Model-based risk-oriented robust systems design with object-process methodology

Yaniv Mordecai*

Technion – Israel Institute of Technology,
Haifa 32000, Israel
E-mail: yanivmor@technion.ac.il
*Corresponding author

Dov Dori

Technion,
Israel Institute of Technology,
Haifa 32000, Israel
and
Massachusetts Institute of Technology,
Cambridge, MA 02139, USA
E-mail: dori@mit.edu

Abstract: We introduce and demonstrate ROSE – risk-oriented systems engineering, a new approach that integrates risk aspects with system analysis, modelling and design. ROSE is designed to improve the integration and coordination of system design and risk management by fusing robust design paradigms with risk analytic techniques in a model-based environment. While system design and risk management are two critical systems engineering processes, their integration is loose, because too often systems engineers and risk analysts use different semantics, techniques, and tools. This unfortunate disconnects renders risk management efforts detached from system design and management. Object-process methodology (OPM) is a bimodal visual and textual conceptual modelling language and an emerging ISO Standard (19450) for system modelling and design. Making use of OPM, ROSE integrates risk identification, modelling, analysis, mitigation, and control aspects into the robust system design process, and later into system deployment, configuration, and management. Using a commercial airliners defence system against shoulder missiles as a case in point, we demonstrate the principles and benefits of ROSE in risk-oriented systems.

Keywords: risk; risk management; risk modelling; robust systems design; system configuration; object-process methodology; OPM; conceptual modelling; model-based risk analysis; MBRA; model-based systems engineering; MBSE.

Reference to this paper should be made as follows: Mordecai, Y. and Dori, D. (2013) 'Model-based risk-oriented robust systems design with object-process methodology', *Int. J. Strategic Engineering Asset Management*, Vol. 1, No. 4, pp.331–354.

Biographical notes: Yaniv Mordecai holds a BSc and MSc (with honours) in Industrial Engineering and Management from Tel Aviv University, Israel (2002, 2010). He is currently a PhD candidate at the Technion – Israel Institute of Technology. He served as an Officer in the Israeli Air-Force where he practiced project management and system engineering in large-scale projects. He currently works for Elbit Systems, as a Command and Control System Engineer. His research interests include systems engineering, model-based systems engineering, risk analysis, interconnectivity and interoperability analysis, operations research, and decision-making.

Dov Dori is an Information and Systems Engineering Professor and the Head of the Enterprise System Modelling Laboratory at the Faculty of Industrial Engineering and Management, Technion, and a Visiting Professor and Research Affiliate at the Engineering Systems Division, Massachusetts Institute of Technology. He received his PhD in Computer Science from Weizmann Institute of Science, Israel (1988), his MSc in Operations Research (with honours) from Tel Aviv University, Israel (1981), and his BSc in Industrial Engineering and Management (with honours) from the Technion, Israel (1975). He won the Technion Klein Research Award, and the Hershel Rich Innovation Award for Object Process Methodology (OPM) and for OPCAT, an OPM software tool. He won the Dudi Ben Aharon Research Award for Document Image Understanding. He has authored over 200 publications.

This paper is a revised and expanded version of a paper entitled ‘Risk-oriented systems engineering: integrating risk into systems modelling with OPM’ presented at Third International Engineering Systems Symposium CESUN 2012, Delft University of Technology, The Netherlands, 18–20 June 2012.

1 Introduction

Robust systems design allows for reduction of development cycle cost and duration, enables gradual product adaptation, and improves response to predicted or evolving market requirements (Diaz, 1998). However, robustness often increases initial time-to-market and development costs, and balancing these objectives is a continuous product management decision-making problem (Krishnan and Ulrich 2001). Nevertheless, it appears that robust design is in general less risky than case-specific design optimisation (Gaury and Kleijnen, 1998).

Robust design leads to operational configurability and flexibility. Lockheed-Martin’s flag military carrier aircraft, C-130 ‘Hercules’, for instance, is renowned for its versatility. Due to its robust design, it can be configured for various missions, including troop airlift, paradropping, medical evacuation, search and rescue, cargo, scientific and weather reconnaissance, aerial refuelling, and aerial firefighting (USAF, 2009).

System robustness is manifested by such qualities as configurability, modularity, and programmability. Controllable system parameters, and activation and deactivation of features or components, are basic configuration control measures. The ability to reshape, reorder or reconstruct the system based on its building blocks, further extends deployment and configuration possibilities. Reprogramming and modification support extend system functionality in run-time, and are useful especially in software and software-defined hardware. Vendors provide clients with various baseline-configured off-the-shelf robust and configurable products, especially software product. Clients shape the actual solution almost completely through configuration, calibration, and capability generation, capitalising on the product’s inherent configurability and programmability.

Robust design is a useful risk management strategy that has a dual effect:

a it helps coping with development risks, such as vague or contradicting requirements

b it endows the system with the flexibility needed to face various operational challenges and working conditions.

However, introducing flexibility also incurs risk, as it may result in increased development cost and duration, and open the door for extreme, insufficiently risk-hedged configurations. Kapelan et al. (2006) demonstrate the close connection of robustness and risk-oriented design in this context. Providing full built-in risk response capability to every configuration is difficult or even impossible due to the combinatorial state-space explosion and the system's emergent and unpredictable features.

Robust systems are by-design flexible, resilient, and configurable. During system operation, its configuration serves as a risk response agent. Thus, while robust design responds to programmatic and development risks, system configuration setting responds to operational risks. This dual effect is captured in a risk design pattern, in which project risks are addressed by robust design, adding flexibility and configurability to mitigate operational risks. Risk management involves intricate subtle trade-offs among contradicting objectives and fine balancing actions achieving stability and Pareto-optimal satisfaction of the system within its environment.

While robust design and risk aware design seem closely related, no consolidated modelling framework, which binds them together, and accounts for both their synergetic effects and balancing issues, appears to exist. While robust design seeks to maximise the capability of the system to act in multiple operating scenarios, risk-oriented design seeks to minimise the probability of failure due to pre-identified risks. These seem like two sides of the same coin.

The purpose of this paper is to introduce such a model-based integrated approach, which fuses robust design with risk-oriented design. This approach is therefore named risk-oriented systems engineering (ROSE). The challenge in creating such an integrated approach is to unify information and knowledge from several domains through a systematic model-based approach, language, and methodology. These have to be intuitive, simple, and formal. For these reasons, we chose object-process methodology (OPM) as our underlying modelling framework.

OPM (Dori, 2002) is a structured conceptual modelling framework with a bimodal textual and visual representation. OPM is an emerging ISO Standard (ISO 19450), and an underlying framework for process and system modelling within ISO Standards. OPM's simple yet robust notation enables modelling layer addition on top of, and in accord with the core system model. Thus, a coordinated and consistent multi-layered model is formed. In particular, using OPM, a risk model component can be naturally added as a model layer.

The rest of this paper is organised as follows. Section 2 provides a literature review on risk management, model-based risk analysis (MBRA), and OPM. Section 3 presents the ROSE approach, its goals and objectives, and its configuration and design modelling theory and methodology. Section 4 demonstrates the application of ROSE for design and operation of a commercial airliners defence system against shoulder missiles. Finally, Section 5 summarises this paper and delineates directions of future and complementary research.

