Agile Modeling of an Evolving Ballistic Missile Defense System with Object-Process Methodology

Yaniv Mordecai
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
yanivmor@technion.ac.il

Dov Dori
Engineering Systems Division
Massachusetts Institute of Technology
Cambridge MA, USA;
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
dori@mit.edu

Abstract—Evolution of requirements and systems has led to the proliferation of agile development as a matter of course in the current technological landscape. The rate of change poses a significant dilemma to stakeholders regarding the amount of effort and resources they should invest in methodological practices, including modeling, design, analysis, and specification. The attempt to move quickly from concept to operation often leads to preference of expedience for gaining early time-to-market over quality and robustness. Rather than referring to the modeling vs. agility challenge as a dilemma, we perceive it as an opportunity for synergy. Building on Object-Process Methodology (OPM)—the new ISO-19450 standard, we propose an agile model-based systems engineering (MBSE) framework that facilitates, and fosters development agility and system evolution. This agile MBSE framework extends the traditional scope of system modeling to cover aspects of functional and structural evolution, phase transition, deployment feedback, and configuration interdependence. We demonstrate the applicability of agile MBSE on a hypothetical ballistic missile defense system, based on publicly available information on Iron Dome—an Israeli ballistic missile defense system with proven operational capability.

Keywords—Agile Model-Based Systems Engineering, Evolving System Models, Agile Development, Object-Process Methodology, Configuration Management, Ballistic Missile Defense.

I. INTRODUCTION

Constant system and technology evolution is a major challenge in the contemporary technological landscape. System evolution is motivated and amplified by increasing software flexibility, software-defined hardware adaptability, and a multitude of available technologies, vendors, and applications. This trend is prevalent in almost all domains, ranging from traditional ones like oil platforms, public transportation services and space missions to cutting edge ones, such as cloud computing, social networks, intelligent and biomedical devices, and nanotechnologies. Differences among domains stem mainly from clock speed—the frequency and magnitude of the domain change cycle.

The current governing approach to system development is evolutionary in essence, even if not considered “agile” in industry terms [1]–[3]. System evolution has become a strategy, as roadmaps for systems, products, or services are defined and maintained by business stakeholders. System scope and specification change constantly, driving system modification, enhancement, adjustment, and modernization. Concurrent design and development of new system versions, with previous versions still in operation, deployment, adoption or retirement is a matter of course. Large-scale legacy systems experience significant leaps forward with relatively long cycles, while smaller-scale agile systems often move far away from their original starting point and initial intentions over short periods of time.

System evolution involves a host of issues, problems, and often crises, due to the increasing variability in architectural, technological, and operational aspects [4]. The simultaneous emergence and evolution of system form, function, and behavior make the tasks of system integration and verification a major effort, presenting a severe obstacle in many cases. System and component configuration management often becomes a Sisyphean, time-consuming effort. Such an effort requires tracking and controlling component versions and cross-version compliance via system baselines for gradual realization, testing, and deployment [5]. NASA’s four baseline points—“functional”, “allocated”, “product”, and “as-deployed” [6]—respectively align with the primary phases in the system or mission lifecycle—requirement specification, design, implementation, and deployment (see Fig. 1).

Even well-organized, complete, and reliable models and design documents often become outdated or unsynchronized as the systems they specify undergo significant modifications and frequent updates. As a result, configuration management, rationalization, and cross-functional impact tracking is becoming increasingly more difficult, making the investment in design and configuration management seem less valuable. Specification and configuration quality deterioration turns into a vicious circle. Gradual configuration buildup is wrongly perceived as resulting from resource and contract constraints rather than a desired and planned maturation. This is an unintended consequence of the traditional waterfall approach, which assumes that a fully functional system can be delivered within one long, predetermined development cycle. While
nowadays this is deemed impossible and wrong both technologically and business-wise, many large scale programs still operate with this paradigm as an underlying philosophy.

