML, is the basic notation for model-based requirements [5]. A *requiagram* captures requirements as objects with several attributes, like title, text, unique identifier, kind, status, and risk level. These attributes be imported from documents, spreadsheets, and requirements databases. *Requiagrams* can capture requirement hierarchy and traceability, among other inter-requirement relations. *Requiagrams* can also assign system model artifacts to satisfy the requirements. Requirements can be extended into new diagrams of their own – typically Use Case diagrams or Activity Diagrams (ADs), which are defined as more abstract, high-level diagrams, but essentially any notation can be used, including the more design-oriented Internal Block Diagram (IBD) and Sequence Diagram (SD). A SysML *requiagram* is shown in Fig. 1.

However, *Requiagrams* are not so much about requirement specification. They focus on requirements relationship and contextualization – similar to Use Case diagrams (a particular use case is typically specified in an Activity Diagram or Sequence Diagram) [5]. Deriving a specific behavioral or structural diagram to specify a particular requirement is possible with some SysML modeling tools, but the only standardized diagram mapping to a requirement in SysML 1.6 is for a Test Case diagram [5].

Research on how well-written and well-organized requirements should look like is abundant, e.g., [6–8]. Intent Specification [7] is an approach to construct requirements while rationalizing and explaining why each requirement is needed. Intent Specifications are based on a 5-level hierarchy of the system's purpose, the design principles, the blackbox requirements, the functional requirements, and finally the physical realization. This approach has been originally conceived for software safety specifications but has become a requirements engineering approach [9] and applied in safety-related systems architecting [10]. It has not been applied as part of model-based systems engineering [11]. Requirements Flowdown – the decomposition, derivation, and allocation of requirements to system architecture –is a primary benefit of Model-Based Systems Engineering (MBSE) [12].

Various modeling and diagramming approaches were reviewed as suitable frameworks for requirements specification as part of a holistic requirements engineering process [8]. When we surveyed the depictions of the various notations, we could clearly see that the state of mind was bent towards new or unprecedented systems, and therefore most if not all of the specifications could be viewed as requirements. In the current landscape of constantly evolving systems, it is no longer a viable approach to assume that most of the design is new. In fact, in many cases, the smaller the effort for satisfying stakeholder and customer needs, the better the solution: it costs less, delivers faster, and entails smaller and fewer risks. System architects might defer such holistic requirements specification frameworks if only a small portion of it is needed as the context for a modification or extension to satisfy a small set of change requests.

Model-Based Requirements Engineering (MBRE), a more rigorous application of MBSE to Requirements Engineering, is first described as a comprehensive paradigm in [1]. MBRE as a SysML-based specification of requirements in context associates the requirements with their stakeholders in an explicit manner, with Use Cases. Use Cases place requirements in context by specifying the situations and scenarios in which specific stakeholders benefit from the realization of the requirement by the system. Requirements are managed and elaborated with *requiagrams*, followed by appropriate diagrams (Block Definition Diagram - BDDs, IBDs, AD, and SD) for specific requirement modeling

Although the <<satisfy>> relation is thoroughly utilized to clarify which model elements are responsible for satisfying or implementing each requirement (as also shown in Fig. 1), even with the comprehensive MBRE approach it is difficult to distinguish the context of the requirement from its prospect when observing the detailed diagram that specifies each relation. There are some schematic ways that are used heuristically and informally such as layout arrangements and textual tags, but no formal way is used to clearly separate the context of the requirement (as opposed to the high-level context defined as association with stakeholder points of view) from required structure or behavior.

Object-Process Methodology (OPM) is another MBSE approach and modeling language that has preceded SysML [13] and has been both compared and integrated with SysML for various conceptual modeling problems [14–20]. Modeling and simulation of evolving complex systems was discussed and introduced with Object-Process Methodology (OPM) in [21]. However, the focus was on deriving variant designs rather than complying with evolving requirements – the requirements were not explicitly defined within the model. A framework for specifying system requirements that trace back