2 Literature review

2.1 Risk management

Risk management is a key success factor in both the project management and systems engineering domains (PMI, 2000; Haskins et al., 2007). Risk management aims at reducing the probability of occurrence of risky processes and their adverse impact on stakeholder objectives and assets. Risk management focus changes during a system's lifecycle. Project risk management (PRM) focuses on reducing delays and cost overruns, while satisfying specification (spec) and quality requirements (Chapman and Ward, 2003). Operational risk management (ORM) is concerned with assuring such system objectives as reliability, safety, security, availability, and business continuity (some of the so-called 'ilities') in operational settings subject to risk (Hoffman, 2002; Haimes, 2009). Several guides and standards with general applicability or relevance to particular domains have been published (PMI, 2000; Stoneburner, 2002; ISO & IEC, 2004; NASA, 2007; Sage and Rouse, 2011).

System risk analysis requires quantitative, probabilistic techniques (Cooke, 1991; Bedford and Cooke, 2001), and dedicated system-oriented methods (Haimes, 2008), in addition to classical risk analysis methods, such as fault-tree analysis (FTA), failure mode effect critical analysis (FMECA) (Haimes, 2009), and hazard and operability (HAZOP) (Redmill et al., 1997). Analytical risk-integrated system modelling attempts to define the system's (multi-)objective function, while capturing risk, using mathematical building blocks such as input, output, state variables, decision (control) variables, and random variables. System vulnerability is a manifestation of specific inherent states of that system. State transitions occur in response to the inputs and other building blocks (Haimes, 2009).

2.2 Model-based risk analysis

Several integrated system design and risk management frameworks have been developed. We survey the central ones below:

1. CORAS – an object-oriented UML-based framework for risk assessment of security critical systems, employing several risk analysis methods and computerised tools (Fredriksen et al., 2002; Lund et al., 2011).
2. BRPIM – business process risk integrated modelling – a framework integrating operational risk modelling and organisational business process planning and modelling (Sienou et al., 2008).
3. Quantitative risk assessment for component-based systems (Grunske and Joyce, 2008).
4. RiskM – a multi-perspective method integrating IT risk consideration, assessment and mitigation, into business process modelling and execution (Strecker et al., 2010).

CORAS is considered the most thorough and wide-ranging MBRA framework (Lund et al., 2011). Its underlying modelling framework – the common de-facto UML standard – increases its spread and accessibility. Nevertheless, CORAS has some major disadvantages, which hinder its integration with model-based systems engineering (MBSE), including the following:

1 CORAS diagrams are specialised for its risk management process phases. Thus, some diagrams are outside of the UML conventions. Moreover, extensive use of free text elements reduces model formality, automated processing capability, and further logical and quantitative analysis.

2 The framework does not address possible risk trade-offs, i.e., the ability to exchange one set of risk impacts for another, such as paying for insurance against a certain risk, or increasing resource allocation to risk mitigation efforts.

3 Structural relations among risk barriers and the components they belong to are missing. Therefore, it is difficult, for instance, to allocate components and external entities as risk sources or risk mitigation instruments.

4 CORAS focuses on logical cause-and-effect analysis rather than on quantitative analysis of expected impact, utility, and cost.

5 Uncertainty-related properties of system components, like hardware reliability measures or lifespan, cannot be easily integrated into the model.

6 CORAS focus on security underutilises and neutralises UML's potential. As CORAS has not been extended to general systems, (e.g., through SysML, another UML-profile), implementing CORAS in non-IT systems is not straightforward, even for information security purposes.

Additional limitations of some of the abovementioned methods include the following:

1 Overemphasis on IT and lack of support for socio-technical, electro-mechanical, aerodynamic, real-time communication and processing, and command and control systems.

2 lack of model formalism, automated processing and simulation capabilities, consistency verification, complexity management, and hierarchical analysis.

3 Insufficient support of quantitative analysis and assessment, probabilistic modelling, and uncertainty-related properties.

Interestingly, MBRA methods focus on expressive descriptions and visualisations of risk aspects rather than quantitative aspects, as opposed to the classical risk analysis methods and techniques.

2.3 Conceptual modelling with OPM

OPM (Dori, 2002) is a holistic, integrated approach for complex dynamic systems design and development. Using a minimal set of symbols, OPM integrates the functional, structural, and procedural aspects of a system in one view, expressed both graphically and textually. OPM copes with complexity via detail-level decomposition, contrasted with aspect decomposition, characteristic of UML/SysML.

OPM building blocks are objects and processes, collectively called things. Objects are things that exist and can be stateful (i.e., have states). Processes are things that occur, and transform objects: they generate and consume objects, or affect stateful objects by changing their state. These building blocks are connected by links of two types: structural and procedural. Structural links specify relations between objects, or between processes. Conversely, procedural links connect processes with objects or states. OPM supports the designation of entities as systemic or environmental, and as physical or informatical. A brief description of OPM notation, accompanied by illustrations and comments, is provided in Appendix.

An OPM model consists of a set of hierarchically organised object-process diagrams (OPDs). The hierarchical structure alleviates system complexity through three mechanisms:

1 unfolding and folding of structural hierarchies of things (primarily objects)

2 zooming into or out of the inner details of things (primarily processes)

3 expressing or suppressing the states of objects.

Each OPD is obtained by in-zooming or unfolding an object or a process in its ancestor OPD. The graphical representation of an object is a rectangle, while a process is represented by an ellipse. Object states are represented by round-angle rectangles (‘rountangles’) within the owning object. The OPD hierarchical structure is accompanied by a corresponding set of structured textual model description sentences, written in object-process language (OPL), a subset of English. With OPM’s free CASE tool – OPCAT, OPL sentences are automatically generated in response to visual edits of the model. The textual formulation is equivalent to the graphical view, allowing for bimodal textual and visual description and understanding of the model.

3 Risk-oriented systems engineering

In this section, we present ROSE a new approach for risk-integrated systems modelling and design. The challenge is to develop a method, which closes the gaps we have observed in other methods. The main goal of this approach is to enable systems and risk analysts to collaborate and cooperate through an integrated system-risk model. Furthermore, we aim to promote the synergetic effects of integrated robust system design and risk management. In order to be robust, flexible, and extendable, the method should satisfy the following set of objectives:

1 providing an integrated system-risk model, allowing for functional system analysis and design, enhanced by risk analysis and mitigation planning

2 enabling applicability to various types of systems and risks types

3 creating high-fidelity models that enable the designation of model artefacts as risk sources, risked assets and objectives, or risk mitigation agents

4 supporting simulation and automated scenario generation for analysis and derivation of quantitative measures of risk processes in the system

5 encompassing a lifecycle perspective for capturing project/development risk and operational risk, which provides for risk-integrated system design, development, deployment, monitoring, control, configuration, and maintenance

6 catering to modelling agility for initial solution generation, extensions, refinements, and constant improvement of the integrated modelling solution

