Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2003-570

SEST: SE Specification Tool-set

A.J.J. Lemmers, M.F.R. Keuning and M. Jokipii*

* Instrumentointi Oy

This report is based on a presentation held on Fall Simulation Interoperability Workshop, at Orlando, USA on 15-19 September 2003.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	V.1.D.2
Owner:	National Aerospace Laboratory NLR
Division:	Flight
Distribution:	Unlimited
Classification title:	Unclassified
	November 2003

Approved by author:	Approved by project manager:	Approved by project managing department:
		2/12

Summary

EUCLID RTP 11.13 is a major European initiative to promote the use of Synthetic Environments. The title of the project ‘Realising the Potential of Networked Simulations in Europe’ reflects the fact that although Synthetic Environments are currently being used to support defence programmes in Europe, their full potential is not being realised. Part of the EUCLID RTP 11.13 approach to resolving obstacles is the definition of methodologies and the implementation of tools for federation conceptual modelling and federation system requirements definition.

Contents

(19 pages in total)

1 Introduction

EUCLID RTP 11.13 is a major European initiative to promote the use of Synthetic Environments (SEs). The title of the programme 'Realising the Potential of Networked Simulations in Europe' reflects the fact that although SEs are currently being used to support defence programmes in Europe, their full potential is not yet being realised. The aim of the project is to 'overcome the obstacles that prevent SEs being exploited in Europe by developing a process and an integrated set of prototype tools that will reduce the cost and timescale of specifying, creating and utilising SEs for collective training, mission rehearsal and simulation based acquisition'.

In order to promote the use of distributed simulation exercises across Europe, it is important to realise a common European process with common tools and standards. The Federation Development and Execution Process (FEDEP) has been used as a baseline for this process and has been modified and extended whereever required. The resulting Synthetic Environment Development & Exploitation Process (SEDEP) is used as the steppingstone within the programme, where each of the 8 steps of the process is researched and demonstrated by example and prototype tooling. Figure 1 gives an overview of SEDEP vs. FEDEP, in Ref. [2] an detailed explanation of the SEDEP is given.

Step 2 of the SEDEP, "Define Federation System Requirements", has the purpose to specify the requirements of a system to provide an appropriate representation of the mission and simulation space that applies to the Federation problem space. It is also in this step that Federation user requirements are transformed into a set of highly specific Federation system requirements that will be used as success criteria during Federation testing. At the end of step 2 a complete specification of SE is available.

The Federation Conceptual Model (FCM) is the centrepiece of the SE specification and is supported by additional federation and evaluation system requirements. The Unified Modelling Language (UML) is used to shape the FCM. Stereotypes add specific semantics to the standard UML specification. Structure and well-formedness rules ensure a common approach, structure, and notation that is consistent across multiple developments. Additional federation and evaluation requirements are organised in an open flexible format that ensures commonality in structure and notation.

Although the results of RTP 11.13 will be demonstrated using prototype tools, it is the explicit goal to be tool-vendor independent. The programme realises this through the definition of open eXtensible Mark-up Language (XML) based data interchange formats. The standard XML

Metadata Interchange (XMI) for UML format is used to store and interchange an FCM. For the additional federation and evaluation requirements a new open format has been defined. This paper describes:

- An overview of the SEDEP process and how the SE specification fits in,
- The process to create an SE specification,
- Conceptual Modelling using the Unified Modelling Language (UML),
- Definition of additional federation and evaluation requirements,
- Design of the SE Specification Tool-set,
- Implementation of prototype SEST using the COTS tools DOORS and Rational Rose.