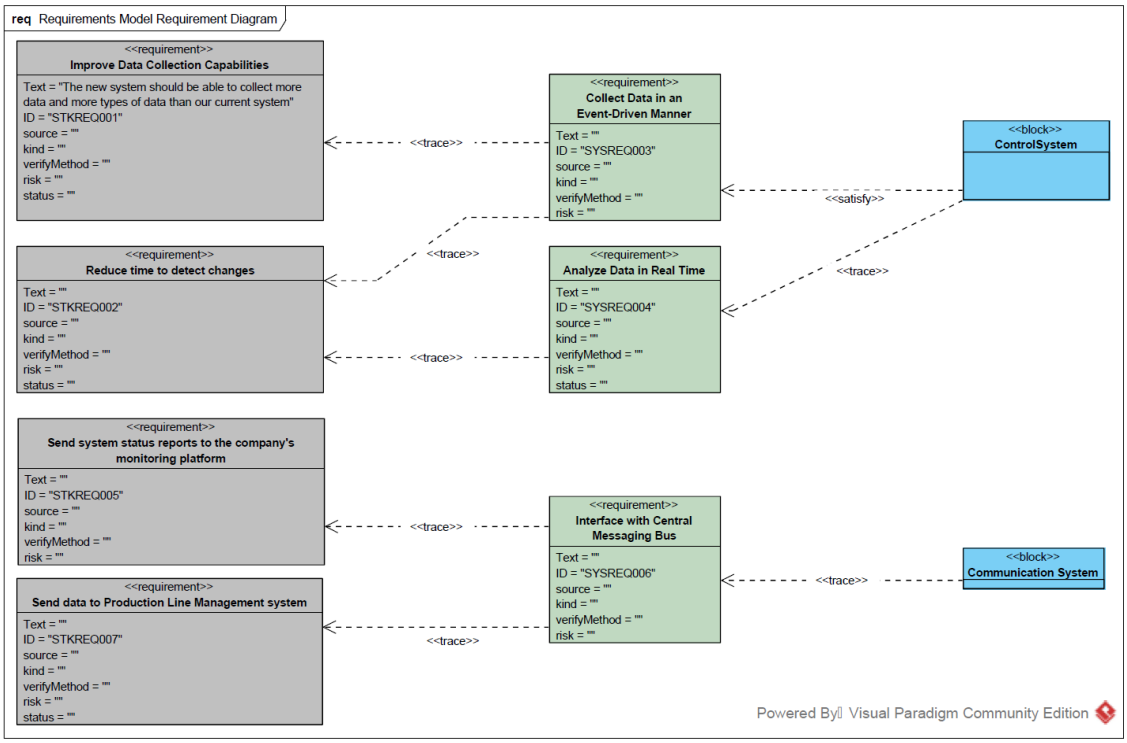



Fig. 1. SysML Requirements Diagram for real-time event detection. Stakeholder requirements focus on incremental improvement of data collection (STKREQ001), detection time (STKREQ002), and extended connectivity (STKREQ005 and STKREQ007). System Requirements trace to Stakeholder Requirements and provide a more formal concrete specification of necessary structure and behavior. Two subsystems – Control System and Communication System – are assigned to satisfy the System Requirements.

to stakeholder requirements and for allocating the system requirements to elements of form and function was presented in [22]. The limitation in that approach is that it captures both the stakeholder requirements and the system requirements as textual statement objects – just like SysML's requiagrams. While stakeholder requirements might be kept in their original textual form, system requirements could benefit from a more formal problem specification, especially since the environmental context should have been better captured.

Another approach, "True Model-Based Requirements", uses SysML BDDs, IBDs, and SDs for specifying requirements [4]. The limitation in that approach is that it is not possible or that is very difficult to tell the context from the prospect. For example, a sequence diagram that represents some requirement does not distinguish between existing and required objects, messages, and sequences. A more recent approach studies the limitations of Use Case Diagrams, and explored UML Swimlane ADs in the context of business process modeling for MBRE [23]. Similarly, the framework is not suited to specify which portions of the specification constitute the reference for the requirement. The authors also argue that MBRE research is still sparse.

Evolving architectures and agile systems engineering are becoming commonplace, even in aerospace and defense [21,24–26], due to the constant change in stakeholder needs and thanks to the growing versatility and reusability of modern technologies. Context-adaptive systems exhibit structural or behavioral adaptability. The broad environmental, situational, and circumstantial contexts of such systems must be constantly analyzed for changes over time and emergence of new patterns [27]. Many assets in such architectures precede and facilitate the realization of incoming requirements.

To conclude, it appears that the MBRE movement is still evolving and trying to shift the focus from recording textual requirements in models to using formal notation to specify the requirements intent. This is still not a standardized practice in SysML or OPM but some frameworks suggest progress in this direction, with some evidence that it is indeed more useful and informative for requirements understanding than recording their textual descriptions in the model. The attempt to make this paradigm shift from model-based requirement documentation to model-based requirement specification gives rise to the problem of separating the prospect and intent of the requirement from its context and background. In this paper we focus on trying to remove this limitation by suggesting appropriate semantics and analysis methods to facilitate this context-prospect distinction and leverage it.

## III. The Need for Context—Prospect Distinction

Consider the following simple requirement:

**"The pod shall be able to scan 40m²/sec".**

From this statement alone, we cannot tell which part of the statement is already part of the architecture or another requirement. We also miss many contextual details. Questions that might arise are:

- What kinds of surface are to be scanned by this system? (e.g. terrain, water surface, ocean floor)
- Is the pod already operational or also defined as one of the expected deliverables?
- Does the scanning capability already exist, and if so, what is the current coverage?
- Is the scanning capability a pod function or a stand-alone/external functionality?

- Is the required coverage of 40 $m^2$/sec the result of some electromechanical or physical constraint or a derivative of an operational need? Is 40 $m^2$/sec below or above the current capability? Is it a lower or upper bound on the physical or technological capability?
- Is the object carrying the pod able to support coverage of 40 $m^2$/sec, e.g., can it move at a speed of 20 m/sec, assuming the pod can cover a 2m/sample width?

Suppose the current operational capability is based on a 10 $m^2$/sec single-scanner pod. The requirement is only about the 40 $m^2$/sec coverage. Several solutions may be possible: a) improving the scanner by a factor of 4; b) improving the scanner by a factor of 2 and mounting one more scanner on the pod; c) mounting 3 more scanners on the pod; d) replacing the existing scanner with a different scanner; e) replacing the entire pod; f) applying a software configuration change to allow the scanner to scan 4 times faster, with some compromise on quality; etc. Unfortunately, if we do not have the answers to all of the above questions, which constitute the necessary context for figuring out the requirement and what is the minimum viable solution to satisfy it, we are prone to miss critical information, make wrong assumptions, come up with an inadequate design, or require significantly more time and effort to fill in all the missing information.

