ODE Profile V2

White Paper

www.deis-project.eu

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 732242 (DEIS).
# Table of Contents

1 Executive Summary .................................................................................................................................................. 3

2 The ODE v2 Meta-Model......................................................................................................................................... 4

2.1 Structured Assurance Case Meta-Model (SACM) 2.0......................................................................................... 6

2.2 ODE v2 Product Metamodels ............................................................................................................................ 6

2.2.1 ODE::Base Package ....................................................................................................................................... 7

2.2.2 ODE::Design Package ................................................................................................................................... 8

2.2.3 ODE::Dependability Package ......................................................................................................................... 9

2.2.4 ODE::Dependability::Domain::Package, ODE::Dependability::HARA and ODE::Dependability::Requirements Package .......................................................................................................................... 10

2.2.5 ODE::FailureLogic Package .......................................................................................................................... 12

2.2.6 ODE::TARA Package ..................................................................................................................................... 14

3 Acquiring and Deploying ODE v2 .......................................................................................................................... 16

4 Summary and Outlook .............................................................................................................................................. 19
List of Figures

FIGURE 1 - OVERVIEW OF THE ODE v2 METAMODEL ................................................................. 5
FIGURE 2 - ODE::BASE PACKAGE .................................................................................................. 7
FIGURE 3 - ODE::DESIGN PACKAGE ............................................................................................. 8
FIGURE 4 - ODE::DEPENDABILITY PACKAGE ............................................................................... 9
FIGURE 5 - ODE::DEPENDABILITY::DOMAIN PACKAGE ............................................................... 10
FIGURE 6 - ODE::DEPENDABILITY::HARA PACKAGE ................................................................. 10
FIGURE 7 - ODE::DEPENDABILITY::REQUIREMENTS PACKAGE .................................................. 11
FIGURE 8 - ODE::FAILURELOGIC PACKAGE ............................................................................... 12
FIGURE 9 - ODE::FAILURELOGIC::FMEA PACKAGE .................................................................. 12
FIGURE 10 - ODE::FAILURELOGIC::FTA PACKAGE .................................................................. 13
FIGURE 11 - ODE::FAILURELOGIC::MARKOV PACKAGE .......................................................... 13
FIGURE 12 - ODE::TARA PACKAGE ............................................................................................. 14
FIGURE 13 - CREATING A NEW ECore MODELING PROJECT IN ECLIPSE ................................. 16
FIGURE 14 - ECLIPSE PROJECT CREATION WIZARD ............................................................ 17
FIGURE 15 - IMPORTING ECore FILES INTO PROJECT............................................................. 17
FIGURE 16 - ADDING METAMODEL ELEMENTS TO DIAGRAM ..................................................... 18
FIGURE 17 - RESULTING DIAGRAM ............................................................................................ 18

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Long Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDI</td>
<td>Digital Dependability Identity</td>
</tr>
<tr>
<td>EMF</td>
<td>Eclipse Modeling Framework</td>
</tr>
<tr>
<td>FMEA</td>
<td>Failure Mode and Effect Analysis</td>
</tr>
<tr>
<td>FTA</td>
<td>Fault Tree Analysis</td>
</tr>
<tr>
<td>GSN</td>
<td>Goal Structuring Notation</td>
</tr>
<tr>
<td>ODE</td>
<td>Open Dependability Exchange Metamodel</td>
</tr>
<tr>
<td>SACM</td>
<td>Structured Assurance Case Metamodel</td>
</tr>
</tbody>
</table>

Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>E-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yiannis Papadopoulos</td>
<td>University of Hull</td>
<td><a href="mailto:Y.I.Papadopoulos@hull.ac.uk">Y.I.Papadopoulos@hull.ac.uk</a></td>
</tr>
<tr>
<td>Ioannis Sorokos</td>
<td>University of Hull</td>
<td><a href="mailto:i.sorokos@hull.ac.uk">i.sorokos@hull.ac.uk</a></td>
</tr>
<tr>
<td>Jan Reich</td>
<td>Fraunhofer IESE</td>
<td><a href="mailto:Jan.reich@iese.fraunhofer.de">Jan.reich@iese.fraunhofer.de</a></td>
</tr>
</tbody>
</table>
1 Executive Summary

This document provides an overview of the implementation of the second version of the Open Dependability Exchange (ODE) meta-model, which is the meta-model for the DEIS core concept of Digital Dependability Identities (DDI). In D3.2, an overview of the changes to the ODE were provided already. This document serves as an in-depth guide to the ODE v2 metamodel, as well as a user’s guide to the metamodel. The guide should serve as a useful starting point for interested tool providers to connect with the ODE.

