Formal Design Analysis Framework: an Aspect-Oriented Architectural Framework

Dissertation Defense
Lirong Dai

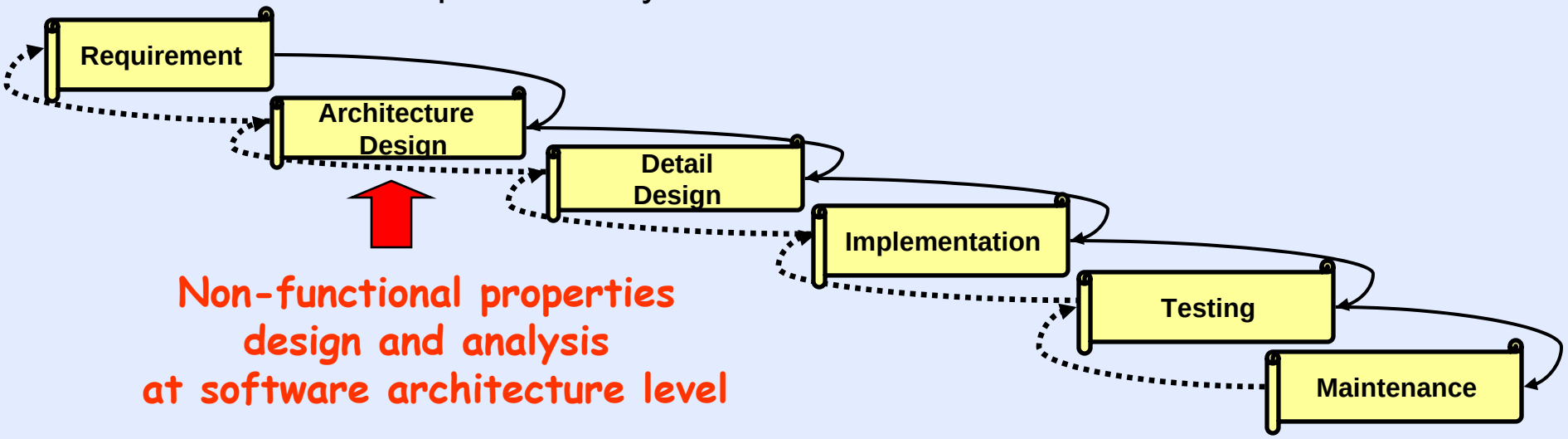
Ph.D. Supervisory Committee: Dr. Kendra Cooper (Supervisor)
Dr. Lawrence Chung
Dr. Gopal Gupta
Dr. I-Ling Yen
The University of Texas at Dallas, USA
October 19, 2005

Outline

- Motivation
- Research Problem
- Related Work
- Solution: Formal Design Analysis Framework (FDAF)
- FDAF Approach Illustrations
- Summary
- Contributions to Knowledge
- Conclusions and Future Work

Motivation

- Software development goal:
  - Deliver complete, consistent, and correct products with low time and cost
- Challenges presented:
  - Systems continue to increase in size and complexity
  - Realizing non-functional requirements
    - Functional requirements: functions and tasks a system must support
    - Non-functional requirements: constraints on these functions and tasks, important factor to decide a system's success (e.g., performance, security, reliability, etc.)
- Investigation on software development methodologies
  - Software development life cycle



Motivation

- Software architecture
  - Structure of a system, including: *software elements, elements' external visible properties, relationships among elements* [Bass L.]
  - Represents the earliest design decisions, hardest to change later, critical to be right
  - Strongly impacts final product's quality, "The right architecture paves the way for system success", [Shaw & Garlan]
- Architectural design of non-functional properties
  - Realizes a system's non-functional requirements
- Architectural analysis of non-functional properties
  - Enforces an assessment step at an early stage to discover and fix defects
  - Software design flaws cost the economy an estimated $59.5 billion annually [NIST]

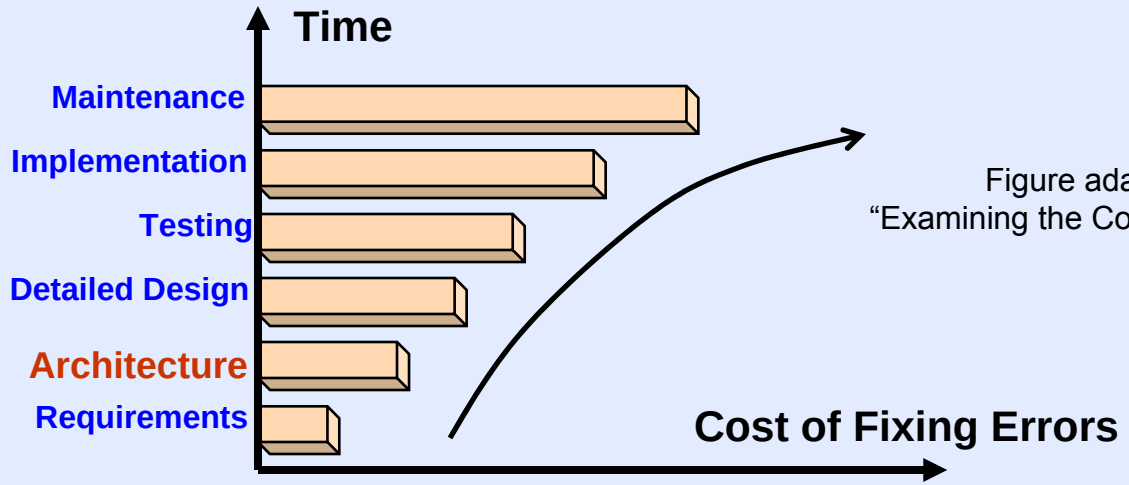


Figure adapted from: Scott W. Ambler,
"Examining the Cost of Change", Agile Modeling Essay

Motivation

- An approach supports the effective design and analysis of non-functional properties for software architecture is needed:
  - Improve the readability and understandability of the design
    - Visual modeling is helpful
      - ✓ Unified Modeling Language (UML) is becoming a de-facto standard
      - ✗ Semi-formal, no automated analysis supported
  - Provide automated analysis to improve the quality of the design
    - Accurate, reliable analysis results, reduce the effort/human error/ cost
      - ✓ Formal methods support this, e.g., Architectural Description Languages (ADLs), Promela
      - ✗ Difficult to read, write, understand, and modify designs
  - Support the maintainability and evolvability of the design
    - Many sources of change, maintenance accounts for 50% of total development cost
      - ✓ Aspect-oriented paradigm supports this
        - Adaptation of aspect-oriented programming principles to the design phase
        - Tangling concerns encapsulated in *aspects*
        - Aspect added one at a time, each aspect model can be developed independently, reduce maintenance time and cost when change happens
        - Weaving process to generate a complete design if necessary

Research Problem

- Represent non-functional properties in a software architecture
- Model crosscutting non-functional properties in UML
- Automatically analyze a UML based architecture design using existing formal methods
- Support the maintainability and evolvability of an architecture design with non-functional properties

Related Work

- Non-functional property design and analysis work in the literature
  - Focuses on performance and security
  - Approaches classified as: Semi-formal, Formal, Integrated semi-formal and formal, Aspect-oriented
- Comparisons of semi-formal approaches with the FDAF

Approaches / Criteria	UML-HPM Framework	UML Performance Modeling F.	UML-Ψ Performance Simulator	UML-MAC Framework	SecureUML	SMASC	FDAF
NFP	Performance	Performance	Performance	Security	Security	Security	Performance Security
NFP Architectural Representation	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Supported
Modeling Notations	UML	UML	UML	UML	UML	UML	UML, formal languages
Analysis	Not supported	Manual analysis	Automated simulation	Not supported	Not supported	Not supported	Automated analysis
Maintainability and evolvability of Design	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Supported using aspect-oriented paradigm
Activity Level	Architecture	Architecture	Architecture	Detailed design	Architecture	Detailed design	Architecture

Related Work

- Comparisons of formal approaches with the FDAF

Approaches / Criteria	QoS-GSPN Model	PEPA Nets	Software Architecture Model	Multi Level Security Architecture	Security Check	CVS-Server Security Architecture	FDAF
NFP	Performance	Performance	Security	Security	Security	Security	Performance Security
NFP Architectural Representation	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Supported
Modeling Notations	Petri Nets	Colored Stochastic Petri Nets	Petri Nets, Temporal Logic	Alloy	Discrete Time Labeled Transition System	Z	UML, formal languages
Analysis	Automated simulation	Automated analysis	Not supported	Automated analysis	Automated analysis	Automated analysis	Automated analysis
Maintainability and Envolvability of Design	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	Supported using aspect-oriented paradigm
Activity Level	Detailed design	Detailed design	Architecture	Architecture	Detailed design	Architecture	Architecture

Related Work

- Comparisons of integrated semi-formal and formal approaches with the FDAF

Approaches / Criteria	UML/ Æmilia ADL Approach	UML/ Petri Nets Approach	UML/ Theorem Prover Approach	UML/Promela Approach	FDAF
NFR	Performance	Performance	Security	Security	Performance Security
NFP Architectural Representation	Not supported	Not supported	Not supported	Not supported	Supported
Modeling Notations	UML, Æmilia	UML, Petri Nets	UML, First-Order Logic	UML, Liner Time Temporal Logic	UML, formal languages including Rapide, Armani, Æmilia Promela, Alloy
Analysis	Automated analysis	Automated analysis	Automated analysis	Automated analysis	Automated analysis
Automated Translation from UML to Formal	Not supported	Not supported	Supported	Supported	Supported
Maintainability and Evolvability of Design	Not supported	Not supported	Not supported	Not supported	Supported using aspect-oriented paradigm
Activity Level	Architecture	Detailed Design	Detailed Design	Requirement	Architecture

Related Work

- Comparisons of aspect-oriented approaches with the FDAF

Approaches / Criteria	COMQUAD Component Model	Re-QoS Management Method	Aspect-Oriented Secure Application	FDAF
NFP	No specific NFP selected, general approach	No specific NFP selected, general approach	Security	Performance Security
NFP Architectural Representation	Not supported	Not supported	Not supported	Supported
Modeling Notations	UML	Not supported	UML	UML, formal languages
Analysis	Not supported	Not supported	Not supported	Automated analysis
Maintainability and Evolvability of Design	Aspect-oriented approach used, but did not address how to use the approach to support this	Aspect-oriented approach used, but did not address how to use the approach to support this	Aspect-oriented approach used, but did not address how to use the approach to support this	Supported using aspect-oriented paradigm
Activity Level	Detailed design	Implementation	Implementation	Architecture

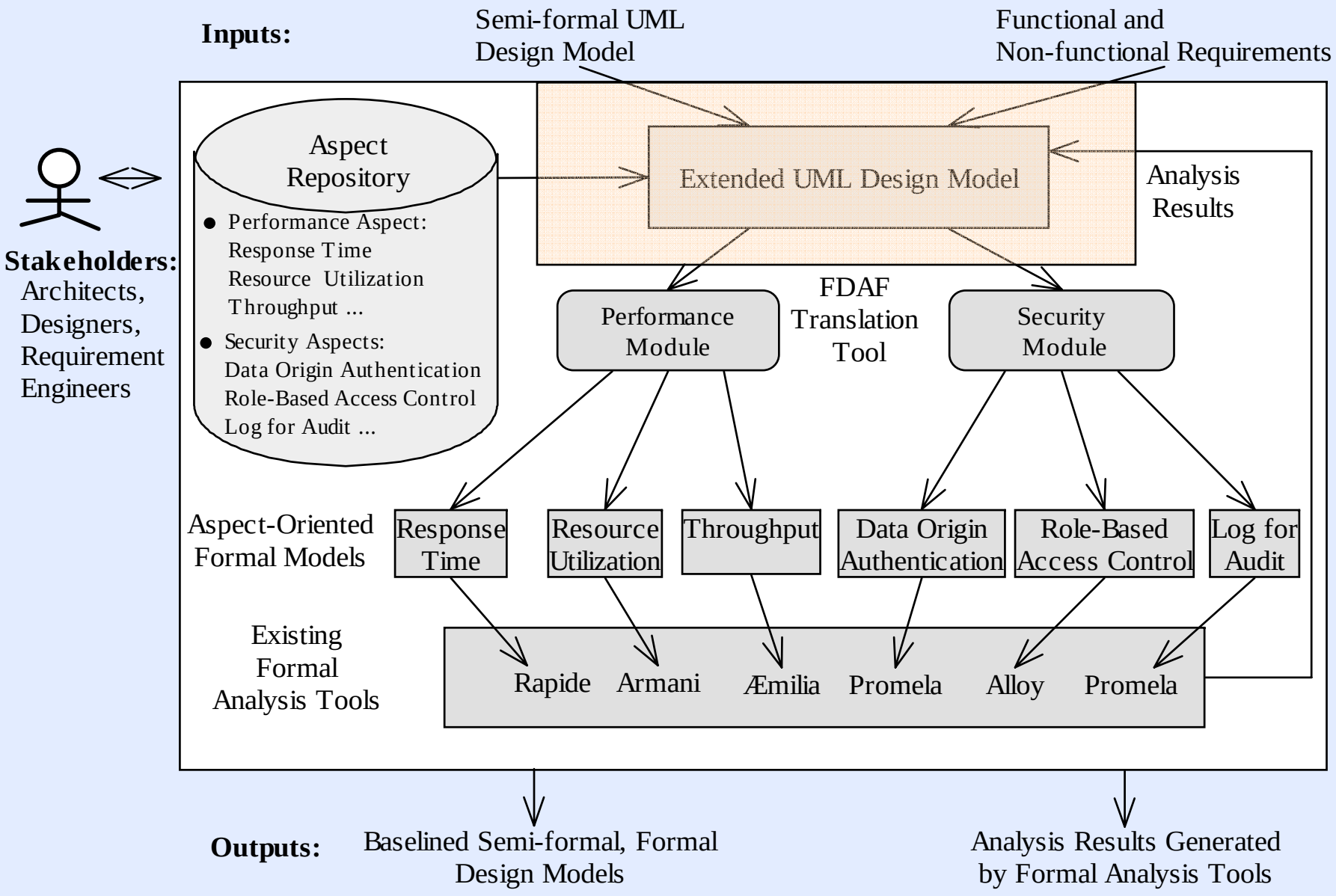
Solution:
Formal Design Analysis Framework