For the above requirement, the pod, scanning function, and initial 10 $m^2$/sec scanning capability are given context aspects that we cannot ignore. Still, it is complicated to explicitly describe them as part of the requirement. The answers to such questions are unlikely to be included in the requirement specification proper because it will render a completely unreadable and potentially more confusing requirement specification than the original concise requirement we start with. Porting the insensitivity to the context-prospect duality and separability into model-based requirements is out of the question. Specifically, as we have just called out this distinction, it can no longer be disregarded.

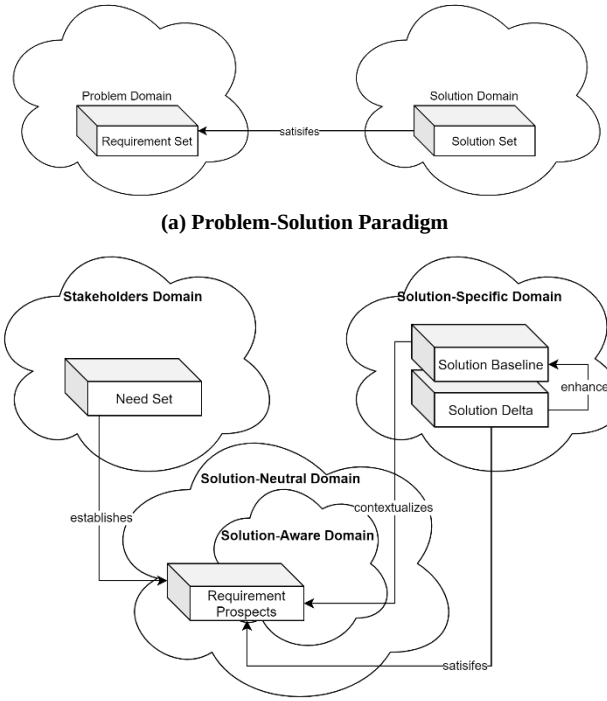
## IV. Context-Aware Model-Based Requirements Specification (CAMBRE)

The core idea that underlies CAMBRE is the distinction in the model between the context and the prospect. It is quite similar to how problem-solving is taught in math and physics. First, we list all the inputs to the problem, under the "Given" section. In fact, the word *data* originally means given in Latin (in plurlal form). In Hebrew, for instance, *data* and *given* in plural form have the same word – *netunim*.

The context in our framework covers everything that enables the realization of the requirement (the *how*), sets the environmental context (the *who*, *where* and *when*), or justify the requirement (the *why*). The prospect of the requirement is the *what*. It can be specified at the blackbox, functional, logical, or physical levels. This approach covers all aspects of intent specifications. CAMBRE is integrated into conceptual modeling, which allows for continued discussion, elaboration, and solution design to realize the requirements, as part of the same model – and the two are integrated, rather than detached.

CAMBRE follows conceptual architecture theory described in [28], a derived concept representation framework [29], and a more rigorous model-based system architecting framework [30]. It consists of elaborating the conceptual architecture in five domains: a) stakeholder needs, b) solution-neutral domain, c) solution-specific domain, d) integrated concept domain, and e) operations domain. A model-driven distinction of requirement context and prospect begins with separating the requirements from the problem domain, and departure from the classic yet restricting problem domain—solution domain dichotomy, as shown in in Fig. 2(a).

According to [29,30] needs are defined in the Stakeholders domain and they are separate from requirements. Requirements are defined in the solution-neutral domain and solutions are defined in the solution-specific domain. We refine this model by including a solution-aware sub-domain in the solution-neutral domain. By including this solution-aware sub-domain we allow requirements to be either strictly solution-neutral or solution-aware. Solution-awareness is still solution-neutrality because it refers only to the information from the solution domain that is sufficient as context for any requirement of interest. This revised concept representation and solution-aware paradigm is illustrated in Fig. 2(b).




**(a) Problem-Solution Paradigm**

**(b) Solution-Aware Paradigm**

Fig. 2. Transition from a problem—solution paradigm (a) to a solution-aware paradigm (b).

Real systems include elements of structure, behavior, operands (inputs, outputs, resources), system states, and relations among any of the above [31,32]. Such concepts must be mapped into conceptual models as part of the system concept representation process. Thereafter, model entities that represent system architecture elements can serve as both the contexts and the solutions for system requirements.

In order to develop a solution-aware requirements model concurrently, we propose Algorithm 1: The CAMBRE Protocol. This is a generic method that can be applied with any modeling language or modeling framework, provided that: a) a requirement can be elaborated into a separate diagram; b) the model can be split into designated domains (solution-neutral and solution-specific domain, for that matter); c) an architectural element can appear in multiple domains without being considered as two separate entities; d) given a model with multiple domains, each element that appears in one domain can be associated with its copy in another domain, if

such a copy exists; and e) the model can provide a textual specification that can be used as requirement specification.

ALGORITHM 1: THE CAMBRE PROTOCOL

1. Given a set of Stakeholder Needs [INPUT],
2. For each Need,
   - 2.1. Define solution-aware requirement placeholders that correspond to the need.
   - 2.2. For each requirement
     - 2.2.1. Create a container diagram from each requirement in the solution-neutral domain.
     - 2.2.2. For each baseline architecture element that is necessary to contextualize the requirement:
       - 2.2.2.1. If the element does not exist in the model, add it to the solution-specific domain.
       - 2.2.2.2. Import the baseline element into the requirement model.
       - 2.2.2.3. Specify solution-neutral architectural elements that constitute the prospect of the requirement.
3. Return set of solution-aware requirements, such that for each requirement two specifications are provided:
   - 3.1. Context: All elements that appear in the solution-neutral domain and in the solution-specific baseline domain.
   - 3.2. Prospect: All elements that appear only in the solution-neutral domain.

