Project Number 318772

D4.3 Integration of formal evidence and expression in MILS assurance case

Version 1.3
1 October 2015
Final

Public Distribution

Fondazione Bruno Kessler, University of York

Project Partners: **Fondazione Bruno Kessler**, **fortiss**, **Frequentis**, **Inria**, **LynuxWorks**, **The Open Group**, **RWTH Aachen University**, **TTTech**, **Université Joseph Fourier**, **University of York**

Project Partner Contact Information

Fondazione Bruno Kessler Alessandro Cimatti Via Sommarive 18 38123 Trento, Italy Tel: +39 0461 314320 Fax: +39 0461 314591 E-mail: cimatti@fbk.eu	**fortiss** Harald Ruess Guerickestrasse 25 80805 Munich, Germany Tel: +49 89 36035 22 0 Fax: +49 89 36035 22 50 E-mail: ruess@fortiss.org
Frequentis Wolfgang Kampichler Innovationsstrasse 1 1100 Vienna, Austria Tel: +43 664 60 850 2775 Fax: +43 1 811 50 77 2775 E-mail: wolfgang.kampichler@frequentis.com	**LynuxWorks** Yuri Bakalov Rue Pierre Curie 38 78210 Saint-Cyr-l'Ecole, France Tel: +33 1 30 85 06 00 Fax: +33 1 30 85 06 06 E-mail: ybakalov@lnxw.com
RWTH Aachen University Joost-Pieter Katoen Ahornstrasse 55 D-52074 Aachen, Germany Tel: +49 241 8021200 Fax: +49 241 8022217 E-mail: katoen@cs.rwth-aachen.de	**The Open Group** Scott Hansen Avenue du Parc de Woluwe 56 1160 Brussels, Belgium Tel: +32 2 675 1136 Fax: +32 2 894 5845 E-mail: s.hansen@opengroup.org
TTTech Wilfried Steiner Schonbrunner Strasse 7 1040 Vienna, Austria Tel: +43 1 5853434 983 Fax: +43 1 585 65 38 5090 E-mail: wilfried.steiner@tttech.com	**Université Joseph Fourier** Saddek Bensalem Avenue de Vignate 2 38610 Gieres, France Tel: +33 4 56 52 03 71 Fax: +33 4 56 03 44 E-mail: saddek.bensalem@imag.fr
University of York Tim Kelly Deramore Lane York YO10 5GH, United Kingdom Tel: +44 1904 325477 Fax: +44 7976 889 545 E-mail: tim.kelly@cs.york.ac.uk	**Inria** Axel Legay Inria - Campus de Beaulieu 35042 Rennes, France Tel: +33 2 99 84 73 15 Fax: +33 2 99 84 71 71 E-mail: axel.legay@inria.fr

Contents

Document Control

Version	Status	Date
0.1	Document started	22 January 2015
0.2	Started description of patterns	29 January 2015
0.3	Added description of implementation and platform patterns	6 February 2015
0.4	Added description of composition pattern	10 February 2015
0.5	First draft introduction	13 February 2015
0.6	Added description of tool support	17 February 2015
1.0	First release of deliverable	27 February 2015
1.1	Added Starlight Example	12 March 2015
1.2	Added description of Process argument patterns. Small modifications to other argument patterns to link to these process arguments.	26 August 2015
1.3	Added description of TTEthernet argument pattern.	1 October 2015

Executive Summary

This document describes how to integrate formal verification evidence, including evidence obtained with compositional reasoning, as part of the assurance case. In addition, it presents an approach to more formally underpinning the argument module interfaces with assume/guarantee reasoning. This provides a more compelling assurance case, increasing confidence in both the argument and the techniques used. It also improves the ability to re-use argument modules, thus meeting overall aims for distributed MILS.

1 Introduction

The D-MILS approach provides an end-to-end support to the modeling, verification, and deployment of a high-assurance system. The approach is driven by the architecture which is modeled in the MILS-AADL language, it is used to perform a compositional verification deriving system-level properties from the components properties, and whose implementation is ensured by the configuration compiler and the platform. Therefore, the assurance case of a D-MILS system follows this process, detailing how the properties of the system are derived from the components, which techniques and tools have been used in the verification, how the platform configuration is generated, and which guarantees are provided by the platform.

This report describes exactly how assurance cases can be created for D-MILS systems following this process and using information from the MILS-AADL model of the system and formal verification activities. This is achieved firstly through specifying a set of assurance case patterns in section 2 that describe the required structure of the argument and evidence that must be provided in the assurance case generated for any D-MILS system. The assurance case patterns are specified using the GSN pattern notation, as described in [1]. The pattern specification includes details of how to instantiate the patterns for a target system by identifying the source of the required instantiation information from system models.

Section 3 describes how the assurance case for a D-MILS system may be created, in large part, automatically by using the D-MILS model-based assurance case tool developed for this project. The tool creates an assurance case argument automatically using the D-MILS assurance case patterns and the relevant system models. We describe in section 4 an example of automatically creating an assurance case argument for the starlight system.

2 Patterns

This section describes the assurance case patterns for a D-MILS system. A pattern is provided for each of the constituent assurance case modules that will make up the overall D-MILS assurance case. The relationship between the different modules is illustrated in figure 1.

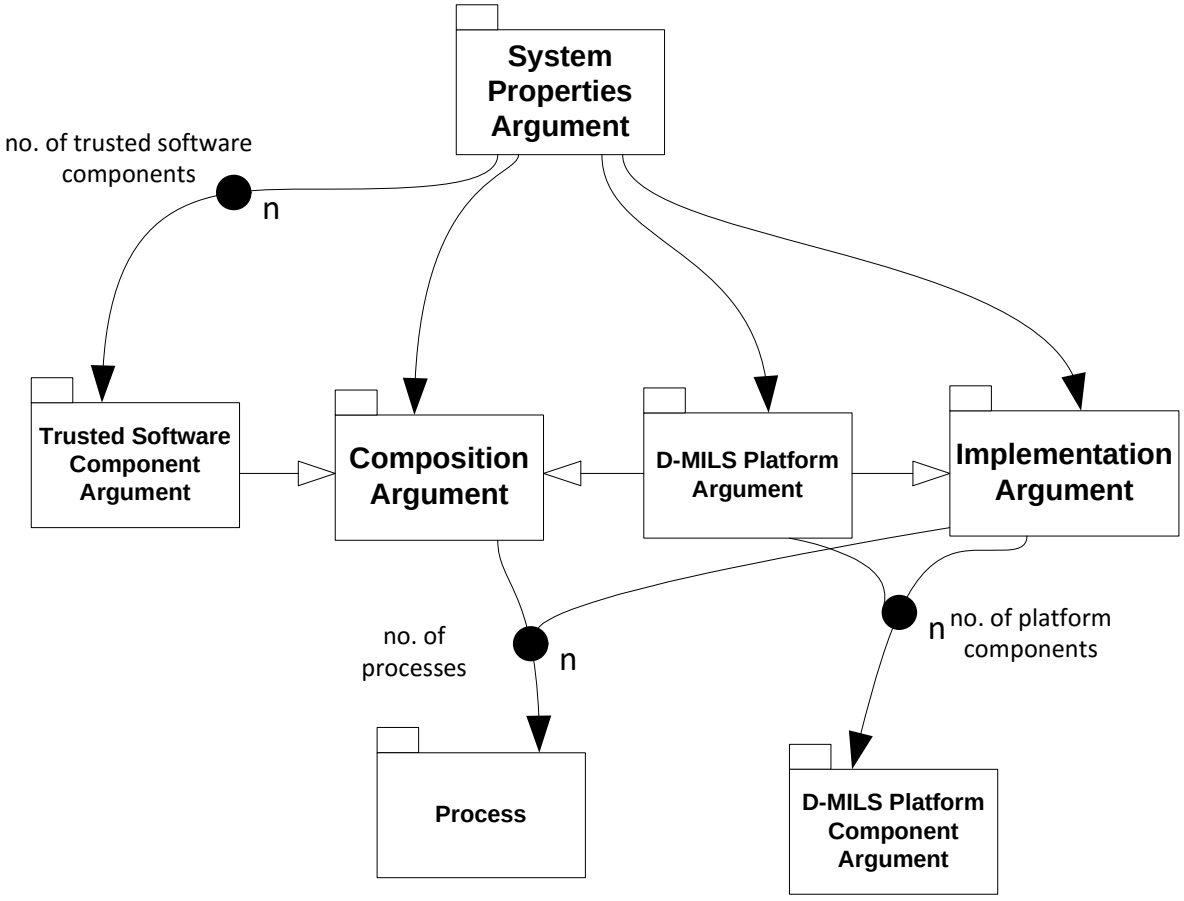

Figure 1: The modules of a D-MILS assurance case

For each D-MILS assurance case pattern we present the following information:

- A general description including the following items:
  - Structure - The pattern structure using GSN.
  - Intent - What the pattern should be used for.
  - Participants - further information regarding key elements of the pattern
  - Applicability - Under what circumstances the pattern may and may not be applied
  - Consequences - Description of what remains to be addressed in the assurance case once this pattern is applied
  - Related Patterns

- Pattern Instantiation - How the patterns may be instantiated using information regarding the target system
- Module Interface - The assurance guarantees made by arguments using this pattern and the dependencies on claims from other assurance case modules required to support those guarantees. The module interface also specifies the context and assumptions within which the assurance guarantees are made.

2.1 System Properties

2.1.1 Pattern description

Structure

The pattern structure is shown in figure 2.

Intent

This pattern should be used to create arguments that a D-MILS system enforces its required properties. The pattern may be used for any properties of interest for the D-MILS system assurance case including, but not limited to, security, safety, functional and real-time properties.

Participants

Goal: sysProps The required properties to be enforced are the informal properties defined for the system, often resulting from hazard analysis or security analysis of the system. These informal properties should be identified using existing established requirements capturing techniques.

Con: sysDescr The system is defined by its MILS-AADL model.

Ass:sysProps The defined system properties must be complete and correct with respect to the threats, vulnerabilities and hazards of the system. It is not within the scope of the D-MILS assurance case to argue about these high-level properties and therefore an assumption is made that this is the case. It is necessary to demonstrate elsewhere that the system properties correctly reflect the system analysis.

Goal: propEnforced A claim of this type is made for each of the system properties.

Con: formalProps Formal properties are defined that correspond to each of the informal system properties

Goal: propAdd For each formal property relating to the informal property under consideration, a claim of this type is created.

Goal: propSuff It must be demonstrated that the formal properties address the informal property.

Goal: propSat This claim, that the MILS-AADL model satisfies the formal property, is supported by the composition assurance case module (see the composition pattern, section 2.2).

Goal: propsSat The verification proves that the formal properties are satisfied by the MILS-AADL model. It must be demonstrated that the implementation of the system supports this, by demonstrating that the MILS-AADL model is correctly implemented and that the assumed properties of the components and platform are enforced.

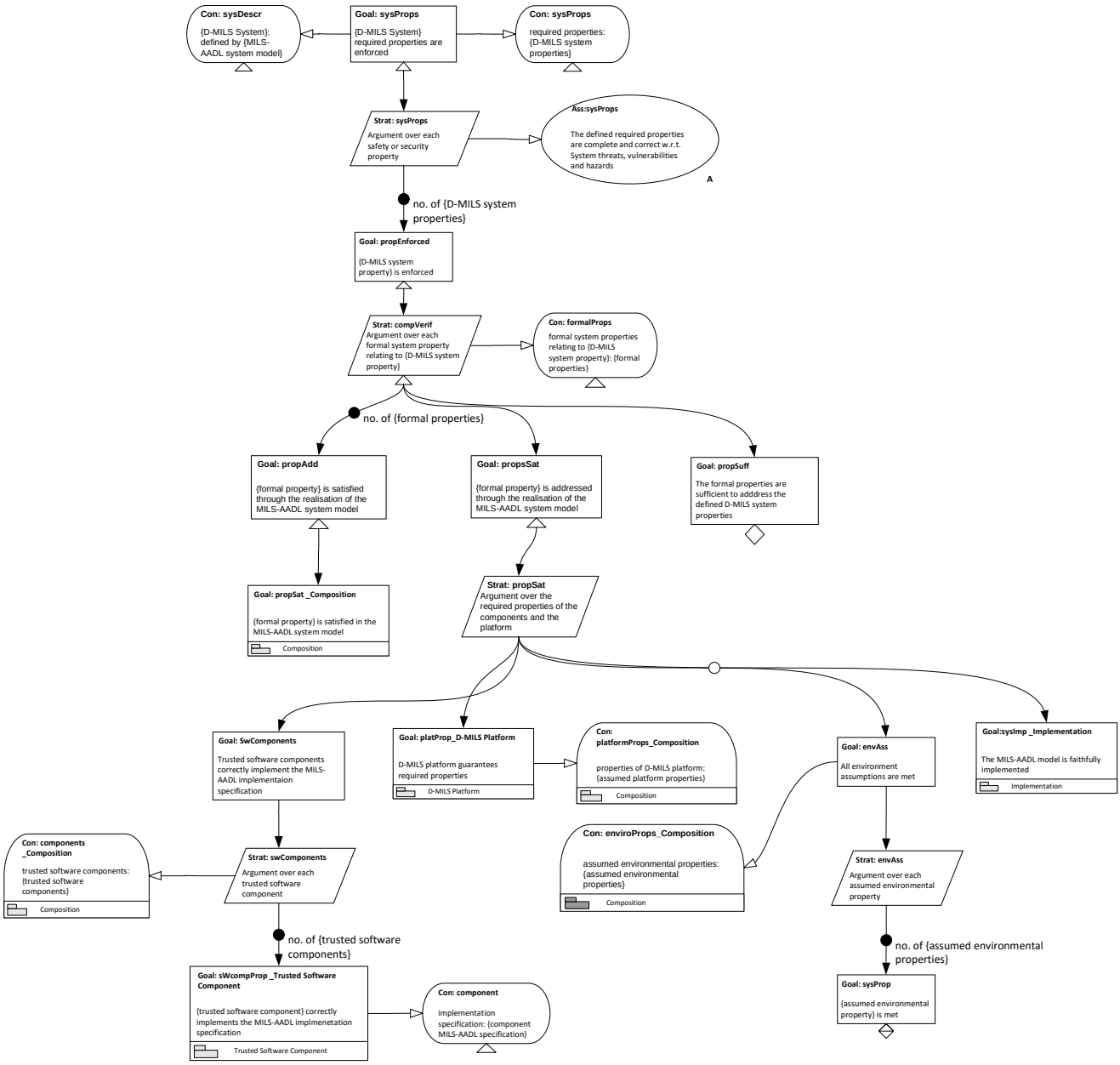

Figure 2: D-MILS system properties assurance case argument pattern

Goal: SwComponents The verification demonstrates that the implementation specification for the software components, defined by the MILS-AADL, satisfies the formal requirement of the component. It must therefore be assured that the software component correctly implements that specification.

Con: components Trusted software components are those components for which formal requirements have been specified.

Goal: sWcompProp A claim of this type is made for each of the trusted software components related to the system property. This claim, that the trusted software component correctly implements its MILS-AADL specification, is supported by the trusted software component module for that component (see the trusted software component pattern, section 2.3).

Goal: platProp The verification of the system properties assumes that the D-MILS platform itself has certain properties. It must be demonstrated that these properties are guaranteed by the platform. The assumed properties of the D-MILS platform include standard D-MILS features, but could also include additional properties if these have been assumed by the verification. This claim is supported by the D-MILS platform assurance case module (see the D-MILS platform pattern, section 2.5).

Goal: envAss The formal system property may be defined by a contract that includes assumptions about the environment to the system. Where such assumptions are present it is necessary to assure that the assumptions are met.

Goal: sysProp A claim of this type is made for each of the environmental assumptions. An argument and evidence must be provided to demonstrate that the assumption holds.

Goal: sysImp It has to be demonstrated that the MILS-AADL model, upon which the formal properties were proved, is faithfully implemented. This claim is supported by the implementation assurance case module (see the implementation pattern, section 2.4).

Applicability

This pattern is applicable to any system that meets the requirements of a D-MILS system and has been developed and analysed using the D-MILS approach.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: propSat A claim of this type will be created for each formal property. Each claim will form a public goal of the composition assurance case module. The way in which these claims are supported is described by the composition pattern (section 2.2).

Goal: propSuff Argument and evidence must be provided to demonstrate that the formal properties address the informal property. Depending upon the notation and tools used for the formal analysis, a number of possible strategies may exist, such as running queries on the formal specification to check the reachability of states, or considering execution traces. To facilitate the support of this claim it is important to capture design justifications at the time at which the formal requirement specification is produced.

Goal: sWcompProp A claim of this type will be created for each trusted software component. Each claim will form a public goal of a separate trusted software component assurance case module. The way in which these claims are supported is described by the trusted software component pattern (section 2.3).

Goal: platProp This claim will form a public goal of the D-MILS platform assurance case module. The way in which this claim is supported is described by the D-MILS platform pattern (section 2.5).

Goal: sysProp A claim of this type will be created for each assumption of the formal system property. Argument and evidence must be provided to demonstrate that the assumption is met by the system's environment. The way in which this is assured will vary depending upon the precise nature of the assumption.

Goal: sysImp This claim will form a public goal of the implementation assurance case module. The way in which this claim is supported is described by the implementation pattern (section 2.4).

Related Patterns

The argument created from this pattern requires support from arguments created using the Composition, Trusted Software Component, D-MILS Platform and Implementation argument patterns.