7 Improved capability to identify and mitigate risks.

Table 1 High-level requirements for model-based, risk-integrated system design

#	*Title*	*Description*	*Comments and extensions*
1	Multiple perspectives	Provide perspectives specific to (groups of) stakeholders involved in the group process.	A perspective should correspond to the abstractions, concepts and (visual) representations known and meaningful to the targeted (groups of) stakeholders. All perspectives should be integrated with each other to foster cross-perspective communication and cooperation.
2	Environmental and organisational context	Account for both system-related risks and chances. Link them to the surrounding action system composed of all relevant environmental and organisational entities.	This requirement extends the organisational context to the environment. Organisational entities include corporate goals, organisational units, and business processes. The environment consists of both the physical and natural environment, and the organisational and business one.
3	Multiple organisational levels	Account for cause-and-effect relationships of system risks and chances, at multiple organisational levels.	
4	Quantitative values, qualitative descriptions	Provide means for both qualitative risk description and risk quantification.	Risk quantification is not always possible, or economically justifiable. Often, qualitative assessment, accompanied by risk description, is sufficient for understanding the risk.
5	Compliance	Support compliance validation and auditing procedures for compliance with regulations, standards, guidelines, and internal controls.	Compliance implies representing the concepts built into regulations, standards, and frameworks, as well as risk monitoring and control mechanisms.
6	Multiple phases	Account for the multiple phases of the risk management process.	Dedicated support for risk modelling and management has to be built into each major system lifecycle phase and into the transition between phases.

Source: Adjusted from Strecker et al. (2010)

Strecker et al. (2010) define six key domain-specific high-level requirements that a method aimed at supporting IT risk assessment should satisfy. Having found these as useful reference requirements for a risk-integrated system design and modelling framework, we adjusted them for our purposes, as summarised in Table 1.

OPM was selected as the basis for ROSE for its following advantages:

1 OPM unifies the static-structural and dynamic-procedural aspects using a single diagram type at varying levels of detail. This reduces clutter and incompatibilities even in highly complex systems.

2 Inherent complexity management is achieved by decomposing system specification into self-similar OPDs at increasing levels of detail, obtained through recursive seamless refinement-abstraction mechanisms.

3 OPM combines semantically equivalent graphical and textual views, which make OPM appealing to both sides of the human brain. Consequently, professionals and practitioners alike can understand OPM models quickly and easily.

4 OPM enables extending the core system model to additional aspects while maintaining full coordination with the core model, as well as the capability to generate metamodels. Generic, multi-purpose models and patterns can later be instantiated and adapted for specific systems and problems.

5 There is a freely available CASE tool – OPCAT, which implements almost all OPM concepts and allows fast adaptation and implementation.

6 OPM is currently in the process of becoming an ISO publically available specification (PAS), ISO 19450, and a basis for system and process modelling in ISO enterprise standards. This enables accelerated dissemination of OPM as a basis for enterprise modelling in general and for risk modelling in particular.

7 Sharon and Dori (2009) demonstrated the superiority of OPM as project-product modelling language over UML, xUML, and, SysML.

ROSE achieves a dual effect by adopting robust design as:

1 a *development* risk reduction approach

2 a facilitator of *operational* risk response capability.

When a system is robustly designed, it includes flexible, resilient, and configurable features. During the operation of the system, the configuration of the system serves as a risk response enabler. Robust design mitigates programmatic and development risks, while system configuration setting responds to operational risks.

With this dual effect principle in mind, we first describe the formal logic of risk-integrated system modelling using OPM as a metamodelling framework. The metamodelling process produces a pure, case-independent formal model, which can be instantiated and implemented for various cases.

The system is the main entity in the model. As we refer to the entire set of development and operational phases of the system, its lifecycle is the most general process. The topmost diagram in the OPM hierarchy, called system diagram, captures the entire essence of the system, provides an initial clear understanding of the problem or system under study, and serves as an anchor for additional modelling. The topmost OPD therefore consists of the main object *system*, and the primary generic process **Lifecycle** (*Arial bold* font is used to denote model entities: objects, processes, and states). Figure 1 is the top-level system diagram of our metamodel. The textual OPL description for this diagram appears in Model spec 1.

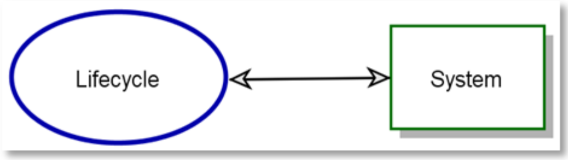
We now analyse the **Lifecycle** process and specify its subprocesses:

a **Project** – in which the system is designed and developed

b **Operation** – in which the system is used, operated, and maintained.

Some intermediate phases, such as mass production, may be regarded as pertaining to either one of these two major phase but in this paper's scope we can ignore them.

Figure 1 System diagram (see online version for colours)



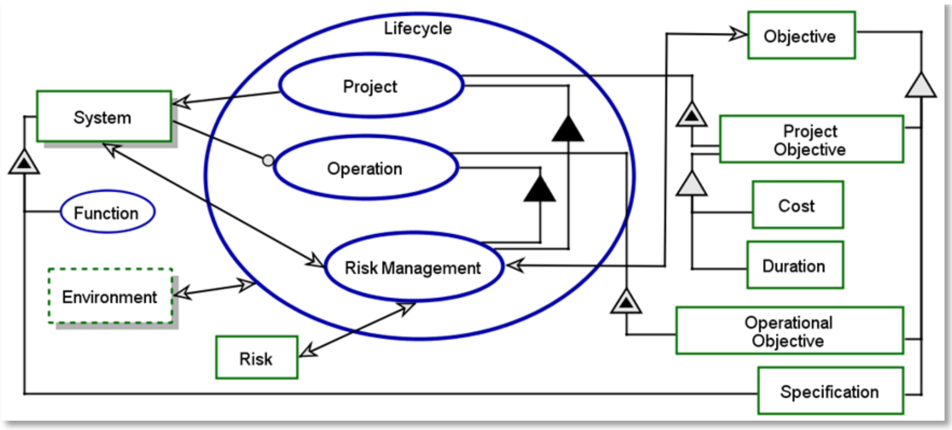
Model spec 1 System diagram – OPL (see online version for colours)

System is physical.

Lifecycle affects System.

We now zoom into the **Lifecycle** process, extending the system diagram with a separate yet related OPD at a lower level. **Lifecycle** comprises two primary subprocesses: **Project** and **Operation**. The OPDis illustrated in Figure 2. The OPL text follows in Model spec 2. In both the OPD and its corresponding OPL description, **Project** is not linked to **System** in the same manner **Operation** is linked to **System**. In OPL: 'project yields system', while 'operation requires system'.

Figure 2 Lifecycle in-zoomed – OPD (see online version for colours)



Model spec 2 Lifecycle in-zoomed – OPL (see online version for colours)

Lifecycle

System exhibits Specification.

Specification is an Objective.

Lifecycle affects Environment.

Lifecycle consists of Project, and Operation.

Project and Operation consist of Risk Management.

Project yields System.

Project exhibits Project Objective.

Project Objective and Operational Objective are Objectives.

Cost is a Project Objective.

Duration is a Project Objective.