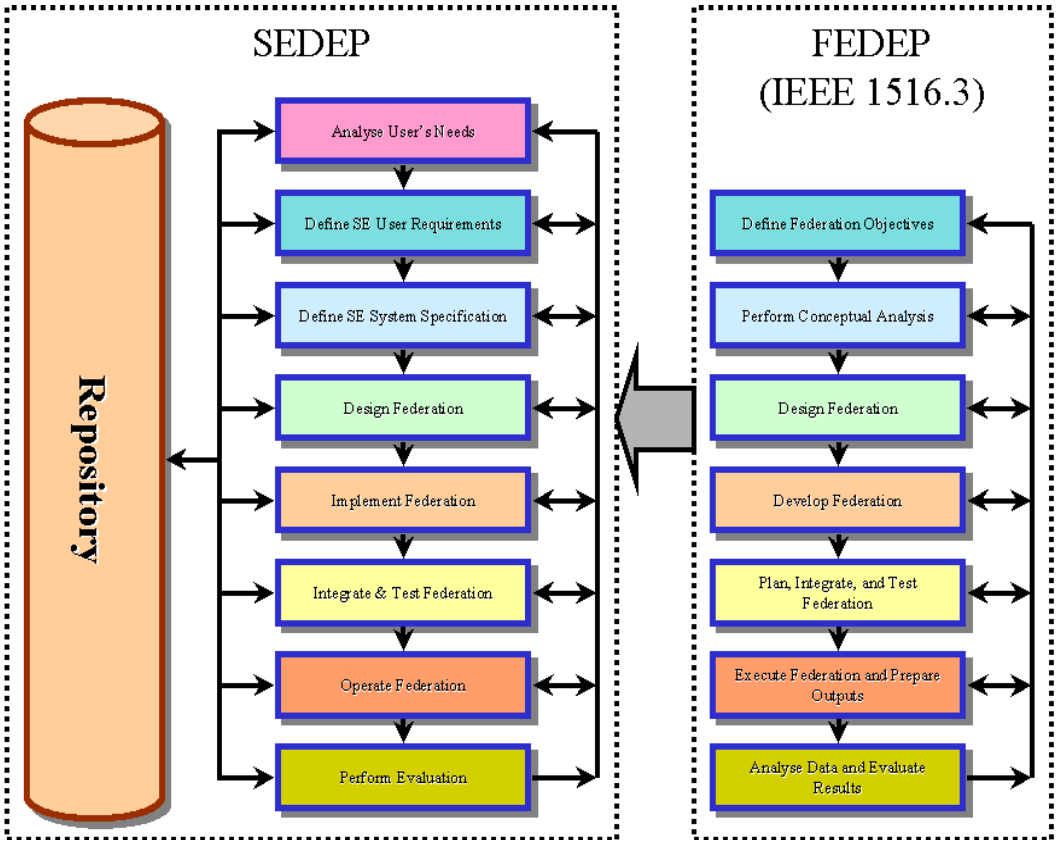

Figure 1: SEDEP vs. FEDEP

2 SEDEP Overview

The development and exploitation of Synthetic Environments requires the definition of a process to:

- Provide support to encourage the use of SE Technology on military programmes;
- Provide guidance for SE developers and users to plan and perform the different activities necessary to produce the required products and results;
- Promote good practice for developing SEs on time and within budget;
- Promote reuse of products (federation, federates, components) and results;
- Provide a framework for a tool set to reduce the cost and time for producing and using SE.

In EUCLID 11.13 it was decided that rather than developing a new SE process from scratch, to use the Federation Development and Execution Process (FEDEP) as enhanced and extended where necessary. This is because the FEDEP has already been widely adopted by the SE community and is already supported by several COTS tools.

The resulting EUCLID 11.13 process is known as the Synthetic Environment Development & Exploitation Process (SEDEP). The use of the term SEDEP has been chosen to reinforce its close links with the FEDEP whilst promoting its more general use for developing SEs by dropping its association with the High Level Architecture (HLA). It should be noted that the current steps of the FEDEP exist as a sub-set of the SEDEP. It is intended that long term, the two processes will merge and that there will only be a single SE process. Figure 1 illustrates how the steps of both processes are related in the most current incarnations of both processes. For the FEDEP the latest draft recommended practice Ref. [3] is used for reference, this is expected to become an IEEE recommended practice (IEEE 1516.3).

3 Step 2: Define SE Specification

Step 2 of the SEDEP is concerned with the definition of the SE Specification (SES). The SES exists of two complementary sets of specifications: Federation Conceptual Model (FCM) and the Federation System Requirements (FSR). The SES is derived from the information obtained from SEDEP step 1, i.e. Scenario User Requirements, Constraints User Requirements, and Evaluation User Requirements. Tool support is essential to produce easily maintainable and consistent models and requirements. To support the definition of the SES, two tools are identified: Conceptual Model Tool (CMT) and Federation Requirements Tool (FRT). The CMT pertains to the definition of the FCM and the FRT pertains to the definition of the FSR. The complete SES serves as input to SEDEP step 3. *Figure 2* gives an overview of SEDEP step 2 activities and information resources.