2.1.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
D-MILS System	Name of the system	system name in MILS-AADL file
MILS-AADL system model	Reference to the MILS-AADL model	Manual
D-MILS system properties	informal system properties	Requirements document or database
D-MILS system property	a member of the set of D-MILS system properties	As for D-MILS system properties
formal properties	formal system properties relating to a D-MILS system property	Guarantee condition of contract defined for system in MILS-AADL specification
formal property	a member of the set of formal properties	As for formal properties
trusted software components	subcomponents of the system for which a formal requirement relating to the system property has been specified	subcomponent name in MILS-AADL specification
trusted software component	a member of the set of trusted software components	As for trusted software components
component MILS-AADL specification	the MILS-AADL specification for a trusted software component	The subject implementation
assumed D-MILS platform properties	Additional assumptions made about the properties of the D-MILS platform over and above the generic platform properties described in the DMILS platform pattern	Unknown
assumed environmental properties	the assumptions made as part of the formal property	Assume condition of contract defined for system in MILS-AADL specification
assumed environmental property	a member of the set of assumed environmental properties	As for assumed environmental properties

Choices

None

Optional

Goal: envAss and its supporting argument are required only when the formal system property contains assumptions.

2.1.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {D-MILS System} required properties are enforced

Dependencies

- {formal property} is satisfied in the MILS-AADL system model
- {trusted software component} correctly implements the MILS-AADL implementation specification
- D-MILS platform guarantees required properties
- The MILS-AADL model is faithfully implemented

Context

- D-MILS System
- required properties
- formal system properties relating to D-MILS system property
- trusted software components
- implementation specification
- properties of D-MILS platform
- assumed environmental properties

Assumptions

- The defined required properties are complete and correct with respect to System threats, vulnerabilities and hazards

2.2 Composition

2.2.1 Pattern description

Structure

The pattern structure is shown in figure 3.

Intent

This pattern should be used to create arguments that formally defined properties of a D-MILS are satisfied by a MILS-AADL model of that system.

Participants

Goal: propSat It is necessary to demonstrate that each of the formal properties specified in the MILS-AADL model is satisfied. This is done using a formal verification approach.

Goal: verifResults The results of the formal verification are used to demonstrate that the formal property is satisfied. The type of formal technique used to verify the property will depend on the property itself.

Con: components As part of the verification of the formal properties, formal requirements may be specified for components of the system that refine the formal system property. Components for which a requirement has been specified are referred to as trusted software components. The trusted software components are referred to from elsewhere in the D-MILS assurance case.

Con: enviroProps As part of the verification of the formal properties, assumptions may be made regarding properties of the environment of the D-MILS system. The assumed environmental properties are referred to from elsewhere in the D-MILS assurance case.

Con: platformProps As part of the verification of the formal properties, it may be necessary to make additional assumptions made about the properties of the D-MILS platform over and above the generic platform properties described in the DMILS platform pattern. The assumed platform properties are referred to from elsewhere in the D-MILS assurance case.

Goal: formalConf As well as presenting the results of the formal verification, it is also necessary to demonstrate that there is sufficient confidence in the correctness of those formal verification results.

Goal: verification The verification technique applied will be selected based upon the type of property to be verified. It must be demonstrated that the process of verification using the technique generates trustworthy results.

Goal: activityTrust A claim is made that the activity of performing the verification using the applied technique is sufficiently trustworthy. This claim is supported by an assurance case module for the verification process, which is an instantiation of the generic process argument pattern (see the process pattern, section 2.6. Where a process model of the verification activity is provided, the appropriate process model is selected according to the name of the technique used (as specified in the MILS-AADL model of the system).

Goal: errorModel It must be demonstrated that the MILS-AADL error model is complete and correct.

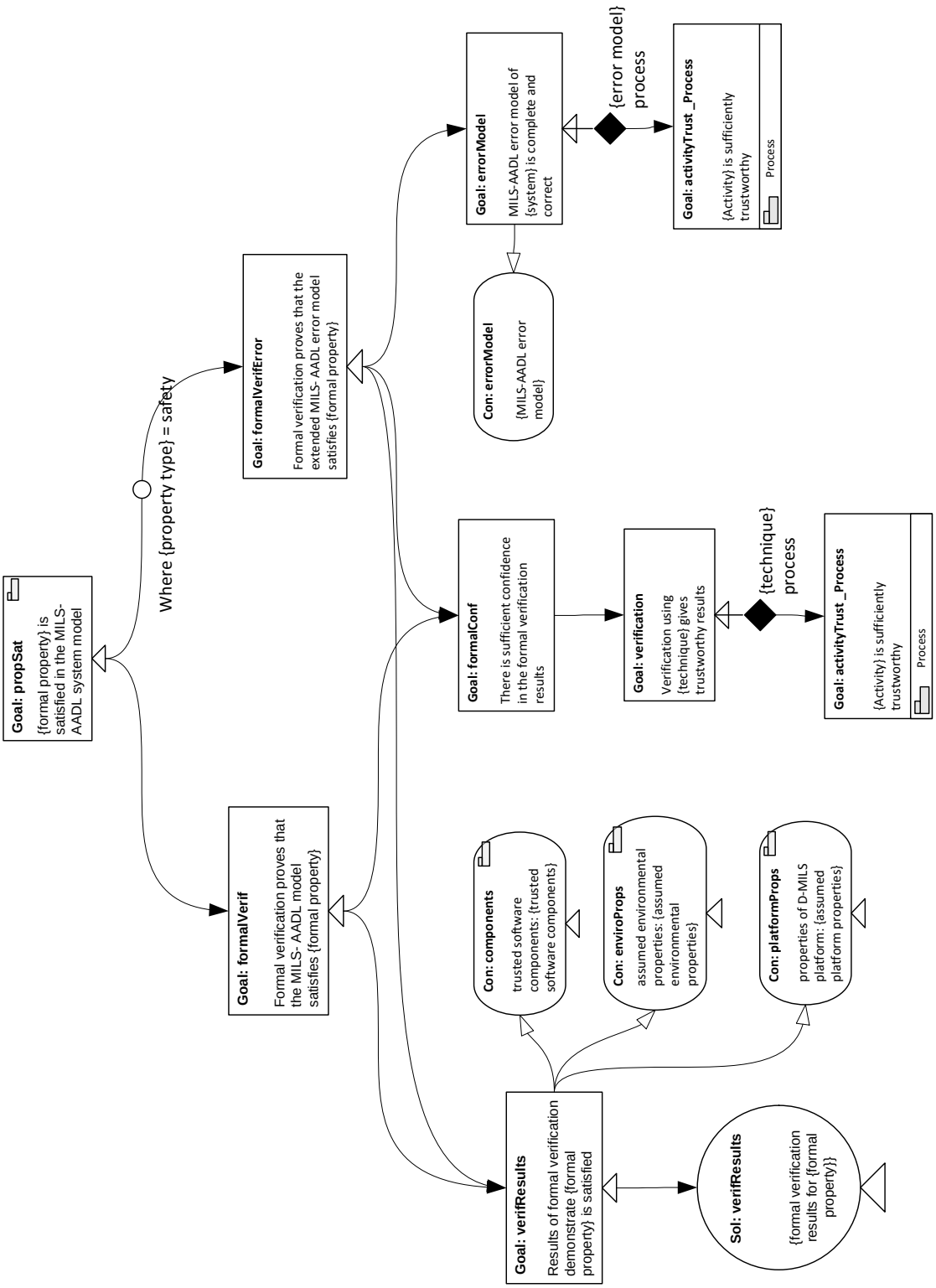

Figure 3: D-MILS composition assurance case argument pattern

Goal: activityTrust A claim is made that the activity of generating the error model is sufficiently trustworthy. This claim is supported by an assurance case module for the error model process, which is an instantiation of the generic process argument pattern (see the process pattern, section 2.6.

Applicability

This pattern is applicable to any D-MILS system that uses a formal verification approach to prove properties of the system.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: activityTrust Two instances of this goal will be created, one for the verification activity, and one for the error model process. These claims will form public goals of the process assurance case module. The way in which this claim is supported is described by the process pattern (section 2.6).

Related Patterns

The argument created from this pattern supports the argument created using the D-MILS system properties pattern and requires support from arguments created using the process argument pattern.

2.2.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`formal property`	formal system property relating to a D-MILS system property	Guarantee condition of contract defined for system in MILS-AADL specification
`trusted software components`	subcomponents of the system for which a formal requirement relating to the system property has been specified	subcomponent name in MILS-AADL specification
`assumed environmental properties`	the assumptions made as part of the formal property	Assume condition of contract defined for system in MILS-AADL specification
`assumed platform properties`	properties assumed as part of the system property verification	Unknown
`formal verification results for {formal property}`	reference to the results of the formal verification of the formal property	Manual
`technique`	the name of the formal technique used for the verification of the formal property	technique name defined for the requirement in MILS-AADL specification
`property type`	the type of the formal property	the type defined for the reference requirement for the formal requirement in the MILS-AADL specification

Choices

{technique} is used to determine which process model to use when instantiating Goal: activityTrust.

Optional

Goal: formalVerifError and its supporting evidence are required only when {property type} to be verified is a safety property.

2.2.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {formal property} is satisfied in the MILS-AADL system model

Dependencies

- {Activity} is sufficiently trustworthy

Context

- assumed environmental properties
- assumed platform properties
- trusted software components
- MILS-AADL error model

Assumptions

None

2.3 Trusted Software Components

2.3.1 Pattern description

Structure

The pattern structure is shown in figure 4.

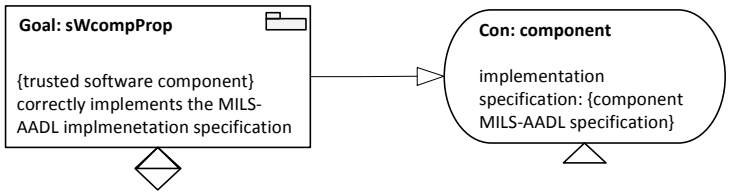

Figure 4: Trusted software components assurance case argument pattern

Intent

This pattern should be used by developers of trusted software components for a D-MILS systems to create arguments that the software component correctly implements the MILS-AADL specification. It is the responsibility of the developing organisation to provide argument and evidence appropriate to the domain of application, certification regime and required integrity. This would be expected to take account of relevant software assurance standards and guidance.

Participants

Goal: sWcompProp The verification of properties for the D-MILS system relies upon trusted software components correctly implementing their specification as defined in the MILS-AADL specification.

Con: component The specification for the component is defined by the MILS-AADL component implementation.

Applicability

This pattern is applicable to any software component for which a formal requirement is specified as part of the MILS-AADL specification.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: sWcompProp It must be demonstrated using evidence generated during the development, analysis and verification of the software component that the specification is correctly implemented.

Related Patterns

The argument created from this pattern can provide support to arguments created using the D-MILS system properties argument pattern.

2.3.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`trusted software component`	a member of the set of `trusted software components`	As for `trusted software components`
`trusted software components`	subcomponents of the system for which a formal requirement relating to the system property has been specified	subcomponent name in MILS-AADL specification
`component MILS-AADL specification`	the MILS-AADL specification for a `trusted software component`	The subject implementation

Choices

None

Optional

None

2.3.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- trusted software component correctly implements the MILS-AADL implmenetation specification

Dependencies

None

Context

- implementation specification

Assumptions

None

2.4 Implementation

2.4.1 Pattern description

Structure

The pattern structure is shown in figure 5.

Intent

This pattern should be used to create arguments that a MILS-AADL model is faithfully implemented by a D-MILS system.

Participants

Goal: impConfig The verified MILS-AADL model is implemented via the creation of a configuration description that is used to configure the D-MILS system. This includes the required configuration and scheduling of the subjects and nodes, and the network connecting the nodes.

Con: config The configuration is described by a MILS Configuration Normal Form (MNCF) file.

Goal: wellFormed It is necessary to demonstrate that the generated configuration is correct with respect to the MILS-AADL specification and satisfies all the constraints (the system model, the platform model and the additional defined constraints). This can be demonstrated by considering the process by which the configuration is generated, and also verifying the consistency of the resulting configuration.

Goal: configGen The correctness of the configuration is demonstrated through consideration of the compilation process used to generate the configuration.

Goal: activityTrust A claim is made that the configuration compilation activity is sufficiently trustworthy. This claim is supported by an assurance case module for the compilation process, which is an instantiation of the generic process argument pattern (see the process pattern, section 2.6).

Goal: configConsist It is necessary to demonstrate that the different views of the configuration, corresponding to the different types of platform component, are consistent with each other. This is done by identifying the objects that are present in multiple views of the configuration and checking that the representation of those objects is consistent between the views. For a D-MILS system the objects present in multiple views include remotely addressable objects (RAOs)and virtual links.

Goal: platform Once a perfected configuration is chosen, that chosen configuration must then be correctly realised by the D-MILS platform itself. The configuration is realised by the constituent components of the D-MILS platform through the use of target-specific configurations created for each platform component.

Goal: targetConfigs It must be shown that the target specific configuration for each of the constituent platform components is syntactically and semantically equivalent to the chosen configuration. This claim is a designated as a public goal, allowing it to be referenced from other assurance case modules. This can be a particularly useful guarantee for other modules that make use of the target-specific configurations.

Goal: activityTrust The generation of the target-specific configurations is done by a process that uses an adapter specific to the target component. It is necessary to demonstrate that this adaption process is trustworthy. This claim is supported by an assurance case module for the target adaption process, which is an instantiation of the generic process argument pattern (see the process pattern, section 2.6).

Goal: configImp It must be demonstrated that the target specific configurations are correctly interpreted and implemented by each of the platform components.

Goal: compConfig This claim, that the platform component correctly implements its configuration, is supported by the {platform component} module for the relevant component.

Applicability

This pattern is applicable to any D-MILS system that uses the D-MILS platform configuration compiler to generate the configuration for the system.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: activityTrust Two instances of this goal will be created, one for the configuration compilation activity, and one for target adaption. These claims will form public goals of the process assurance case module. The way in which this claim is supported is described by the process pattern (section 2.6).

Goal: compConfig A claim of this type will be created for each platform component. Each claim will form a public goal of the relevant {platform component} assurance case module.

Related Patterns

The argument created from this pattern supports the argument created using the D-MILS system properties pattern and requires support from arguments created using the {technique} tool argument pattern.

2.4.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`MILS configuration-normal-form (MCNF) config file`	Reference to the configuration file	Manual
`system model`	Reference to the Prolog system model	Manual
`platform components`	constituent components of the D-MILS platform	`platform description`
`platform component`	a member of the set of `platform components`	As for `platform components`
`types of platform components`	The different types of platform component for which configuration views are created	Manual
`component config file`	Reference to the target-specific configuration file for the `platform component`	Manual

Choices

None

Optional

None

2.4.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- The MILS-AADL model is faithfully implemented
- The target specific configurations are syntactically and semantically correct with respect to the configuration

Dependencies

- {Activity} is sufficiently trustworthy
- {platform component} configuration files are correctly interpreted and implemented in the {platform component}

Context

- constraints: system model, platform description, additional constraints
- configuration
- target platform components
- objects in multiple configurations
- types of platform component
- component configuration files

Assumptions

None

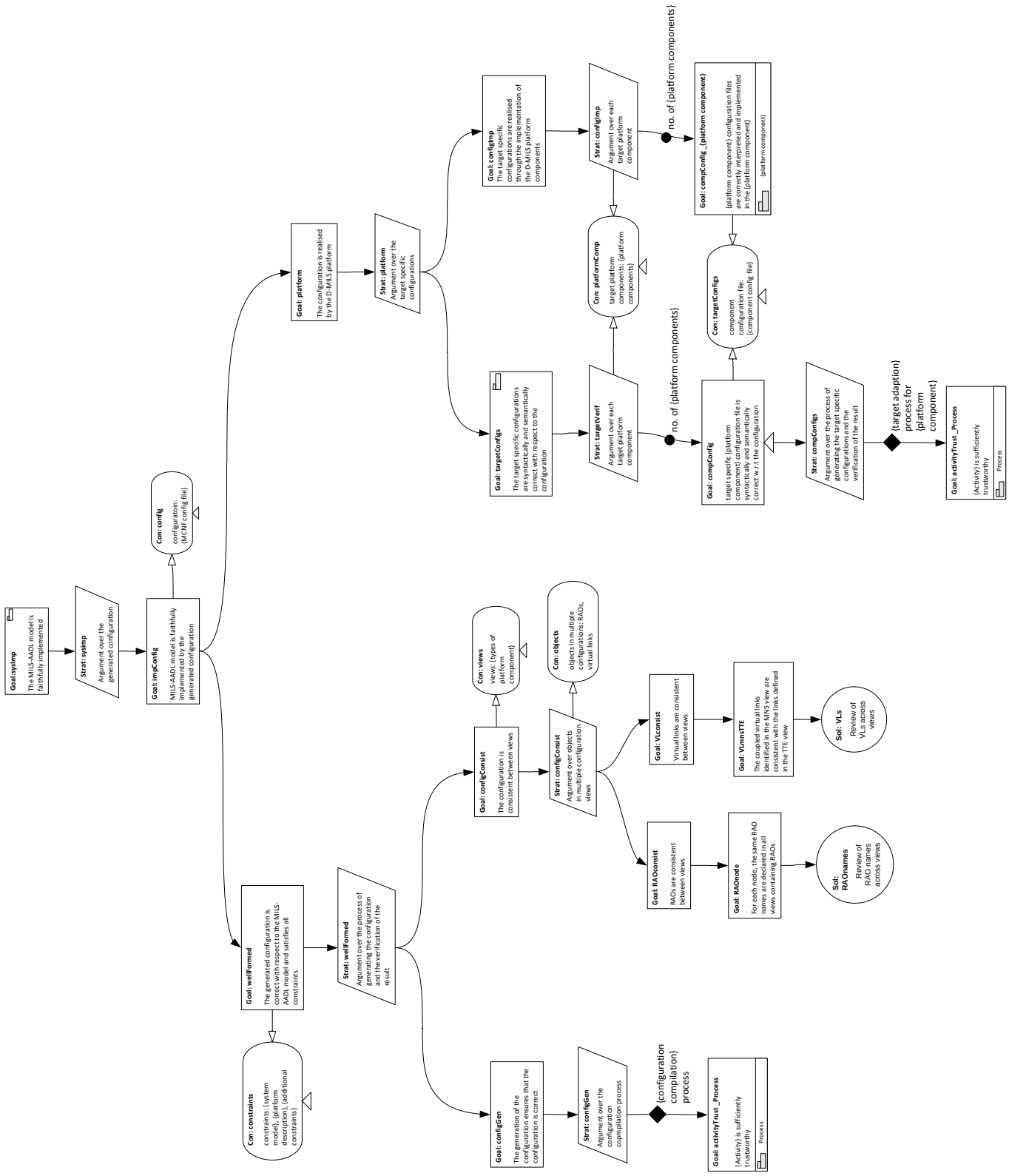

Figure 5: D-MILS implementation assurance case argument pattern

2.5 D-MILS Platform

2.5.1 Pattern description

Structure

The pattern structure is shown in figure 6.