Common modeling languages, such as UML [7] and SysML [8], and architecture frameworks, such as the US Department of Defense Architecture Framework (DoDAF) [9], [10] NATO Architecture Framework (NAF) [11], [12], and the Unified Profile for DoDAF and MoDAF (UK Ministry of Defense Architecture Framework) [13] are not designed for agile, evolutionary system lifecycles. They lack the capability to accommodate system evolution in both their programmatic and product layers [14], [15].

Given this current state of affairs, we suggest a different approach—agile MBSE—to streamlining systems evolution and maturation that unifies the system’s functional requirements specification and evolutionary dimensions within a single, overarching, holistic model. The agile MBSE model enables formal, interoperable definition, architecting, design, development, documentation, and maintenance of the system and its configuration. We study a hypothetical ballistic missile defense system (BMDS), whose specification consists of published information on Iron Dome – an Israeli BMDS with proven capability. Iron Dome itself is briefly described in Section II. The basis for this agile MBSE model is Object-Process Methodology – OPM [16], a holistic conceptual modeling paradigm, approved as ISO standard 19450 [17], briefly reviewed in Section III. Section IV introduces our framework for integrating system evolution into the nominal model. In Section V we model the evolution of our hypothetical BMDS as a case in point, and Section VI summarizes the paper.

II. THE IRON DOME BALLISTIC MISSILE DEFENSE SYSTEM

The Iron Dome is a genuine ballistic missile defense system (BMDS) manufactured by RAFAEL Advanced Defense Systems. It was developed as a defensive weapon against the increasing ballistic missile threat on Israeli cities by terrorist organizations in Lebanon and Gaza. Iron Dome intercepts short range rockets and 155 mm artillery shells with ranges of up to 70 km [18]. Iron Dome is operated by the Israeli Air Force since 2011 and has shot down more than 1200 rockets launched from Gaza to civilian concentrations in Israel, mostly during operations Pillar of Defense (2012) and Protective Edge (2014) [19]. Iron Dome consists of (i) A radar that detects and tracks threatening ballistic objects, (ii) a command and control center (CCC), responsible for planning successful defense plans, monitoring Iron Dome’s sensors, actuators, and resources, overseeing the defense mission, and collecting data for debriefing and analysis, (iii) an array of launchers scattered in the vicinity of the CCC and radar, armed with Tamir interceptors; and (iv) the Tamir interceptor, which eventually targets, hits, and destroys ballistic missiles in midair.

III. OBJECT PROCESS METHODOLOGY (OPM)

OPM is a leading holistic conceptual modeling methodology for architecting, design, and analysis of complex systems, products, services, and processes [16], [21]. OPM models capture structure, function, and behavior within the same unified static-dynamic view at various levels of detail [22]. This unification has proven to alleviate system complexity and simplify its management [23]. OPM is bimodal: both graphical and textual. Each model notion captured visually in an OPM diagram (OPD) is accompanied by a formal textual description in Object-Process Language (OPL), a subset of English. OPM’s modeling elements are Process (ellipse), Object
(rectangle), and state (routangle), along with structural and procedural links, capturing various types of relations. An OPM metamodel of objects, processes, and structural links is shown in Fig. 3. Procedural links and states are shown in Fig. 4. The OPL sentences were embedded in the diagram next to their respective visualizations.

OPDs are uniform, self-similar, hierarchically-organized, and interdependent. An OPD can be extended to lower level OPDs through (i) unfolding, (ii) in-zooming, and (iii) view derivation. Unfolding caters to hierarchical breakdown and elaboration of structure and function. In-zooming is used for procedural and behavioral elaboration of processes into ordered subprocesses or for discovering the internal processes and sub-objects of black-box objects. View derivation highlights or elaborates interesting or important aspects around pivotal model elements, such as end-to-end state transitions. All OPDs in the model are cross-validated, associated, integrated, and consistent with the rest.