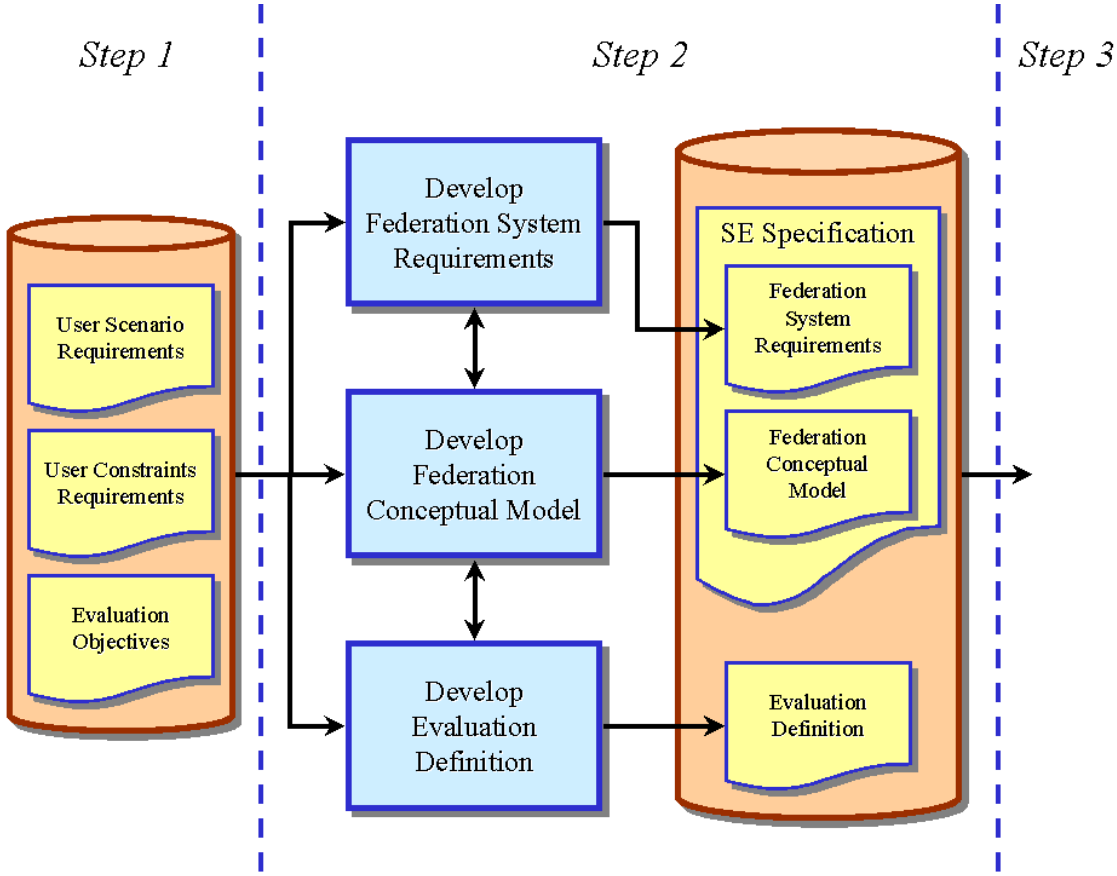

Figure 2: SEDEP step 2

All information produced by the tools will be stored in the RTP11.13 repository. Section 9 describes how information resources are linked together to provide data for verification and consistency checking.

During the Conceptual Model development, the problem solver produces a conceptual representation of the intended problem space based on his interpretation of the constraints user requirements and the scenario user requirements. The product resulting from this activity is known as a Federation Conceptual Model. The Federation Conceptual Model is a description of the SE elements and relations that need to be included in the SE in order to achieve all mission and simulation space objectives. These SE elements and relations are described without any reference to the specific simulations and tools that will be used in the federation.The Federation Conceptual Model provides an implementation-independent representation that serves as a vehicle for transforming objectives into functional and behavioural capabilities; the model also provides a crucial traceability link between the constraints user requirements and scenario user requirements, and the design and implementation. This model can be used as the structural basis for many federation design and development activities (including scenario development) and can highlight correctable problems early in the federation development process when properly validated.

Static relationships can be expressed as ordinary associations or as more specific types of associations such as generalisations (“is-a” relationships) or aggregations (“part-whole” relationships). Dynamic relationships should include (if appropriate) the specification of temporally ordered sequences of object interactions with associated trigger conditions. Object characteristics (attributes) and interaction descriptors (parameters) may also be identified to the extent possible at this early stage of design.

Once the Federation Conceptual Model is completed, it needs to be carefully evaluated before the next step (Federation Design) can start. The evaluation should include a review of key processes and should ensure the adequacy of the domain representation. Revisions to the original federation user requirements and federation scenario may be defined and implemented as a result of feedback.