Intent

This pattern should be used to create arguments that a D-MILS platform guarantees the required properties.

Participants

Goal: platProp In order to be able to guarantee properties of a D-MILS system, it is necessary to be able to rely on the D-MILS platform providing certain properties. The required platform properties are those properties that are assumed during the compositional verification of the D-MILS system properties. These platform properties will always include guarantees regarding inter- and intra-nodal interference.

Goal: interComms One of the platform properties that must be guaranteed is that communication between nodes over the network occurs only as defined in MILS-AADL model of the system.

Goal: network Inter-nodal communication is controlled by ensuring that network communication only occurs in accordance with Global Information Flow Policy (GIFP). The GIFP is a target-specific configuration file. This is assured through the operation of the MILS Networking System (MNS) and the TTEthernet network.

Goal: MNS This claim, that the MNS will only permit communications that conform to the GIFP, is supported by the MILS networking system assurance case module.

Goal:TTEgtee This claim, that the TTEthernet network can guarantee that only those connections that have been specified between nodes are possible, is supported by the TTEthernet assurance case module.

Goal: GIFPcorrect Since the GIFP is being used to defined permitted communication between nodes, it must be demonstrated that the GIFP is correct with respect to the MILS-AADL model. This is demonstrated using a claim (Goal: targetConfigs) provided in the implementation module (see the implementaion pattern, section 2.4).

Goal: interComp To ensure the security and integrity of the D-MILS system, it must be demonstrated that the inter-nodal communication guarantees cannot be compromised either maliciously or through error. This is assured by considering the vulnerabilities of the network and demonstrating that those vulnerabilities are mitigated.

Goal: noNetworkAccess The first vulnerability that requires mitigation is access by subjects other than the MNS to the network. If other subjects access the network then they could send and receive messages that aren't in accordance with the GIFP. This is mitigated through assuring that only the MILS Network Subsystem (MNS) can control the TTEthernet card. Without control of the TTEthernet card it is not possible for a subject to communicate over the network. This is guaranteed through a number of design features.

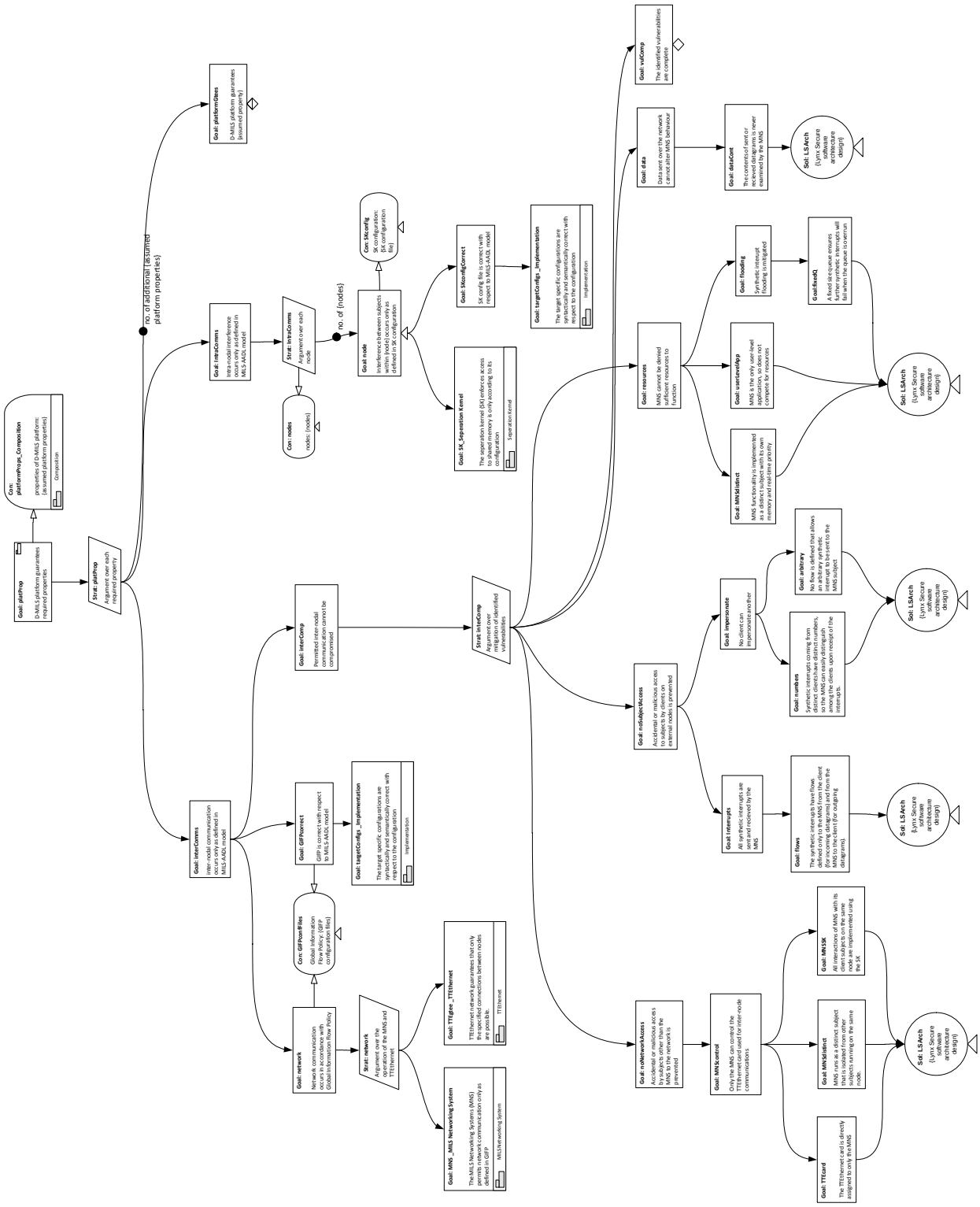

Figure 6: D-MILS platform assurance case argument pattern

Goal: noSubjectAccess The second vulnerability that requires mitigation is clients on other nodes gaining access to a node's subjects to which they are not permitted. This is mitigated in two ways. Firstly through assuring that all interrupts arriving at a node over the network are handled by the MNS (which is guaranteed to enforce the GIFP). Secondly by assuring that a client can't impersonate another client, thus being given improper access to a subject. This is again guaranteed through a number of design features.

Goal: resources The third vulnerability to address is that the MNS could be compromised through being denied sufficient resources to operate. Once again there are features of the platform design that address this vulnerability.

Goal: data The final vulnerability is that there is a possibility that MNS behaviour could be altered by malicious or erroneous data that is sent over the network. It is assured that this cannot occur by ensuring that the MNS never examines the contents of the datagrams sent on the network.

Goal: vulComp It must be demonstrated that there is confidence that there are no more vulnerabilities other than those already addressed in the argument. This could be assured through consideration of vulnerability analysis performed on the platform.

Goal: intraComms Another platform property that must be guaranteed is that interference within a node only occurs as defined in the MILS-AADL model. An argument is made over each of the nodes in the system.

Goal: node For each node it must be assured that the interference between the subjects running on that node only occurs as defined by the configuration. This is enforced using the node's separation kernel.

Goal: SK This claim, that the separation kernel ensures that memory access only occurs as defined by the configuration, is supported by the separation kernel module for the SK of the relevant node.

Goal: SKconfigCorrect Since the SK configuration file is being used to define the permitted interference between subjects, it must be demonstrated that the SK config file for the node is correct with respect to the MILS-AADL model. This is demonstrated using a claim (Goal: targetConfigs) provided in the implementation module (see the implementaion pattern, section 2.4).

Sol: platformGtees Where additional platform properties have been assumed for the verification of the system properties (in additional to the guarantees regarding inter- and intra-nodal interference), it must be demonstrated that these properties are met by the D-MILS platform.

Sol: LSArch Evidence for the presence of many required design features can be provided by information contained in the design of the Lynx Secure software architecture. For each of the required features, the solution should be instantiated with reference to the relevant specific design information.

Applicability

This pattern is applicable to any D-MILS system.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: MNS This claim must be supported by argument and evidence in the MILS networking system assurance case module.

Goal: TTEgtee This claim must be supported by argument and evidence in the TTEthernet assurance case module.

Goal: targetConfigs This claim will form a public goal of the implementation assurance case module. The way in which this claim is supported is described by the implementation pattern (section 2.4).

Goal: SK This claim must be supported by argument and evidence in the MILS networking system assurance case module.

Goal: vulComp Argument and evidence must be provided concerning the completeness of the network vulnerabilities.

Goal: platformGtees A claim of this type will be created for each additional required platform guarantee. Each claim will require argument and evidence that the assumed property is guaranteed by the D-MILS platform.

Related Patterns

The argument created from this pattern supports the argument created using the D-MILS system properties pattern and requires support from arguments created using the implementation argument pattern.

2.5.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
GIFP configuration file	Reference to the configuration file	Manual
Lynx Secure software architecture design	Reference to design features	Unknown
nodes	The constituent nodes of the D-MILS system	platform description
node	a member of the set of nodes	As for nodes
assumed platform properties	properties assumed as part of the system property verification	Unknown
assumed property	a member of the set of assumed platform properties	As for assumed platform properties

Choices

None

Optional

None

2.5.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- D-MILS platform guarantees required properties

Dependencies

- The MILS Networking Systems (MNS) permits network communication only as defined in GIFP
- TTEthernet network guarantees that only the specified connections between nodes are possible.
- The target specific configurations are syntactically and semantically correct with respect to the configuration
- The separation kernel enforces access to shared memory is only according to its configuration

Context

- properties of D-MILS platform
- Global Information Flow Policy
- nodes
- separation kernel configuration

Assumptions

None

2.6 Process

2.6.1 Pattern description

Structure

The pattern structure is shown in figure 7.

Intent

This pattern is used to create an argument for any process used during the development and analysis of a D-MILS system. The trustworthiness of the processes adopted is an important part of the assurance case. Process arguments are used as part of the argument made within other modules of the D-MILS assurance case. This pattern is instantiated for the process relevant to the argument at that point. Where process models have been created, these process models can be used to automatically instantiate the process argument pattern. Processes are considered to be activities, which can be composed of further sub-activities. All activities (and sub-activities) may have any number of participants. Participants may be classified as tools, people or organisations. Activities may require and produce any number of artefacts. Activities are undertaken using a defined technique.

Participants

Goal: activityTrust There are many points within the D-MILS assurance case where the trustworthiness of an activity (process) must be assured.

Goal: activityParts It must be demonstrated that each of the participants in an activity are sufficiently trustworthy to undertake that activity.

Goal: partTrust A claim of this type is made for each of the participants in an activity. The appropriate supporting claim for this must be selected based upon the type of the participant (tool, person or organisation).

Goal: tool This claim is made where the participant in an activity is a tool. This claim is supported by an argument of the form presented in the Tool argument pattern (see section 2.7).

Goal: person This claim is made where the participant in an activity is a person. This claim is supported by an argument of the form presented in the Person argument pattern (see section 2.8).

Goal: organisation This claim is made where the participant in an activity is an organisation. This claim is supported by an argument of the form presented in the Organisation argument pattern (see section 2.9).

Goal:activityReqs It must be demonstrated that each of the artefacts required in order to perform the activity is sufficiently trustworthy.

Goal: reqArtTrust A claim of this type is created for each artefact required by an activity. This claim is supported by an argument of the form presented in the Artefact argument pattern (see section 2.12.3).

Goal:activityTech It must be demonstrated that the techniques used to perform the activity are sufficiently trustworthy.

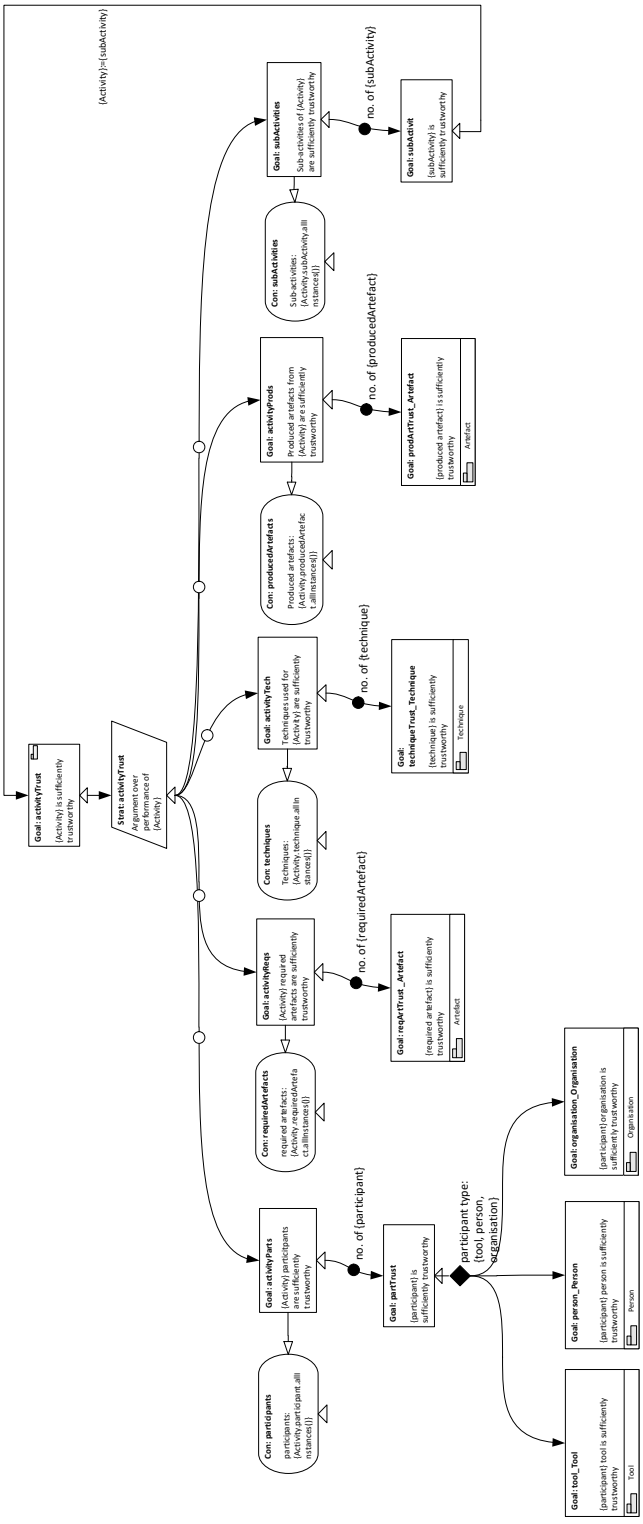

Figure 7: D-MILS process assurance case argument pattern

Goal: techniqueTrust A claim of this type is created for each technique applied in performing an activity. This claim is supported by an argument of the form presented in the Technique argument pattern (see section 2.11).

Goal:activityProds It must be demonstrated that each of the artefacts generated as a result of performing an activity is sufficiently trustworthy.

Goal: ProdArtTrust A claim of this type is created for each artefact produced by an activity. This claim is supported by an argument of the form presented in the Artefact argument pattern (see section 2.12.3).

Goal:subActivities It must be demonstrated that each of the sub-activities of an activity is sufficiently trustworthy.

Goal: subActivit A claim of this type is created for each sub-activity. This claim is supported by applying the same process argument pattern (as described here) to each sub-activity i.e. the argument for a sub-activity has the same form as that for the activity itself.

Applicability

This pattern is applicable to any process. An example process model is provided below, however note that it is not required to produce a process model in order to apply this pattern.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: tool This claim must be supported by argument and evidence as defined by the Tool assurance case pattern.

Goal: person This claim must be supported by argument and evidence as defined by the Person assurance case pattern.

Goal: organisation This claim must be supported by argument and evidence as defined by the Organisation assurance case pattern.

Goal: reqArtTrust This claim must be supported by argument and evidence as defined by the Artefact assurance case pattern.

Goal: techniqueTrust This claim must be supported by argument and evidence as defined by the Technique assurance case pattern.

Goal: prodArtTrust This claim must be supported by argument and evidence as defined by the Artefact assurance case pattern.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS composition and implementation patterns and requires support from arguments created using the tool, person, organisation, artefact and technique argument patterns.