## V. DEMONSTRATION WITH OBJECT-PROCESS METHODOLOGY

Object Process Methodology (OPM) is a common modeling language that is also an ISO standard: ISO-19450 [33,34]. OPM uses objects to represents elements of structure, and processes to represent elements of behavior. States are specified within objects and links reflecting procedural and structural relations connect objects, processes, and states to each other. OPM's modeling notation and modeling software, OPCloud [35], accommodate the necessary conditions defined above for supporting CAMBRE. The mapping of each condition to OPM's existing features and modeling capabilities is summarized in TABLE I. OPM's textual modality, OPL – a formal textual specification that accompanies the graphical model – is of special importance.

The input to the process is a set of stakeholder needs, which are captured within the Stakeholders domain, and mapped to requirements in the Solution-Neutral Domain. Next, we illustrate an example with the first requirement, defined as an "Input Requirement" and identified as Req001. The requirement is unfolded in a new diagram, as illustrated in Fig. 3. This diagram is constructed by importing the Baseline object and its relevant portions that are sufficiently important or relevant as context for the requirement. In this case, the requirement pertains to a new sub-process of the main system process, that receives new input – as well as the existing input – and generates new output. The new process does not affect the existing output. Keeping the existing output in the diagram can be considered redundant, but keeping it in the diagram without a direct relation to the requirement, clarifies that it is not associated to the required process. Note that in order to keep the requirement as solution-neutral as possible, no mention of the structural element that implements the new process is specified. Such an allocation of function to form belongs in the solution-specific domain.

The baseline for the model, which contains the context for the above requirement – and more, is illustrated in Fig. 4. Other aspects that are specified in the baseline, and are completely unrelated to Req001, are not included in Fig. 3, because they are not sufficiently important. This way, only critical and necessary aspects of the baseline that constitute the context are included, and the requirement specification diagram is minimized.

TABLE I. OPM COMPLIANCE WITH CONDITIONS FOR SOLUTION-AWARE REQUIREMENT SPECIFICATION

| Condition | Supporting OPM Feature |
|---|---|
| (a) Requirement specification diagram | A requirement is an object that is classified or typed as a Requirement. A requirement specification diagram is an unfolded diagram of the object. A diagram derived from an object that is classified as a requirement is therfore a requirement specificaiton diagram. |
| (b) Designated model domains | A domain is an object that is classified or typed as a Domain. Each domain is unfolded in a diagram. A diagram derived from an object that is classified as a domain is therfore a domain specificaiton diagram. |
| (c) Referring to elements from the baseline domain in the requirements domain | Any object or process can be brought into any diagram after being defined once anywhere in the model. A relation between objects or processes that is necessary as an architectural element can be defined first as an object that is classified as a relation, unfolded in a separate diagram where the explicit relation appears, and |
| (d) Classifying baseline elements in requirement specifications as context | Any statement in the solution-neutral domain, that also appears under the baseline domain, is marked as part of the context. The rest of the statements are marked as the prospect. |
| (e) Textual requirement specification | OPM features a built-in textual specification in a natural language called OPL – Object-Process Language. Every construct in the model is acompanied by a textual statement in OPL. |

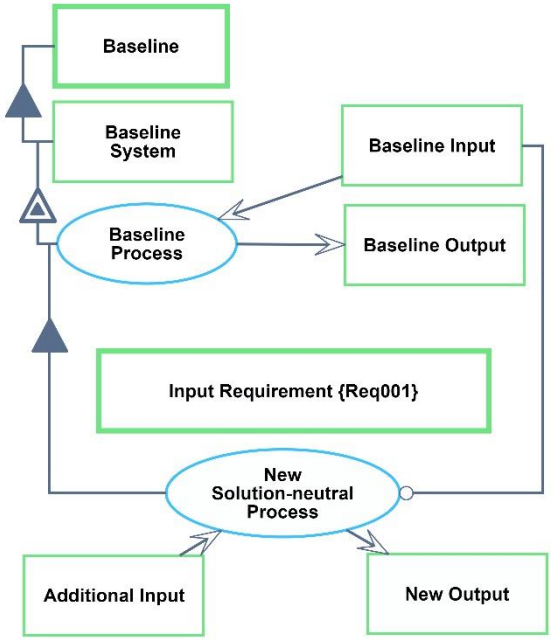



Fig. 3. Elaborating a requirement, by importing Baseline constructs as context for the required solution-neutral process. The requirement deisgnator object is placed in the middle of the diagram to graphically separate the given context (above it) from the required prospect (below it). Relations to the context are clearly visible, such as specifying the new process as a subprocess of the main system process, and using the existing input, in addition to some new input.

We now compare the textual specification of the requirement specification diagram (Fig. 3) with that of the baseline diagram (Fig. 4), and identify the context. Any statement that appears in both diagrams is considered a context statement. Any statement that does not appear in the specification of the baseline is considered as the prospect of

the requirement. The mapping of the complete OPL textual specification of Req001's diagram to OPL statements of the baseline diagram is shown in TABLE II. If no equivalent statement is found, 0 is returned. The remaining baseline specifications that are not part of Req001's context are provided for reference in 0This subset was created by filtering out all the baseline statements that were matched for in TABLE II. For example, all the statements concerning the interaction with the external environment are irrelevant in the context of Req001.