Together with the Federation Conceptual Model, the additional Federation System Requirements have to be elaborated. These requirements, based on the original user requirements (step 1), should provide the level of guidance needed to design and develop the federation. The Federation System Requirements should also explicitly address the issues of fidelity and evaluation, so that these requirements can be considered during selection of federation participants. This needs close co-operation with the Conceptual Model development activity. There will be many references between the Federation System Requirements and the items of the Federation Conceptual Model. In addition, any programmatic or technical constraints on the federation should be refined and described to the degree of detail necessary to guide SE development.

4 Conceptual Model Format

The profiling of UML for purpose of constructing Conceptual Models for Synthetic Environments exists of two customisations:

- Specific stereotypes.
- Rules about well-formedness of a Conceptual Model.

The following stereotypes are defined specifically for SE Conceptual Models.

Name	**Base Class**
SEelement	(Abstract) Class
SEdata	Class
SErelation	Package

SEelement: A class with the <<SEelement>> stereotype represents an element that participates in the specified SE. <<SEelement>> classes have the capability to communicate <<SEdata>> with other <<SEelement>> classes. <<SEelement>> classes may be composed of other <<SEelement>> classes. An <<SEelement>> class may be abstract, this will be indicated by specifying that the class is an abstract class. An abstract <<SEelement>> class cannot be instantiated (as is common semantics for an abstract class), but specifies common properties of <<SEelement>> classes that inherit from it.

SEdata: A class with the <<SEdata>> stereotype represents a data type that can be exchanged by <<SEelement>> classes. An <<SEdata>> data type may be a complex data type.

SErelation: A package with the <<SErelation>> stereotype contains a specification of a relation between <<SEelement>> classes. Each <<SErelation>> package specifies the exchange of one kind of <<SEdata>> data type. An <<SErelation>> package must provide three model elements that together specify a relation using a so-called push-model Observer pattern (a.k.a. Publish/Subscribe pattern):

1. One non-abstract class with the <<SEdata>> stereotype.
2. One abstract class that inherits from the abstract class SEdataProvider. This class represents a specific data provider for the <<SEdata>> data type specified by this <<SErelation>> package. This class must aggregate the <<SEdata>> data type with a UML composition association.
3. One abstract class that inherits from the abstract class SEdataConsumer. This class represents a specific data consumer for the <<SEdata>> data type specified by this <<SErelation>> package. This class must have a UML dependency association to the specific SEdataProvider class.

As to the well-formedness rules, an SE Conceptual Model is separated into three aspects of concern. This separation of concern is realised by grouping model aspects together, it does not change the actual model. The three aspects of concern are listed in the following table:

SE Elements	These are the elements that participate in the SE. These elements have the capability to communicate with other SE Elements and may be composed of other SE Elements.
SE Relations	For each type of relation between SE Elements a SE Relation Model is defined. A standard framework for defining these models ensures that in each CM these relations are modelled in the same way.
SE Auxiliary Data Models	This is an optional handhold for storing auxiliary data models that support the definition of the CM. For example the SEDRIS Spatial Reference Model might be included.

Besides being involved in <<SErelation>> specifications, <<SEelement>> classes can have two types of associations with each other. First, generalisation associations facilitate modelling of commonalities by means of an abstraction hierarchy. Second, aggregation associations facilitate composition of SE elements into more complex SE elements.
Abstractions may be modelled with or without the use of sub-packaging. Sub-packaging is encouraged to maintain complexity to a reasonable level and to enhance reusability. Sub-packaging does not change semantics of the model it merely improves structure. When a sub-package is defined for an abstraction, these packages should provide the following:

- An <<SEelement>> class with the same name as the package. In most cases this class will be an abstract <<SEelement>> class, this is however not a requirement.
- All <<SEelement>> classes that inherit from the abstraction class. They may of course be contained in further abstraction sub-packages.
- A class diagram with the name “Main”, illustrating the newly created level of abstraction.

With respect to modelling composition of <<SEelement>> classes the only requirement is that the composite class cannot be abstract. The parts of a composition may be abstract if and only if there exists at least one non-abstract <<SEelement>> class that inherits from it. Composition may be illustrated on the “Main” class diagram of the package that contains the composite class. For complex compositions it is encouraged to use separate diagrams. Note that a diagram is a view on the actual model and as such does not add semantics to the model. An example is shown in Figure 3.