- FDAF solutions to the research problem:
  - Represent non-functional properties in software architecture
    - Aspects, definitions provided for aspects with guidance on using aspects
  - Model non-functional properties in UML
    - A UML extension is proposed, defined with syntax and semantics
  - Automatically analyze a UML based architecture design using existing formal analysis tools
    - Automated formalizing (part of) UML into a set of formal languages
  - Support maintainability and evolvability of an architecture design with non-functional properties
    - Aspect-oriented paradigm

Formal Design Analysis Framework

- Overview
- FDAF Aspects
- UML extension for aspect modeling
- Automated analysis of UML aspect architecture design
- Validation
- Case study: Domain Name System (DNS)

FDAF Overview

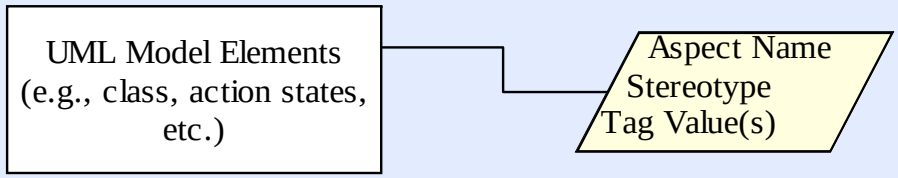


FDAF Aspects

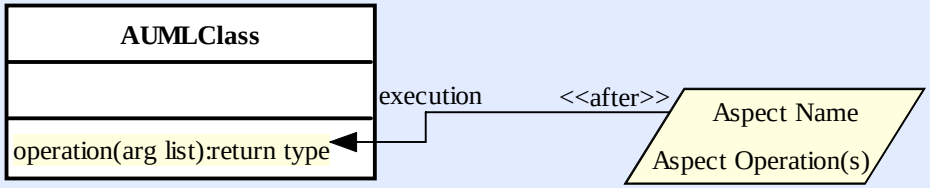
- Property aspects
  - Describe properties of systems using certain format of data (e.g., numerical values)
  - Do not trace to specific code modules
  - Defined using UML stereotypes with tags
  - E.g., performance aspects, including response time, resource utilization, throughput, etc.
- Substantive aspects
  - Provide additional capabilities
  - Constructed in code
  - Defined using UML class diagram (static view) and sequence diagram (dynamic view)
  - E.g., security aspects, including data origin authentication, role-based access control, log for audit, etc.

FDAF UML Extension for Aspect Modeling

- Introduce a new graphical icon, a parallelogram notation, to present aspects in a UML design
- Two modeling mechanism for property aspects and substantive aspects
  - Property aspect modeling mechanism
    - Annotates a property aspect to a UML model element
    - Weaving (concrete syntax):

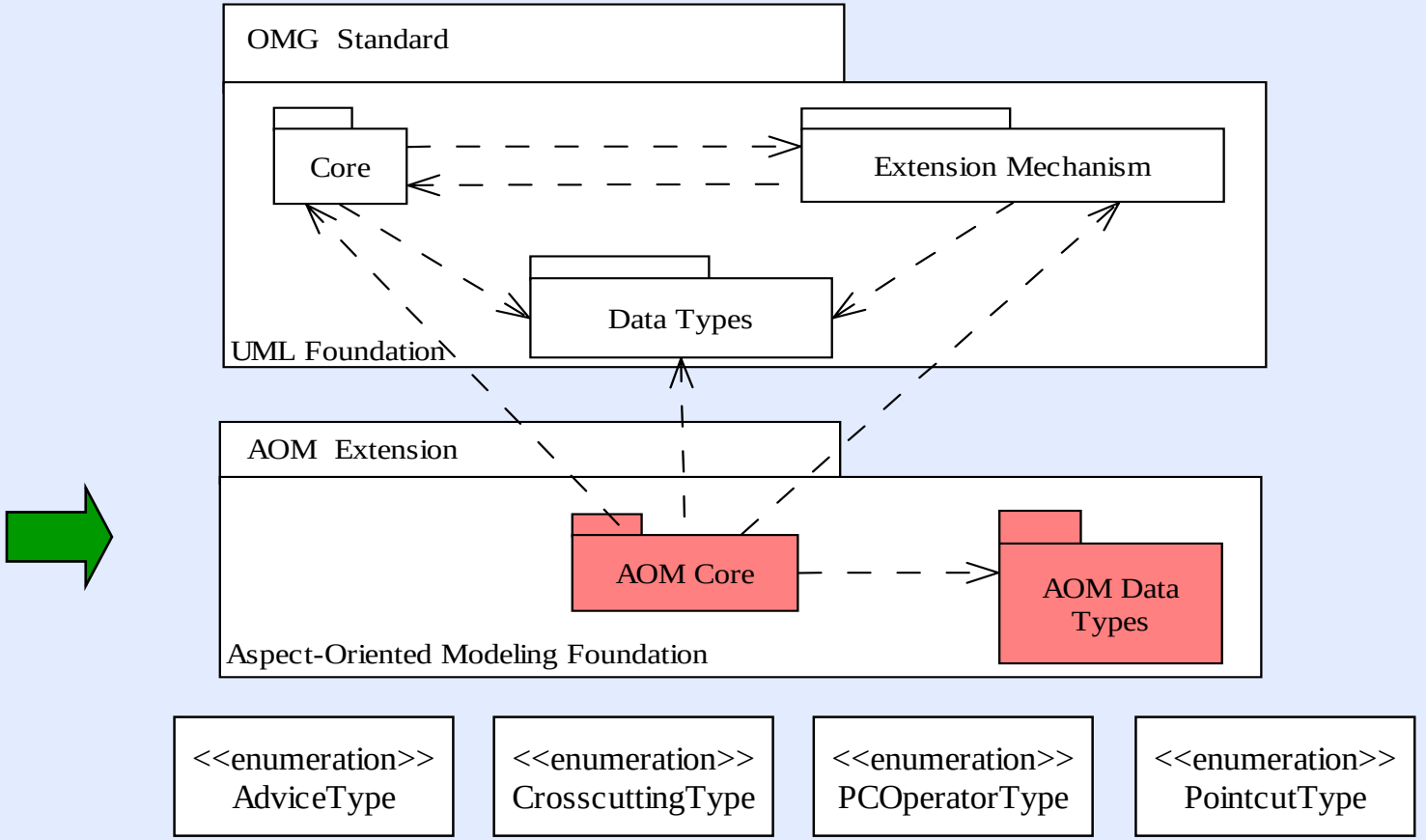


  - Substantive aspect modeling mechanism
    - Generates an aspect-oriented design that can be mapped into aspect-oriented implementation, e.g., AspectJ
    - Abstracts aspect concepts join point, advice etc., up to design level with syntax and semantics (UML action semantics and OCL)
    - Weaving (concrete syntax):



High-Level Package View of the UML Extension for Aspect-Oriented Modeling

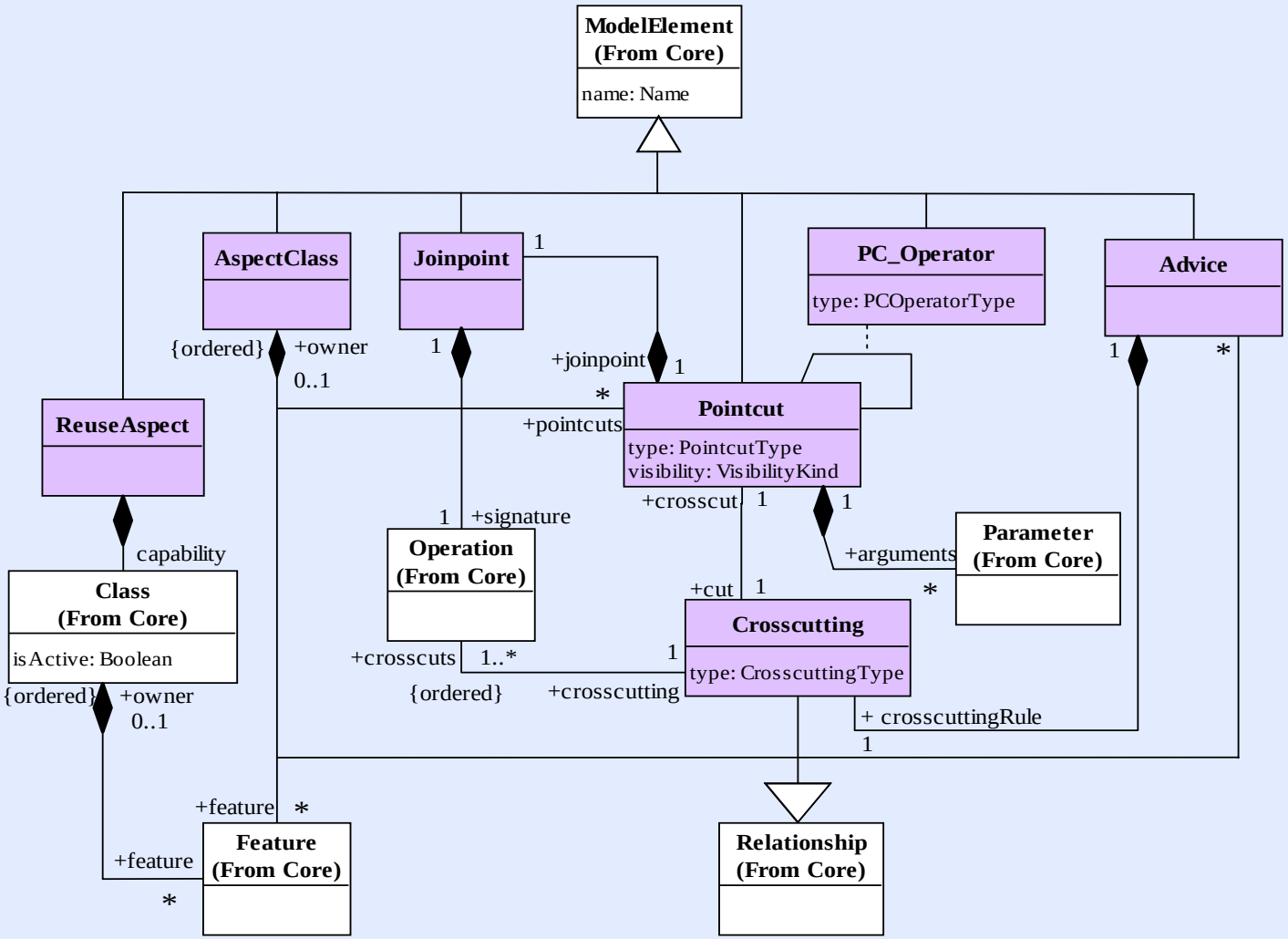
(Substantive Aspects)



AOM Data Types Package

Aspect-Oriented Modeling Core Package

(Substantive Aspects)



Abstract Syntax

FDAF Automated Analysis of UML Aspect Architecture Design

- Architecture design defined in UML class diagram and UML swim lane activity diagram, then extended with aspects
- Formalizing this part of UML into formal languages using translational semantic approach [Meyer]
  - Algorithm defined and verified with correctness
  - Algorithm implemented for the automated translation
  - Using existing formal analysis tools
  - Supported formalization include:
    - Rapide architecture description language
    - Armani architecture description language
    - Æmilia architecture description language
    - Promela
    - Alloy