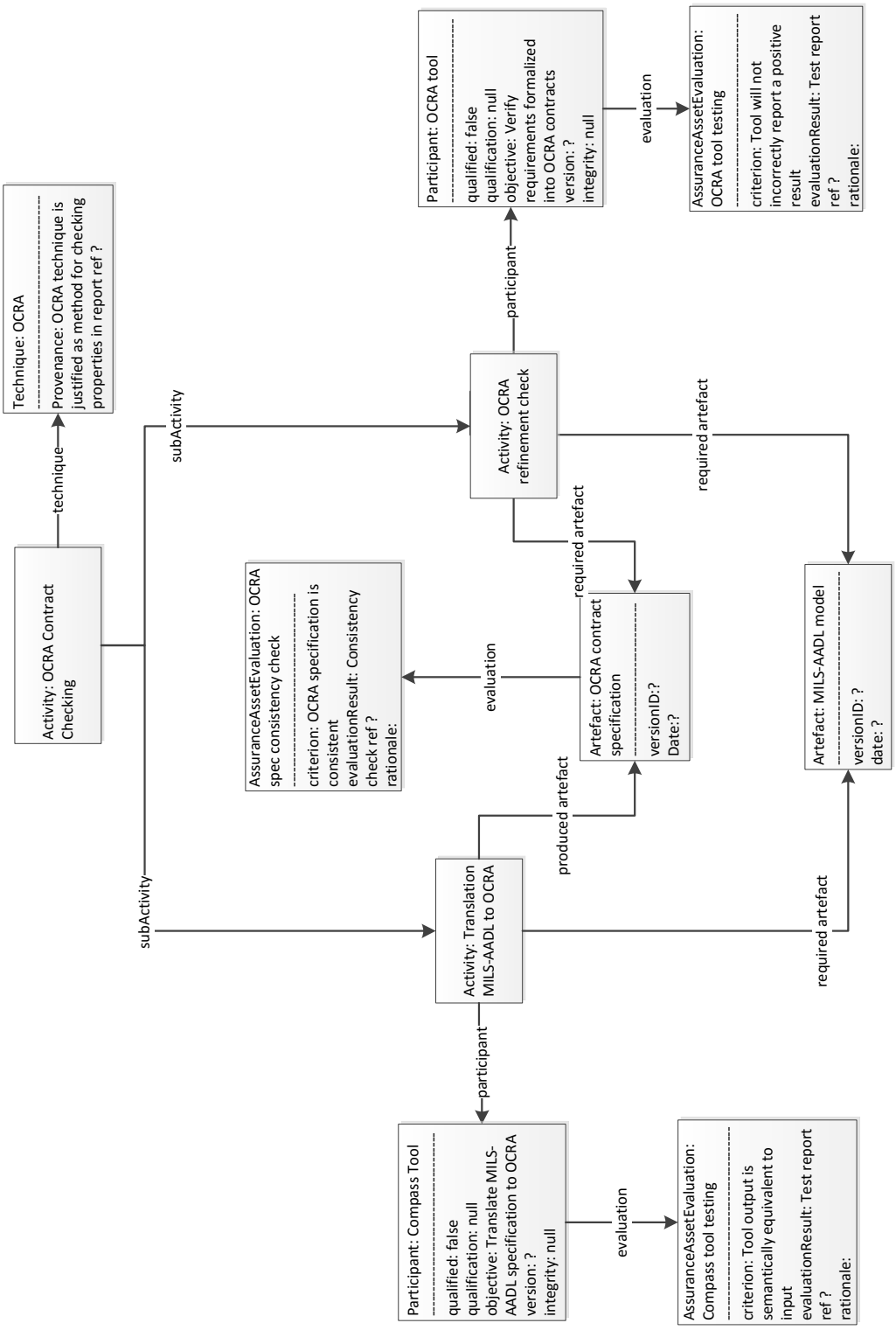

Figure 8: Example process model for the activity of checking OCRA contracts

2.6.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`Activity`	Name of the process activity	Process model
`participant`	A participant in an activity	Process model
`required artefact`	An artefact required to perform an activity	Process model
`technique`	A technique applied in performing an activity	Process model
`produced artefact`	An artefact generated from an activity	Process model
`subActivity`	Name of a sub-activity	Process model

Choices

The appropriate child goal of Goal: partTrust should be chosen depending on the type of the participant.

Optional

All of the child goals of Strat: activityTrust are optional and should only be instantiated if the relevant elements exist as part of the process.

2.6.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {Activity} is sufficiently trustworthy

Dependencies

- {participant} tool is sufficiently trustworthy
- {participant} person is sufficiently trustworthy
- {participant} organisation is sufficiently trustworthy
- {required artefact} is sufficiently trustworthy
- {technique} is sufficiently trustworthy
- {produced artefact} is sufficiently trustworthy

Context

- participants
- required artefacts
- techniaues
- produced artefacts
- sub-activities

Assumptions

None

2.7 Tool

2.7.1 Pattern description

Structure

The pattern structure is shown in figure 9.

Intent

This pattern is used to argue about the trustworthiness of tools used as part of a process included in the assurance case.

Participants

Goal: tool It must be demonstrated that the tool is trustworthy to use in its role as part of an activity. This is done by showing that it is appropriate for that task and that it has the necessary assurance.

Goal: toolProv The required integrity of the tool must be determined. This will be determined from consideration of the criticality of the activity for which the tool is used. This will depend, amongst other things, on the assurance requirements placed upon the system as a whole.

Goal: toolQualify If a tool has been qualified (to some standard) then the argument should demonstrate that the objectives of that standard have been met. The nature of this argument will depend upon the standard being applied. The integrity level achieved for the tool through qualification should be at least that which was determined to have been required.

Goal: objectives The objectives of using the tool as part of the activity should be clearly defined. A claim is made that the defined objectives are met. To show the objectives are met it is necessary to perform some evaluation of the tool against those objectives. An argument about the development of the tool may also be provided to give confidence that the objectives are met.

Goal: evaluations A number of evaluations may be performed on the tool, such as tests, to show that the tool meets its objectives. The type and number of evaluations performed will depend upon the required integrity of the tool.

Goal: evaluation A claim of this type is created for each evaluation performed on the tool.

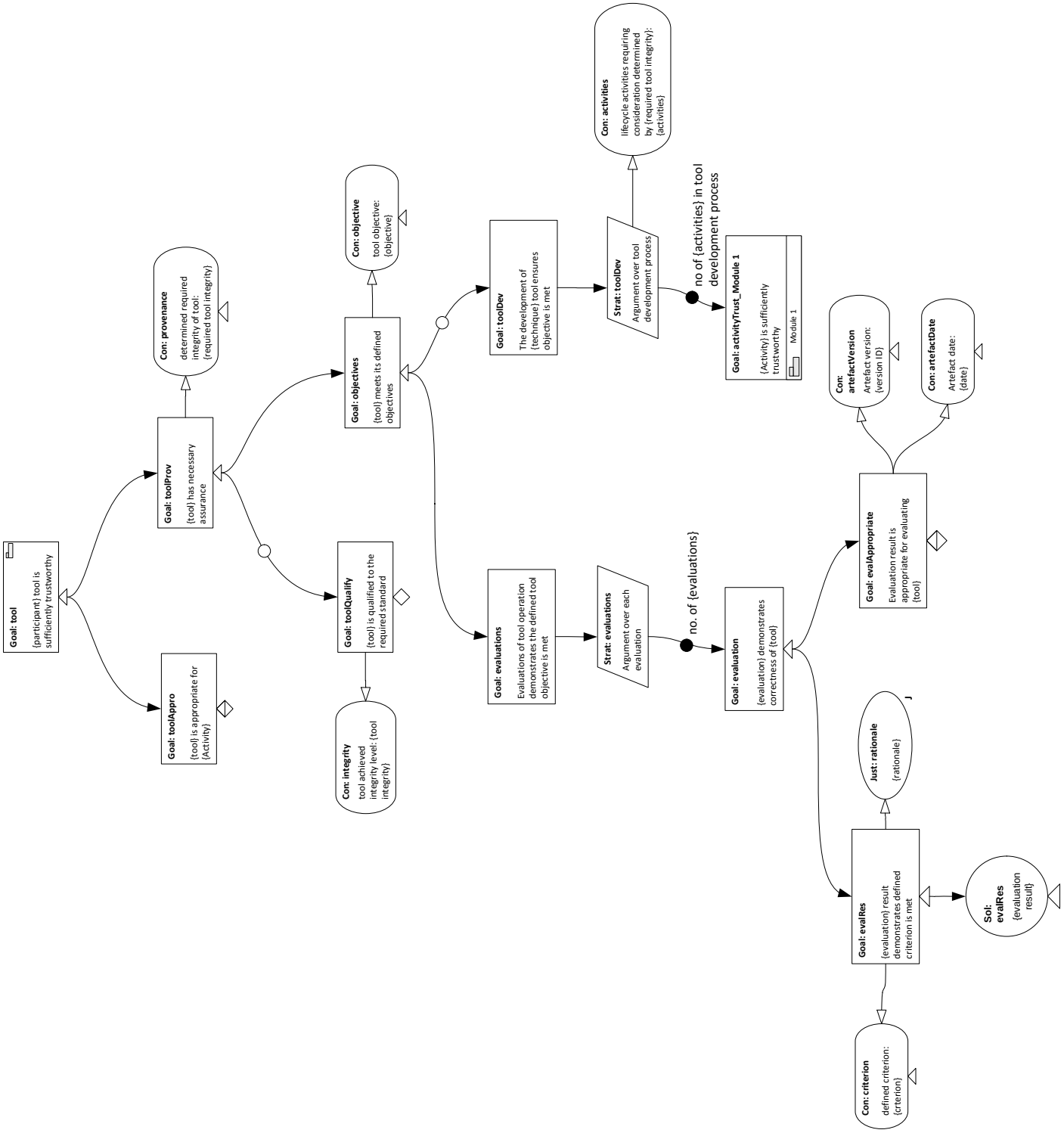

Figure 9: D-MILS tools assurance case argument pattern

Goal: evalRes Each evaluation performed on a tool must have criterion defined that specify explicitly what the evaluation is trying to demonstrate. A claim should be made that the results of the evaluation are able to demonstrate that the criterion are met. Rationale explaining the relevance of the evaluation result may also be provided.

Goal: evalAppropriate It must be shown that the evaluation results used are appropriate for the tool being evaluated. In particular it must be shown that the evaluation was performed on the correct version of the tool.

Goal: toolDev Depending upon whether the tool has been qualified or not, and on the required level of integrity of the tool, a claim may be made that the tool has been developed in a trustworthy manner. This claim requires consideration of the process used to develop the tool. This requires that each of the activities in the development lifecycle of the tool is considered. Although for high integrity tools each lifecycle activity may be considered, for lower integrity, just a subset of key activities may be included in the argument.

Goal: activityTrust For each activity in the tool development lifecycle that is being considered, a claim regarding the trustworthiness of that activity must be made. This claim is supported by an argument of the form presented in the Process argument pattern (see section 2.6).

Applicability

This pattern is applicable to any tool used as part of a process.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: toolAppro Argument and evidence must be provided to demonstrate that the chosen tool is appropriate to use for its intended role as part of the activity.

Goal: toolQualify Argument and evidence must be provided to demonstrate that the tool has been qualified to the defined standard. This is an argument of conformance to the defined objectives defined in the standard.

Goal: evalAppropriate Argument and evidence must be provided to demonstrate that the referenced evaluation result is an appropriate item of evidence for the tool under consideration.

Goal: activityTrust This claim must be supported by argument and evidence as defined by the Process assurance case pattern.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS process pattern and may also require support from arguments created using the process pattern.

2.7.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`participant`	A participant in an activity	Process model
`tool`	The name of the tool	Process model
`Activity`	Name of the process activity	Process model
`required tool integrity`	The integrity level that has been determined for the tool	Manual
`tool integrity`	The integrity level that has been demonstrated for the tool	Process model
`objective`	The objective of the tool in this activity	Process model
`evaluation`	Name of the evaluation performed on the tool	Process model
`criterion`	Criterion to define what the evaluation will demonstrate	Process model
`rationale`	Rationale for the appropriateness of the evaluation	Process model
`version ID`	Version number of the tool being evaluated	Process model
`date`	Date on which the evaluation was performed	Process model

Choices

None.

Optional

Goal: toolQualify is only required for qualified tools. Goal: toolDev may not be required depending upon the required integrity of the tool and whether it is qualified.

2.7.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {participant} tool is sufficiently trustworthy

Dependencies

- {Activity} is sufficiently trustworthy

Context

- determined required integrity of tool
- tool objective
- tool achieved integrity level
- defined criterion
- rationale
- Artefact version
- Artefact date
- lifecycle activities requiring consideration

Assumptions

None

2.8 Person

2.8.1 Pattern description

Structure

The pattern structure is shown in figure 10.

Intent

This pattern is used to argue about the trustworthiness of people involved in a process included in the assurance case.

Participants

Goal: person It must be demonstrated that the person participating in the activity is trustworthy undertaking their role. This is done by showing that the person has the required attributes to perform the activity and also that the level of supervision and review provided is sufficient.

Goal: personAtt It must be demonstrated that the person participating in the activity has both the capability and experience to perform the activity.

Goal: capability The capability of the person is demonstrated through the provision of evidence, such as the person's qualifications or training record.

Goal: experience The experience of the person in carrying out the activity is demonstrated through the provision of evidence, such as the person's CV.

Goal: supervision It is often necessary to supervise a person performing an activity and/or review their work. It must be demonstrated that the level of supervision and review provided is sufficient. The level of supervision and review required will depend upon the capability and experience of the person performing the task and also upon the level of integrity required of the activity being performed. Where supervision is required, the trustworthiness of the supervisor may also need to be demonstrated, in this case the person argument pattern may also be applied to the supervisor.

Applicability

This pattern is applicable to any person involved in performing a process activity.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: supervision Argument and evidence must be provided to demonstrate that the participant is provided with sufficient supervision and review, as appropriate for the person's attributes and the required integrity of the activity.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS process pattern.

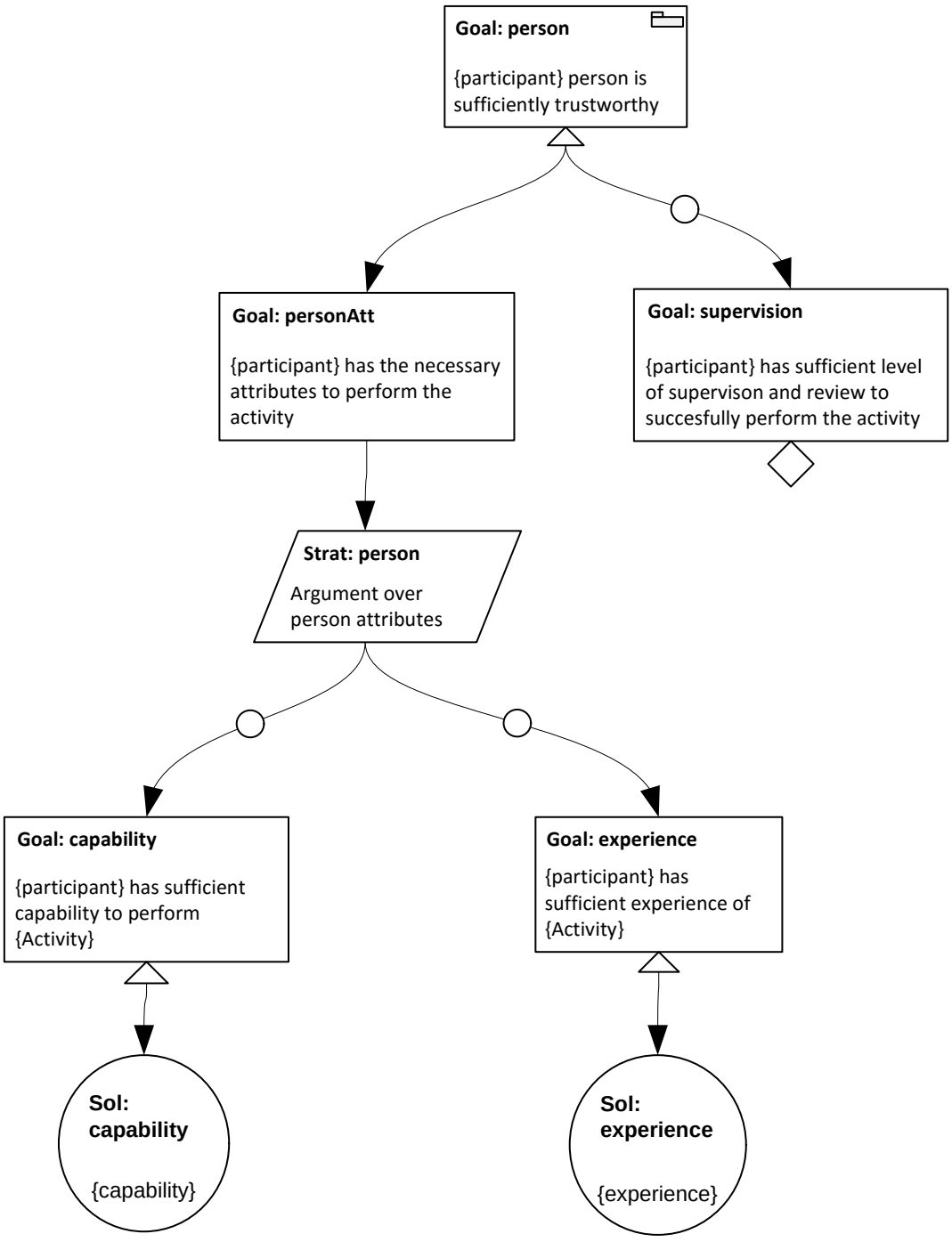

Figure 10: D-MILS person assurance case argument pattern

2.8.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`participant`	A participant in an activity	Process model
`Activity`	Name of the process activity	Process model
`capability`	Description of the person's capability to perform the activity or reference to evidence of capability	Process model
`experience`	Description of the person's experience in carrying out the activity or reference to evidence of that experience	Process model

Choices

None

Optional

Supervision and review of the person may not always be necessary, and Goal: supervision is therefore optional. The person performing the task may not have demonstrable capability or experience in that activity, therefore Goal: capability and Goal: experience are also marked as optional. In cases where either of these claims cannot be supported, it is likely that Goal: supervision will need to be supported.

2.8.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {participant} person is sufficiently trustworthy

Dependencies

None

Context

None

Assumptions

None

2.9 Organisation

2.9.1 Pattern description

Structure

The pattern structure is shown in figure 11.