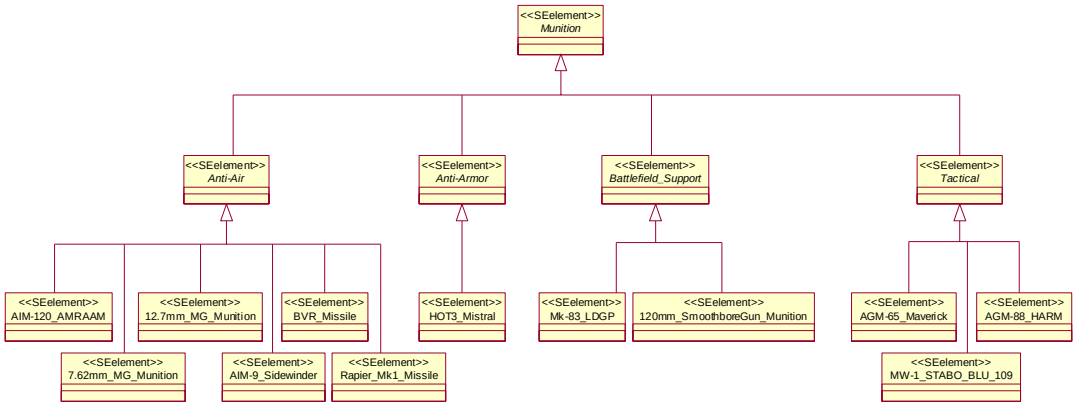

Figure 3: SE Element Decomposition Example

No abstraction or composition structures are prescribed. It is foreseen that template structures can be easily reused from previous SE developments. Such template can be customised to the particular needs of the SE under development.

An <<SErelation>> specification specifies for a particular data type which <<SEelement>> classes produce data of this type and which <<SEelement>> classes consume data of this type. The <<SErelation>> specifications are modelled under the aspect “SE Relations”. The following rules apply:

- Each <<SErelation>> specification specifies the exchange of exactly one <<SEdata>> data type between <<SEelement>> classes.
  - As such each <<SErelation>> package contains exactly one <<SEdata>> data type.
  - And each <<SEdata>> data type must be a specialisation of “<<SEdata>> SEdata”.
- Each <<SErelation>> contains exactly one provider class that must be a specialisation of the class “SEdataProvider”.
  - This provider class aggregates the <<SEdata>> data type of this <<SErelation>> specification.
  - And all <<SEelements>> that can provide data of the <<SEdata>> data type of this <<SErelation>> specification inherit from this provider class.
- Each <<SErelation>> contains exactly one consumer class that must be a specialisation of the class “SEdataConsumer”.
  - This consumer class has a dependency association to the provider class of this <<SErelation>> specification.
  - And all <<SEelements>> that consume data of the <<SEdata>> data type of this <<SErelations>> specification inherit from this consumer class.

– Each <<SErelation>> specification contains a class diagram with the name Main. This diagram illustrates the specification according to the rules as stated above. An example is shown in *Figure 4*.