The document structure is as follows; first, an overview of the ODE v2 is provided, followed by a brief reminder on the SACM metamodel. Next, changes to each package of the ODE v2 are provided and their motivation discussed. Finally, the document concludes with a short guide on acquiring a copy of the ODE v2 metamodel from the DEIS online repository and using recommended modeling tools to create models compliant to it.
2 The ODE v2 Meta-Model

The ODE is the meta-model for the DEIS core concept of the DDI and thus a central artifact of the project. It needs to be fit for the DEIS core challenges and objectives as described in the proposal and refined in WP2, WP5 and WP6. The general concept of the ODE v2 as well as its refinement based on engineering stories is presented in deliverable 3.2.

In the second version, the relationship between the ODE and the Structured Assurance Case Metamodel (SACM) was modified to address semantical concerns. The ODE is now clearly distinguished in its role as a collection of ‘Product’ metamodels. In contrast, the SACM provides a generic, higher-level structure for encapsulating assurance claims and argumentation and can reference other models, including those derived from Product metamodels such as the ODE v2.

Figure 1 presents an overview of both the SACM (highlighted in purple) and the ODE v2 (highlighted in green). The overview indicates that while there has been some reduction and simplification, the ODE (and SACM) remain quite complex metamodels, spanning across a plethora of metamodeling elements.
Figure 1 - Overview of the ODE v2 Metamodel
2.1 Structured Assurance Case Meta-Model (SACM) 2.0

The Structured Assurance Case Meta-Model (SACM) is a modelling language specified by the Object Management Group (OMG) to create model-based assurance cases. SACM supports existing system assurance case approaches such as the Goal Structuring Notation (GSN) and Claims-Arguments-Evidence (CAE).

An assurance case is a set of auditable claims, arguments and evidence created to support the claim that a defined system/service will satisfy typical requirements such as safety and/or security. An assurance case in this context is a machine-readable model that facilitates the exchange of information between various systems stakeholders such as suppliers and integrators, and between the operator and regulator, with the knowledge related to the safety and security of the system being communicated in a clear and defendable way. Each assurance case should communicate the scope of the system, the operational context and the safety and/or security arguments, along with the corresponding evidence.

SACM is a complex metamodel defined by the specifiers of existing system assurance case approaches (i.e. GSN and CAE), based on the collective knowledge and experiences of safety and/or security practitioners over the period of 20 years. SACM contains more features than GSN and CAE and is therefore more powerful in terms of expressiveness. The full specification of SACM can be found at https://www.omg.org/spec/SACM.

A paper explaining the usage of SACM via examples has also been submitted to the Journal of Systems and Software, entitled “Model Based System Assurance Using the Structured Assurance Case Metamodel”. Further details on SACM have already been provided in the D3.2 and D4.1 deliverables. Moreover, there have been no changes made to SACM for the ODE v2 profile, so the reader is referred to the above documents for more information.

2.2 ODE v2 Product Metamodels

During the transition to ODE v2, the ODE::Integration package was removed from the design. It was determined that, given the integration capabilities already established in SACM, the ODE::Integration package was redundant.

Detailed changes to each of the ODE v2 packages now follow, beginning with the Base package.
2.2.1 ODE::Base Package

Figure 2 presents the current state of the ODE::Base package. The main change is the removal of the ‘isCitation’ and ‘isAbstract’ attributes of the BaseElement. The reason for this change is the recognition that SACM already provides mechanisms for modeling assurance artifact citations and abstractions, so the above attributes became redundant in the ODE. Additionally, the ‘citedElement’ association from BaseElement onto itself was also removed. As a corollary to the removal of the two attributes, all PackageInterface and PackageBinding elements have been removed from all other ODE product packages.

The other change to the package is the inclusion of the commonly used enumeration TimeUnit. Given its use in other packages, it was moved here to be available to all other packages.

Besides the above changes, the package is otherwise unchanged.
2.2.2 ODE::Design Package

The ODE::Architecture package has undergone some simplification. Its updated state can be seen in Figure 3. To begin with, it has been renamed to 'Design' and its internal containment element 'DesignPackage' for etymological reasons.