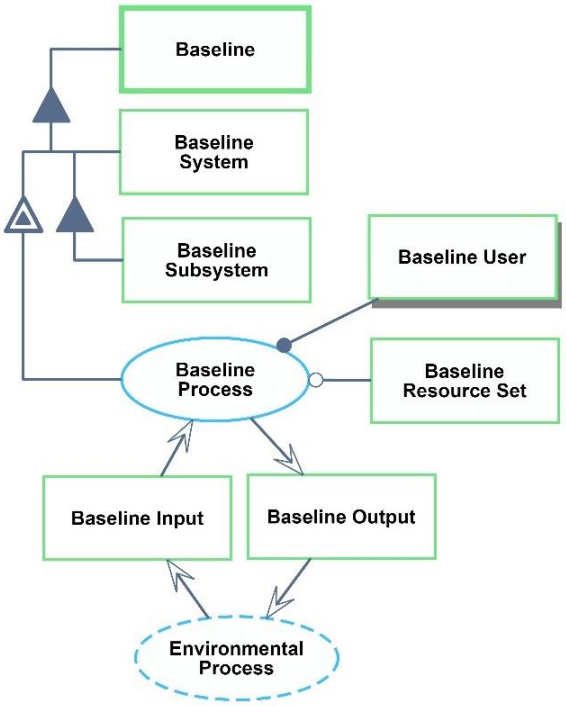



Fig. 4. The Baseline diagram includes additional information that is not included in the context of the requirement specified in Fig. 3.

TABLE II. OPM TEXTUAL SPECIFIATION OF A REQUIREMENT SPECIFICATION DIAGRAM, WITH A POINTER TO THE EQUIVALENT OPL STATEMENT IN THE BASELINE DIAGRAM, IF IT EXISTS.

| Input Requirement {Req001} | Baseline |
|---|---|
| **Input Requirement, Req001,** is informatical and systemic. | 0 |
| **Additional Input** is informatical and systemic. | 0 |
| **Baseline** is informatical and systemic. | 1 |
| **Baseline System** is informatical and systemic. | 2 |
| **Baseline Input** is informatical and systemic. | 3 |
| **Baseline Output** is informatical and systemic. | 4 |
| **New Output** is informatical and systemic. | 0 |
| **Baseline** consists of **Baseline System**. | 9 |
| **Baseline System** exhibits **Baseline Process**. | 8 |
| **Baseline Process** consists of **New Solution-neutral Process**. | 0 |
| **New Solution-neutral Process** is informatical and systemic. | 0 |
| **New Solution-neutral Process** requires **Baseline Input**. | 0 |
| **New Solution-neutral Process** consumes **Additional Input**. | 0 |
| **New Solution-neutral Process** yields **New Output**. | 0 |
| **Baseline Process** of **Baseline System** is informatical and systemic. | 11 |
| **Baseline Process** of **Baseline System** consumes **Baseline Input**. | 14 |
| **Baseline Process** of **Baseline System** yields **Baseline Output**. | 15 |

Based on the mapping in TABLE II. we can split the OPL textual specification into two segments: Context and Prospect. Each sentence is auto-numbered serially within its segment: [ReqID].[0XX] for context statements, and [ReqID].[1XX] for prospect statements, which are the sub-requirements within the context of the main requirement. The context is introduced before the prospect – as an introduction rather than an addendum. The result is illustrated in TABLE IV.

TABLE III. OPL STATEMENTS IN THE SPECIFICATION OF THE BASELINE, WHICH ARE NOT PART OF THE REQUIREMENT SPECIFICATION

| Baseline |
|---|
| **Baseline Resource Set** is informatical and systemic. |
| **Baseline User** is physical and systemic. |
| **Baseline Subsystem** is informatical and systemic. |
| **Baseline System** consists of **Baseline Subsystem**. |
| **Baseline User** handles **Baseline Process** of **Baseline System**. |
| **Baseline Process** of **Baseline System** requires **Baseline Resource Set**. |
| **Environmental Process** is informatical and environmental. |
| **Environmental Process** consumes **Baseline Output**. |
| **Environmental Process** yields **Baseline Input**. |

TABLE IV. TEXTUAL REQUIREMENT SPECIFIATION OF A REQUIREMENT, WHICH SPLITS THE SCOPE OF THE REQUIREMENT INTO A CONTEXT SET AND A PROSPECT SET. EACH STATEMENT IS UNIQUELY IDENTIFIED WITHIN ITS SCOPE.

| | Statement ID | Statement |
|---|---|---|
| | **Req001** | **Input Requirement {Req001}** |
| Context | Req001.001 | Baseline is informatical and systemic. |
| | Req001.002 | Baseline System is informatical and systemic. |
| | Req001.003 | Baseline Input is informatical and systemic. |
| | Req001.004 | Baseline Output is informatical and systemic. |
| | Req001.005 | Baseline consists of Baseline System. |
| | Req001.006 | Baseline System exhibits Baseline Process. |
| | Req001.007 | Baseline Process of Baseline System is informatical and systemic. |
| | Req001.008 | Baseline Process of Baseline System consumes Baseline Input. |
| | Req001.009 | Baseline Process of Baseline System yields Baseline Output. |
| Prospect | Req001.101 | Input Requirement, Req001, is informatical and systemic. |
| | Req001.102 | Additional Input is informatical and systemic. |
| | Req001.103 | New Output is informatical and systemic. |
| | Req001.104 | Baseline Process consists of New Solution-neutral Process. |
| | Req001.105 | New Solution-neutral Process is informatical and systemic. |
| | Req001.106 | New Solution-neutral Process requires Baseline Input. |
| | Req001.107 | New Solution-neutral Process consumes Additional Input. |
| | Req001.108 | New Solution-neutral Process yields New Output. |