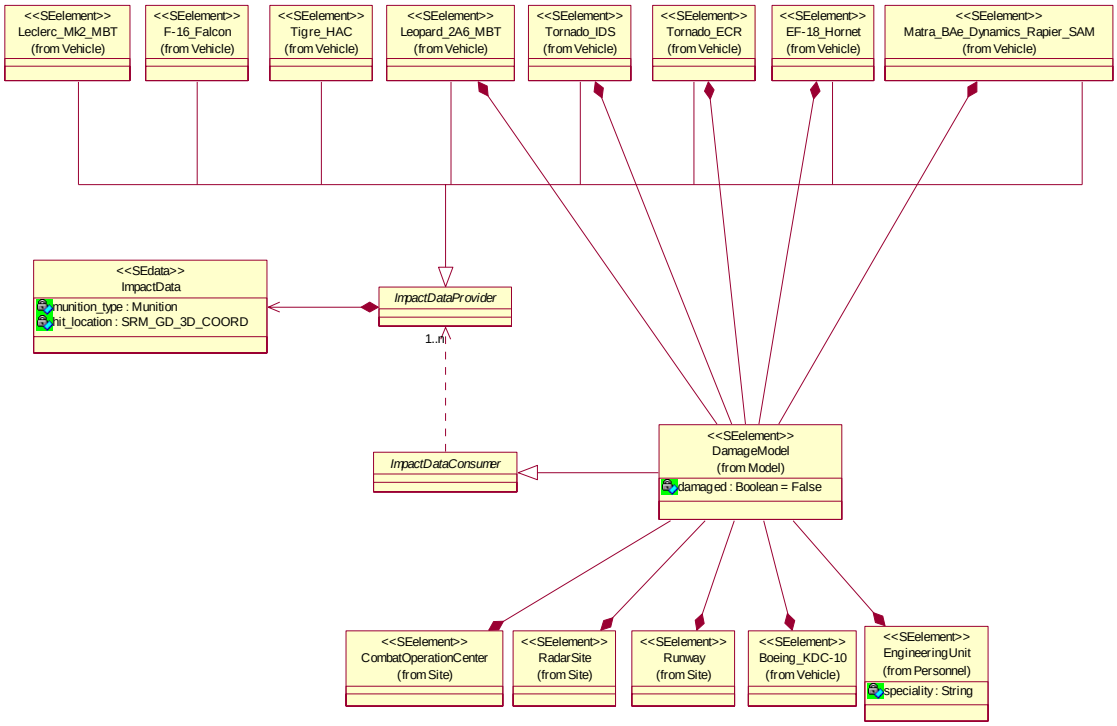

Figure 4: SE Relation Example

5 Definition of Federation System Requirements

At present a variety of different languages are being used in specifying requirements for software systems. The characteristics of the used languages vary largely from strictly formal languages (e.g. Object Constraint Language, OCL) to informal (i.e. natural language, e.g. English).

No explicit language is prescribed for Federation System Requirements (FSR) definition as there is currently no single language that is capable of expressing every imaginable system requirement. Thus the problem solver is allowed to apply the specification language that is the most suitable. Different languages may be used for different requirements, however the applied language must be identified as part of the requirement specification.

Although no strict rules or languages are defined to use for FSR definition, a general framework has been designed to guide the problem solver into defining a more exact and complete set of system requirements. The problem solver is enforced to define requirement attributes, like e.g. traceability, testability, language, etc., which are considered essential not only for internal consistency, completeness and exactness, but also to support activities like verification, validation and change management (see *Figure 5*).

Also the notion of flexible requirement category templates has been introduced. These templates represent hierarchies of requirement categories that guide the problem solver to consider all aspects in requirements analysis. As different applications require different categories, e.g. for one federation development safety might be an issue whereas for another it might not be applicable, these templates can be custom defined (or derived from existing templates).
From formal languages the RTP 11.13 prototype implementation of the requirements tool has support for MathML. MathML is a generic format for defining mathematical expressions and it is based on XML. MathML itself is not easy to read and thus it needs additional functionality for visualising and editing the mathematical expressions. MathML is needed e.g. in defining algorithms for to describe dynamic behavior of some simulation element.