OPM’s freely available CASE tool, OPCAT [24] provides a modeling and model management GUI, auto-generates formal textual specifications in response to graphical model editing, and simulates models by executing them for behavior validation and verification. OPM has been adopted by various complex socio-technical system development programs in aerospace and defense, information systems, medicine, biochemistry, and space exploration. In addition, OPM has been shown to accommodate various model-based aspects, such as standards authoring [25], project management [15], [26], requirements engineering [27], risk analysis [28], and decision making [29].

OPM is a viable alternative to wearisome text-based systems specifying. It provided integrity and consistency, and mitigates costly deficiencies discovered through field trial and error. OPM also challenges multi-diagram, multi-symbol languages by capturing the three major system aspects—structure, behavior, and function—using a minimal universal ontology of stateful objects and processes that transform them, organized in a set of diagrams of a single kind.

IV. AGILE MBSE WITH OPM

Our Agile MBSE approach consists of the notion that the system and the model evolve concurrently, such that the model is aligned with the system or the system-to-be at all times. Evolving System Model (ESM), a key concept describing the modeling pattern introduced in this paper, is an intentionally ambiguous phrase, as the process Evolving can refer to both System and Model. As we show next, OPM, our underlying conceptual modeling framework, caters to system evolution, evolutionary modeling, and, consequently, for evolving systems engineering.

We clarify our approach by first presenting a simple modeling pattern in Fig. 5. We define System as comprising several Subsystems. System has some high-level emergent Functionality. Functionality is abstract and emergent, while its constituent Functions are actual operations of the Subsystems. Unlike Functionality, which relates to the functional system aspect, Scenario relates to the behavioral aspect: It is a set of Functions, aimed at accomplishing some goal or objective, and can encompass several systems and users. The distinction between Functionality and Scenario provides for modeling two system aspects—the functional and the procedural—to coexist within the same model. This way, we can model emergence—emergent traits of a system as its functionality—a specific combination of ordered functions whose value to some beneficiary is greater than the sum of the values of the individual functions.
The above pattern is useful for early system definition and conceptualization. However, applying this pattern to existing complex systems or ongoing system development programs involves complications and ambiguities. In order to streamline the transition to working with the ESM pattern, we add an evolution layer, as shown in Fig. 6. This layer consists of: a) **Subsystem Variants**, b) **Function Variants**, and c) **Scenario Variants**. **Function Variants** are the actual manifestations of the conceptual **Functions**, and are assigned (often at a later stage) to **Subsystem Variants**. **Scenario Variants** are the manifestations of conceptual **Scenarios**, and they consist of **Activities** that consist in turn of **Function Variants**.

![Fig. 6. Evolving System Model Pattern: Scenario, Subsystem, Function, Input, Output and Resource Set all have Variants as specilizations.](image)

Using this pattern, several variants of the same subsystem can be defined in the same model, and functions can be assigned to these variants. Multiple **Function Variants** may be available under a single deployment as part of a **Baseline**, – a situation called overloading, or functional polymorphism. **Function Variants** are reusable: **Activities**, the building blocks of procedural **Scenario Variants**, consist of these **Function Variants** in multiple scenarios. **Scenario Variants** are the manifestations of the way **Scenarios** evolve over time, due to **Functionality** additions and requirement changes.

The modeling pattern’s final extension includes of **Input Variants**, **Output Variants**, and **Resource Set Variants**. **Input** and **Output** may evolve as **Functions** evolve, although it is a good practice to avoid extreme interface changes. **Resource Sets**, required for **Activity** or **Function** to occur, can also evolve, though this happens less often. **Input Variant** and **Output Variant** management is especially important for business-level or system-to-system interfaces and interactions. Input and output formats evolve over time to accommodate changes to baseline requirements. For example, the same input message can have several versions to support input from several resources while keeping the enveloping structure and name of the input unit. This practice is known as object **polymorphism**.