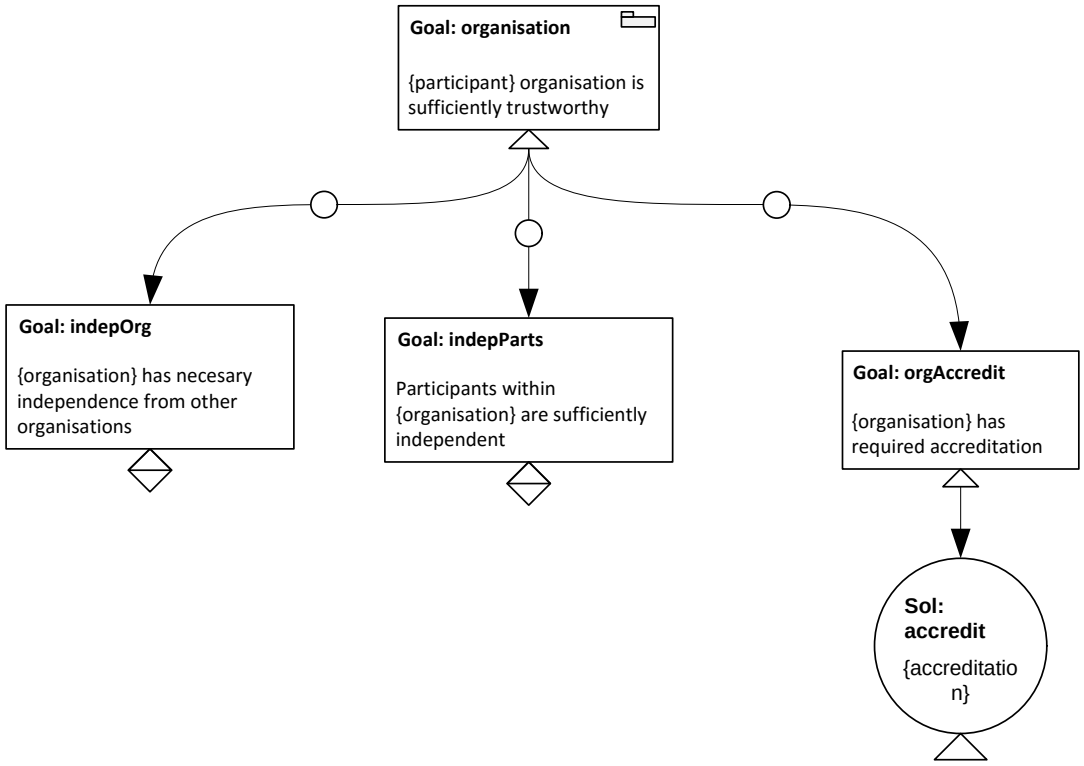

Figure 11: D-MILS organisation assurance case argument pattern

Intent

This pattern is used to argue about the trustworthiness of organisations involved in a process included in the assurance case.

Participants

Goal: organisation It must be demonstrated that the organisation responsible for carrying out the activity is trustworthy. This is done by showing that the organisation is sufficiently independent form other organisations involved in the activity, that all participants from the same organisation are sufficiently independent from each other, and that the organisation has any accreditation that is required.

Goal: indepOrg Where multiple organisations are involved in fulfilling different activities within a process, it may be required to demonstrate that the organisations are independent from each

other. This will particularly be the case where one organisation is validating or checking the work of another. The level of independence required for activities will often depend upon the required integrity of the process being performed.

Goal: indepParts Where multiple people from the same organisation are involved in carrying out activities, it may be required to demonstrate that those individuals, despite being members of the same organisation, are sufficiently independent from each other. This will particularly be the case where one person is validating or checking the work of another. The level of independence required for activities will often depend upon the required integrity of the process being performed.

Goal: orgAccredit It must be demonstrated that an organisation carrying out an activity has all the relevant required accreditation. The nature and level of accreditation required will be determined based upon the required integrity of the process being performed and the requirements of relevant standards for the domain of the target system.

Applicability

This pattern is applicable to any organisation involved in performing a process activity.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: indepOrg Argument and evidence must be provided to demonstrate that the organisation is sufficiently independent from other organisations involved in the process, as determined by the nature of the activities performed by the organisations, the required integrity of the process and requirements of the relevant standards.

Goal: indepParts Argument and evidence must be provided to demonstrate that the participants from within an organisation involved in an activity are sufficiently independent from other participants from the same organisation, as determined by the nature of the activities performed by the organisations, the required integrity of the process and requirements of the relevant standards.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS process pattern.

2.9.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`participant`	A participant in an activity	Process model
`organisation`	Name of the organisation	Process model
`accreditation`	Description of the organisation's accreditation or reference to accreditation	Process model

Choices

None

Optional

All of the child goals of Goal: organisation are optional. Whether these claims are required in the assurance case will be determined based upon consideration of the nature of the activities performed by the organisation, the required integrity of the process and requirements of the relevant standards.

2.9.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {participant} organisation is sufficiently trustworthy

Dependencies

None

Context

None

Assumptions

None

2.10 Artefact

2.10.1 Pattern description

Structure

The pattern structure is shown in figure 12.

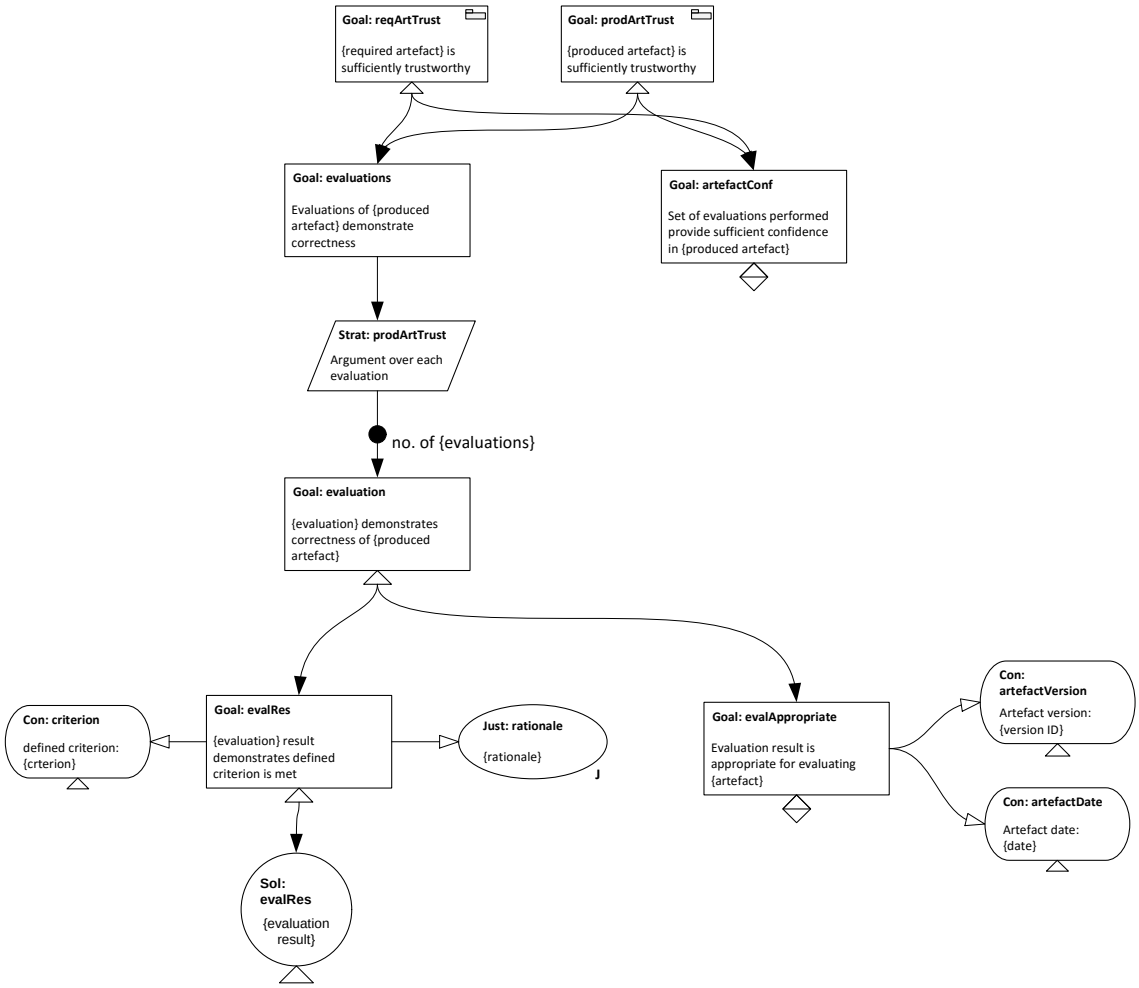

Figure 12: D-MILS artefact assurance case argument pattern

Intent

This pattern is used to argue about the trustworthiness of artefacts either required by, or produced by, activities included in the assurance case.

Participants

Goal: reqArtTrust It must be demonstrated that all artefacts required in order to perform an activity are trustworthy. This is done performing evaluations of the artefact.

Goal: prodArtTrust It must be demonstrated that all artefacts produced as a result of performing an activity are trustworthy. This is done performing evaluations of the artefact.

Goal: evaluation For each evaluation performed on an artifact a claim is created that the evaluation demonstrates the correctness of that artefact.

Goal: evaluations A number of evaluations may be performed on an artefact, such as tests or reviews, to show that the artefact is correct. The type and number of evaluations performed will depend upon the required integrity of the artefact.

Goal: evaluation A claim of this type is created for each evaluation performed of the artefact.

Goal: evalRes Each evaluation performed on an artefact must have criterion defined that specify explicitly what the evaluation is trying to demonstrate. A claim should be made that the results of the evaluation are able to demonstrate that the criterion are met. Rationale explaining the relevance of the evaluation result may also be provided.

Goal: evalAppropriate It must be shown that the evaluation results used are appropriate for the artefact being evaluated. In particular it must be shown that the evaluation was performed on the correct version of the artefact.

Goal: artefactConf It must be shown that the evaluation, or set of evaluations, that are performed on the artefact are appropriate to provide the required level of confidence in the correctness of the artefact.

Applicability

This pattern is applicable to any artefact required by or produced by a process activity.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: artefactConf Argument and evidence must be provided to demonstrate that the evaluations performed on the artefact provide the required confidence in that artefact's correctness.

Goal: evalAppropriate Argument and evidence must be provided to demonstrate that the referenced evaluation result is an appropriate item of evidence for the artefact under consideration.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS process pattern.

2.10.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`required artefact`	An artefact required by an activity	Process model
`produced artefact`	An artefact produced by an activity	Process model
`evaluation`	Name of the evaluation performed on the artefact	Process model
`criterion`	Criterion to define what the evaluation will demonstrate	Process model
`rationale`	Rationale for the appropriateness of the evaluation	Process model
`version ID`	Version number of the artefact being evaluated	Process model
`date`	Date on which the evaluation was performed	Process model

Choices

None

Optional

None

2.10.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {required artefact} is sufficiently trustworthy
- {produced artefact} is sufficiently trustworthy

Dependencies

None

Context

- defined criterion
- rationale
- artefact version
- artefact date

Assumptions

None

2.11 Technique

2.11.1 Pattern description

Structure

The pattern structure is shown in figure 13.

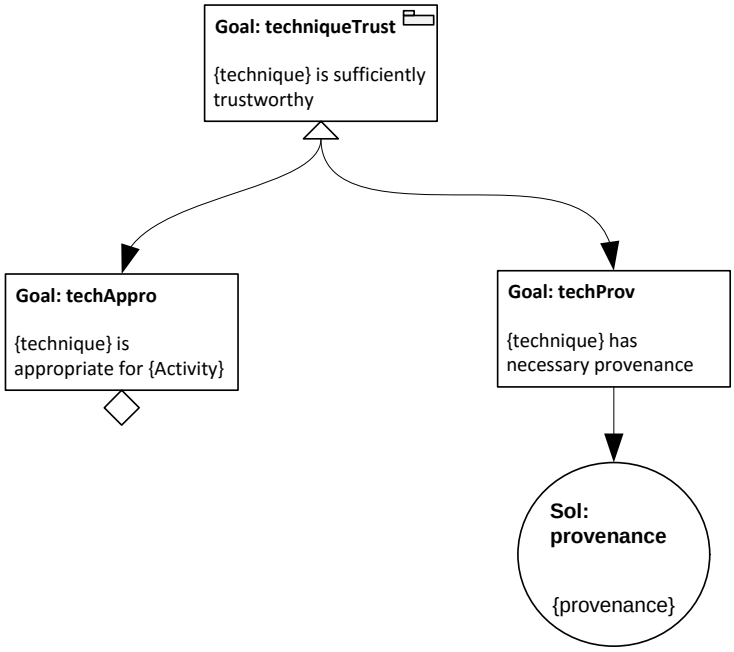

Figure 13: D-MILS technique assurance case argument pattern

Intent

This pattern is used to argue about the trustworthiness of techniques applied when performing activities included in the assurance case.

Participants

Goal: techniqueTrust It must be demonstrated that any techniques used to perform an activity are trustworthy. This is done by considering the appropriateness and provenance of the technique applied.

Goal: techAppro It must be demonstrated that any technique chosen to perform the activity is appropriate. The appropriateness of different techniques will depend upon the nature of the activity, but may also depend upon factors such as the standards being applied or the required integrity of the activity.

Goal: techProv It must be demonstrated that the chosen technique has the necessary provenance, this should include consideration of the extent of its application for the activity in other systems and the robustness and explicitness of its definition. The technique should also be systematic in its application and have objective acceptance criteria.

Applicability

This pattern is applicable to any technique applied when performing a process activity.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: techAppro Argument and evidence must be provided to demonstrate that the the technique chosen to use to perform the activity is appropriate.

Related Patterns

The argument created from this pattern supports the arguments created using the D-MILS process pattern.

2.11.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`technique`	The name of the technique applied applied for an activity	Process model
`provenance`	Description of the technique's provenance or reference to evidence of capability	Process model

Choices

None

Optional

None

2.11.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- {technique} is sufficiently trustworthy

Dependencies

None

Context

None

Assumptions

None

2.12 TTEthernet

2.12.1 Pattern description

Structure

The pattern structure is shown in figure 14.

Intent

This pattern is used to argue about the assurance of the TTEthernet network used as part of a D-MILS platform.

Participants

Goal: TTEgtee It must be demonstrated that the TTE network ensures that only the desired connections between nodes, as defined in the GIFP are possible. This is done by considering how the GIFP is implemented on the network, and also how the implemented configuration is enforced. And also how that configuartion is enforced by the network.

Goal: TTEconfigImp To demonstrate that the configuration is correctly implemented by the TTE hardware, the way in which the harware is configured is considered. It must also be demonstrated that once configured, the hardware cannot be changed inadvertently.

Goal: hwConfigDownload The correctness of the hardware configuration is achieved by downloading the configuration data to the node controllers and switches using an application on a central workstation.

Goal: configChange It is ensured that the configured hardware cannot be inadvertently changed by requiring switch pins to be manually set before switch configuration can be changed. Nodes and switches can only be reconfiguration if they are re-programmed.

Goal: TTEconfigCorrect It must be demonstrated that the configuration file used to configure the TTE network is correct with respect to the GIFP. This claim is supported by a claim in the implementation module about the correctness of the target specific configurations that are generated (see section 2.4).

Goal: TTEcommsConsist It is ensured that network communication is consistent with the implemented configuration through a fault tolerant design of the TTE switches and endpoint cards, by guaranteeing temporal behaviour of network frames, and by ensuring that malicious attacks on the network are mitigated.

Goal: TTEfaultTolDes It must be demonstrated that the network employs a fault tolerant design. This is done by demonstrating that faulty frames cannot be generated and guaranteeing the availability of frames.

Goal: faultIntercept An faulty frames are intercepted by a monitor. This is supported by the evidence from the design of the high integrity TTE switches and endpoint cards.

Goal: availableFrames Path redundancy is to ensure the availability of frames is demonstrated by the design of the TTE network.

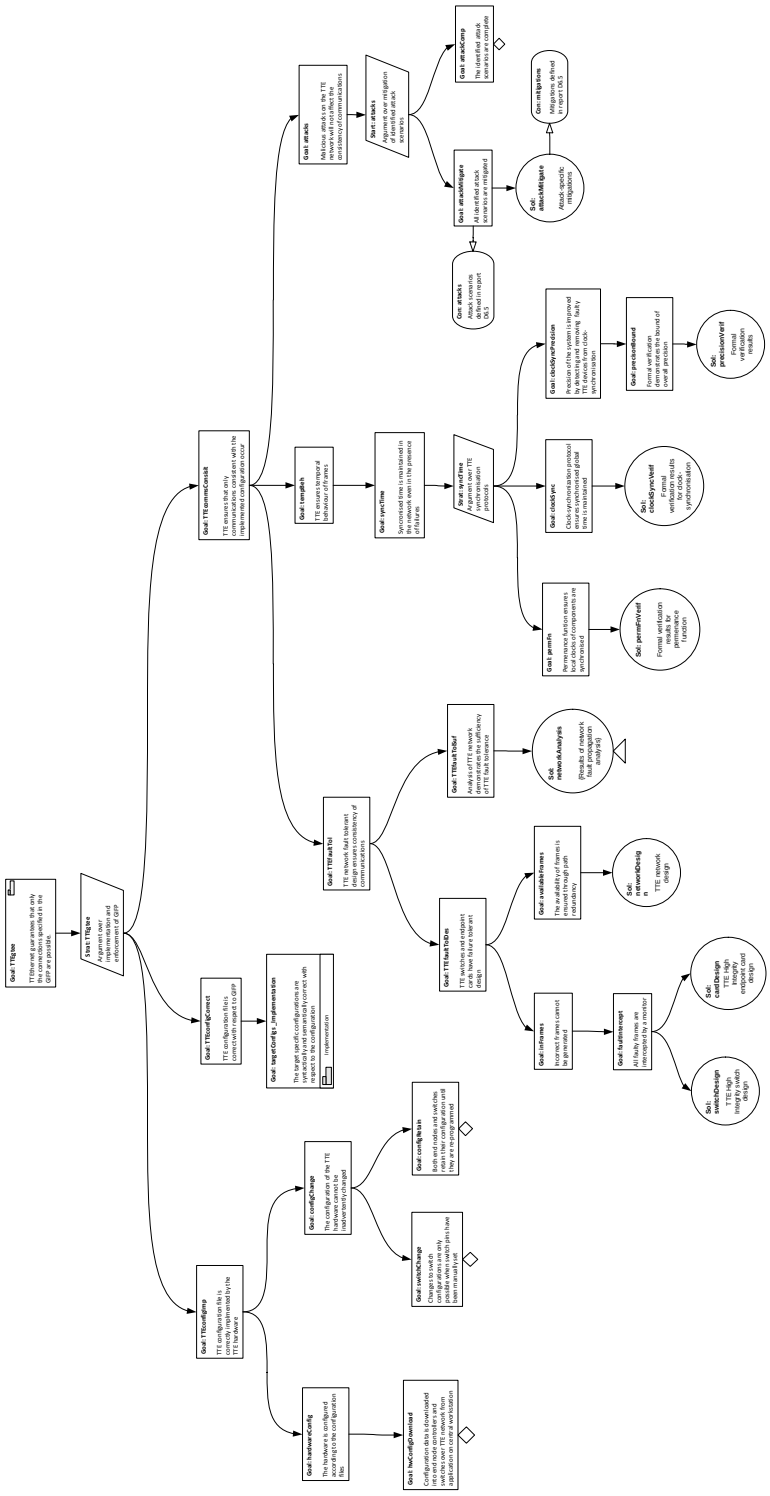

Figure 14: TTEthernet assurance case argument pattern

Goal: TTEfaultTolSuf The sufficiency of the TTE fault tolerant design is demonstrated by analysis performed on the TTE network.