The System and Function elements’ role has been enhanced, as all other elements within the package compose onto or inherit from them. This change shifts the focus onto the structure of the system architecture as opposed to its properties. The change is also motivated by the desire to reduce inheritance dependencies, which introduced unnecessary complexity. The removal of the common supertype from within the package led to the distribution of the DependabilityRequirement and AssuranceLevel associations across System and Function.

Another significant change is the association of ODE::FailureLogic::FailureModels with Systems and Functions, whereas the more specific ODE::FailureLogic::Failures associate with individual Ports. This change clarifies and makes more precise the relationship between the functional/systemic failure analysis and individual failure behavior at the level of function/system/component interfaces.

Another notable addition is the association of Systems and Functions optionally as Dependability::TARA::Assets. This enables both former elements to be included in a security risk analysis. See TARA package below for more information.
Finally, we should note that in Figure 3 and subsequent metamodel figures, elements internal to the package being discussed, in this case the ODE::Design package, are depicted in yellow color. Elements from other packages, e.g. Dependability::TARA::Asset in Figure 3, are depicted in blue color.

2.2.3 ODE::Dependability Package

The Dependability package, seen in Figure 4, has undergone some restructuring, although most major elements and concepts remain unchanged. Under the new structure, the DependabilityPackage containment element now directly contains containment elements from other subpackages of Dependability, as well as Measures, MeasureTypes and MaintenanceProcedures. This change reduces some complexity from the previous scheme, which required foreign containment elements to be composed under inheritance from a common supertype element.

In the new version, subpackage containment is direct and further extension of the Dependability package is simpler as well, as seen by the inclusion of the new Dependability::TARA::TARAPackage for security risk analysis.
2.2.4 ODE::Dependability::Domain::Package, ODE::Dependability::HARA and ODE::Dependability::Requirements Package

Figure 5 - ODE::Dependability::Domain Package

Figure 6 - ODE::Dependability::HARA Package
The Domain, Hazard and Risk Analysis (HARA) and Requirements subpackages of ODE::Dependability remain largely the same, with minor restructuring changes, seen in Figure 5, Figure 6 and Figure 7 respectively. In all subpackages, common supertype elements were removed, distributing their attributes and associations across their inheritors.

A new SecurityRequirement element now extends the ODE with capabilities for modeling requirements originating from the security domain (from standards and system development). Each SecurityRequirement can be associated with a number of Dependability::TARA::SecurityCapabilities, see TARA package below for more information.

The main change in the HARA subpackage is the association of a Hazard with zero or more ODE::FailureLogic::Failures, as opposed to the previous version’s limit of one. This change simplifies the modeling of Hazards which can be caused by a combination of failures. In the previous version, in such situations, an intermediate ODE::FailureLogic::OutputFailure caused by a combination of the other failures was required to delegate failure propagation to the Hazard.
2.2.5 ODE::FailureLogic Package

Figure 8 - ODE::FailureLogic Package

Figure 9 - ODE::FailureLogic::FMEA Package
The ODE::FailureLogic package has undergone significant restructuring and simplification. Figure 8, Figure 9, Figure 10 and Figure 11 present the current state of the package and its subpackages. Of particular note is the reduction of the separate Failure subtypes (InternalFailure, InterfaceFailure, InputFailure, OutputFailure) into a single type. The single Failure type maintains the semantics captured by the previous subtypes within its properties. For instance, the Failure::originType describes whether the failure is internal within a function/system/component or at the interface level.

Similarly, the FailureMode and CCF (Common Cause Failure) elements have been absorbed within the Failure type and represented with its class, isCCF and ccfFailures properties. The latter two denote whether the Failure represents a CCF and which are the other Failures that can be caused by the CCF.
The element MinimalCutSets now accurately describes the synonymous concept without the need to employ elements from the ODE::FTA package. In the previous version, complex combinations of minimal cut sets needed the ODE::FailureLogic::FTA::Gate element to be represented.

The SecurityViolation element is a new addition to the FailureLogic package. By inheriting from Failure, it enables modeling the direct effect a security ODE::TARA::Attack has on the system (see TARA package below).

The ODE::FTA package has been simplified by reducing the various types of events represented in a fault tree down to a generic Cause element. Boolean logic event connectors are represented by the Gate element, which, as a subtype of Cause, can be used to chain hierarchies of Causes together into a fault tree.