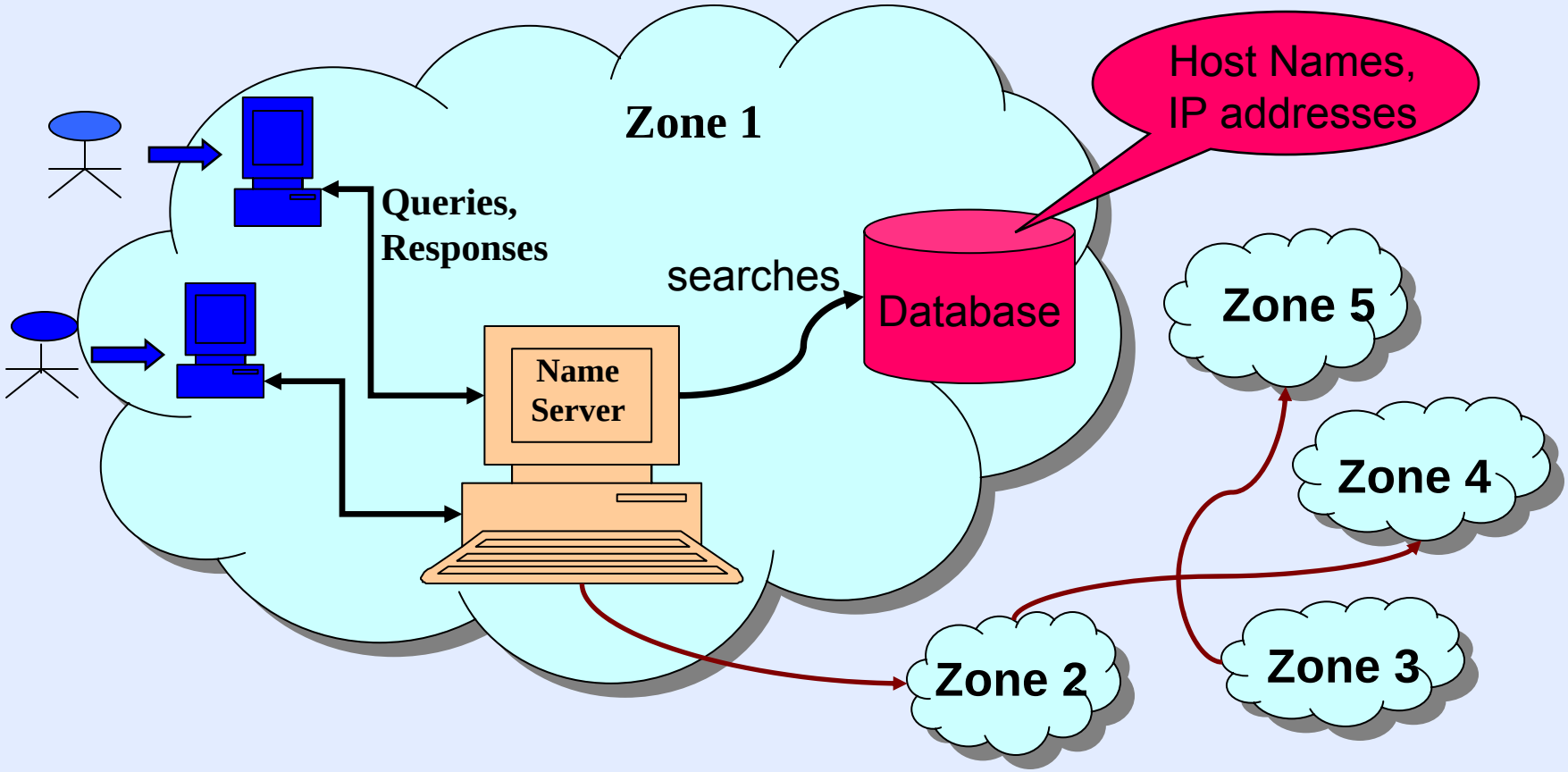
FADF Validation

- All translation algorithms have been proven with correctness
- FDAF tool support
  - Extended on open source ArgoUML
    - Implementation language Java
    - Extension: around 110 Java classes, 9250 line source code
  - Extension include:
    - Aspect repository with a set of predefined aspects
    - Weaving an aspect into a design
    - Automatically generate Rapide, Armani, Æmilia, Promela, Alloy for an extended UML architecture design
    - Interfaces with formal analysis tools
  - Source code available upon request

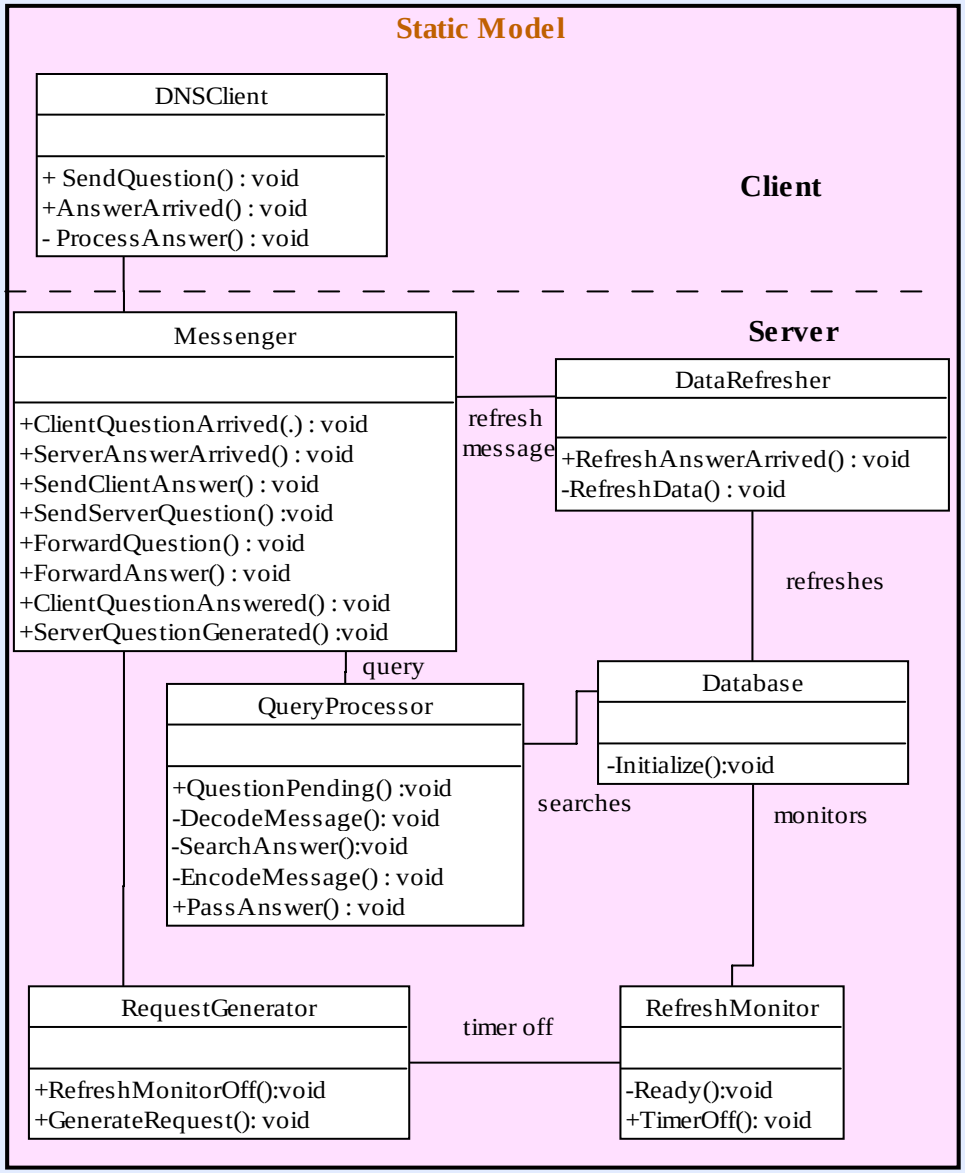
FDAF Case Study: Domain Name System

DNS Functionalities:

1. Convert host name www.cnn.com to IP address “207.25.71.28”
2. Convert IP address “207.25.71.28” to host name www.cnn.com



DNS Architecture Static Model



DNS Architecture Dynamic Model

Database:

DataRefresher:

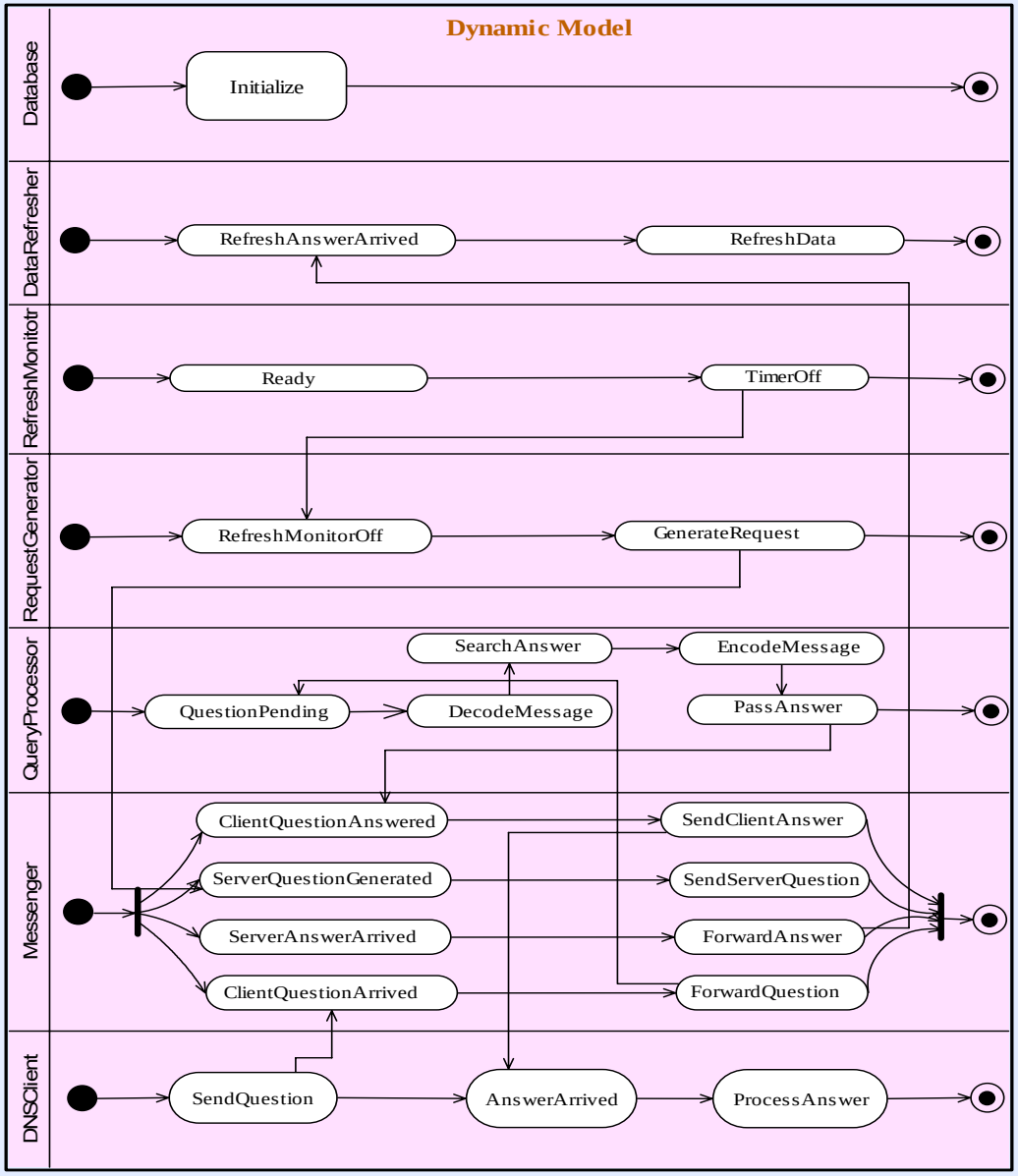
RefreshMonitor:

RequestGenerator:

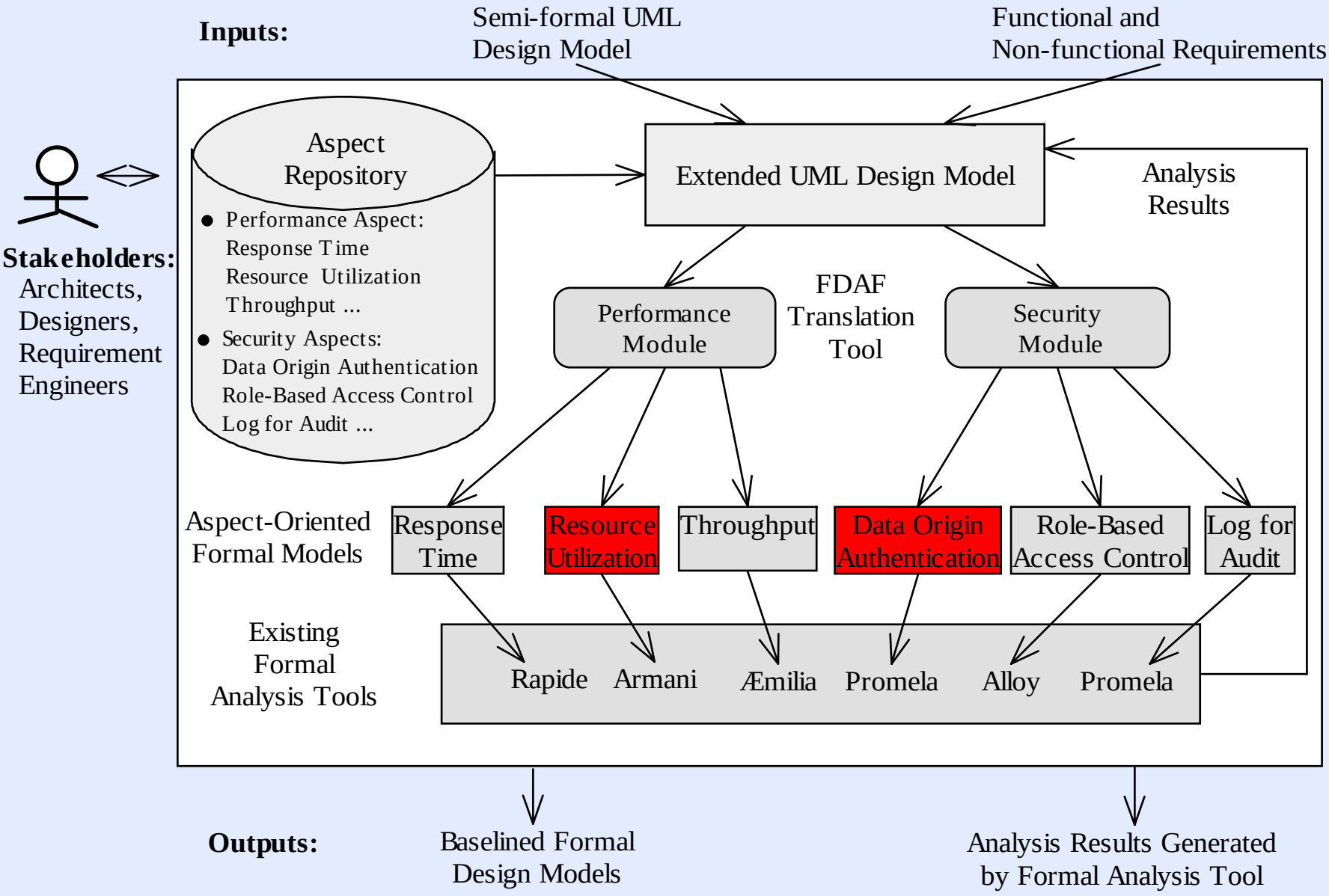
QueryProcessor:

Messenger:

DNSClient:

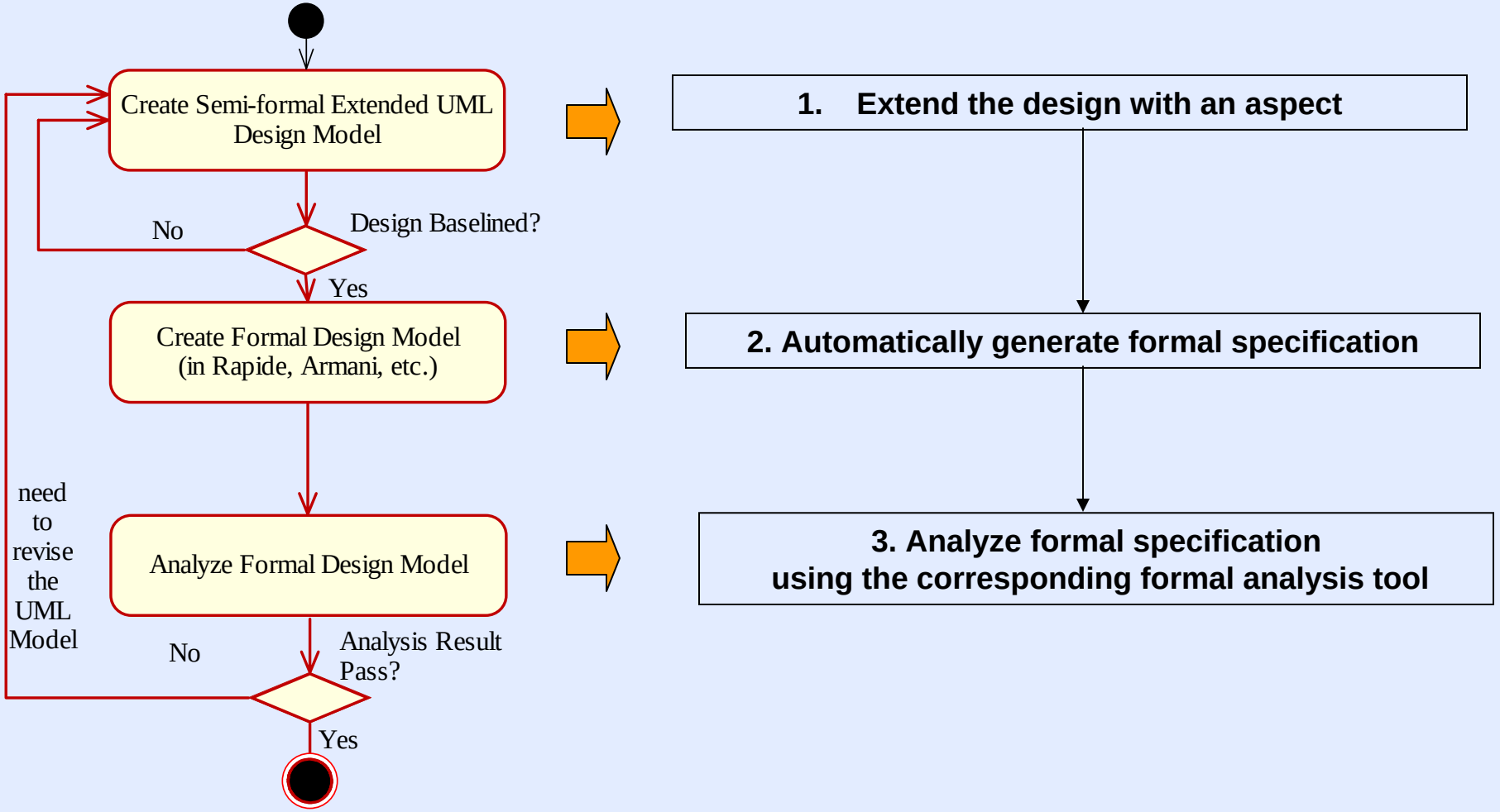


FDAF Approach Illustration Overview



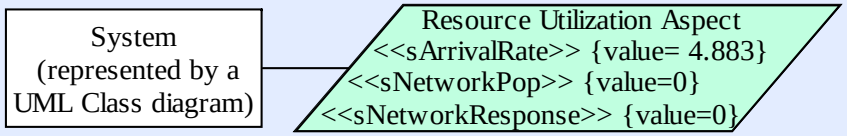
FDAF Approach Illustrations

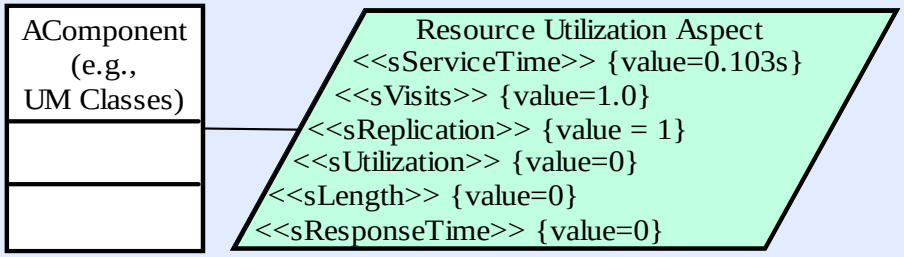
(Definition, design and analysis of supported aspects)

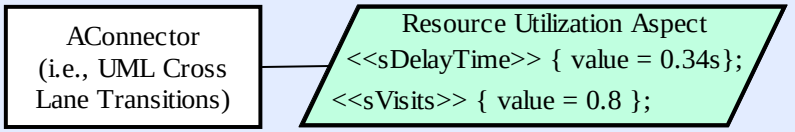


FDAF Approach Illustration: Resource Utilization Performance Aspect

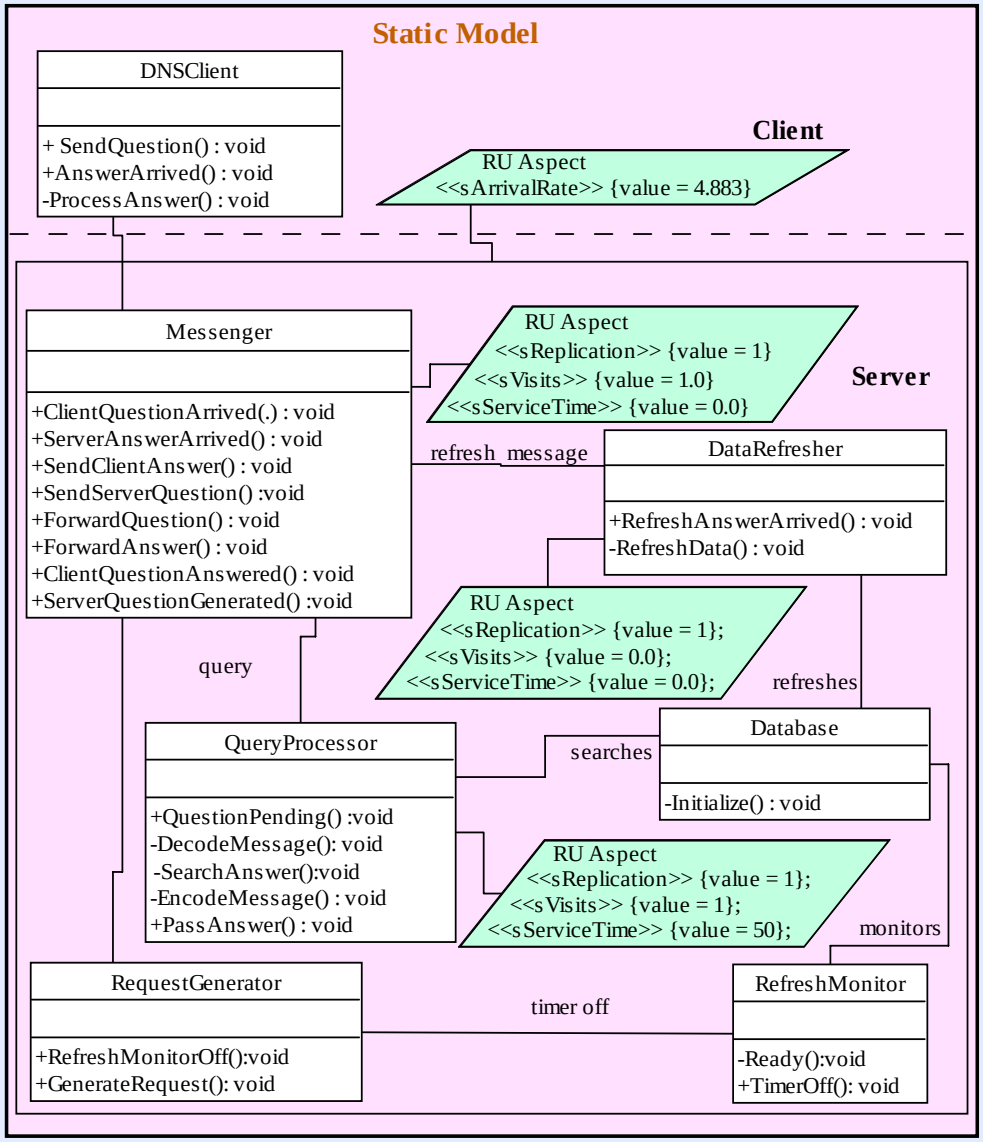
- Property aspect
- Aspect definition
  - Resource utilization aspect for systems
    - <<sArrivalRate>>, arrival rate of jobs
    - <<sNetworkPop>>, mean numbers of jobs
    - <<sNetworkResponse>>, mean response time
  - Resource utilization aspect for components
    - <<sServiceTime>>, service time
    - <<sVisits>>, percentage of jobs visit
    - <<sReplication>>, number of component copies
    - <<sUtilization>>, busy percentage time;
    - <<sLength>>, queue length
    - <<sResponseTime>>, response time
  - Resource utilization aspect for connectors
    - <<sDelayTime>>, transmission delay time
    - <<sVisits>>, percentage of jobs visit



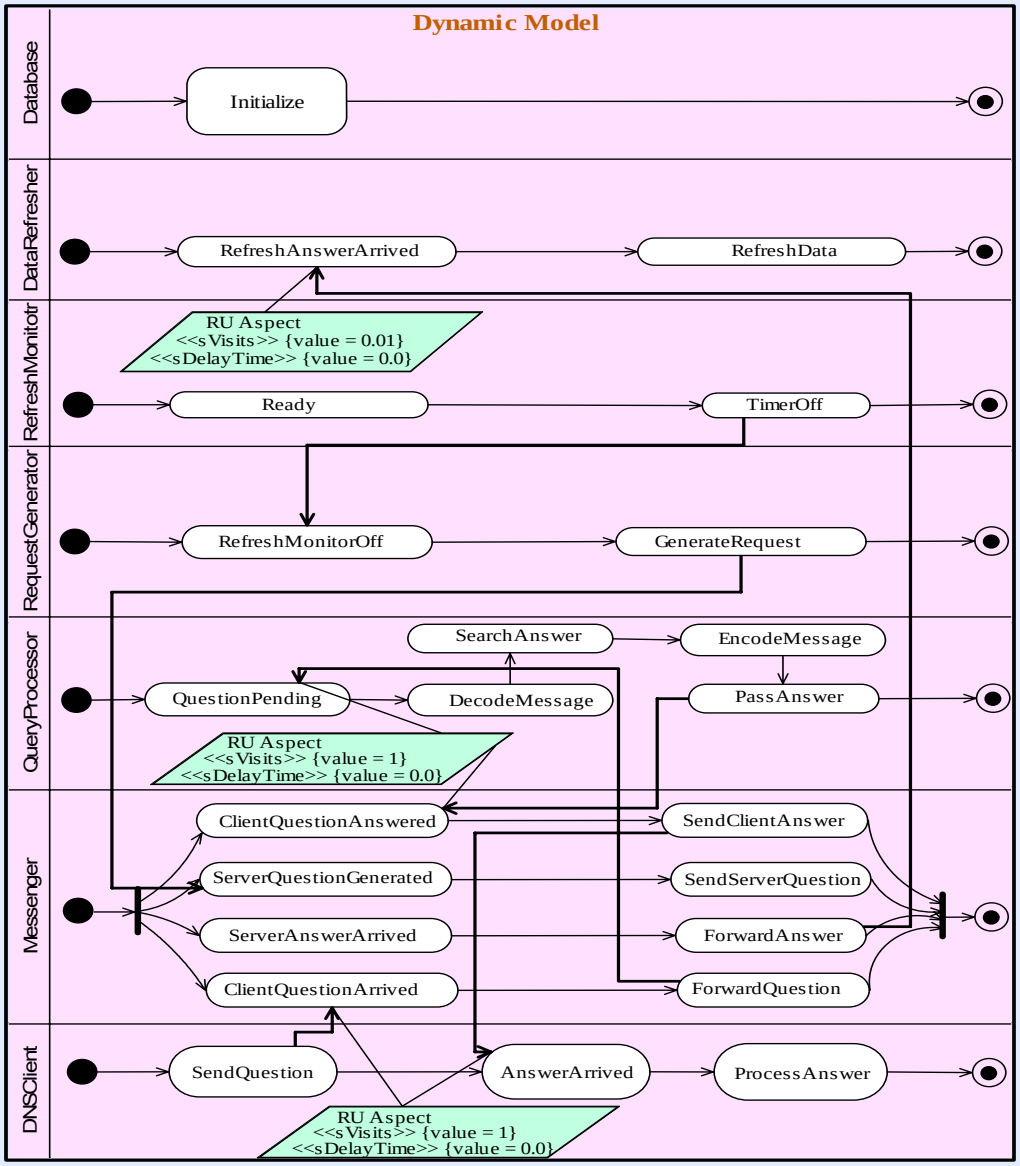




FDAF Step One: Extend DNS architecture design with resource utilization aspect



FDAF Step One: Extend DNS architecture design with resource utilization aspect (continued)