Sol: networkAnalysis The results of the fault propagation analysis performed on the network are used as evidence of the sufficiency if the TTE fault tolerant design.

Goal: syncTime To ensure the temporal behaviour of network frames it must be demonstrated that even in the presence of failures, synchronised time is maintained across the network. This is done through the use of synchronisation protocols. These are a permanence function to synchronise the local clocks of components and clock-synchronization protocol to synchronise global time. In addition the precision of the synchronisation is improved by detecting and removing faulty TTE devices. All of these claims are supported by the results of formal verification.

Goal: attackMitigate The identified attack scenarios for TTE along with their mitigations are detailed in D-MILS deliverable D6-5 [4].

Applicability

This pattern is applicable to TTE networks used as part of a D-MILS platform.

Consequences

Once this pattern is instantiated, the following claims will be created that require support:

Goal: hwConfigDownload Argument and evidence must be provided to demonstrate that configuration data is successfully downloaded to nodes and switches.

Goal: switchChange Argument and evidence must be provided to demonstrate that no changes can be made to switches unless switch pins are set.

Goal: configRetain Argument and evidence must be provided to demonstrate that nodes and switches retain their configuration.

Goal: attackComp Argument and evidence must be provided to demonstrate that the identified attack scenarios are complete.

Related Patterns

The argument created from this pattern supports the argument created using the D-MILS Platform pattern and requires support from arguments created using the implementation argument pattern.

2.12.2 Pattern instantiation

When instantiating the pattern the following information regarding the target system is required. The source describes the suggested source for the required information, see section 3 for details on automatically instantiating the argument pattern with the required information.

Roles

Role	Description	Source
`Results of network fault propagation analysis`	reference to the results of the fault propagation analysis	Manual

Choices

None

Optional

None

2.12.3 Module Interface

This pattern forms a self-contained assurance case module that is intended to be used as part of a larger assurance case. This interface defines the assurance claims that this module can guarantee (to the assurance case), the dependencies that the module has on other modules (these dependencies must be assured for the guaranteed claim to be supported), and context and assumptions within whose scope the dependencies must be assured.

Guarantees

- TTEthernet guarantees that only the connections specified in the GIFP are possible.

Dependencies

- The target specific configurations are syntactically and semantically correct with respect to the configuration

Context

None

Assumptions

None

3 Automated Instantiation

In order to further facilitate the creation of assurance cases for D-MILS systems we have developed a tool that can be used to automatically generate large parts of the assurance case. For those parts of the assurance case that cannot be automatically constructed, the explicit assurance claims requiring further development are presented to the user and become assurance obligations that must be discharged in order to create a complete assurance case.

The tool that has been developed uses a model-based approach to instantiate the D-MILS assurance case patterns using information about the target system extracted directly from the system models. This section describes the operation of the D-MILS model-based assurance case (MBAC) tool, with particular focus on how the user is expected to interact with the tool to create an assurance case argument for the system. An example of how the tool is used to create an assurance case argument for an example D-MILS system is presented in section 4.

In the following sections we describe each of the elements of the tool illustrated in figure 15.

3.1 GSN Pattern Models

The GSN patterns described in this document, which define the required structure of the D-MILS assurance case argument, are taken as input to the MBAC tool. This requires that the graphical pattern structure is converted to a model of the GSN pattern captured as an XML file. We refer to the GSN model files as GSNML files. The GSN pattern model must conform to the GSN meta-model that we defined in [3]. In order to facilitate the creation of GSNML files we have created a graphical model editor that allows GSN pattern structures to be easily and intuitively developed graphically, and then exported automatically as GSNML files conformant to the GSN metamodel. The GSN graphical model editor runs as an stand-alone Eclipse application.

Since the D-MILS patterns have already been defined, and the corresponding GSNML files created, it will often be unnecessary for the system developer to use the GSN graphical editor to create pattern models. However, there may be situations where the system developer wishes to make changes to, or add more detail to, existing patterns. In such cases the graphical editor is available. This may particularly be the case for third party component developers who may wish to reflect application-specific features in the patterns.

The GSNML files must be added to the project workspace of the MBAC program.

3.2 System Models

The system models are the models of the system that contain the information necessary to instantiate the argument patterns. The pattern descriptions in section 2.5 detailed the sources of the implementation information regarding the target system. These source models are taken as input to the MBAC tool. The system models must be captured in XML files, and conform to a defined meta-model (which is used in creating the weaving model - see section 3.3). The system model XML files must be added to the project workspace of the MBAC program.

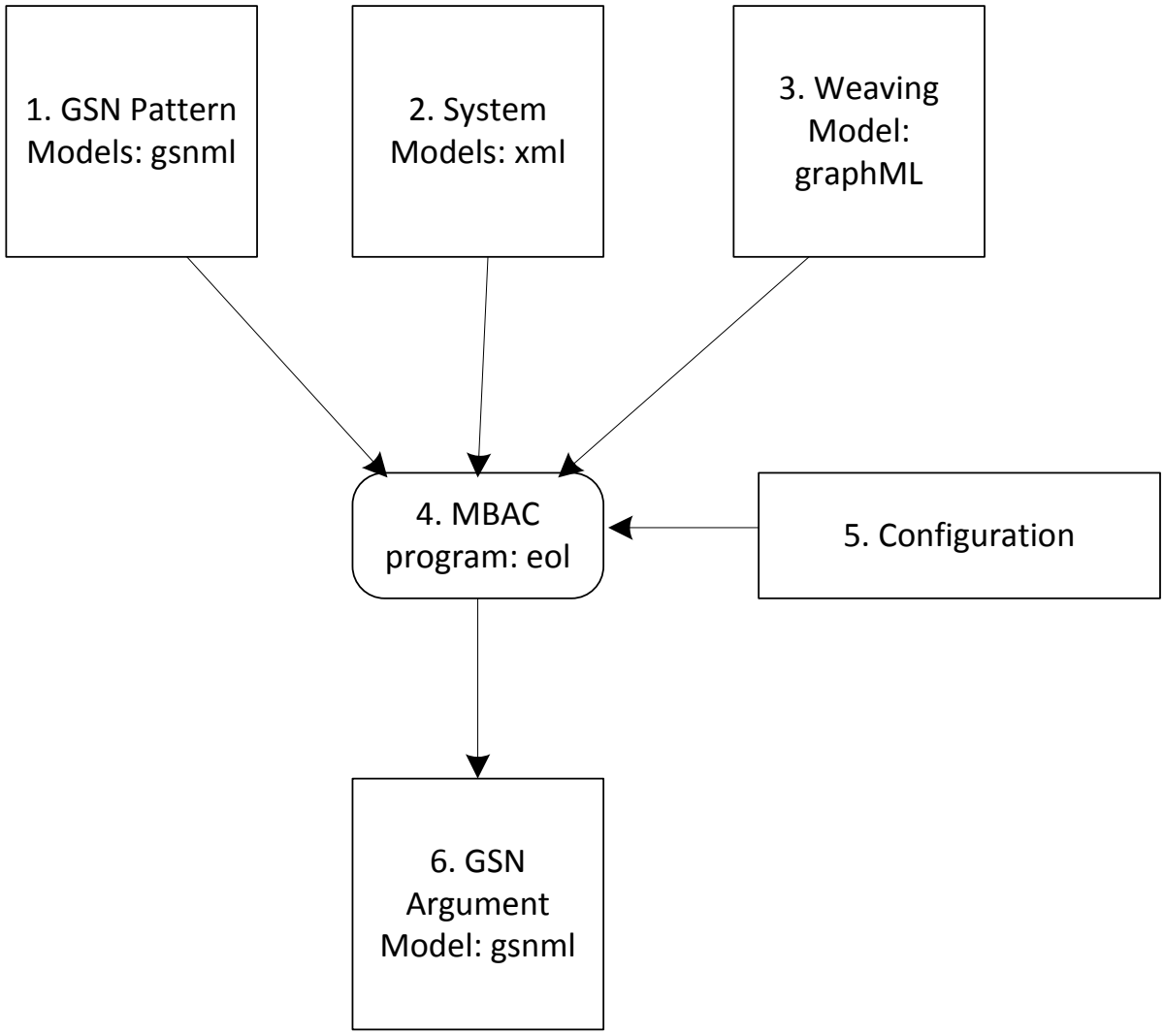

Figure 15: The D-MILS model-based assurance case tool

3.3 Weaving Model

The weaving model is a model of the dependencies that exist between the models that are taken as input by the MBAC tool. The dependencies are captured in the weaving model by specifying relationships between metamodels. This includes dependencies between elements of the GSN pattern models and elements of the system metamodels, as well as (where multiple system models are used) the dependencies between the system metamodels.

The current version of the tool uses an interim solution for creating weaving models that involves creating the weaving models graphically and importing them to the tool as graphML files. Future development of the tool will include the creation of weaving models directly from the metamodels, rather than graphically.

To create the weaving model graphML files, the open source yEd diagramming tool[1] is used. The weaving model is represented using nodes and edges with the following properties.

GSN Pattern Role Node

- Type = RoleBinding

System Metamodel Element Node

- Type = AADLmodelElement
- properties: String ElementType={metamodel element name}, String ModelName={model name}

GSN Pattern Role Mapping Edge

- Type = Mapping

System Metamodel Dependency Edge

- Type = metaDep
- properties: path={metamodel EReferences}

The weaving model created in yEd is automatically generated as a graphML file. The graphML file must be added to the project workspace of the MBAC program.

If no changes are made to the existing D-MILS patterns and only system models conforming to the same metamodels are used, then the weaving model will not need to be changed by the system developer. Any changes to the patterns or to the metamodels will require the user to use yEd to update the weaving model. Note that changes to the system models **should not** normally require changes to the weaving model.

3.4 MBAC program

The MBAC program is an Epsilon Object Language (eol) program the runs on the Eclipse platform. It is executed using a run configuration that specifies the input files. If no changes have been made to the D-MILS patterns or to the metamodels, then the system developer will only to ensure that the system models for the target system are included in the run configuration. The MBAC program is provided as an Eclipse project and requires the Epsilon plugin[2] and GraphML integration driver[3]. The initial run configuration is included within the project.

[1] The yEd Graph editor is available to download from http://www.yworks.com/en/products/yfiles/yed/
[2] http://download.eclipse.org/epsilon/interim/
[3] http://epsilonlabs.googlecode.com/svn/trunk/org.eclipse.epsilon.emc.graphml.updatesite

3.5 GSN Argument Model

The output of the MBAC tool is a GSN argument model of the assurance case argument for the target system that has been instantiated using information extracted from the system models. The argument model is generated as a GSNML file. This GSNML file can then be used to present information to the user in a number of ways. Firstly, the argument model can be represented graphically as a GSN structure. Secondly, the model can be queried in order to provide a particular view on the assurance case. For example it is possible to just select those argument elements that remain undeveloped, requiring additional support from the system developer (these can be considered to be outstanding assurance requirements). It is also possible to create an assurance case module interface from the argument model, defining the public and away goals of the assurance case module as well as the module's context and assumptions. This would be particularly useful where multiple parties are developing different assurance case modules. Finally an instantiation table can also be generated that summarises how the pattern has been instantiated in tabular form, rather than having to consult the entire argument structure.

The GSN argument model can also be used as the basis for performing verification of the assurance argument structure, as well as validation of the argument with respect to the system models. These verification and validation activities have not yet been implemented and will be explored further in the remainder of the project.

4 Starlight example

In this section we describe an example automated instantiation of the D-MILS assurance case patterns for a small example D-MILS system. The system used is the Starlight example which is described in [2]. We describe each step of the example instantiation and present a summary of the results.

4.1 GSN Pattern Models for Starlight

The GSN graphical model editor was used to create the pattern structures for the system properties and composition assurance case patterns that were described in section 2.5. The editor created GSNML files for the patterns. Figure 16 shows the part of the file created for the composition pattern to illustrate the structure of the GSNML files.

```
<?xml version="1.0" encoding="UTF-8"?>
<gsnmetamodel:Case xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
xmlns:gsnmetamodel="http://gsnmetamodel/1.0">
  <contains xsi:type="gsnmetamodel:GSN_Module" id="composition 1">
    <ArgumentElements xsi:type="gsnmetamodel:GSN_Goal" id="Goal: propSat" tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Role" role="formal property"/>
      <contents xsi:type="gsnmetamodel:Literal" literal="is satisfied in the MILS-AADL system model"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_Goal" id="Goal: formalVerif " tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Literal" literal="Formal verification proves that the MILS-AADL model satisfies"/>
      <contents xsi:type="gsnmetamodel:Role" role="formal property"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_SupportedBy" hasSource="//@contains.0/@ArgumentElements.0"
    hasTarget="//@contains.0/@ArgumentElements.1"/>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_Goal" id="Goal: verifResults" tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Literal" literal="Results of formal verification demonstrate"/>
      <contents xsi:type="gsnmetamodel:Role" role="formal property"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_SupportedBy" hasSource="//@contains.0/@ArgumentElements.1"
    hasTarget="//@contains.0/@ArgumentElements.3"/>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_Solution" id="Sol: verifResults">
      <contents xsi:type="gsnmetamodel:Role" role="formal verification results for formal property"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_SupportedBy" hasSource="//@contains.0/@ArgumentElements.3"
    hasTarget="//@contains.0/@ArgumentElements.5"/>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_ContextAsAssertion" id="Con: components" tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Literal" literal="trusted software components: "/>
      <contents xsi:type="gsnmetamodel:Role" role="trusted software components"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_InContextOf" hasSource="//@contains.0/@ArgumentElements.3"
    hasTarget="//@contains.0/@ArgumentElements.7"/>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_ContextAsAssertion" id="Con: enviroProps" tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Literal" literal="assumed environmental properties:"/>
      <contents xsi:type="gsnmetamodel:Role" role="assumed environmental properties"/>
    </ArgumentElements>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_InContextOf" hasSource="//@contains.0/@ArgumentElements.3"
    hasTarget="//@contains.0/@ArgumentElements.9"/>
    <ArgumentElements xsi:type="gsnmetamodel:GSN_ContextAsAssertion" id="Con: platformProps" tobeInstantiated="true">
      <contents xsi:type="gsnmetamodel:Literal" literal="properties of D-MILS platform:"/>
```

Figure 16: GSNML file for the composition assurance case pattern

4.2 Starlight System Model

The model used to instantiate the patterns was the MILS-AADL specification for the Starlight system. The specification is taken as input to the MBAC tool as an XML file (an extract is shown in figure 17).

The corresponding metamodel for the MILS-AADL model was created by extending the existing AADL metamodel to include the required MILS-specific elements. Figures 18 and 19 show these extensions.

4.3 Weaving Model

The weaving model used for the instantiation is shown graphically in figure 20. The rectangles are elements of the system metamodel (MILS-AADL metamodel) and the rounded nodes are role elements in the D-MILS assurance case pattern models. Two types of connector are shown:

- the solid lines represent connections that exist between elements in the MILS-AADL metamodel. These connections include a definition of a path attribute that defines the reference path from the source to the target element in the metamodel.
- the dotted lines represent dependencies between the assurance argument roles and the metamodel elements. These connections can include the definition of a constraint on the selection of model elements of the defined type.

4.4 Starlight Assurance Argument Models

Using the models defined above as input to the MBAC tool the argument files were generated for the system properties and composition assurance case modules. These outputs are described below.

4.4.1 Starlight System Properties Assurance Argument Model

The generated gsnml file for the instantiated system properties module argument is provided in Appendix A. This output is represented graphically as a GSN structure in B.

From this instantiation the following assurance claims still requiring support are identified. The satisfaction of these claims are assurance obligations on the system properties assurance case module.