## VI. EXAMPLE: ADDING DRONE INTERCEPTION CAPABILITIES TO A MISSILE DEFENSE SYSTEM

In this section we review a more concrete example for CAMBRE, concerning the development and deployment of drone interception capabilities within existing missile defense systems. Drones have become a major threat in border conflicts, due to their low price, high availability, and low footprint. At the same time, drones are employed by the defending party for surveillance, reconnaissance, and public media coverage, as well as for intercepting enemy drones and other threats [36,37].

The operational need for a quick response to the drone intrusion threat calls for creative enhancements to existing border protection platforms, rather than a greenfield solution. For instance, command and control centers can provide battle management capabilities, and radars can be taught to detect and track enemy drones. In some cases, kinetic and cybernetic measures may be repurposed for drone interception.

Consider an existing missile and rocket interception system. In order to incorporate drone interception capability into such a platform, we must know which elements of the platform can be utilized while specifying the operational system requirements. For example, can we include "identify friend-or-foe" (IFF) indicators for drones in the theater, if our own drones are not equipped with an IFF transponder?

A fictitious baseline of a missile interception system is specified in Fig. 5. We focus on four key requirements that are necessary for satisfying the need to intercept only the threatening drones: detection, threat assessment, decision making, and interception. For this example, we focus on the detection phase. We assume that the existing radar has the capability to detect drones, but we need the radar to classify the target as a drone or missile, so that the system and the operators will know how to engage it.

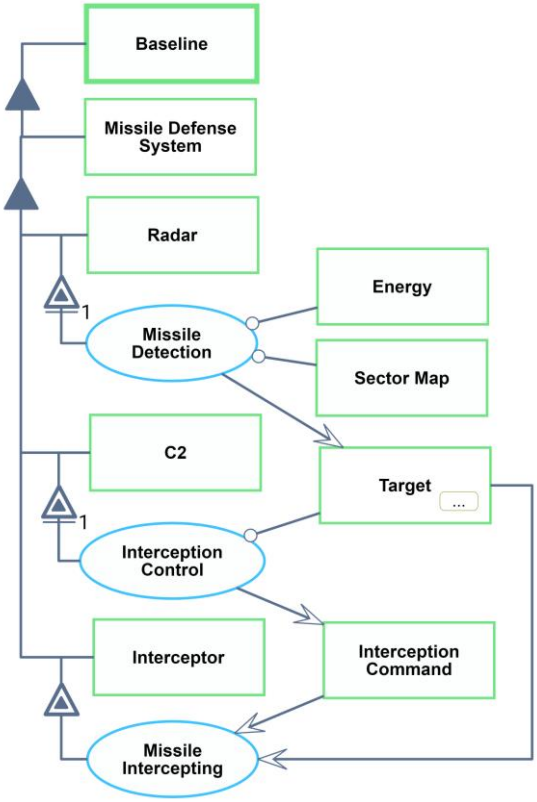



Fig. 5. Baseline architecture for a missile defense system

The Drone Detection Req specification diagram is illustrated in Fig. 6. The specification is provided in TABLE V. Changes to existing functions and operands emerge from the interaction with the baseline. For example, the Missile Detection function is still part of the context (Req001.007) but it is now required to classify targets as missiles (Req001.111).

This additional value that emerges from applying the CAMBRE approach is significant: it ensures that changes to the existing architecture as a result of integrating a new capability are not missed, because they themselves are captured as part of the prospect of the requirement. Note that at this stage, change requirements are still solution-neutral (yet solution-aware): the implementation may be carried out in different ways that may not necessarily impact the existing solution (for instance, missile classification will be the default, and drone classification will override it if necessary). However, from a solution-neutral, problem-oriented perspective, it is still essential for the stakeholders to recognize such implications on the architecture, performance, operations, etc.