Polymorphism is a common design practice even for non-evolutionary needs, but it becomes a critical principle to employ when evolutionary conditions set in. Thus, our pattern supports both polymorphic object and process structure design as well as adoption of this approach to cope with systemic or environmental evolution. The model supports the definition of an abstract, variant-indifferent layer and of the decomposition of abstract objects and process to their actually-used variants. Thus, the same models can have one function to represent all of its variants and service all internal and external users, while function variants are mapped to and used by specific users. A complete formal and auto-generated OPL textual specification of the finalize ESM pattern appears in Fig. 7. The text is compact, plain, and easily understandable by both experienced and unexperienced readers, users, and reviewers.

<table>
<thead>
<tr>
<th>ID</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td><strong>System</strong> exhibits many <strong>Baselines</strong> and many <strong>Functionals</strong>.</td>
</tr>
<tr>
<td>1100</td>
<td><strong>Baseline</strong> relates to many <strong>Subsystem Variants</strong>.</td>
</tr>
<tr>
<td>1200</td>
<td><strong>Functionality</strong> consists of many <strong>Functions</strong>.</td>
</tr>
<tr>
<td>2000</td>
<td><strong>System</strong> consists of many <strong>Subsystems</strong>.</td>
</tr>
<tr>
<td>2100</td>
<td><strong>Subsystem</strong> exhibits many <strong>Functions</strong>.</td>
</tr>
<tr>
<td>2110</td>
<td><strong>Function</strong> requires <strong>Resource Set</strong>.</td>
</tr>
<tr>
<td>2120</td>
<td><strong>Function</strong> consumes <strong>Input</strong>.</td>
</tr>
<tr>
<td>2130</td>
<td><strong>Function</strong> yields <strong>Output</strong>.</td>
</tr>
<tr>
<td>3000</td>
<td><strong>Input Variant</strong> is an <strong>Input</strong>.</td>
</tr>
<tr>
<td>4000</td>
<td><strong>Output Variant</strong> is an <strong>Output</strong>.</td>
</tr>
<tr>
<td>5000</td>
<td><strong>Resource Set Variant</strong> is a <strong>Resource Set</strong>.</td>
</tr>
<tr>
<td>6000</td>
<td><strong>Subsystem Variant</strong> is a <strong>Subsystem</strong>.</td>
</tr>
<tr>
<td>7000</td>
<td><strong>Subsystem Variant</strong> exhibits many <strong>Function Variants</strong>.</td>
</tr>
<tr>
<td>7100</td>
<td><strong>Function Variant</strong> is <strong>Function</strong>.</td>
</tr>
<tr>
<td>7200</td>
<td><strong>Function Variant</strong> requires <strong>Resource Set Variant</strong>.</td>
</tr>
<tr>
<td>7300</td>
<td><strong>Function Variant</strong> consumes <strong>Input Variant</strong>.</td>
</tr>
<tr>
<td>7400</td>
<td><strong>Function Variant</strong> yields <strong>Output Variant</strong>.</td>
</tr>
<tr>
<td>8000</td>
<td><strong>Scenario</strong> consists of many <strong>Functions</strong>.</td>
</tr>
<tr>
<td>9000</td>
<td><strong>Scenario Variant</strong> is <strong>Scenario</strong>.</td>
</tr>
<tr>
<td>10000</td>
<td><strong>Scenario Variant</strong> zooms into many <strong>Activities</strong>.</td>
</tr>
<tr>
<td>10100</td>
<td><strong>Activity</strong> is instance of <strong>Function Variant</strong>.</td>
</tr>
</tbody>
</table>

![Fig. 7. Evolving System Model Pattern: Formal textual OPL specification corresponding to the OPD in Fig. 6.](image)

The generic ESM pattern introduced here provides a single notation for both the abstract pattern and the actual model. Thus, the pattern can be easily instantiated as an actual model of the system of interest, without any notational transformation. In addition, similar artifacts such as subsystems, functionalities, scenarios, and their internal parts can be easily added. Once the model is defined according to this evolvable pattern, it can be easily extended to include additional variants. System engineers need not capture multiple variants without adequate built-in configuration control. Rather than avoiding variant representation and adhering to the nominal model, this pattern allows them to conveniently represent variability while preserving model consistency.