FDAF Step Two: Automatically generate Armani formal specification

- Armani
  - Monroe, Carnegie Mellon University
  - First-order predicate logic-based language
  - Underlying performance analysis theory: queueing network theory

- Mapping from extended UML class diagram and UML swim lane activity diagram to Armani
  - UML class diagram is used to describe the static structure
    - One UML class is translated into one Armani component, may be extended with resource utilization aspect
  - UML swim lane avtivity diagram organizes actions and activities for classes
    - UML action states and transitions are not translated
    - UML crossing swim lane transitions are translated into Armani architectural connections, may be extended with resource utilization aspect for connectors
  - Resource utlization aspect is translated into performance properties used in Armani

Mapping algorithm for translating extended UML class diagram into Armani

***Algorithm Am.1**:*

Input: A UML class diagram CD, where CD.Classes = {C_1, C_2, C_3 … C_m | C_i is a UML class, $1 \le i \le m$} (|CD.Classes| = m);

Output: S, which is an Armani architecture specification, where S.Style.ComponentTypes = {SCT_1, SCT_2, SCT_3 … SCT_n | SCT_i is an Armani architecture component type, $1 \le i \le n$}, S.System.Components = {SC_1, SC_2, SC_3 … SC_l | SC_i is an Armani system component, $1 \le i \le l$},and satisfies m = n = l.

Translate (CD) : S
- S ← ∅;
- S.Style.Name ← a randomly generated string;
- S.System.Name ← a randomly generated string;
- S.System.SystemResourceUtilizationAspect ← CD.SystemResourceUtilizationAspect;
- **while** CD.Classes ≠ ∅
  - newAmComponentType : amComponentType;
  - newAmComponent : amComponent;
  - uC ∈ CD.Classes;
  - newAmComponentType.Name ← uC.Name;
  - **while** uC.Operations ≠ ∅
    - newAmPort : amPort;
    - uOp ∈ uC.Operations;
    - amPort.Name ← uOp.Name;
    - newAmComponentType.Ports ← newAmComponentType.Ports ∪ {newAmPort};
    - uC.Operations ← uC.Operaions - {uOp}
  - newAmComponent.Name ← uC.Name + “Instance”;
  - newAmComponent.ComponentType ← newAmComponentType;
  - newAmComponent.ComponentResourceUtilizationAspect ← uC.ComponentResourceUtilizationAspect;
  - S.Style.ComponentTypes ← S.Style.ComponentTypes ∪ { newAmComponentType};
  - S.System.Components ← S.System.Components ∪ {newAmComponent };
  - CD.Classes ← CD.Classes - {uC};
- S.System.Style ← S.Style;

Time complexity: O(N*M), N (number of classes), M (maximum number of operations a class has)

Algorithm for translating UML swim lane activity diagram is presented in Section 4.2.2.2 of Chapter 4 in the dissertation

Partial Algorithm Proof:

/ Pre: uC is a UML class* $\wedge$ *|uC.Operations| = X* $\wedge$ *newAmComponentType is an Armani architecture component type*
$\wedge$ *|NewAmComponentType.Ports| = Y */*

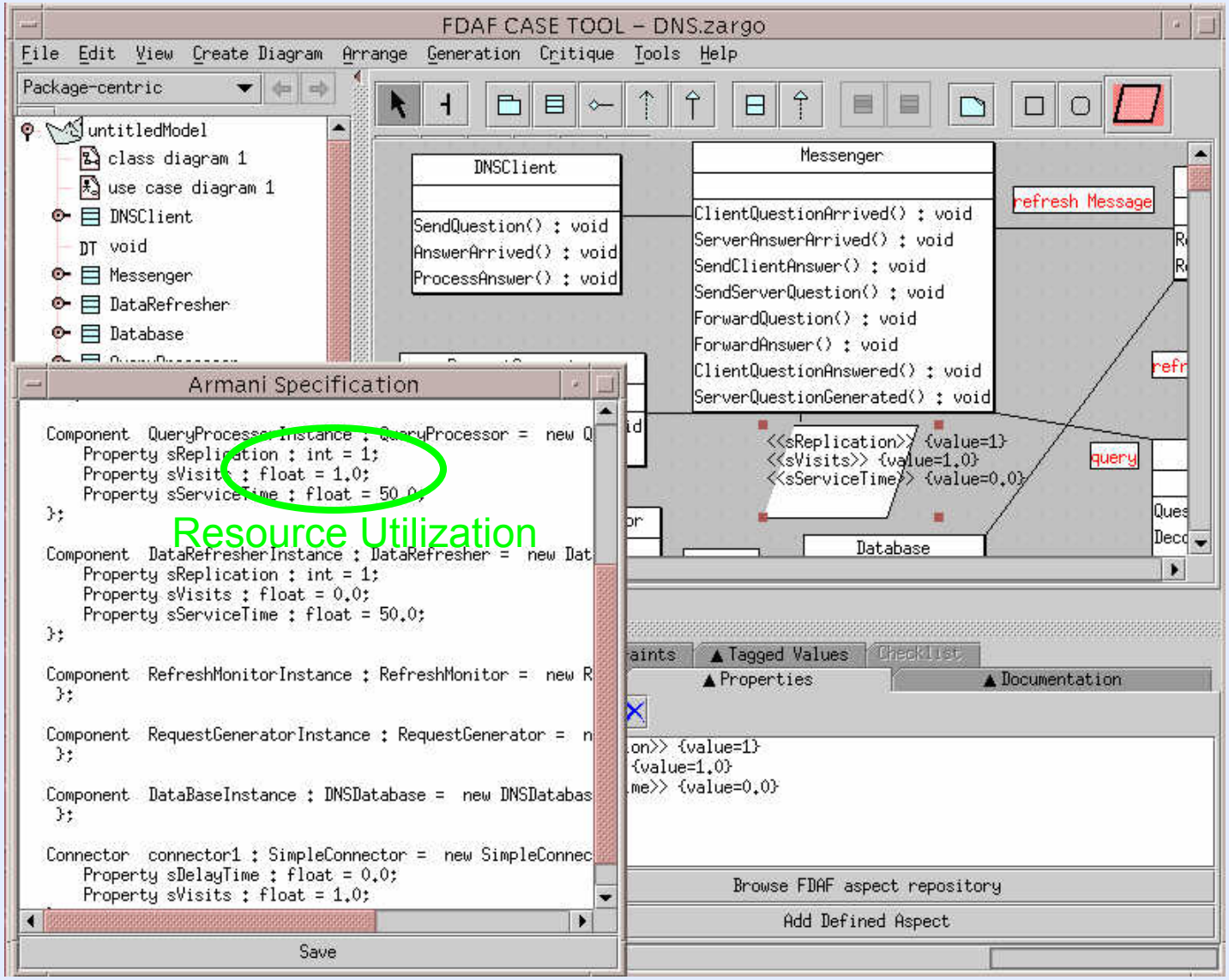
while uC.Operations $\neq \varnothing$
newAmPort : amPort;
uOp $\in$ uC.Operations;
amPort.Name $\leftarrow$ uOp.Name;
newAmComponentType.Ports $\leftarrow$ newAmComponentType.Ports $\cup$ {newAmPort};
uC.Operations $\leftarrow$ ucOperaions - {uOp};

*/*Post: X - |uC.Operations| = |NewAmComponentType.Ports| - Y */*

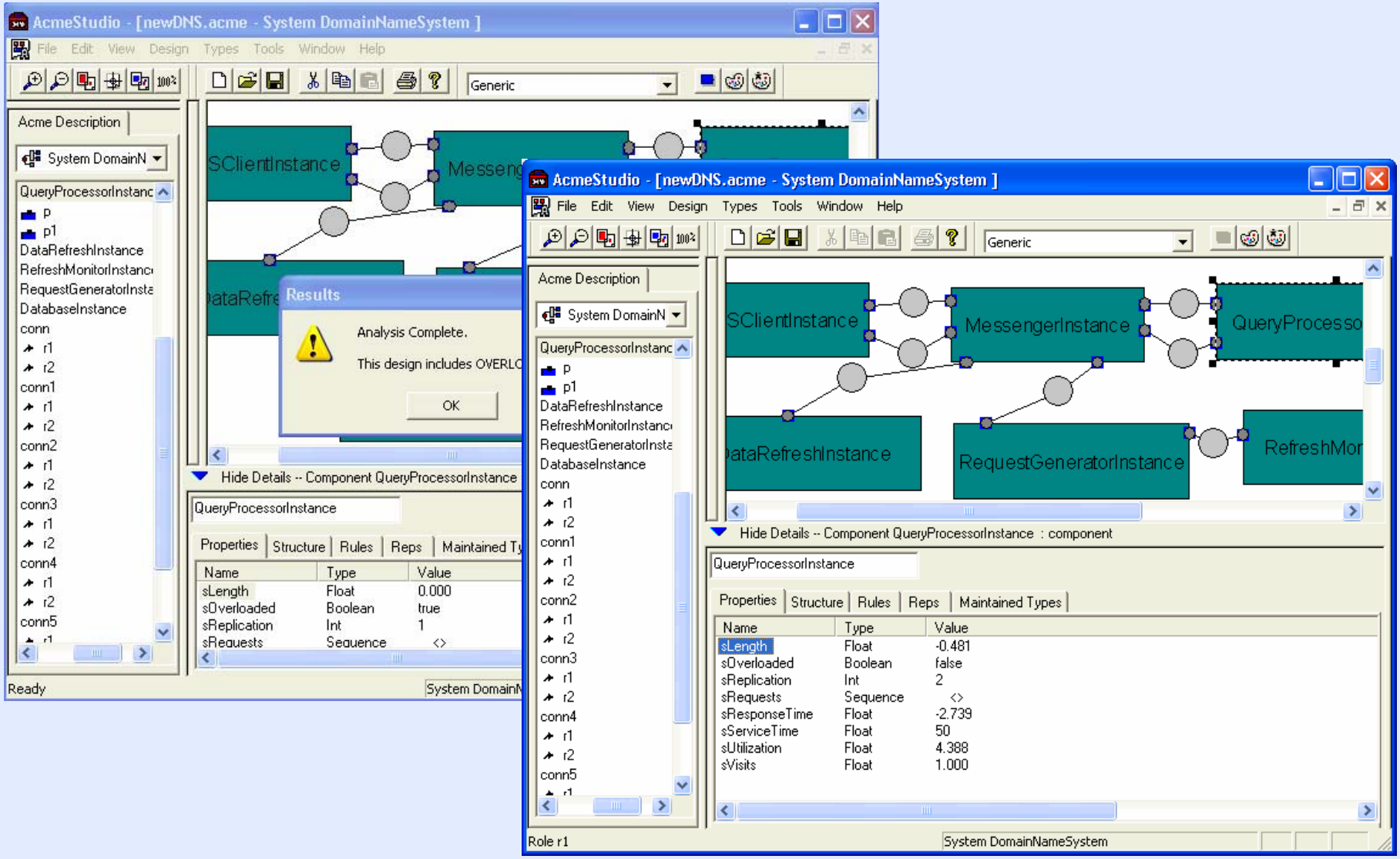
Proof:

1. On the initial entry to the loop:
   |uC.Operations| = X $\wedge$ *newAmComponentType is an Armani architecture component type* $\wedge$ *|NewAmComponentType.Ports| = Y* (from Pre), and *X - |uC.Operations| = X - X = 0*, *|NewAmComponentType.Ports| - Y = Y - Y = 0*, thus, ***P*** : *X - |uC.Operations| = |NewAmComponentType.Ports| - Y* is true.
2. Suppose that P is true before an arbitrary loop iteration. Assume |uC.Operations| = A, |NewAmComponentType.Ports| = B, and X - A = B - Y.
   Then after the iteration:
   From *NewAmComponentType.Ports* $\leftarrow$ *NewAmComponentType.Ports* $\cup$ *{newAmPort};*
   |NewAmComponentType.Ports| = |NewAmComponentType.Ports| + 1 = B + 1
   From *uC.Operations* $\leftarrow$ *uC.Operations - {uOp};* |uC.Operations| = |uC.Operations| - 1 = A - 1
   Thus, X - (A - 1) = B + 1 – Y, After the iteration, Post is true.
3. On exit from the loop:
   P $\wedge \neg$(uC.Operations $\neq \varnothing$) = uC.Operations = $\varnothing \wedge$ P $\rightarrow$ X = |NewAmComponentType.Ports| - Y
   $\rightarrow$ X - |uC.Operations| = |NewAmComponentType.Ports| - Y = post
4. So long as the loop has not terminated,
   P $\wedge$ (uC.Operations $\neq \varnothing$) = X - |uC.Operations| = |NewAmComponentType.Ports| - Y $\wedge$ (uC.Operations $\neq \varnothing$)
   $\rightarrow$ |uC.Operations| > 0 $\rightarrow$ t > 0 (t is an integer-valued function)
5. After each iteration, one element in uC.Operations is deleted from the set. This implies that |uC.Operations| (i.e., t) is decreased by a positive integer amount. Thus the loop will terminate, and the part of algorithm is correct.