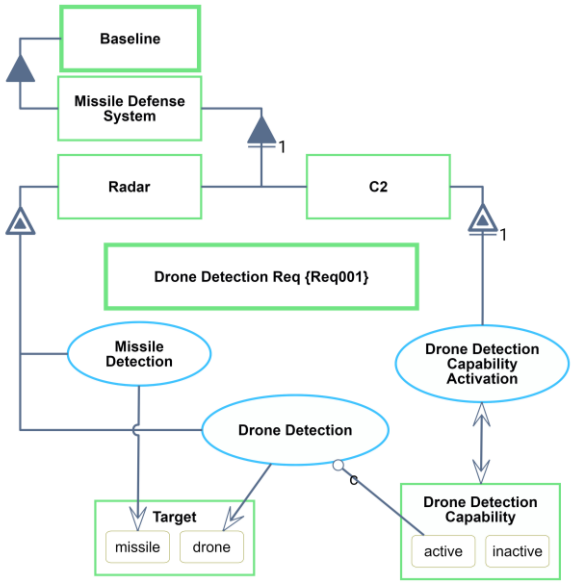



Fig. 6. Baseline architecture for a missile defense system

TABLE V. TEXTUAL REQUIREMENT SPECIFIATION FOR DRONE DETECTION, INCLUDING IMPLICATIONS ON THE EXISTING DESIGN

| | Statement ID | Req |
|---|---|---|
| | **Req001** | **Drone Detection Requirement {Req001}** |
| Context | Req001.001 | Baseline is informatical and systemic. |
| | Req001.002 | Missile Defense System is informatical and systemic. |
| | Req001.003 | Target is informatical and systemic. |
| | Req001.004 | Radar is informatical and systemic. |
| | Req001.005 | C2 is informatical and systemic. |
| | Req001.006 | Baseline consists of Missile Defense System. |
| | Req001.007 | Missile Detection of Radar is informatical and systemic. |
| Prospect | Req001.101 | Drone Detection Req, Req001, is informatical and systemic. |
| | Req001.102 | Target can be missile or drone. |
| | Req001.103 | Drone Detection Capability is informatical and systemic. |
| | Req001.104 | Drone Detection Capability can be active or inactive. |
| | Req001.105 | Missile Defense System consists of C2 and Radar. |
| | Req001.106 | Radar exhibits Drone Detection and Missile Detection. |
| | Req001.107 | C2 exhibits Drone Detection Capability Activation. |
| | Req001.108 | Drone Detection of Radar is informatical and systemic. |
| | Req001.109 | Drone Detection of Radar occurs if Drone Detection Capability is at state active, otherwise Drone Detection of Radar is skipped. |
| | Req001.110 | Drone Detection of Radar yields Target at state drone. |
| | Req001.111 | Missile Detection of Radar yields Target at state missile. |
| | Req001.112 | Drone Detection Capability Activation of C2 is informatical and systemic. |
| | Req001.113 | Drone Detection Capability Activation of C2 affects Drone Detection Capability. |

## VII. CONCLUSION

We asserted that the current methods for specifying requirements fail to capture the distinction between the existing context of requirements and the essence of necessary changes or additions to the architecture. Such a distinction is critical for evolutionary design of complex systems – a

practice that becomes more common even in aerospace and defense – domains that have traditionally evolved linearly or in big increments (e.g., according to the spiral model [38]).

We have defined the context and prospect of a requirement as two complementary concepts – the former anchors the latter in the given reference architecture. We have introduced a language-agnostic protocol for context-aware model-based requirements specification, CAMBRE, and have shown how this protocol can be implemented in a model-based architecting process using OPM, leveraging the key feature of automated textual specification in a natural language that accompanies the diagrams. We have demonstrated CAMBRE on a concrete example of extending missile defense systems with drone interception capabilities – a real operational need.

Future research involves several aspects, including incorporation of context analysis methods, linguistic touch-up of prospect statements, impact analysis due to heavy contextualization, contextualization between requirements, and a SysML implementation of CAMBRE.

## REFERENCES


[1] Holt J, Perry SA, Brownsword M. Model-Based Requirements Engineering. The Institution of Engineering and Technology (IET); 2012.

[2] Friedenthal S, Moore A, Steiner R. OMG Systems Modeling Language (OMG SysML™) Tutorial. 2006.

[3] OMG. OMG Systems Modeling Language (OMG SysML) Version 1.3. 2012.

[4] Salado A, Wach P. Constructing True Model-Based Requirements in SysML. Systems 2019;7:19. https://doi.org/10.3390/systems7020019.

[5] Object Management Group. OMG Systems Modeling Language Version 1.6. Needham, MA, USA: 2019.

[6] Komendantova N, Mrzyglocki R, Mignan A, Khazai B, Wenzel F, Patt A, et al. Multi-hazard and multi-risk decision-support tools as a part of participatory risk governance: Feedback from civil protection stakeholders. Int J Disaster Risk Reduct 2014;8:50–67. https://doi.org/10.1016/j.ijdrr.2013.12.006.

[7] Leveson NG. Intent specifications: An approach to building human-centered specifications. IEEE Trans Softw Eng 2000;26:15–35. https://doi.org/10.1109/32.825764.

[8] Hull E, Jackson K, Dick J. Requirements Engineering. 3rd Editio. Springer-Verlag London; 2011. https://doi.org/10.1007/978-1-84996-405-0.

[9] Cheng BHC, Atlee JM. Research Directions in Requirements Engineering. Proc ACM/IEEE Int Conf Softw Eng (ICSE'07 Futur Softw Eng 2007. https://doi.org/https://doi.org/10.1109/FOSE.2007.17.

[10] Leveson NG. Engineering a Safer World. Cambridge, MA, USA: MIT Press; 2011.

[11] Google Scholar. "Model Based Intent Specification" 2021. https://scholar.google.com/scholar?q=%22model-based+intent+specification%22 (accessed January 21, 2021).

[12] McDermott TA, Hutchinson N, Clifford M, Van Aken E, Slado A, Henderson K. Benchmarking the Benefits and Current Maturity of Model-Based Systems Engineering across the Enterprise. Hoboken, NJ, USA: Stevens Institute of Technology; 2020.

[13] Dori D. Object-Process Methodology: A Holistic Systems Approach. Berlin, Heidelberg: Springer-Verlag; 2002.

[14] Estefan J. Survey of model-based systems engineering (MBSE) methodologies. Rev. B. 2008.