The gradual exposure of the ESM pattern and its extensions is intentional: it resembles and demonstrates actual system
model construction and extension to accommodate system evolution. It also shows OPM’s capability to cater to model evolution and maturation in conjunction with system design and architecture evolution, which is a key component in agile MBSE. The capability of this modeling framework to capture variations and modifications to an existing design reinforces the conceptual model’s viability and value as a systems engineering tool. This modeling capability is powerful as it can work equally well in slowly- and quickly-evolving systems alike. Moreover, it complements and facilitates agile system development by providing a clear visual representation of the conceptual and actual specification and design of the solution, as well as the relations to and discrepancies (“deltas”) of the current system as expressed by its model.

V. AGILE MBSE: A BMDS EXAMPLE

In this section we demonstrate the use of ESM as part of the agile MBSE approach on a BMDS with evolutionary characteristics. Our BMDS is a hypothetical one, but our specification is loosely based on publicly available information about Iron Dome. The various reports suggest that Iron Dome has allegedly gone through tremendous upgrades to enable interception of emerging threats, extended target ranges, reduced interceptor consumption, improved salvo handling, integration with other BMDSs, UAV interception, etc. [20]. However, this paper has no intention to claim or argue that our hypothetical system reflects the way the real Iron Dome system operates.

To complement this structural-functional view, we add a procedural view showing the Nominal Scenario, including the Ballistic Missile Launching and Flight, Defense Planning Activity based on the functionality of the BMDS, The Interception Activity of Interceptor, and the successful Ballistic Missile Destruction, which consumes both the Ballistic Missile and the Interceptor. This view is illustrated in Fig. 9.

This naive simplified model assumes that every ballistic missile is detected by the system and then intercepted. However, as published, in order to save interceptors and avoid an economic attrition war, BMDSs only intercept ballistic missiles if they are going to hit a defended area and are within feasible interception conditions. Therefore, the first modification includes additional functionality to support the Defended Area Management, as well as Defended Area Hitting Probability Assessment. This addition is illustrated in Fig. 10. In Fig. 11. The assessment function becomes part of Defense Planning in the Nominal Scenario. The results of Detection and Hit Probability Assessment either enable the flow of the process or restart it. Defended Area Management is not part of the nominal scenario, but rather part of a separate BMDS Preparation scenario, which includes the deployment of the system’s components, connecting them, and performing a set of preparation activities, some of which are supported by system functions. That scenario, however, is beyond the scope of this paper.

The Interception Planning function modification covers the infeasible Interception option, in which case Defense Planning restarts as well, while feasible planning triggers Intercepting as shown in Fig. 11. The earlier version, which did not include these modifications in the assessment and interception planning processes, could be typical to very early test-range-only proof of concept—an early system version. At a later stage, however, it would be mandatory to include these capabilities in order for the BMDS to be both safe and beneficial to operate.

Based on publications on speculated or actual UAV interception capability, we demonstrate how this additional functionality and its constituent functions would be variants of their nominal ballistic missile defense ancestors. This makes sense, because the system is expected to be indifferent to the
kind of threat it detects—missile or UAV—and at the same
time identify it, assess it, and plan its interception accordingly.
Thus, UAV Scenario is a variant of the nominal scenario, while
UAV Defense Planning is a variant of the nominal Defense Planning.
The internal activities consist of variants of the BMDS
functions. This modeling approach enables the encapsulation of
the variant UAV Scenario within the Nominal Scenario, and the
derivation of the relevant variants in both the procedural aspect
and the functional aspect. The UAV scenario-functionality dual
variant is illustrated in Fig. 12.

The variant UAV Defense functionality is now unfolded in a
dedicated diagram, defining its constituent functions. The
baseline constituent functions of Ballistic Missile Defense
functionality are selectively adopted as they are, replaced by
corresponding function variants, or removed altogether if they
are irrelevant. Additional functions are added wherever
necessary. An unfolded view of UAV Defense appears in Fig. 13.