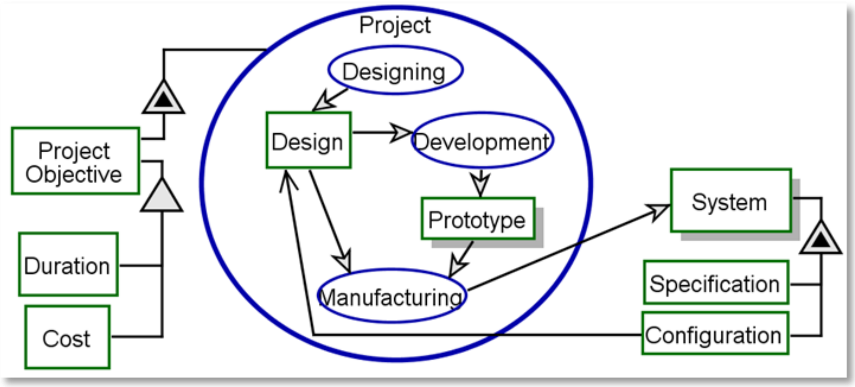
Operation requires System.

Operation exhibits Operational Objective.

Risk Management affects Risk, Objective, and System.

Risk Management is as a subprocess of both the **Project** and **Operation**. This model focuses on the integration of **Risk Management** into the general system model, and therefore sets aside other aspects, which are also common to the **Project** and **Operation** phases, such as quality management, documentation, and training, among others. The commonality of **Risk Management** is shown as a common subprocess (part) of **Project** and **Operation** in **Lifecycle**.

Figure 3 Lifecycle/project in-zoomed – OPD (see online version for colours)



Model spec 3 Lifecycle/project in-zoomed – OPL (see online version for colours)

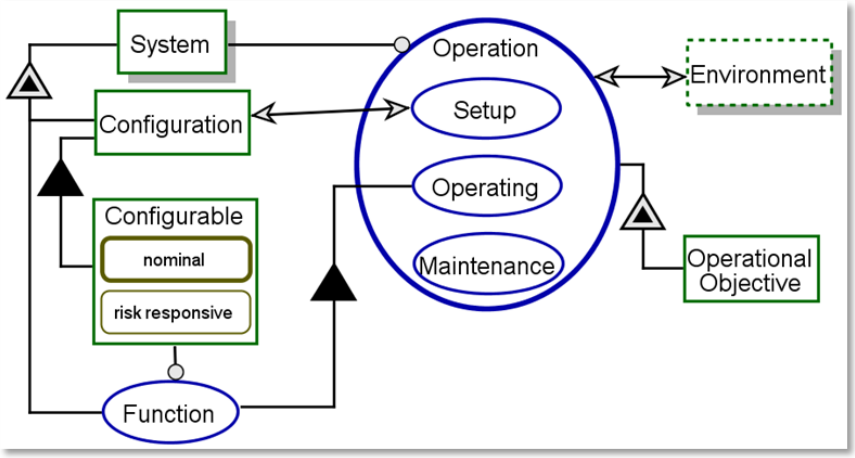
Lifecycle/Project

System exhibits Configuration.

Configuration relates to Design.

The OPDs at the next level down zoom into each subprocess in the primary process: **Project** (OPD in Figure 3, OPL in Model spec 3), **Operation** (OPD in Figure 4, OPL in Model spec 4), and **Risk Management** (OPD in Figure 5, OPL in Model spec 5). Excerpts from the corresponding OPL texts, referring to system design and configuration management, appear at the bottom of each figure.

Figure 4 Lifecycle/operation in-zoomed – OPD (see online version for colours)



Model spec 4 Lifecycle/operation in-zoomed – OPL (see online version for colours)

Lifecycle/Operation

Operation consists of Setup, Operating, and Maintenance.

Setup affects Configuration.

Configuration consists of many Configurables.

Configurable can be nominal or risk responsive.

System exhibits Function.

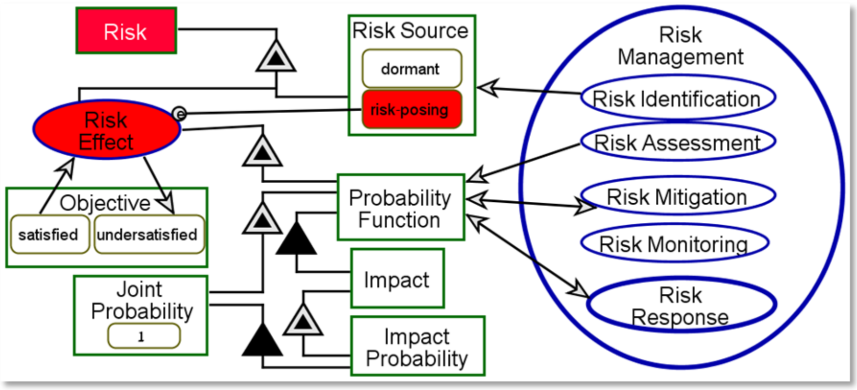
Function requires Configurable.

Operating consists of Function.

As noted, the **System** is generated during the **Project** phase, exhibiting a possible **Configuration**, which corresponds to the **Design** of the system. **Design** itself may also be a deliverable, and it is indeed used within the **Project**, as the output of the **Designing** phase, and the input of the **Development** and **Manufacturing** phases. **Project** also has two programmatic **Objectives**: **Duration** and **Cost**. The **Specification** of the **System**, which is also an **Objective**, is captured in the properties of the **System**, as elaborated in the sequel.

During the **Operation**, the **System's Configuration** may be set up or changed to match various operational modes and needs, and satisfy various **Operational Objectives**. This process is known as **Setup**. **Configuration** is a set of **Configurables** – objects that may be changed and configured during **Operation**. **Configurables** may be in various predetermined states and their attributes can assume certain values, as defined during the **Project** phase. Thus, robustness and configurability are emerging qualities: an aspect of a system is considered robust and configurable when its states, values, or modes, can be changed and adjusted for operational uses during the operation of the system (or product, or service). A component of the system is *robust* if it embodies the system's proven capability to support various working conditions. Thus, possible modes and states of the **System's Configurables** enable system processes that contribute to the operational functioning of the system – its ability to provide value to its beneficiaries.

Figure 5 Lifecycle/risk management in-zoomed – OPD (see online version for colours)



Model spec 5 Lifecycle/risk management in-zoomed – OPL (see online version for colours)

Lifecycle/Risk Management

Risk Management consists of Risk Identification, Risk Assessment, Risk Response, Risk Mitigation, and Risk Monitoring.

Objective can be satisfied or undersatisfied.

Risk exhibits Risk Source, as well as Risk Effect.

Risk Source can be risk-posing or dormant.

Risk Source triggers Risk Effect when it enters risk-posing.

Risk Effect exhibits Probability Function.

Probability Function exhibits Joint Probability.

Joint Probability is 1.

Joint Probability consists of many Impact Probabilities.

Probability Function consists of many Impacts.

Impact exhibits Impact Probability.

Risk Effect changes Objective from satisfied to undersatisfied.

Risk Identification yields Risk Source.