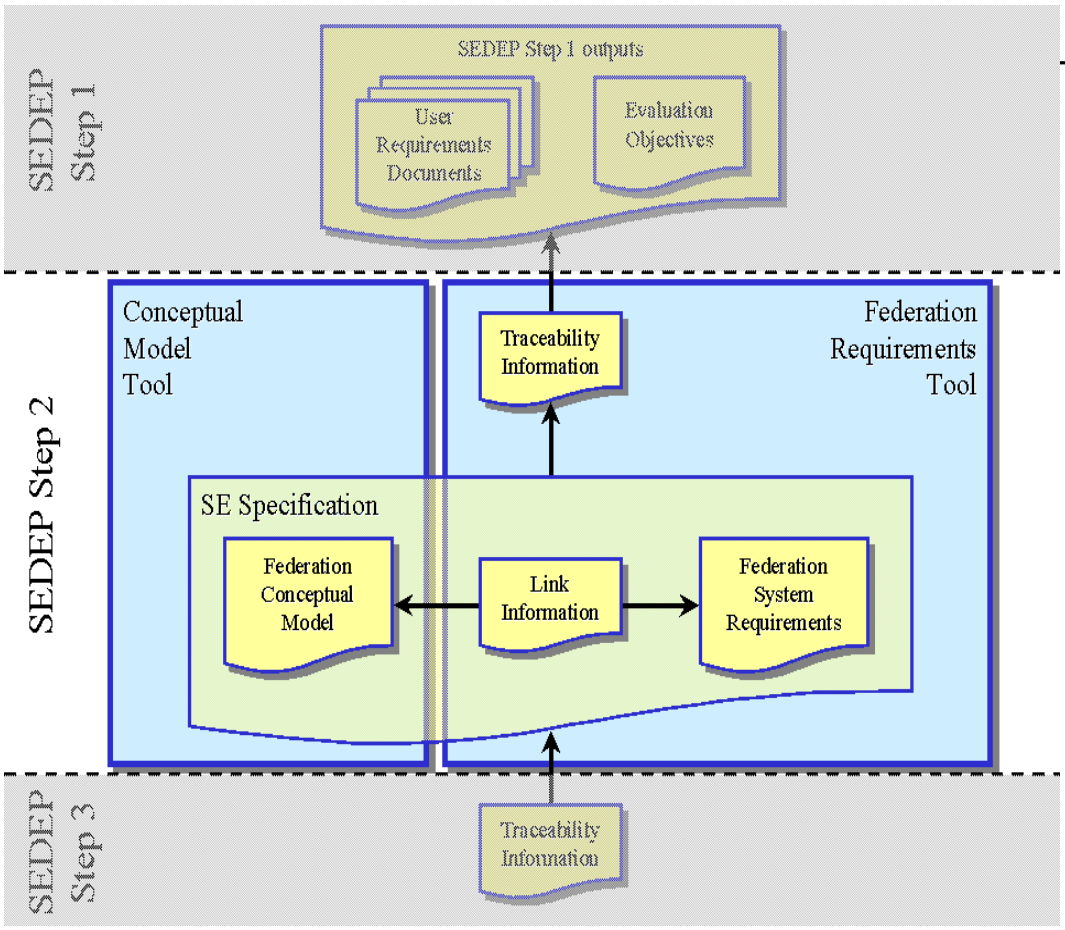

Figure 5: Traceability management

6 Tool Design

To demonstrate the concepts of SEDEP step 2 as outlined in the previous sections, two prototype tools were developed, i.e. the Conceptual Model Tool (CMT) and the Federation Requirements Tool (FRT). Together these tools provide functionality to completely specify the elements, relationships, functionality, constraints and other supplementary requirements of a Synthetic Environment. Specifications on behalf of evaluation needs are also supported by the CMT and FRT and are handled as any other specifications. However, the definition of measures, criteria, algorithms, questionnaires, and checklists for the benefit of evaluation, also referred to as Evaluation Definition is not covered by the CMT and FRT but by the Evaluation Definition Tool (EDT), which is out of scope of this paper.

The core of the Conceptual Model Tool (CMT) and the Federation Requirements Tool (FRT) is realised by the Commercial Off-The Shelf (COTS) tools Rational Rose and DOORS from Telelogic. These COTS tools are registered within the RTP 11.13 SE Management Tool (SEMT). The SE Management tool is an overarching tool that aides both problem setters and problem solvers through the SEDEP process. A user is able to execute the CMT and FRT from the SEMT. The execution is handled through wrappers. The use of wrappers in execution is invisible for the user.

For reasons of flexibility and reuse it was decided to use a two-wrapper architecture for the CMT and FRT as is illustrated for the CMT by Figure 6 and for the FRT by Figure 7. One 'thick' wrapper provides communication to both the repository and the SEMT. A 'thin wrapper' is provided for each COTS tool. The wrappers communicate with each other through the local file system. The names 'thick' and 'thin' are used here, in analogy with the term 'thin' and 'thick' client as commonly used in the client/server nomenclature, indicate the presence of relatively many and complex functionality in 'thick' wrappers and only few and straightforward simple functionality in 'thin' wrappers.

Rational Rose is at the heart of UML based graphical conceptual modelling. DOORS is inherently equipped with versatile traceability management functionalities. Traceability information is handled in certain linking tables. Because of the lack of traceability support in the selected Conceptual Model Tool also the traces of FCM are managed within DOORS. Although the CMT and FRT are implemented using the mentioned COTS tools for purpose of concept demonstration in the Euclid RTP 11.13 programme, the wrapper architecture offers the flexibility to easily incorporate other COTS tools with similar functionality. The choice to use Rational Rose and DOORS was made mainly because of availability at the involved companies.