The UAV Defense functionality differs from the basic Ballistic
Missile Defense functionality in several ways. First, the detection
and tracking of enemy aircraft and UAVs are made by external
radar systems that are optimized for aircraft kinematics. A
central external subsystem, Air Defense Command, is responsible
for providing the detection and tracking services. Thus, the Radar subsystem becomes irrelevant for the UAV Defense functionality and UAV scenario. Specialized UAV Damage Assessment and UAV Interception Planning functions for UAVs replace the original ones used for Ballistic Missile. In addition, aircraft interception mandates approval request from Air Defense Command, while ballistic missile interception does not require such approvals. Hence, an Air Defense Approval Requesting function is also added. Finally, the UAV’s physical interception requires the interceptor to carry out aerodynamic maneuvers that are different from the ones needed for ballistic missile interception. Hence, a function variant of Intercepting – UAV Intercepting, has been added.

Zooming into the UAV Scenario in Fig. 14, we define activities and activity variants utilizing the constituent functions comprising the UAV Defense functionality. Since this is a scenario variant, some of its constituent activities are variants as well. Environmental activities that occur as part of the scenario are not defined as variants, because they are not a part of the system. However, if, for example, we want to emphasize that an external activity, which is performed by an external system, is a variant of the other system, it can be captured in the model as well. Such a distinction might not even impact the system-of-interest. In this model we see that the UAV takeoff and UAV flight are similar to the Ballistic Missile launching and Ballistic Missile flight, but this does not mean that they are variants. UAV Engaging, on the other hand, can be a variant of another function. If, for example, the Air Defense Command already implements a similar scenario with another defense system, parts of that scenario could be reused for the BMDS. BMDS designers may prefer to omit this detail if they don’t see it as informative or relevant to the BMDS.

In this section we have shown how the ESM pattern, a central component of our evolving agile MBSE approach, is applied to the evolving BMDS model. We began with a model of a nominal system for ballistic missile interception, and gradually extended it to also cover a) selective ballistic missile engagement based on area defense policy, and b) UAV interception. We adjusted the model in both the functional and procedural aspects by defining functionality and function variants for the former, and scenario and activity aspects for the latter. In the end of this evolution cycle, the baseline ballistic missile defense scenario and the emerging UAV defense scenario are both operational, as are their corresponding functional building blocks.

VI. SUMMARY

We propose a fresh approach to agile model-based systems engineering (MBSE) that integrates the evolutionary dimension as part of the core system model. This time- and version-aware framework supports rich, informative, valuable, and usable evolvable system model (ESM) creation and management by gradually constructing a configuration layer on top of a nominal system model. In preparation for extending the system model, we formalize the distinction between the functional and procedural model aspects and provide a supporting modeling pattern. While this distinction may not be mandatory for configuration buildup, formulating the ESM with this distinction enhances the evolutionary model aspects of form and function.

We have demonstrated the applicability of this modeling pattern on a hypothetical, simplified ballistic missile defense system, based on publicly available information about the Israeli Iron Dome. In our example, the operational capability is extended to allow the interception of unmanned air vehicles (UAV). We show that UAV-oriented functional and procedural views are derived from their nominal model counterparts, such that the resulting model reflects an integrated defense system that performs both missions interchangeably.

Agility and evolution on the one hand, and formal modeling and design on the other, are not conflicting objectives or paradigms. Conversely, they reinforce each other when the system is modeled in an evolvable manner, using our ESM pattern, based on Object-Process Methodology as the underlying conceptual modeling framework. One can view this modeling pattern as a variant of nominal OPM modeling, adjusted for the ESM scenario. Further extension and elaboration include ESM simulation and behavior mapping, which increase the reliability, consistency, and verifiability of the model and its realization.

ACKNOWLEDGMENT

This research was supported by Israel Aerospace Industries – IAI as part of academic collaboration with the Technion – Israel Institute of Technology. The authors would like to thank the SysCon 2015 referees for their useful feedback.

REFERENCES