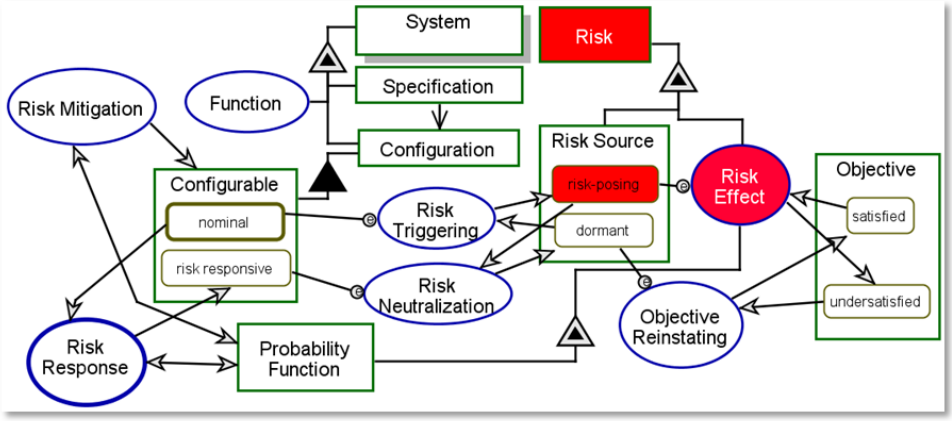
Risk Assessment yields Probability Function.

Risk Mitigation affects Probability Function.

Risk Response affects Probability Function.

The **Risk Management** process, common to both **Project** and **Operation**, is initially modelled in Figure 5 as a pattern. **Risk** is defined as an abstract idea, featuring a **Risk Source** and a **Risk Effect** (a separation of source and effect that many risk modelling methodologies fail to make). The **Risk Management** process mainly consists of the following subprocesses: **Risk Identification**, **Risk Assessment**, **Risk Mitigation**, **Risk Monitoring**, and **Risk Response**. Each phase generates or handles various parts and aspects of the **Risk** construct. Any object in the system or in the environment may constitute a **Risk Source** for particular **Risks**, and, at the same time, be targeted by other **Risks**. A **Risk Source** must be in its **risk-posing** state in order to trigger the **Risk Effect** process, which, when executed, causes some **Objective(s)** of the **System** to become **undersatisfied**. Risk-related entities shown in Figure 5 are coloured in red to make them easily distinguishable.

Figure 6 Lifecycle/risk management/risk response in-zoomed – OPD (see online version for colours)



Model spec 6 Lifecycle/risk management/risk response in-zoomed – OPL (see online version for colours)

Lifecycle/Risk Management/Risk Response

Risk Mitigation yields Configurable.

Risk Response changes Configurable from nominal to risk responsive.

Configurable triggers Risk Triggering when at nominal.

Configurable triggers Risk Neutralisation when at risk responsive.

Risk Effect exhibits Probability Function.

Risk Triggering changes Risk Source from dormant to risk-posing.

Risk Neutralisation changes Risk Source from risk-posing to dormant.

We are now in a position to metamodel the mutual effects of risk management on design and configuration, and vice versa. Indeed, risk management involves intricate subtle trade-offs among contradicting objectives and fine balancing actions to maintain stability and overall Pareto-optimal satisfaction of the system within its environment. The metamodel is illustrated in Figure 6. Risk-related entities are coloured in red for a better distinction. The corresponding OPL supplement is described in Model spec 6. This textual description is the essence of our approach, as it guides system analysts and risk analysts alike, in integrating risk into system models and configuration considerations.

This part of the model captures risk handling in both the **Project** and the **Operation**, and clarifies the relations between these risk handling patterns. As we have claimed in the beginning of this section, design-related, project-time risk handling, namely **Risk Mitigation**, facilitates and designates **Configurables** in the **System**, acting as risk reduction and mitigation mechanisms and agents. Operation-related risk handling, namely **Risk Response**, includes the setting up or modification of the **System's** mode or state through the selection of appropriate values for the **Configurables** which were embedded in the **System** during the **Project**. In other words, if a system has no configuration change capability whatsoever (an extremely rare situation), no risk response activity may be applied by the system, or via the system, except for the termination or removal of the system (which, with proper modelling, ought to be supported by adequate configurable components and states of the system). However, if every item in the system is configurable so that it may be adjusted to match various working conditions, then the number of possible risk response actions depends on the combination of possible configurable states that provide the desired response to an emerging risk. Due to combinatory, this number can be very big.

The importance and criticality of risk-oriented design is manifested by including risk in the system model. As long as only system functionality is considered during the design of the system, with only tacit reference to risk mitigation, some project aspects may not be designed to mitigate design and development risks, or to support risk response during operation. Such mitigation might still be possible through unintended emergence, but it cannot be guaranteed. Risk-oriented design explicitly considers system components as sources and/or targets of risks. A risk-oriented designer consciously and deliberately yields configurable system components, with their possible states, in order to enable risk response, reduction, and mitigation during system design or operation. This mode of design reinforces standard risk identification and assessment processes with the capability to provide built-in response to these risks. Provisions for risk response are clearly marked, so that when needed, designers and users will be able to utilise them for risk response. This can replace attempts to reverse-engineer the system and figure out how tampering with its configurable aspects might provide some extent of risk response.

4 Case study: shoulder-missile defence

In this section, we demonstrate the use of ROSE for design and management of a shoulder missile defence system for commercial airliners. Over the last decade, the threat of shooting down a commercial airliner by various terrorist organisations with a simple and easily obtainable shoulder missile has dramatically increased. This threat has become even more severe since the beginning of the revolts and revolutions in the Arab States in 2011, as governments and military forces lost the reins, and fanatic groups attempted, and probably succeeded, to exploit the chaos and get hold of modern and strategic military technologies and weapon systems. The aerospace and defence industry has tackled this challenge of securing the safety and welfare of the passengers and crew on board by equipping airliners with means to avoid or neutralise such a threat, bearing in mind that the pilot is not necessarily well-trained for such an action.

We specify a conceptual-level system, based on real solutions and designs in the field of missile defence, with the capability to react on its own to identified threats, using one of several available countermeasures: a missile evasion manoeuvre; shoulder missile attack with some weapon, like a missile or a laser cannon; or shoulder missile mission disruption by decoy scattering or radio signal transmission. Obviously, not all of these countermeasures are available in each product, on-board each aircraft, or to each country purchasing such a system. The system supports pilot intervention and manual control, according to the level of hostility in the area, the availability of the different countermeasures on the aircraft, i.e., its countermeasure configuration, and the pilot discretion during an emergency, including the system neutralisation capability, in case of false alarm or threat infeasibility.

The **System Diagram** is illustrated in Figure 7 and in more detail in Figure 8. The primary process, **Shoulder Missile Aircraft Defense**, consists of three subprocesses: **Threat Identification**, **Reacting**, and **Reporting**.