Complete proof for the algorithm is presented in Section 4.2.2.2 of Chapter 4 in the dissertation

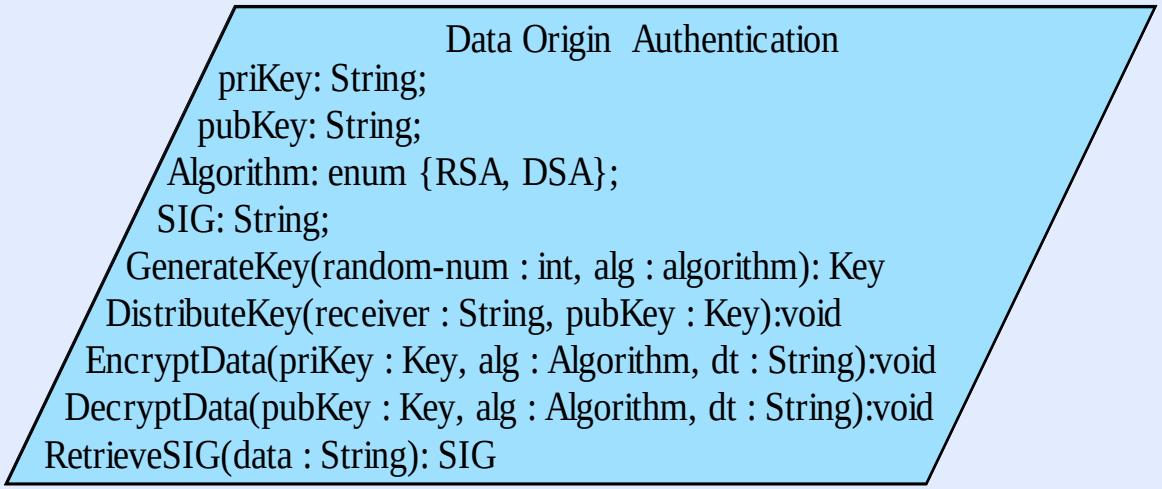

FDAF CASE TOOL – DNS.zargo
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
use case diagram 1
DNSClient
DT void
Messenger
DataRefresher
Database
DNSClient
SendQuestion() : void
AnswerArrived() : void
ProcessAnswer() : void
Messenger
ClientQuestionArrived() : void
ServerAnswerArrived() : void
SendClientAnswer() : void
SendServerQuestion() : void
ForwardQuestion() : void
ForwardAnswer() : void
ClientQuestionAnswered() : void
ServerQuestionGenerated() : void
refresh Message
<<sReplication>> {value=1}
<<sVisits>> {value=1.0}
<<sServiceTime>> {value=0.0}
query
Database
Armani Specification
Component QueryProcessorInstance : QueryProcessor = new Q
Property sReplication : int = 1;
Property sVisits : float = 1.0;
Property sServiceTime : float = 50.0;
};
Component DataRefresherInstance : DataRefresher = new Dat
Property sReplication : int = 1;
Property sVisits : float = 0.0;
Property sServiceTime : float = 50.0;
};
Component RefreshMonitorInstance : RefreshMonitor = new R
};
Component RequestGeneratorInstance : RequestGenerator = n
};
Component DataBaseInstance : DNSDatabase = new DNSDatabas
};
Connector connector1 : SimpleConnector = new SimpleConnec
Property sDelayTime : float = 0.0;
Property sVisits : float = 1.0;
Save
Resource Utilization
Tagged Values
Checklist
Properties
Documentation
on>> {value=1}
{value=1.0}
me>> {value=0.0}
Browse FDAF aspect repository
Add Defined Aspect

FDAF Step Three: Resource utilization aspect analysis results in Armani

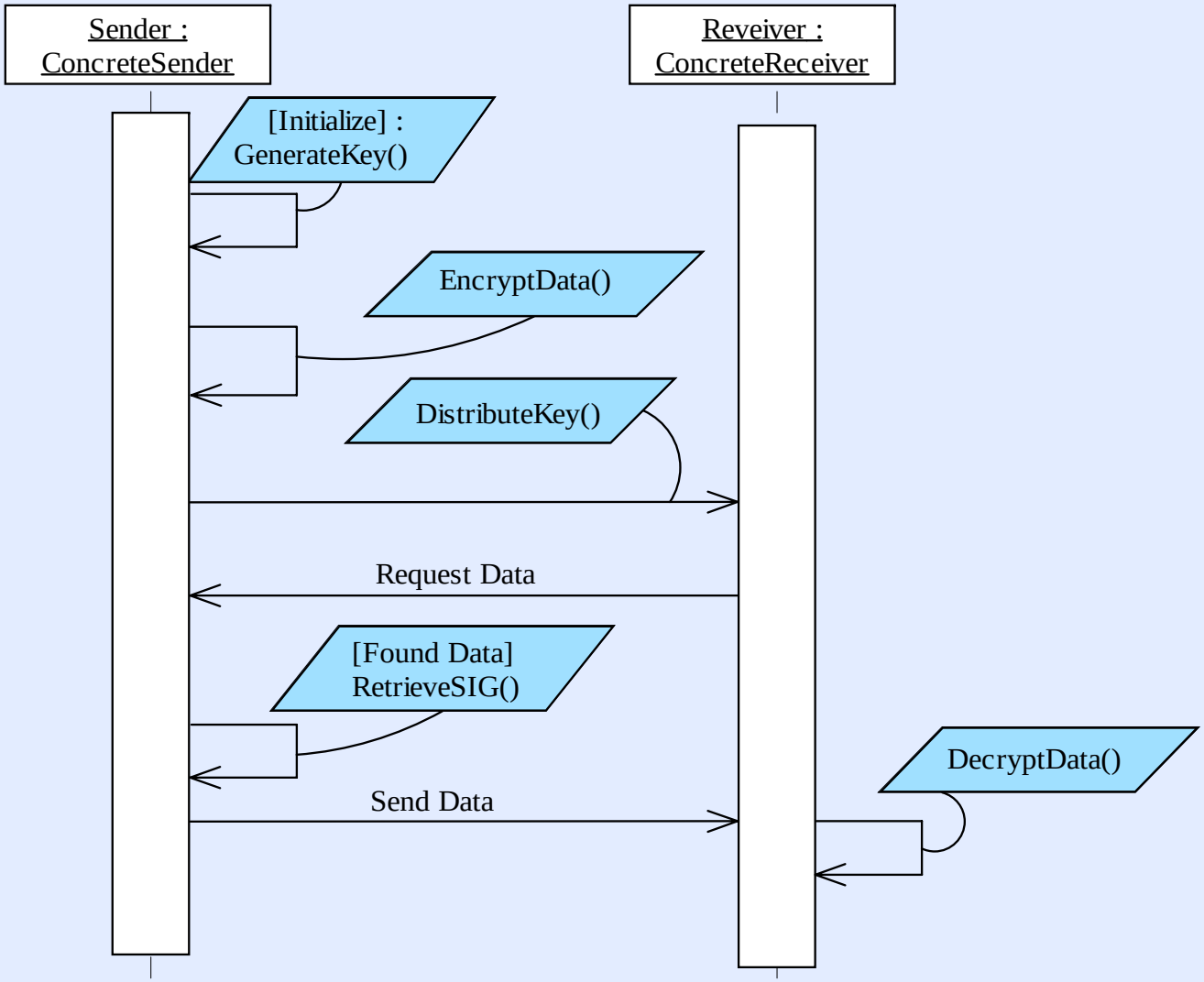


FDAF Approach Illustrations : Data Origin Authentication Security Aspect

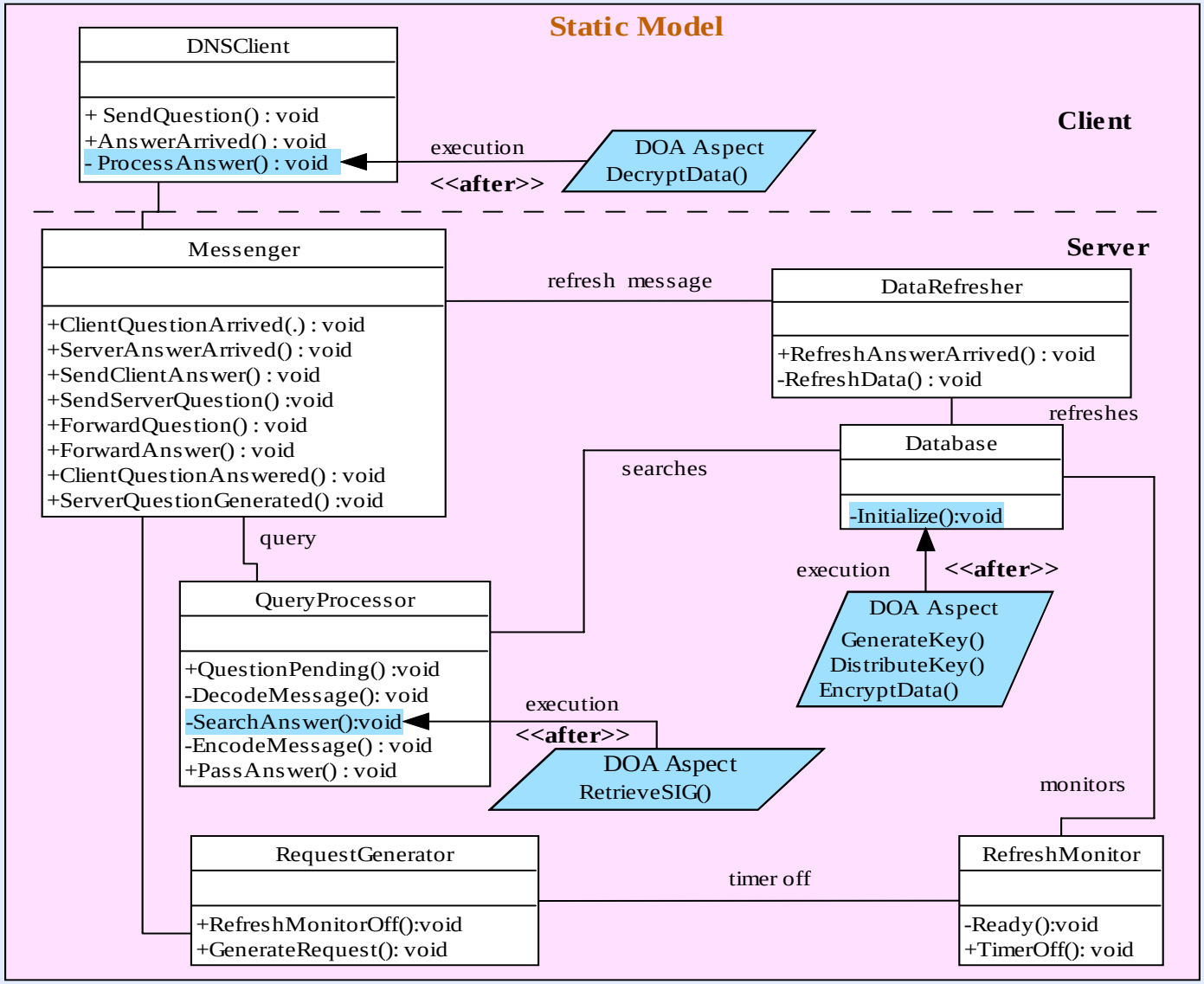
- Substantive aspect
- Aspect definition
  - Adapted from data origin authentication security pattern
  - Security service that verifies an identity claimed by or for an entity
  - Static view