[15] Grobshtein Y, Dori D. Generating SysML Views from an OPM Model: Design and Evaluation. Syst Eng 2011;14:327–40. https://doi.org/10.1002/sys.20181.

[16] Ramos AL, Ferreira JV, Barceló J. Model-based systems engineering: An emerging approach for modern systems. IEEE Trans Syst Man Cybern Part C Appl Rev 2012;42:101–11. https://doi.org/10.1109/TSMCC.2011.2106495.

[17] Mordecai Y, Orhof O, Dori D. Modeling Software Agent Awareness of Physical-Informatical Essence Duality. IEEE Int. Conf. Softw. Sci. Technol. Eng. - SwSTE, Ramat Gan, Israel: IEEE; 2014. https://doi.org/10.1109/SWSTE.2014.12.

[18] Dori D. Model-Based Systems Engineering with OPM and SysML. New York: Springer; 2016. https://doi.org/10.1007/978-1-4939-3295-5.

[19] Shani U, Jacobs S, Wengrowicz N, Dori D. Engaging ontologies to break MBSE tools boundaries through semantic mediation. Conf. Syst. Eng. Res., Huntsville, AL, USA: INCOSE; n.d.

[20] Hause M, Day RL. Frenemies: OPM and SysML Together in an MBSE Model. INCOSE Internaional Symp 2019;29. https://doi.org/https://doi.org/10.1002/j.2334-5837.2019.00629.x.

[21] Mordecai Y, Dori D. Agile Modeling of an Evolving Ballistic Missile Defense System with Object-Process Methodology. 9th IEEE Syst. Conf., Vancouver BC, Canada: IEEE; 2015, p. 839–46. https://doi.org/10.1109/SYSCON.2015.7116855.

[22] Mordecai Y, Dori D. Model-based requirements engineering: Architecting for system requirements with stakeholders in mind. IEEE Int. Symp. Syst. Eng. ISSE, 2017. https://doi.org/10.1109/SysEng.2017.8088273.

[23] Bakar NWA, Musa S, Mohamad AH. A Mini Comparative Study of Requirements Modelling Diagrams towards Swimlane: Evidence of Enterprise Resource Planning System. J Phys Conf Ser 2020;1529. https://doi.org/10.1088/1742-6596/1529/5/052054.

[24] Madni AM, Sievers M. Model-based systems engineering: Motivation, current status, and research opportunities. Syst Eng 2018;21:172–90. https://doi.org/10.1002/sys.21438.

[25] Ramos AL, Ferreira JV, Barceló J. Lithe: An agile methodology for human-centric model-based systems engineering. IEEE Trans Syst Man, Cybern Part ASystems Humans 2013;43:504–21. https://doi.org/10.1109/TSMCA.2012.2207888.

[26] Krupa GP. Application of agile model-based systems engineering in aircraft conceptual design. Aeronaut J 2019;123:1561–601. https://doi.org/10.1017/aer.2019.53.

[27] Sitou W, Spanfelner B. Towards requirements engineering for context adaptive systems. Proc - Int Comput Softw Appl Conf 2007;2:593–8. https://doi.org/10.1109/COMPSAC.2007.223.

[28] Crawley E, Cameron B, Selva D. Systems Architecture: Strategy and Product Development for Complex Systems. Hoboken, NJ, USA: Pearson Higher Education; 2016.

[29] Menshenin Y, Crawley E. A system concept representation framework and its testing on patents, urban architectural patterns, and software patterns. Syst Eng 2020;23:492–515. https://doi.org/10.1002/sys.21547.

[30] Menshenin Y, Mordecai Y, Crawley EF, Cameron BG. Model-Based System Architecting and Decision-Making. In: Madni AM, Augustine N, editors. Handb. Model. Syst. Eng., Springer; 2021.

[31] Dori D. Object-Process Methodology for Structure-Behavior Codesign. In: Embley DW, Thalheim B, editors. Handb. Concept. Model., Berlin, Heidelberg: Springer Berlin Heidelberg; 2011, p. 209–58. https://doi.org/10.1007/978-3-642-15865-0.

[32] Mordecai Y, Fairbanks JP, Crawley EF. Category-Theoretic Formulation of the Model-Based Systems Architecting Cognitive-Computational Cycle. Appl Sci 2021;11. https://doi.org/https://doi.org/10.3390/app11041945.

[33] Dori D. Model-Based Systems Engineering with OPM and SysML. New York: Springer; 2016.

[34] ISO/TC 184. ISO/PDPAS 19450 Automation systems and integration — Object-Process Methodology 2015.

[35] Dori D, Kohen H, Jbara A, Wengrowicz N, Lavi R, Levi-Soskin N, et al. OPCloud: An OPM Integrated Conceptual-Executable Modeling Environment for Industry 4.0. In: Kenett RS, Swarz RS, Zonnenshain A, editors. Syst. Eng. Fourth Ind. Revolut. Big Data, Nov. Technol. Mod. Syst. Eng., Wiley, Hoboken, NJ, USA; 2020.

[36] Koslowski R, Schulzke M. Drones along borders: Border security UAVs in the United States and the European Union. Int Stud Perspect 2018;19:305–24. https://doi.org/10.1093/isp/eky002.

[37] Ahronheim A. Drones give militants new precision weapon in Gaza conflict - analysis. Jerusalem Post 2019.

[38] Boehm B. Some future trends and implications for systems and software engineering processes. Syst Eng 2006;9:1–19. https://doi.org/10.1002/sys.20044.