Figure 7 Shoulder missile defence system – OPD system diagram (see online version for colours)

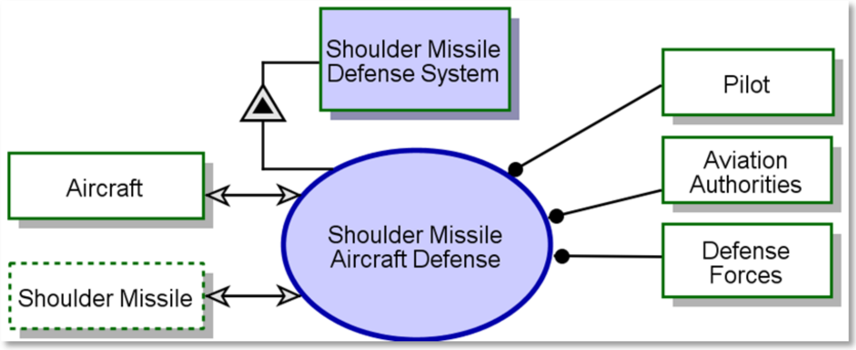
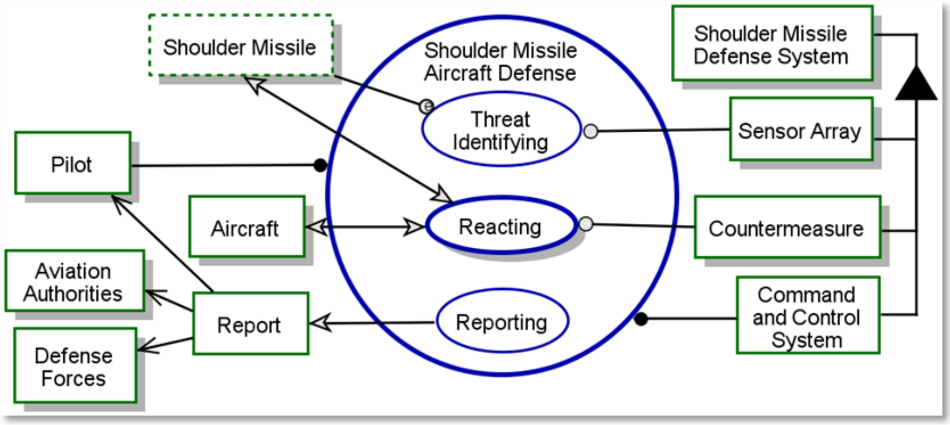

Shoulder Missile Defense System
Pilot
Aviation Authorities
Aircraft
Shoulder Missile Aircraft Defense
Defense Forces
Shoulder Missile

Figure 8 Shoulder missile aircraft defence in-zoomed – OPD (see online version for colours)



In this example, we focus on the **Reacting** process (OPD in Figure 9, OPL in Model spec 7), and on two risks: a **Shoulder Missile** may hit the **Aircraft**, and the **Shoulder Missile Defense System** may trigger **False Activation** due to false-positive identification of unthreatening objects. The system's support of manual operation is represented by the configurable **Pilot Authorisation Required**?. The system may fail to react appropriately if the pilot does not authorise reaction against a real threat, or does authorise reaction against a falsely identified threat.

Model spec 7 Shoulder missile aircraft defence/reacting in zoomed – OPL (see online version for colours)

Shoulder Missile Aircraft Defense/Reacting

Threat Identifying consists of Reaction Triggering.

Shoulder Missile can be threatening or unthreatening.

Threat Identifying yields threatening Shoulder Missile.

Reacting consists of Pilot Authorisation, Countermeasure Selection, and Countermeasure Activation.

Pilot Authorisation Required? can be yes or no.

Pilot Authorisation Required? triggers Pilot Authorisation when it enters yes.

Pilot provides Authorisation.

Authorisation can be positive or negative.

Reaction Triggering occurs if Pilot Authorisation Required? Is yes.

Reaction Triggering invokes Countermeasure Selection.

Countermeasure Selection occurs if Countermeasure is available.

Countermeasure Activation requires available Countermeasure.

Countermeasure Activation affects Aircraft.

Countermeasure Activation changes Shoulder Missile from threatening to unthreatening.

Aircraft Hitting is Risk Effect.

Shoulder Missile is a Risk Source.

Pilot Authorisation Required? is a Risk Source.

Authorisation is a Risk Source.

Aircraft Hitting may occur if Authorisation is negative and Pilot Authorisation Required? is yes.

Aircraft Hitting requires threatening Shoulder Missile.

Shoulder Missile may trigger Aircraft Hitting when threatening.

Aircraft Hitting consumes threatening Shoulder Missile.

False Activation is Risk Effect.

Countermeasure Activation may result inFalse Activation.

Pilot Authorisation Required may triggerFalse Activation when positive.

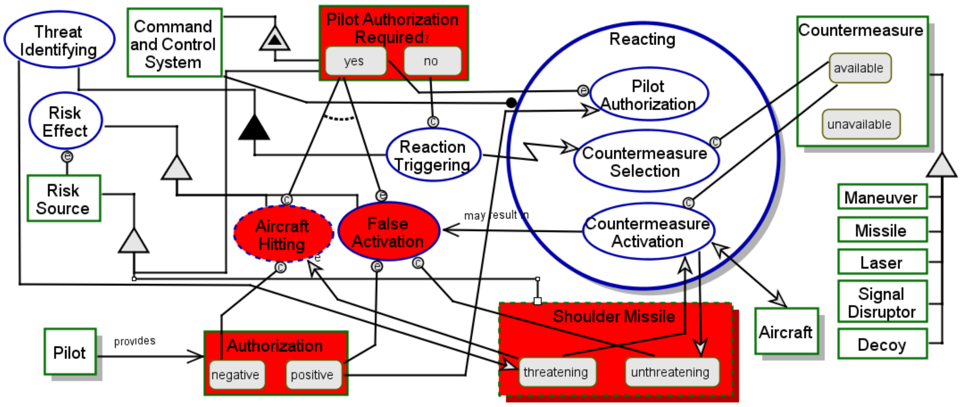
Authorisation may trigger False Activation when positive.

False Activation occurs if Shoulder Missile is unthreatening.

False Activation requires positive Pilot Authorisation Required and positive Authorisation.

The model described here is a design of a system as it is meant to react in an operational scenario. The model includes risk-related notions and provides solutions, which enable flexibility and resilience during the operational phase. In doing so, we follow the pattern of risk-oriented robust design, as we address both risk mitigation during the design and development phase – the project, and the facilitation of a risk response capability for the system’s operational phase.

Figure 9 Shoulder missile aircraft defence/reacting in zoomed – OPD (see online version for colours)



5 Summary

Systems are becoming highly integrated and interconnected, boundaries among business areas become fuzzy and fade out, system lifecycles are shortening, and version updates become more frequent. As these processes accelerate, it is vital to support the system's lifecycle with a suitable risk management methodology. Various risk scenarios arise during the design phase and coped with during the operational phase. Yet, the lack of appropriate alignment and coordination with design and configuration decisions makes it difficult to conduct thorough risk identification, analysis, mitigation, response, and monitoring. The main reason for this is the disparity and lack of integration between the system development (project) phase and risk management, as well as between system operation and risk management.

System engineers and risk analysts alike have been accustomed to practicing well-rooted methodologies, perceptions and working traditions that are at best loosely aligned. Bridging the gap between these two types of practitioners is mandatory in order for current and future increasingly complex and multidisciplinary systems to thrive in a risk-plagued world. To meet risk management challenges throughout the system's lifecycle, we have presented ROSE a new approach to the integration of risk modelling into the process of system design, configuration setting, and operational setup and activation, based on OPM, an emerging systems modelling and design framework.