- The formal properties are sufficient to address the defined D-MILS system properties
- **Assumed environmental properties:**
- ‘true’ is met
- ‘always ((cmd) implies then ((return) releases (not ((cmd or switch_to_high or switch_to_low)))))’ is met
- ‘always (((cmd) implies then ((return) releases (not ((cmd or switch_to_high or switch_to_low))))) and (((last_data(cmd) less high_bound)) implies ((not (switch_to_low)) since (switch_to_high)))’ is met

```
    <dataConnection name="display2user"
          src="//@aadlPackage.0/@aadlPublic/@systemType.1/@features/@dataPort.1"
          dst="//@aadlPackage.0/@aadlPublic/@subjectType.0/@features/@dataPort.1">
    </dataConnection>
    </connections>


  </systemImpl>

  <systemType name="starlight" implementations="//@aadlPackage.0/@aadlPublic/@systemImpl.1">
    <features>
      <dataPort name="cmd" direction="in"/>
      <eventPort name="switch_to_high" direction="in"/>
      <eventPort name="switch_to_low" direction="in"/>
      <dataPort name="return" direction="out"/>
      <dataPort name="outL" />
    </features>

    <annotations>
    <contract name="ev_functional_prop" >
     <technique name="ocra" />
     <assumption name="true" />
     <guarantee name="always ( {cmd} implies in the future {return}" />
    </contract>

    <contract name="data_functional_prop" >
     <technique name="ocra" />
     <assumption name="always ( {cmd} implies then ( {return} releases (not ( {cmd or
     switch_to_high or switch_to_low} )) ))" />
     <guarantee name="always ( {return} implies {last_data(return) =
     computation(last_data(cmd))} )" />
    </contract>

     <contract name="secure_prop" >
     <technique name="ocra" />
     <assumption name="always (
                 ( {cmd} implies then ( {return} releases (not ( {cmd or switch_to_high or
                 switch_to_low} )) ))
                 and
                 ( ( {last_data(cmd) less high_bound} ) implies ( (not {switch_to_low})
                 since {switch_to_high} )))" />
     <guarantee name="any" value="always ({outL greater high_bound})" />
     </contract>
    </annotations>
  </systemType>

  <systemImpl name="starlight.impl" compType="//@aadlPackage.0/@aadlPublic/@systemType.1">
    <subcomponents>
      <subjectSubcomponent name="dispatch"
      classifier="//@aadlPackage.0/@aadlPublic/@subjectType.1"/>
      <subjectSubcomponent name="high"
      classifier="//@aadlPackage.0/@aadlPublic/@subjectType.2"/>
      <subjectSubcomponent name="low"
      classifier="//@aadlPackage.0/@aadlPublic/@subjectType.3"/>
    </subcomponents>

    <connections>
```


Figure 17: XML file for the MILS-AADL specification of the Starlight system

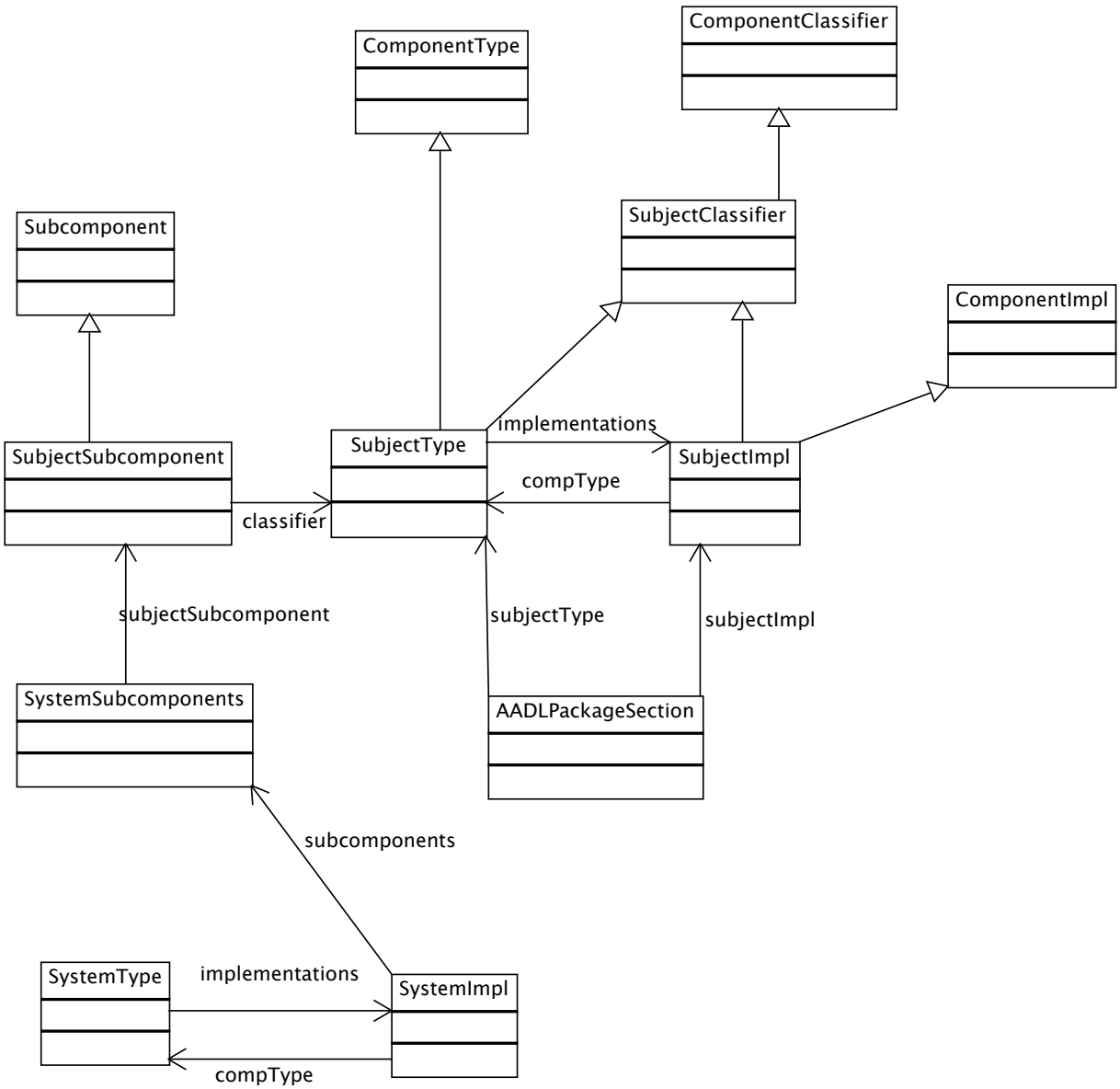

Figure 18: AADL metamodel extended to include Subjects

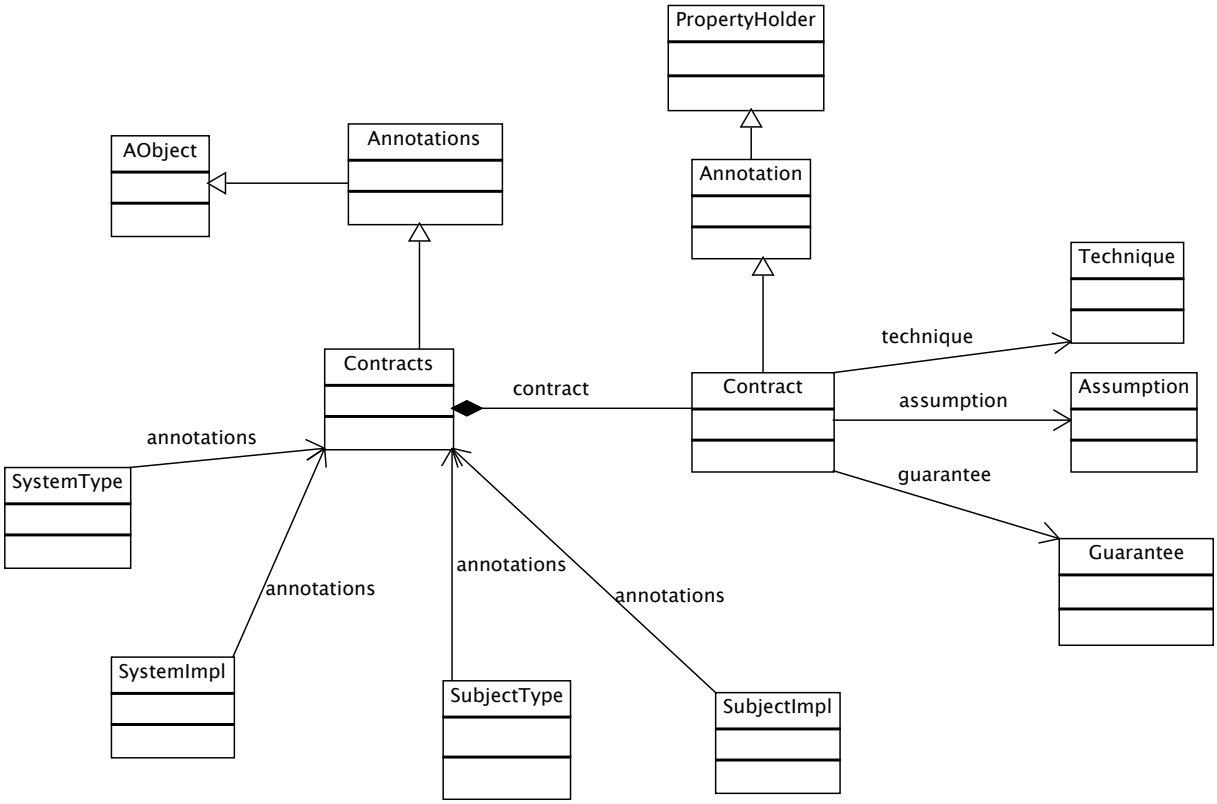

Figure 19: AADL metamodel extended to include Contract definitions

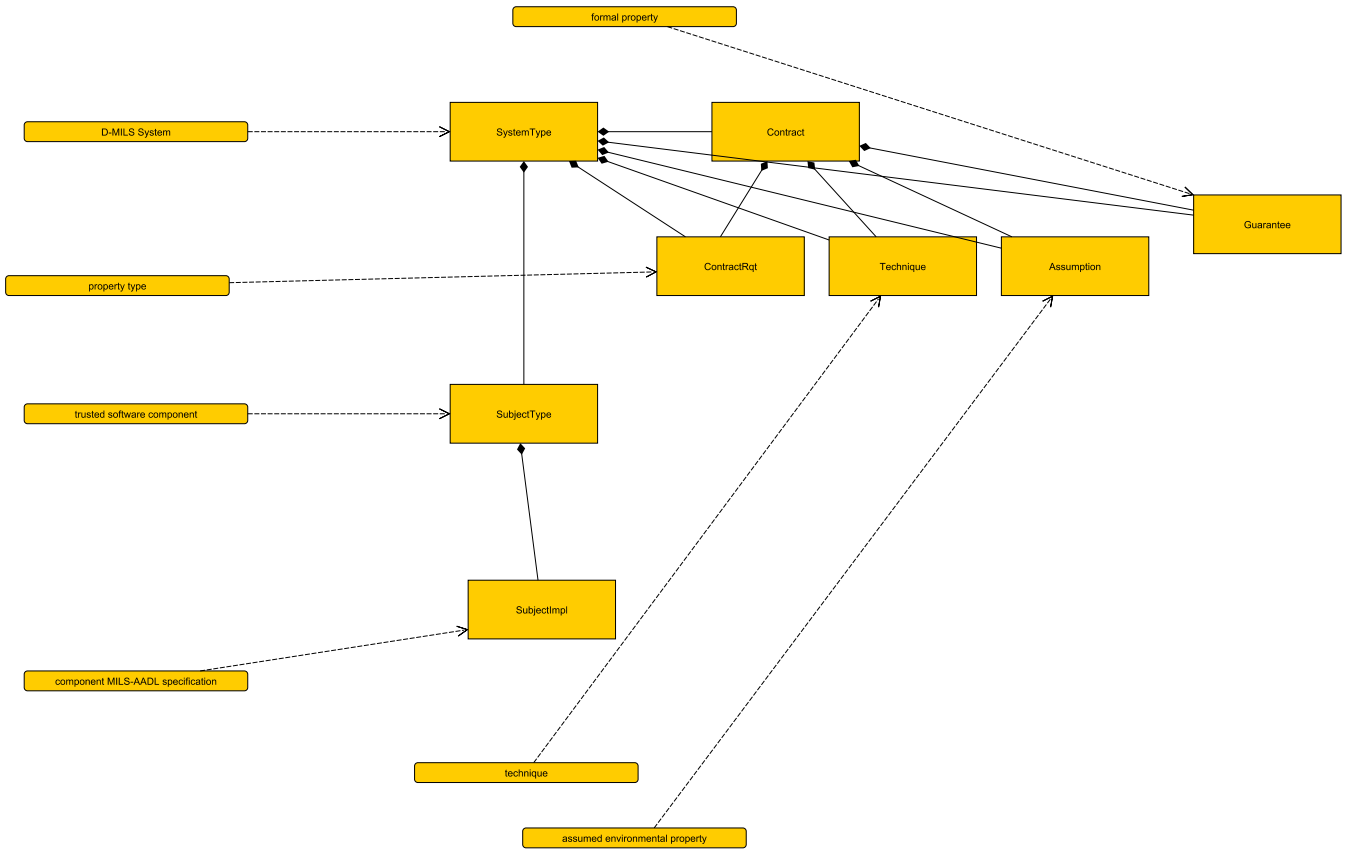

Figure 20: Graphical representation of the weaving model used to instantiate the Starlight system assurance argument

The following assurance claims requiring support in other modules are identfied. The satisfaction of these claims are assurance obligations on the relevant assurance case module.

Composition:

- 'always ((cmd) implies in the future (return)' is satisfied in the MILS-AADL system model
- 'always ((return) implies (last_data(return) = computation(last_data(cmd))))' is satisfied in the MILS-AADL system model
- 'any' is satisfied in the MILS-AADL system model

Trusted Software Components:

- Usubject correctly implements the MILS-AADL implementation specification
- Dsubject correctly implements the MILS-AADL implementation specification
- Lsubject correctly implements the MILS-AADL implementation specification
- Hsubject correctly implements the MILS-AADL implementation specification

D-MILS Platform:

- D-MILS platform guarantees required properties

4.4.2 Starlight Composition Assurance Argument Model

The generated gsnml file for the instantiated composition module argument is provided in Appendix C. This output is represented graphically as a GSN structure in D.

From this instantiation the following assurance claims still requiring support are identified. The satisfaction of these claims are assurance obligations on the system properties assurance case module.

- ocra formal model validation demonstrates it is correct w.r.t. the MILS-AADL model
- ocra formal model validation demonstrates it is correct w.r.t. the MILS-AADL model
- ocra formal model validation demonstrates it is correct w.r.t. the MILS-AADL model

Note that the instantiation has resulted in three identical claims, since in each case the same technique (ocra) has been used. These three claims correspond to a single assurance obligation since all three claims can be supported in an identical manner.

The following assurance claims requiring support in other modules are identfied. The satisfaction of these claims are assurance obligations on the relevant assurance case module.

{technique} Tool:

- ocra translation tool is sufficiently trustworthy
- ocra translation tool is sufficiently trustworthy
- ocra translation tool is sufficiently trustworthy
- ocra tool is sufficiently trustworthy
- ocra tool is sufficiently trustworthy
- ocra tool is sufficiently trustworthy

Note that the instantiation has resulted in identical claims, since only the Ocra technique is used. These claims correspond to two different assurance obligations.

A Starlight system properties GSNML model

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Argumentation>
  <ArgumentElements id="Goal: sysProps.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="required properties are enforced" xsitype="gsnmetamodel:Literal"/>
    <contents role="sys" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sysProps.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="required properties are enforced" xsitype="gsnmetamodel:Literal"/>
    <contents role="starlight" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Con: sysProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="required properties:" xsitype="gsnmetamodel:Literal"/>
    <contents role="Any command sent between a switch_to_high (or the beginning) and a switch_to_low event should
    not be visible to the low-security subject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: sysProps" hasTarget="Con: sysDescr" id="RCon: sysDescr.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: sysDescr.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="D-MILS System: defined by" xsitype="gsnmetamodel:Literal"/>
    <contents role="Starlight MILS AADL model" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Strat: sysSecurity" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
    <contents literal="Argument over each safety or security property" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Ass: sysProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="The defined required properties are complete and correct w.r.t. system threats, vulnerabilities and hazards"
    xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propEnforced.0.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is enforced" xsitype="gsnmetamodel:Literal"/>
    <contents role="Any command sent between a switch_to_high (or the beginning) and a switch_to_low event should not
    be visible to the low-security subject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Strat: compVerif" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
    <contents literal="Argument over each formal property relating to" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Con: formalProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="formal system properties relating to D-MILS system property:" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
    <contents tag="--instantiation error: No role binding specified for role:  --"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propSuff" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="The formal properties are sufficient to address the defined D-MILS system properties" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propAdd.1.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propSat_Composition" id="RGoal: propSat_Composition.1" xsitype="gsnmetamodel:GSN_Suppor
  <ArgumentElements id="Goal: propSat_Composition.1" modRef="Composition" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propSat_Composition" id="RGoal: propSat_Composition.1" xsitype="gsnmetamodel:GSN_Suppor
  <ArgumentElements id="Goal: propSat_Composition.1" modRef="Composition" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propSat_Composition" id="RGoal: propSat_Composition.1" xsitype="gsnmetamodel:GSN_Suppor
  <ArgumentElements id="Goal: propSat_Composition.1" modRef="Composition" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propsSat" id="RGoal: propsSat.1" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: propsSat.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is addressed through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propsSat" id="RGoal: propsSat.1" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: propsSat.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is addressed through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propAdd" hasTarget="Goal: propsSat" id="RGoal: propsSat.1" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: propsSat.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is addressed through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Start: SysSecurity" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
    <contents literal="Argument over the required properties of the components and the platform" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal:SwComponents" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Trusted software components correctly implement the MILS-AADL implmentation specification" xsitype="gsnmetamodel:Literal"/>
```