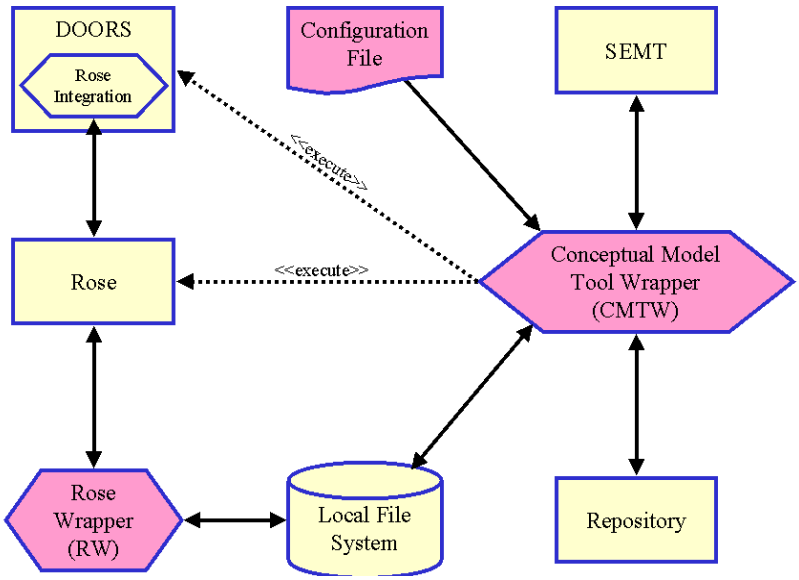

Figure 6: Conceptual Model Tool Architecture

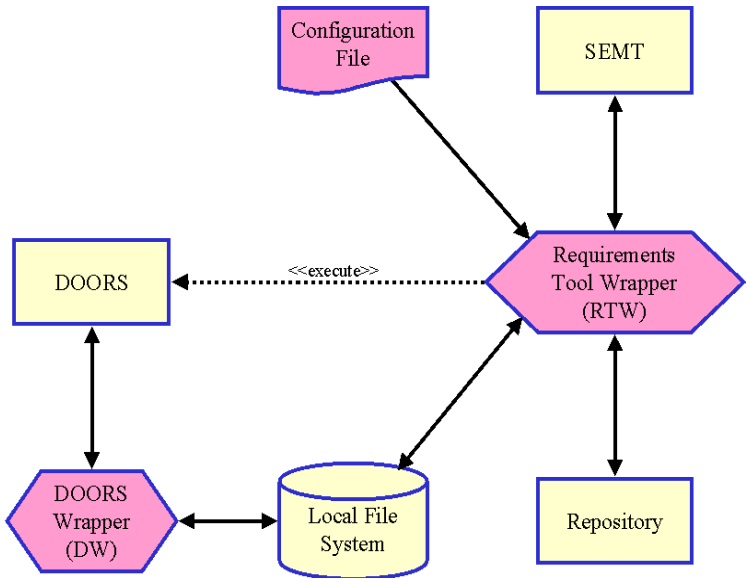

Figure 7: Federation Requirements Tool Architecture

7 Concluding remarks

This paper outlines the results of EUCLID RTP 11.13 with respect to the supporting the activities of SEDEP step 2 (largely similar to FEDEP step 2). Both methodologies and tools have been developed to support the problem solver in performing these activities. The Conceptual Model Tool (CMT) and the Federation Requirements Tool (FRT) provide functionality to completely specify the elements, relationships, functionality, constraints, and other supplementary requirements on a federation that is to be developed. These tools manage two complementary sets of specifications:

- Conceptual Model. A graphical model that captures structure, dependencies and functionality.
- Federation System Requirements. A set of requirements to complement the conceptual model with details and constraints.

All information produced by the tools will be stored in the in the RTP 11.13 repository, such that this information is available for subsequent activities and for possible reuse in other projects.

The Conceptual Model Tool (CMT) key features are:

- The COTS tool Rational Rose is used for concept demonstration.
- Has a flexible wrapper based architecture that allows easy incorporation of different COTS tools.
- Uses the Unified Modelling Language (UML) as the formalised and standardised graphical language to construct the SE Conceptual Model.
- Uses the standardised XML Metadata Interchange (XMI) format to store UML models in the repository.

The Federation Requirements Tool (FRT) key features are:

- The COTS tool Telelogic DOORS is used for concept demonstration.
- Has a flexible wrapper based architecture that allows easy incorporation of different COTS tools.
- Uses a RTP 11.13 developed XML format to store requirements in the repository.
- The tool implementation is shared with the SEDEP step 1 tool for User Constraint Requirements.

8 References

[1] High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.5, December 8 1999.

[2] K. J. Ford, The EUCLID RTP 11.13 Synthetic Environment Development & Exploitation Process, Fall SIW, September 01, Paper 01F-SIW-124.

[3] "Draft Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP) Model" SISO, draft 4, 2002.

[4] "OMG Unified Modeling Language Specification" Object Modeling Group, issue UML V1.4, 2001.

[5] "Extensible Markup Language (XML) 1.0" W3C Recommendation, 2nd edition, 2000.

[6] "OMG XML Metadata Interchange (XMI) Specification" Object Modeling Group, XMI V1.1, 2000.