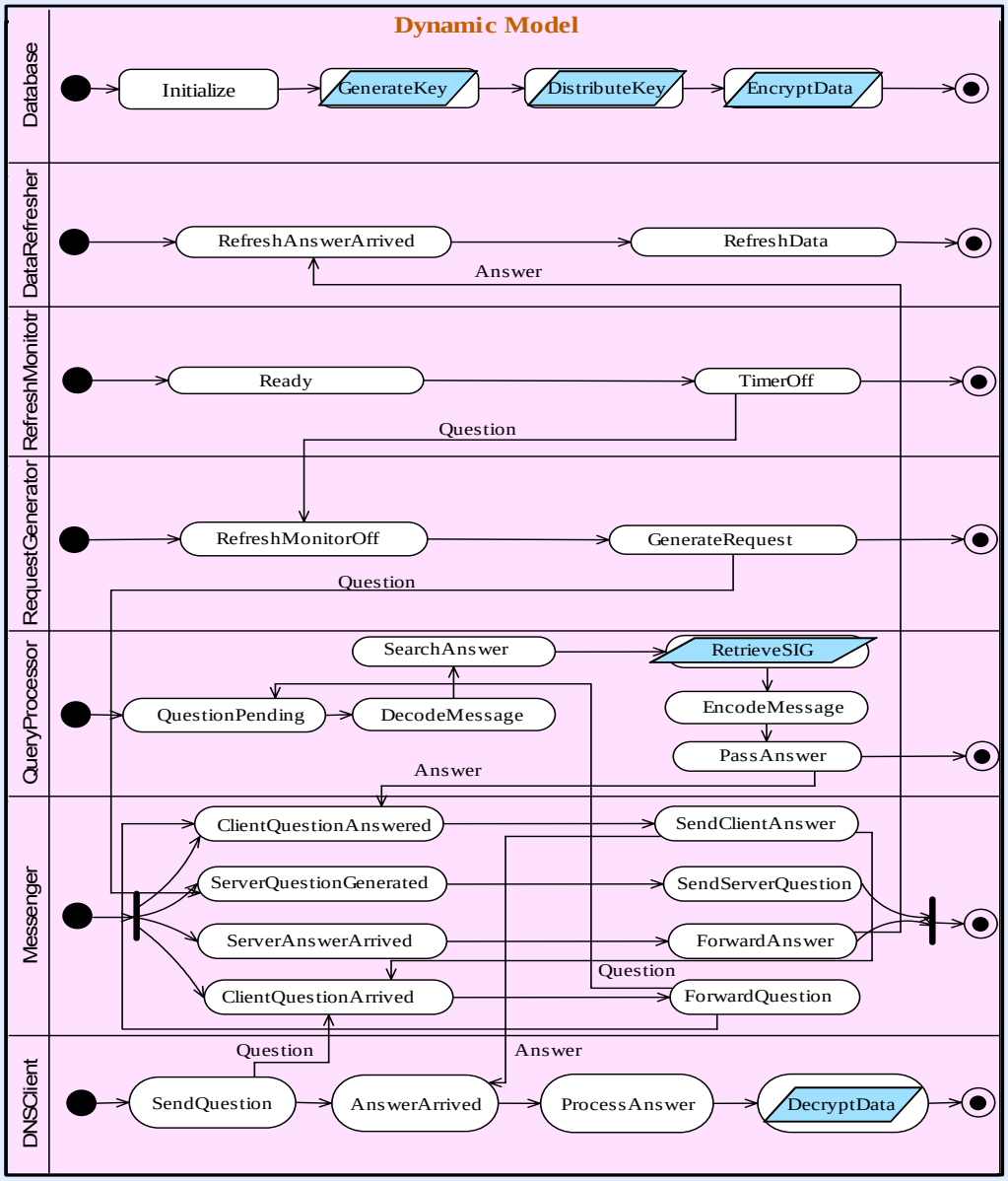
- Dynamic view



FDAF Step One: Extend DNS architecture design with data origin authentication aspect

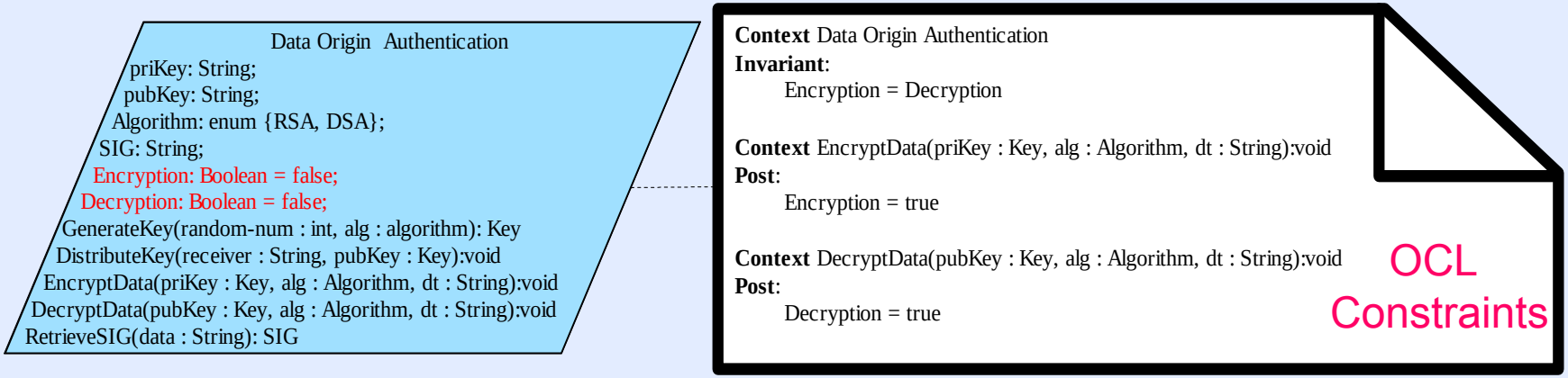


FDAF Step One: Extend DNS architecture design with data origin authentication aspect (continued)



FDAF Step Two: Automatically generate Promela formal specification

- Why formal analysis?
  - Prevent inconsistency of an aspect-oriented design
    - Only “EncrypteData()” is added, without “DecryptData()”, or the other way around
- How?
  - OCL constraints defined for the DOA security aspect by the FDAF
  - Create Promela formal specification for the design
  - Use model checker Spin to discover design errors



FDAF Step Two: Automatically generate Promela formal specification (continued)

- Promela
  - Based on Hoare's communication sequential processes (CSP) language
  - Input language for model checker Spin
  - Spin: simulator, exhaustive verifier

- Mapping form extended UML class diagram and UML swim lane activity diagram into Promela
  - UML class diagram is used to describe the static structure
    - One UML class is translated into one Promela process type
    - OCL constraints are translated into Promela macros and then inserted as Promela assertions
  - UML swim lane activity diagram organizes actions and activities for classes
    - UML states and transitions are translated into Promela process type block statements
    - UML crossing swim lane transitions are translated into message channels between process types

Mapping algorithm for translating extended UML swim lane activity diagram into Promela (partial)

Algorithm P.2:

Input: A UML swim lane activity diagram SAD, where SAD = {U_1, U_2, U_3 … U_n | U_i is a uOrganizationUnit, $1 \leq i \leq n$}, and data origin authentication aspect DOA. The SAD has been extended with the DOA aspect;

Output: S, which is a PSpecification, where S.Proctypes = {P_1, P_2, P_3 … P_y | P_i is a PProctype, $1 \leq i \leq y$}, S.Channels = {Ch_1, Ch_2, Ch_3 … Ch_x | Ch_i is a pMChannels, $1 \leq i \leq x$}, and satisfies y = n;

Translate (SAD = {U_1, U_2, U_3 … U_n}) : S
S : generated by Algorithm P.1;
processProtypes : integer;
processProtypes ← 0;

while SAD ≠ ∅
u ∈ SAD;
uName : String;
uName ← u.Name;
curProctype : pProctype;
size : integer;
size ← |S.Proctypes|;
for int x ← 1 **to** size /*find the corresponding Pormela proctype */
if (S.Proctypes.elementAt[x].Name == uName)
curProctype = S.Proctypes.elementAt[x] ;
processProtypes++;
break ;
if could not find such a proctype
Input Error;
while u.Transitions ≠ ∅
uTran : uTransition;
pBlock : PTBlock;
……

Time complexity: O(N*M), N (number of lanes), M (maximum number of transitions a lane has)
Complete algorithm is presented in Section 3.3.2.2 of Chapter 5 in the dissertation

Partial Algorithm Proof:

/ Pre: SAD is a UML swim lane activity diagram* $\wedge$ *|SAD| = X* $\wedge$ *processProtypes = 0 */*

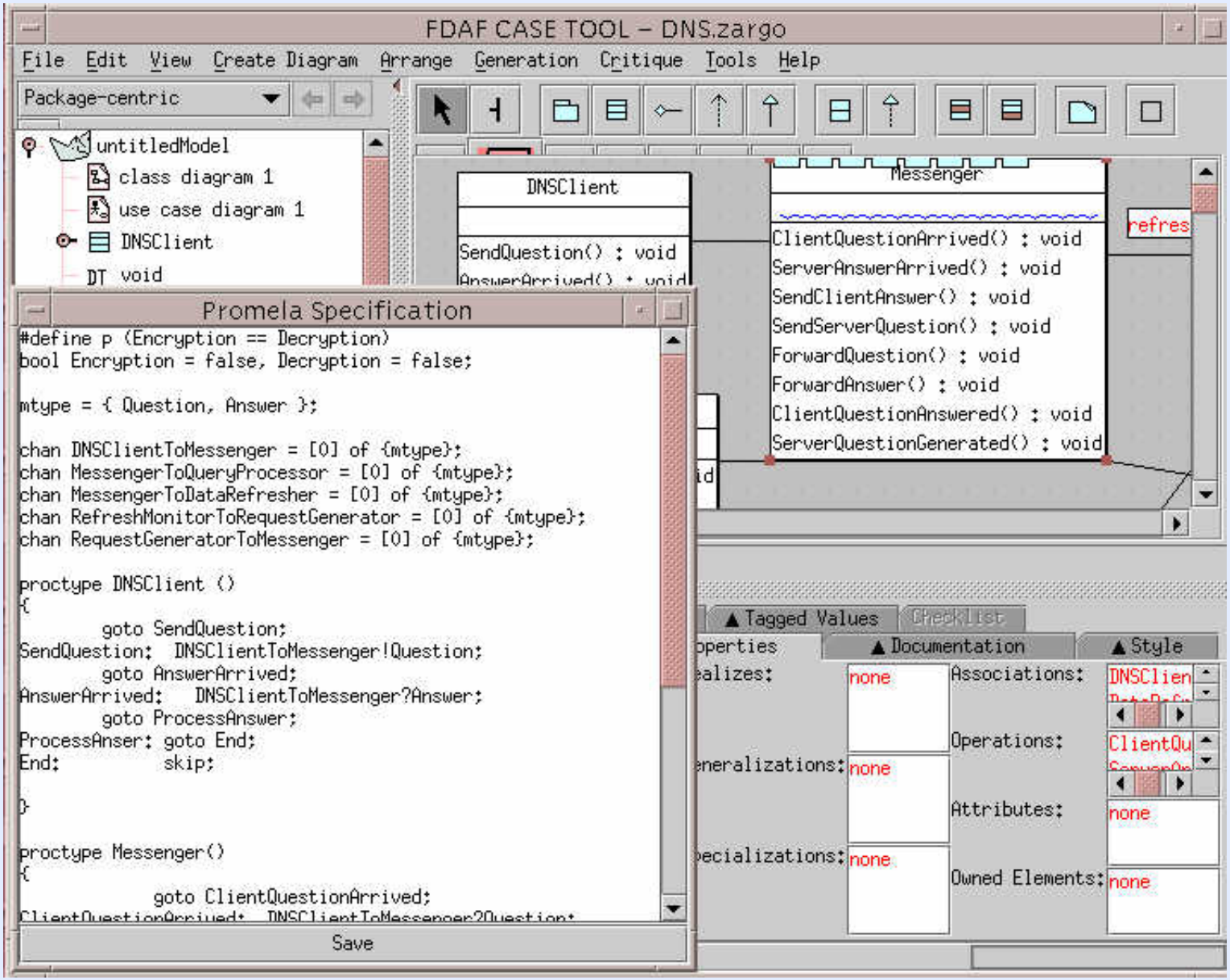
S : generated by Algorithm P.1;
processProtypes : integer;
processProtypes $\leftarrow$ 0;
while SAD $\neq \varnothing$
……

*/*Post: X - |SAD| = processProtypes */*

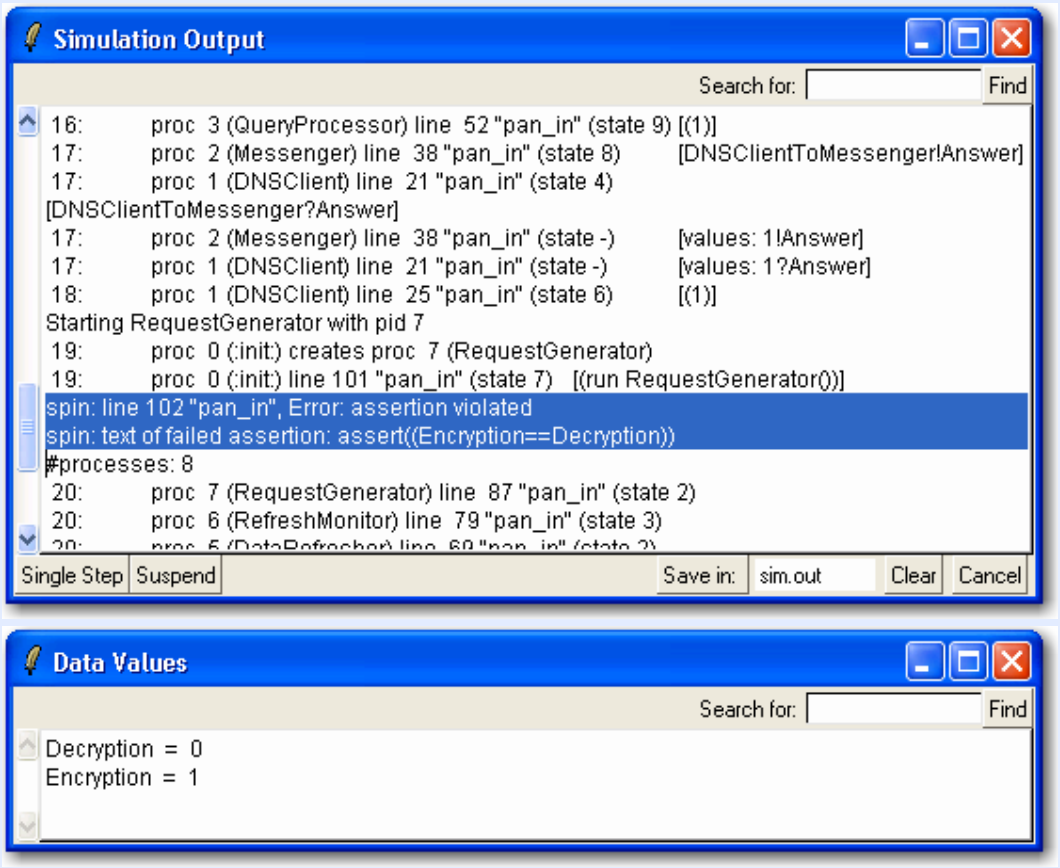
Proof:

1. On the initial entry to the loop:
   |SAD| = X $\wedge$ *processProtypes = 0* (from Pre), and *X* - *|SAD| = X - X = 0*, and processProtypes = 0 thus,
   P : *X* - *|SAD|* = processProtypes is true.
2. Suppose that P is true before an arbitrary loop iteration. Assume |SAD| = A, processProtypes = B, and X - A = B.
   Then after the iteration, two cases:
   Case (i); *if (S.Proctypes.elementAt[x].Name == uName):*
   From *processProtypes++;* processProtypes = processProtypes + 1 = B + 1
   From *SAD* $\leftarrow$ *SAD - {u};* |SAD| = |SAD| - 1 = A - 1
   Thus, X - (A - 1) = B + 1
   Case (ii); *if could not find such a proctype:*
   From *Input Error;* The algorithm terminates;
   Thus, X - A = B
   After the iteration, Post is true.
3. On exit from the loop:
   P $\wedge \neg$(SAD $\neq \varnothing$) = SAD= $\varnothing \wedge$ P $\rightarrow$ X = processProtypes $\rightarrow$ X - |SAD | = processProtypes = post
4. So long as the loop has not terminated,
   P $\wedge$ (SAD$\neq \varnothing$) = X - |SAD | = processProtypes $\wedge$ (SAD $\neq \varnothing$) $\rightarrow$ |SAD| > 0 $\rightarrow$ t > 0 (t is an integer-valued function)
5. After each iteration, one element in SAD is deleted from the set. This implies that |SAD| (i.e., t) is decreased by a positive integer amount.
   Thus the loop will terminate, and the algorithm is correct.