```
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: platProp_D-MILS PLatform" modRef="DMILS Platform" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
    <contents literal="D-MILS platform guarantees required properties" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Con: platformProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="properties of D-MILS platform:" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
    <contents tag="--instantiation error: No role binding specified for role:  --"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: envAss" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="All environmental assumptions are met" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Con: enviroProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="assumed environmental properties:" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
    <contents tag="--instantiation error: No role binding specified for role:  --"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sysImp_Implementation" modRef="Implementation" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
    <contents literal="The MILS-AADL model is faithfully implemented" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Strat: envAss" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
    <contents literal="Argument over each assumed environmental property" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sysProp.1.1" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is met" xsitype="gsnmetamodel:Literal"/>
    <contents role="true" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sysProp.1.2" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is met" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies then ( {return} releases (not ( {cmd or switch_to_high or switch_to_low} )) ))" xsitype="gsnmetamodel:Role"/
  </ArgumentElements>
  <ArgumentElements id="Goal: sysProp.1.3" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is met" xsitype="gsnmetamodel:Literal"/>
    <contents role="always (( {cmd} implies then ( {return} releases (not ( {cmd or switch_to_high or switch_to_low} )) ))
    and ( ( {last_data(cmd) less high_bound} ) implies ( (not {switch_to_low}) since {switch_to_high} )))" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Strat: swComponents" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
    <contents literal="Argument over each trusted software component" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Con: components" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="trusted software components:" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
    <contents tag="--instantiation error: No role binding specified for role:  --"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sWcompProp_Trusted Software Component.1" modRef="Trusted Software Component" toBeInstantiated="false" xsitype="gsnmetamod
    <contents literal="correctly implements the MILS-AADL implementation specification" xsitype="gsnmetamodel:Literal"/>
    <contents role="Usubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sWcompProp_Trusted Software Component.1" modRef="Trusted Software Component" toBeInstantiated="false" xsitype="gsnmetamod
    <contents literal="correctly implements the MILS-AADL implementation specification" xsitype="gsnmetamodel:Literal"/>
    <contents role="Dsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sWcompProp_Trusted Software Component.1" modRef="Trusted Software Component" toBeInstantiated="false" xsitype="gsnmetamod
    <contents literal="correctly implements the MILS-AADL implementation specification" xsitype="gsnmetamodel:Literal"/>
    <contents role="Lsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: sWcompProp_Trusted Software Component.1" modRef="Trusted Software Component" toBeInstantiated="false" xsitype="gsnmetamod
    <contents literal="correctly implements the MILS-AADL implementation specification" xsitype="gsnmetamodel:Literal"/>
    <contents role="Hsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propAdd.1.2" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propAdd.1.3" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied through the realisation of the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: sysProps" hasTarget="Con: sysProps" id="R0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements hasSource="Goal: sysProps" hasTarget="Con: sysDescr" id="R1" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements hasSource="Goal: sysProps" hasTarget="Strat: sysSecurity" id="R2" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements hasSource="Strat: sysSecurity" hasTarget="Ass: sysProps" id="R3" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements hasSource="Strat: sysSecurity" hasTarget="Goal: propEnforced" id="R4" xsitype="gsnmetamodel:GSN_SupportedBy"/>
</Argumentation>
```

B Starlight System Properties Module GSN Structure

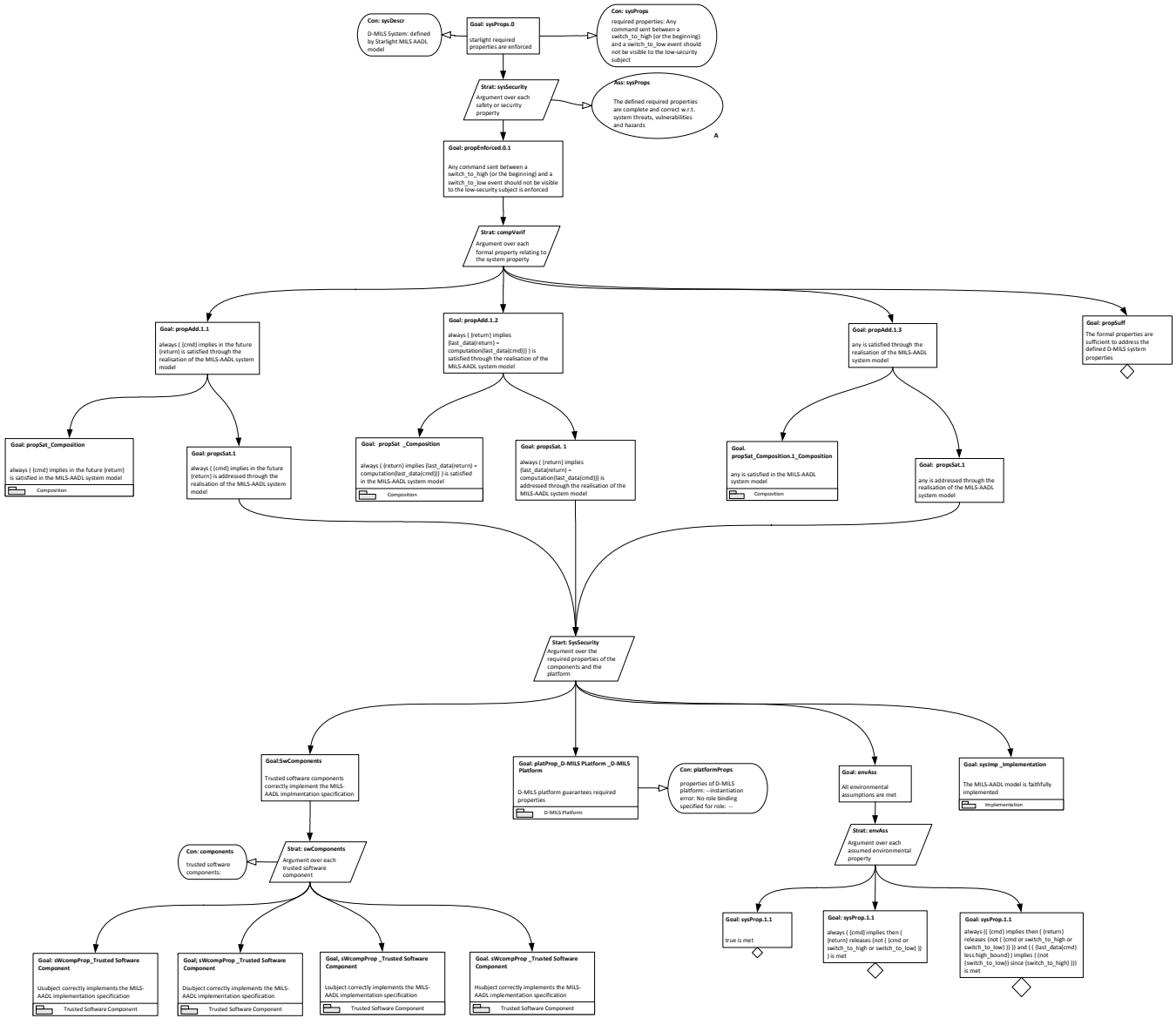

Figure 21: Starlight system properties module GSN structure

C Starlight Composition GSNML model

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Argumentation>
  <ArgumentElements id="Goal: propSat.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propSat.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Goal: propSat.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="is satisfied in the MILS-AADL system model" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propSat" hasTarget="Goal: formalVerif" id="RGoal: formalVerif.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: formalVerif.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Formal verification proves that the MILS-AADL model satisfies" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propSat" hasTarget="Goal: formalVerif" id="RGoal: formalVerif.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: formalVerif.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Formal verification proves that the MILS-AADL model satisfies" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: propSat" hasTarget="Goal: formalVerif" id="RGoal: formalVerif.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: formalVerif.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Formal verification proves that the MILS-AADL model satisfies" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: formalVerif" hasTarget="Goal: verifResults" id="RGoal: verifResults.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: verifResults.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Results of formal verification demonstrate" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies in the future {return}" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: formalVerif" hasTarget="Goal: verifResults" id="RGoal: verifResults.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: verifResults.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Results of formal verification demonstrate" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {return} implies {last_data(return) = computation(last_data(cmd))} )" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: formalVerif" hasTarget="Goal: verifResults" id="RGoal: verifResults.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
  <ArgumentElements id="Goal: verifResults.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
    <contents literal="Results of formal verification demonstrate" xsitype="gsnmetamodel:Literal"/>
    <contents role="any" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements id="Sol: verifResults" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Solution">
    <contents literal="null" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: components" id="RCon: components.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: components.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="trusted software component:" xsitype="gsnmetamodel:Literal"/>
    <contents role="Usubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: components" id="RCon: components.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: components.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="trusted software component:" xsitype="gsnmetamodel:Literal"/>
    <contents role="Dsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: components" id="RCon: components.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: components.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="trusted software component:" xsitype="gsnmetamodel:Literal"/>
    <contents role="Lsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: components" id="RCon: components.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: components.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="trusted software component:" xsitype="gsnmetamodel:Literal"/>
    <contents role="Hsubject" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: enviroProps" id="RCon: enviroProps.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: enviroProps.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="assumed environmental properties:" xsitype="gsnmetamodel:Literal"/>
    <contents role="true" xsitype="gsnmetamodel:Role"/>
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: enviroProps" id="RCon: enviroProps.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: enviroProps.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="assumed environmental properties:" xsitype="gsnmetamodel:Literal"/>
    <contents role="always ( {cmd} implies then ( {return} releases (not ( {cmd or switch_to_high or switch_to_low} )) ))" xsitype="gsnmetamodel:Role"/
  </ArgumentElements>
  <ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: enviroProps" id="RCon: enviroProps.0" xsitype="gsnmetamodel:GSN_InContextOf"/>
  <ArgumentElements id="Con: enviroProps.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="assumed environmental properties:" xsitype="gsnmetamodel:Literal"/>
    <contents role="always (                    ( {cmd} implies then ( {return} releases (not ( {cmd or switch_to_high or switch_to_low} )) ))
  </ArgumentElements>
  <ArgumentElements id="Con: platformProps" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
    <contents literal="properties of D-MILS platform:" xsitype="gsnmetamodel:Literal"/>
    <contents role="" xsitype="gsnmetamodel:Role"/>
    <contents tag="--instantiation error: No role binding specified for role:  --"/>
```

```
</ArgumentElements>
<ArgumentElements id="Goal: formalConf" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="There is sufficient confidence in the formal verification results" xsitype="gsnmetamodel:Literal"/>
  <contents role="" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements hasSource="Goal: formalConf" hasTarget="Goal: verification" id="RGoal: verification.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements id="Goal: verification.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="gives trustworthy results" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements hasSource="Goal: formalConf" hasTarget="Goal: verification" id="RGoal: verification.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements id="Goal: verification.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="gives trustworthy results" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements hasSource="Goal: formalConf" hasTarget="Goal: verification" id="RGoal: verification.0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements id="Goal: verification.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="gives trustworthy results" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: verifTool_technique Tool.0.1" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: verifTool_technique Tool.0.2" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: verifTool_technique Tool.0.3" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: formalModel.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model is a correct representation of the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: formalModel.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model is a correct representation of the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: formalModel.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model is a correct representation of the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Con: formalModel" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_ContextAsReference">
  <contents literal="formal model:" xsitype="gsnmetamodel:Literal"/>
  <contents role="" xsitype="gsnmetamodel:Role"/>
  <contents tag="--instantiation error: No role binding specified for role:  --"/>
</ArgumentElements>
<ArgumentElements id="Strat: formalModel" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Strategy">
  <contents literal="Argument over the translation from MILS-AADL to the formal representation" xsitype="gsnmetamodel:Literal"/>
  <contents role="" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: validate.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model demonstrates it is correct w.r.t. the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: validate.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model demonstrates it is correct w.r.t. the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: validate.0" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_Goal">
  <contents literal="formal model demonstrates it is correct w.r.t. the MILS-AADL model" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: transTool_technique Tool.0" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="translation tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: transTool_technique Tool.0" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="translation tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements id="Goal: transTool_technique Tool.0" modRef="technique Tool" toBeInstantiated="false" xsitype="gsnmetamodel:GSN_AwayGoal">
  <contents literal="translation tool is sufficiently trustworthy" xsitype="gsnmetamodel:Literal"/>
  <contents role="ocra" xsitype="gsnmetamodel:Role"/>
</ArgumentElements>
<ArgumentElements hasSource="Goal: propSat" hasTarget="Goal: formalVerif" id="R0" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: formalVerif" hasTarget="Goal: verifResults" id="R1" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: verifResults" hasTarget="Sol: verifResults" id="R2" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: components" id="R3" xsitype="gsnmetamodel:GSN_InContextOf"/>
<ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: enviroProps" id="R4" xsitype="gsnmetamodel:GSN_InContextOf"/>
<ArgumentElements hasSource="Goal: verifResults" hasTarget="Con: platformProps" id="R5" xsitype="gsnmetamodel:GSN_InContextOf"/>
<ArgumentElements hasSource="Goal: formalVerif" hasTarget="Goal: formalConf" id="R6" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: formalConf" hasTarget="Goal: verification" id="R7" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: verification" hasTarget="Goal: verifTool_technique Tool" id="R8" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: formalConf" hasTarget="Goal: formalModel" id="R9" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Goal: formalModel" hasTarget="Con: formalModel" id="R10" xsitype="gsnmetamodel:GSN_InContextOf"/>
<ArgumentElements hasSource="Goal: formalModel" hasTarget="Strat: formalModel" id="R11" xsitype="gsnmetamodel:GSN_SupportedBy"/>
<ArgumentElements hasSource="Strat: formalModel" hasTarget="Goal: validate" id="R12" xsitype="gsnmetamodel:GSN_SupportedBy"/>
```

```
  <ArgumentElements hasSource="Strat: formalModel" hasTarget="Goal: transTool_technique Tool" id="R13" xsitype="gsnmetamodel:GSN_SupportedBy"/>
</Argumentation>
```

D Starlight Composition Module GSN Structure

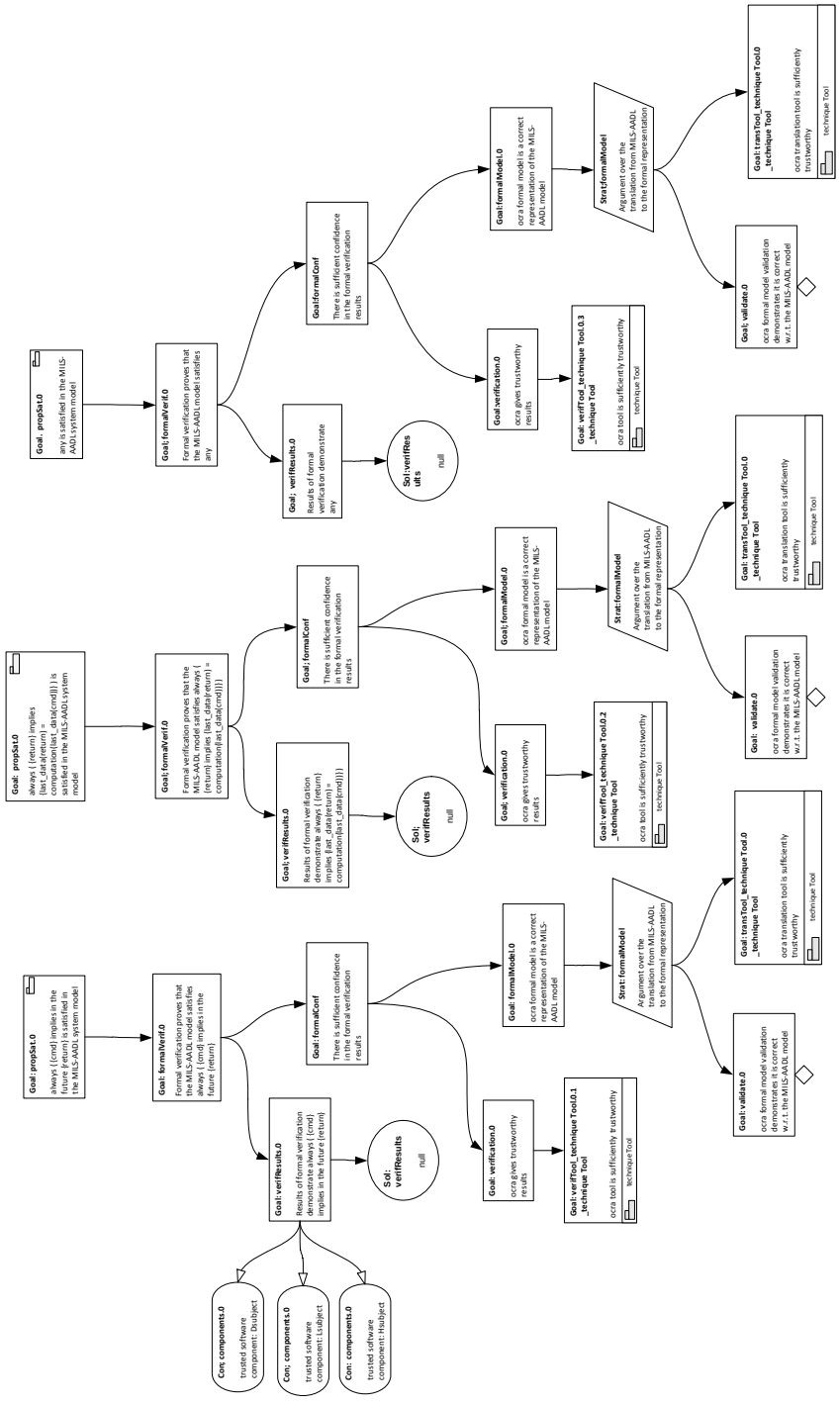

Figure 22: Starlight composition module GSN structure

Acronyms

MCNF MILS configuration-normal-form 19

MNS MILS Network Subsystem 22

References

[1] GSN community standard. Technical report, Origin Consulting (York) Limited, 2011.

[2] Translation of MILS-AADL into Formal Architectural Modeling Framework. Technical Report D2.2, Version 1.2, D-MILS Project, February 2014. http://www.d-mils.org/page/results.

[3] Compositional assurance cases and arguments for distributed MILS. Technical Report D4.2, Version 1.0, D-MILS Project, April 2014. http://www.d-mils.org/page/results.

[4] Security and distributed MILS TTEthernet. Technical Report D6.5, Version 1, D-MILS Project, 2015. http://www.d-mils.org/page/results.