We formed a metamodel that integrates robust design with risk mitigation and risk response, vis-à-vis the risks the system faces. This made it possible to link robust design decisions with project risk mitigation decisions and to embed operational risk response capabilities into the system during the its design phase. ROSE improves the ability of systems engineers, risk analysts, and configuration managers to capture, understand, and utilise robustness and resilience in their system when facing risks and threats of various types. ROSE bridges some major gaps in current MBRA methods:

1 ROSE uses the same single diagram type and model formalism in the underlying methodology (OPM), avoiding specialised risk diagrams, which preserve and underline the disparity between risk analysis and system analysis.

2 ROSE enables capturing trade-offs in risk mitigation and response decision-making using constraints, states, and dependencies among processes and effects.

3 ROSE captures the structure of risk elements, including the risk source and its risk-posing states, the risk effect, and the risk-incurred states of assets and objectives. Any systemic or environmental entity may constitute a risk source or a risk target.

4 ROSE captures both cause-and-effect sequences, and quantitative measures like impact, utility, and cost.

5 ROSE supports uncertainty-related properties of risk and system components; OPM's built in simulation mechanism utilises probabilistic simulation of events.

6 ROSE is suitable for any type of complex large socio-technical system and it supports both physical and informatical aspects.

Two goals guided this work. The first was to promote understanding and awareness of the importance of integrating system modelling and risk modelling by both systems analysts and risk analysts. Our second goal was to provide a methodology and means to model and understand risk management in general and in the context of design and configuration decision-making in particular. Guided by these goals, we derived objectives and developed ROSE to satisfy them and improve developers' capability to identify and mitigate system risks during both design and operation.

This paper is part of a broader research on conceptual modelling of systems with attention to enveloping, crosscutting aspects, such as risk management. In the wider scope of this research, we discuss risk management aspects during the transition from the project phase to the operation phase. Throughout the system lifecycle, the risk model is integrated into the system model and is synchronised with it; as the system model evolves, so does the risk model. Along this line of thought, we are currently researching lifecycle risk modelling and management, including theoretical aspects, conceptual modelling with OPM, risk management extensions to OPM, and implementation of our methodology for various real-life applications.

Acknowledgements

This research was supported by EU FP7 VISIONAIR Project 262044. We would like to thank the anonymous referees who reviewed the manuscript of this paper and provided constructive and useful advice.

References

Bedford, T. and Cooke, R. (2001) *Probabilistic Risk Analysis: Foundations and Methods*, Cambridge University Press, Cambridge UK.

Chapman, C. and Ward, S. (2003) *Project Risk Management: Processes, Techniques and Insights*, 2nd ed., John Wiley & Sons, Chichester, UK.

Cooke, R.M. (1991) *Experts in Uncertainty: Opinion and Subjective Probability in Science*, Oxford University Press, USA.

Diaz, C.a. (1998) 'Product re-configurability and product introduction', *Concurrent Engineering*, Vol. 6, No. 3, pp.172–177.

Dori, D. (2002) *Object-Process Methodology: A Holistic Systems Approach*, Springer-Verlag, Berlin, Heidelberg, New York.

Fredriksen, R. et al. (2002) 'The CORAS framework for a model-based risk management process', in Anderson, S., Felici, M. and Bologna, S. (Eds.): *Lecture Notes*, pp.94–105, Springer-Verlag.

Gaury, E.G.A. and Kleijnen, J.P.C. (1998) 'Risk analysis of robust system design', in *Simulation Conference Proceedings*, Winter, Vol. 2, pp.1533–1540.

Grunske, L. and Joyce, D. (2008) 'Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles', *Journal of Systems and Software*, Vol. 81, No. 8, pp.1327–1345.

Haimes, Y. (2008) 'Models for risk management of systems of systems', *Int. J. System of Systems Engineering*, Vol. 1, Nos. 1/2, pp.222–236.

Haimes, Y. (2009) *Risk Modeling, Assessment, and Management*, 3rd ed., John Wiley & Sons, Hoboken, New Jersey.

Haskins, C., Forsberg, K. and Krueger, M. (2007) *Systems Engineering Handbook*, v3.1 ed., INCOSE.

Hoffman, D. (2002) *Managing Operational Risk: 20 Firmwide Best Practice Strategies*, John Wiley & Sons, New York City.

ISO & IEC (2004) *ISO/IEC 16085: Information Technology – Software Life Cycle Processes – Risk Management*.

Kapelan, Z. et al. (2006) 'Risk- and robustness-based solutions to a multi-objective water distribution system rehabilitation problem under uncertainty', *Water Science and Technology*, Vol. 53, No. 1, pp.61–75.

Krishnan, V. and Ulrich, K.T. (2001) 'Product development decisions: a review of the literature', *Management Science*, Vol. 47, No. 1, pp.1–21.

Lund, M.S., Solhaug, B. and Stølen, K. (2011) *Model-Driven Risk Analysis: The CORAS Approach*, Springer, Berlin, Heidelberg.

NASA (2007) *Systems Engineering Handbook* [online] http://adsabs.harvard.edu/full/1995NASSP6105.....S (accessed 26 January 2013).

PMI (2000) *A Guide to the Project Management Body of Knowledge*, Project Management Institute (PMI), Newtown Square, Pennsylvania, USA.

Redmill, F., Chudleigh, M.F. and Catmur, J.R. (1997) 'Principles underlying a guideline for applying HAZOP to programmable electronic systems', *Reliability Engineering & System Safety*, Vol. 55.3, pp.283–293.

Sage, A.P. and Rouse, W.B. (Eds.) (2011) *Handbook of Systems Engineering and Management*, 2nd ed., John Wiley & Sons, New York.

Sharon, A. and Dori, D. (2009) 'A model-based approach for planning work breakdown structures of complex systems projects', in *Proc. 14th IFAC Symposium on Information Control Problems in Manufacturing*.

Sienou, A. et al. (2008) 'Towards a semi-formal modeling language supporting collaboration between risk and process manager', *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*, pp.119–125.

Stoneburner, G. (2002) *Risk Management Guide for Information Technology Systems*, National Institute of Standards and Technology, Washington DC, USA.

Strecker, S., Heise, D. and Frank, U. (2010) 'RiskM: a multi-perspective modeling method for IT risk assessment', *Information Systems Frontiers*, Vol. 13, No. 4, pp.595–611.

USAF (2009) *C-130 Hercules Fact Sheet*.