The ODE::FMEA package has undergone some simplification as well. Propagation is now represented with the association of each FMEAEntry with a ‘mode’ and ‘effect’ Failure. This change contrasts the previous version’s propagation modeling with explicit elements such as FMEAPropagation and DiagnosableFailurePropagation, which were deemed redundant. The specialization of the Failure type for FMEAs was also removed due to redundancy.

2.2.6 ODE::TARA Package

Figure 12 - ODE::TARA Package
The TARA package models the results of a Threat And Risk Analysis (TARA) for security, seen in Figure 12. ThreatAgents are either Human or NonHuman (typically electronic) sources of Attacks. While individual Attacks may serve many purposes, ThreatAgents will also feature some higher-level AttackerGoal, representing the overall goal of the attacker. The AttackerGoal revolves around negatively impacting the Assets being considered for security, often being the system’s operation and its data for example. Attacks exploit Vulnerabilities of the system.

Cumulatively, the above elements contribute towards the SecurityRisk posed by the various threats identified during the TARA. To combat these threats and reduce risk, SecurityCapabilities and SecurityControls are established. Respectively, these are high-level and low-level security safeguards/counter-measures.

SecurityCapabilities are directly associated with ODE::Dependability::Requirements::SecurityRequirements. SecurityControls are instead directly associated with ODE::Dependability::Measures. After applying these measures, risk is reduced accordingly.

The risk analysis can consider many factors surrounding each of the principal elements mentioned above. For instance, each ThreatAgent can include its assumed skill level, abstract motivation and desire for anonymity; each AttackerGoal, Attack, Vulnerability, SecurityCapability and SecurityControl can be categorized according to the types of security violation / protection they are relevant; each Attack, Asset and SecurityRisk can model the impact of a security attack from both the attacker’s and the asset’s perspective.

Finally, more detailed analysis of the propagation of effects of Attacks on the system are modeled by linking individual Attacks with ODE::FailureLogic::SecurityViolations. This link further enables hybrid security-safety analysis, as complex ODE::FTA::Causes can also be associated with ODE::FailureLogic::Failures from non-security dependability analysis.
3 Acquiring and Deploying ODE v2

In this section, a short guide for downloading and deploying the ODE v2 using recommended tools is provided. The user begins by downloading the metamodel definition files from the DEIS software repository: https://github.com/DEIS-Project-EU/ODEv2.

The metamodel is defined in the Eclipse Modeling Framework (EMF) Ecore format. The format is based on the Extensible Markup Language (XML) and is human-readable. Thus, it can be reviewed using any common text editor. However, for more effective review and use for modeling, the Eclipse platform using the EMF is recommended.

The latest version is available at: http://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-modeling-tools. Installation according to instructions appropriate for the user’s workstation configuration should be followed.

Once the Eclipse modeling environment is correctly installed and the ODE v2 Ecore files acquired, the user can review them by creating a new EMF project in Eclipse, as per Figure 13.

![Eclipse Workspace - Modeling - Eclipse](image)

**Figure 13 - Creating a new Ecore Modeling Project in Eclipse**
Once the project has been created using mostly default settings as per Figure 14, the user can import the metamodel using the corresponding function, seen in Figure 15.

**Figure 14 - Eclipse Project Creation Wizard**

**Figure 15 - Importing Ecore files into project**
Following this step, the user can now freely review and create new diagrams using the metamodel elements, as per Figure 16.

![Figure 16 - Adding metamodel elements to diagram](image)

In Figure 17, the result of adding the ‘Function’ element to the diagram is shown.

![Figure 17 - Resulting diagram](image)

The EMF offers more options for automatically generating code and editors for ODE v2 models; further information can be found at: [http://help.eclipse.org/2018-09/topic/org.eclipse.emf.doc/references/overview/EMF.html?cp=34_0_0](http://help.eclipse.org/2018-09/topic/org.eclipse.emf.doc/references/overview/EMF.html?cp=34_0_0).
4 Summary and Outlook

In this document, the current status of the development of the ODE metamodel was presented. First, the role of the SACM in the updated version was briefly explained. A detailed view of the changes accumulated over the transition from the first version in D4.1 were then discussed. The document concluded with a brief practical guide on deploying the ODE v2 for use with the recommended tools on a local workstation.