Appendix

OPM notation

Table A1 Entities □ ▢ ⬭ (see online version for colours)

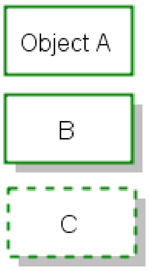
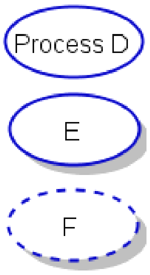
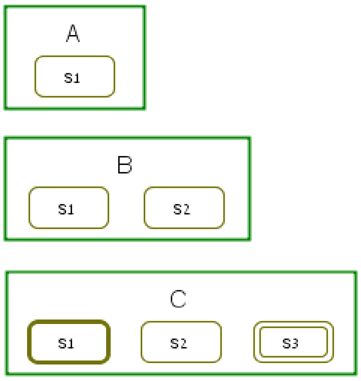
Name	*Symbol*	*OPL*	*Definition*
Object		B is physical. (shaded rectangle) C is physical and environmental (shaded dashed rectangle)	An object is a thing that exists.
Process		E is physical (shaded ellipse) F is physical and environmental (shaded dashed ellipse)	A process is a thing that transforms at least one object. Transformation is object generation or consumption, or effect – a change in the state of an object.
State		A is s1. B can be s1 or s2. C can be s1, s2, or s3. s1 is initial. s3 is final.	A state is situation an object can be at or a value it can assume. States are always within an object. States can be initial or final.

Table A2 Structural links and complexity management (see online version for colours)

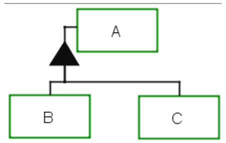
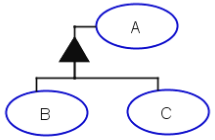
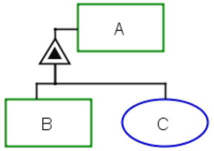
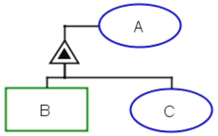
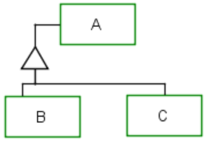
Name		*Symbol*	*OPL*	*Semantics*
Fundamental structural relations	Aggregation-participation		A consists of B and C.	A is the whole, B and C are parts.
			A consists of B and C.	
	Exhibition-characterisation		A exhibits B, as well as C.	Object B is an attribute of A and process C is its operation (method).
			A exhibits B, as well as C.	A can be an object or a process.
	Generalisation-specialisation		B is an A. C is an A.	A specialises into B and C. A, B, and C can be either all objects or all processes.

Table A2 Structural links and complexity management (continued) (see online version for colours)

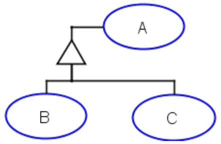
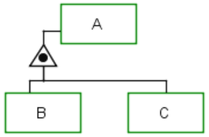
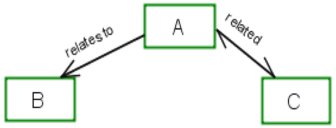
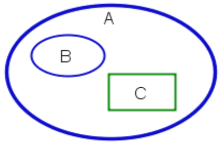
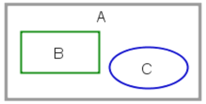
Name		*Symbol*	*OPL*	*Semantics*
Fundamental structural relations	Generalisation-specialisation		B is A. C is A.	
	Classification-instantiation		B is an instance of A. C is an instance of A.	Object A is the class, for which B and C are instances. Applicable to processes too.
Unidirectional and bidirectional tagged structural links			A relates to B. (for unidirectional) A and C are related. (for bidirectional)	A user-defined textual tag describes any structural relation between two objects or between two processes.
In-zooming			A exhibits C. A consists of B. A zooms into B, as well as C.	Zooming into process A, B is its part and C is its attribute.
			A exhibits C. A consists of B. A zooms into B, as well as C.	Zooming into object A, B is its part and C is its operation.

Table A3 Enabling and transforming procedural links (see online version for colours)

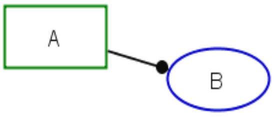
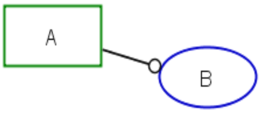
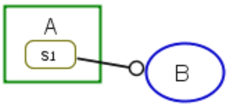
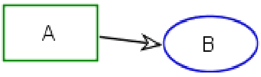
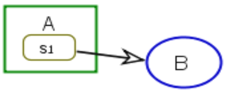
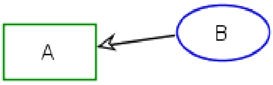
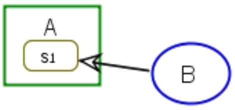
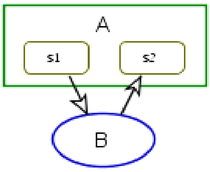
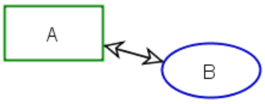
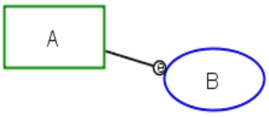
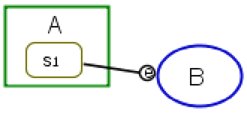
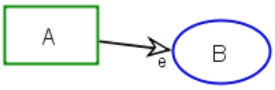
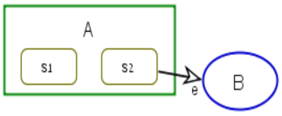
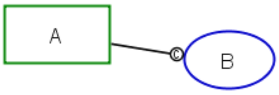
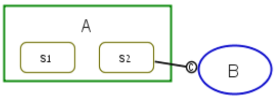
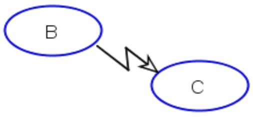
	Name	*Symbol*	*OPL*	*Semantics*
Enabling links	Agent link		A handles B.	Denotes that the object is a human operator.
	Instrument link		B requires A.	'Wait until' semantics: Process B cannot happen if object A does not exist.
	State-specified instrument link		B requires s1 A.	'Wait until' semantics: Process B cannot happen if object A is not at state s1.
Transforming links	Consumption link		B consumes A.	Process B consumes Object A.
	State-specified consumption link		B consumes s1A.	Process B consumes Object A when it is at State s1.
	Result link		B yields A.	Process B creates Object A.
	State-specified result link		B yields s1A.	Process B creates Object A at State s1.
	Input-output link pair		B changes A from s1 to s2.	Process B changes the state of Object A from State s1 to State s2.
	Effect link		B affects A.	Process B changes the state of Object A; the details of the effect may be added at a lower level.

Table A4 Event, condition, and invocation procedural links (see online version for colours)

Name	*Symbol*	*OPL*	*Semantics*
Instrument event link		A triggers B. B requires A.	Existence or generation of object A will attempt to trigger process B once. Execution will proceed if the triggering failed.
State-specified instrument event link		A triggers Bwhen it enters s1. B requires s1 A.	Entering state s1 will attempt to trigger the process once. Execution will proceed if the triggering failed.
Consumption event link		A triggers B. B consumes A.	Existence or generation of object A will attempt to trigger process B once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.
State-specified consumption event link		A triggers B when it enters s2. B consumes s2 A.	Entering state s2 will attempt to trigger the process once. If B is triggered, it will consume A. Execution will proceed if the triggering failed.
Condition link		B occurs if A exists.	Existence of object A is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
State-specified condition link		B occurs if A is s2.	Existence of object A at state s2 is a condition to the execution of B. If object A does not exist, then process B is skipped and regular system flow continues.
Invocation link		B invokes C.	Execution will proceed if the triggering failed (due to failure to fulfil one or more of the conditions in precondition set).