Complete proof for the algorithm is presented in Section 3.3.2.2 of Chapter 5 in the dissertation

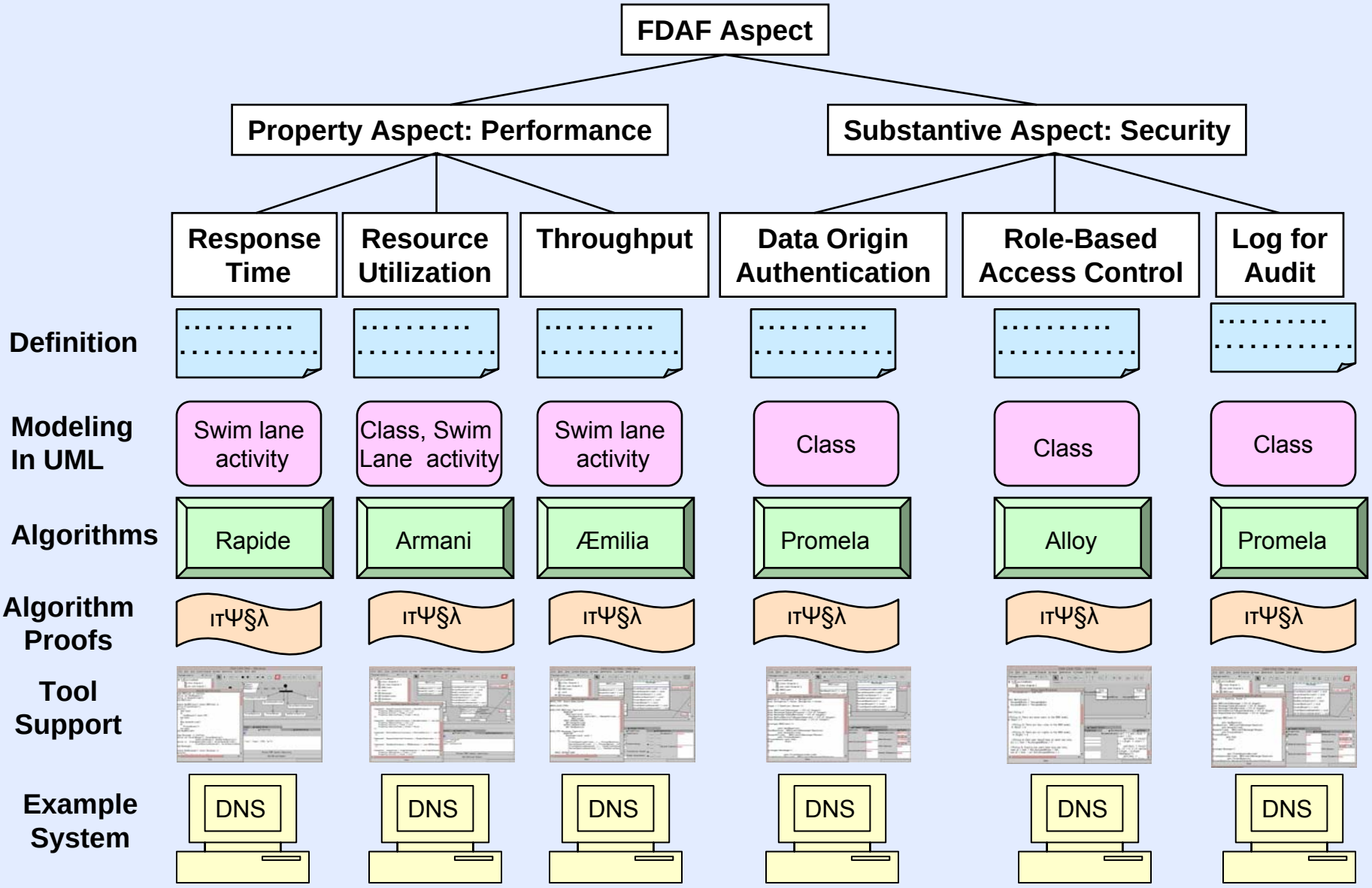

FDAF CASE TOOL – DNS.zargo
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
use case diagram 1
DNSClient
DT void
DNSClient
SendQuestion() : void
Messenger
ClientQuestionArrived() : void
ServerAnswerArrived() : void
SendClientAnswer() : void
SendServerQuestion() : void
ForwardQuestion() : void
ForwardAnswer() : void
ClientQuestionAnswered() : void
ServerQuestionGenerated() : void
refres
Promela Specification
#define p (Encryption == Decryption)
bool Encryption = false, Decryption = false;
mtype = { Question, Answer };
chan DNSClientToMessenger = [0] of {mtype};
chan MessengerToQueryProcessor = [0] of {mtype};
chan MessengerToDataRefresher = [0] of {mtype};
chan RefreshMonitorToRequestGenerator = [0] of {mtype};
chan RequestGeneratorToMessenger = [0] of {mtype};
proctype DNSClient ()
{
goto SendQuestion;
SendQuestion: DNSClientToMessenger!Question;
goto AnswerArrived;
AnswerArrived: DNSClientToMessenger?Answer;
goto ProcessAnswer;
ProcessAnser: goto End;
End: skip;
}
proctype Messenger()
{
goto ClientQuestionArrived;
Save
Tagged Values
Checklist
Documentation
Style
Associations:
Operations:
Attributes:
Owned Elements:
none
DNSClien
ClientQu

FDAF Step Three: Data origin authentication aspect analysis results in Promela



Indicates a defect in the design, a mismatch in the Encryption/Decryption mechanism. Architects need to go back to revise their design.

Summary


FDAF Aspect
Property Aspect: Performance
Substantive Aspect: Security
Response Time
Resource Utilization
Throughput
Data Origin Authentication
Role-Based Access Control
Log for Audit
Definition
Modeling In UML
Swim lane activity
Class, Swim Lane activity
Swim lane activity
Class
Class
Class
Algorithms
Rapide
Armani
Æmilia
Promela
Alloy
Promela
Algorithm Proofs
ITΨ§λ
Tool Support
Example System
DNS

Contributions to Knowledge

- Represents non-functional properties performance and security into aspects that can be understood, modeled, and analyzed at the software architecture design level, enabling the realization of non-functional requirements for software architectures
- Defines a UML extension, with syntax and semantics, for modeling crosscutting non-functional properties
- Formalizes (part of) UML into a set of existing formal languages. With formal analysis tools, the automated analysis of a UML based architecture design is achieved
- Applies the aspect concept to support the maintainability and evolvability of an architecture design with non-functional properties

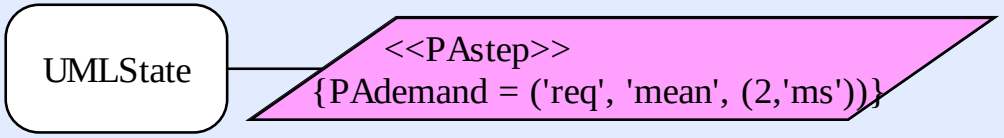
Conclusions and Future Work

- Formal Design Analysis Framework
  - Extensible, flexible aspect-oriented architectural framework supports the design and analysis of non-functional properties for software architectures
- Future work
  - Investigate interaction analysis among security aspects
  - Conform the FDAF with UML 2.0
    - Use composite structure diagram to describe software architecture
  - Extend the framework
    - Additional non-functional properties (e.g., reliability)
    - The design and analysis of product line architectures
  - Support the identifying, analysis, and negotiation among multiple, (possibly) conflicting non-functional properties
    - The Non-Functional Requirement Framework [Chung]
    - Architecture Tradeoff Analysis Method (ATAM) [Kazman]
    - Partial goal satisfaction [Letier]

Thank You!

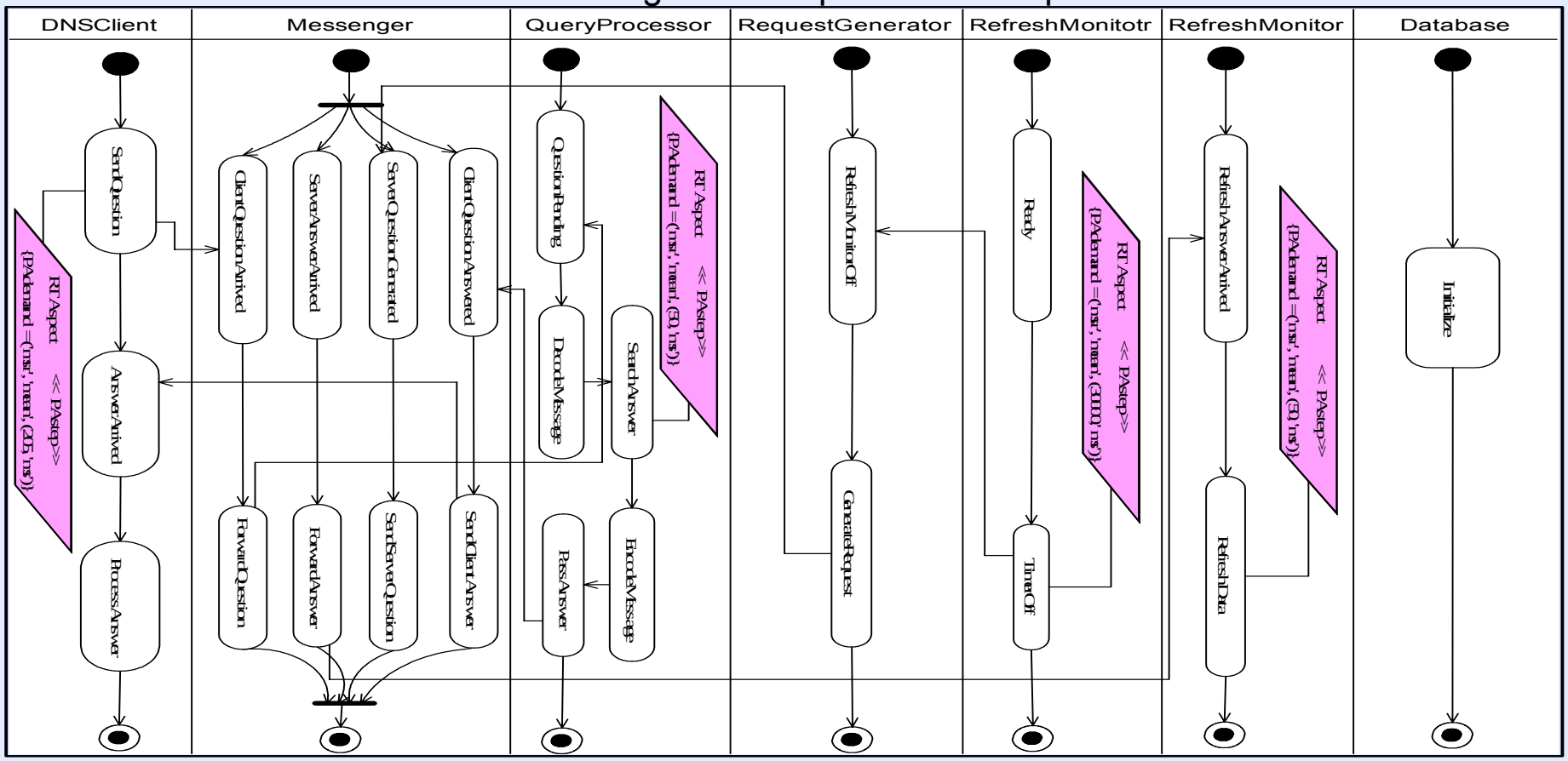
FDAF Approach Illustrations : Response Time Performance Aspect

- Property aspect
- Aspect definition
  - Stereotype <<PAstep>> and tag PAdemand, defined in RTUML



- DNS illustration

1. Extend DNS architecture design with response time aspect



Publications

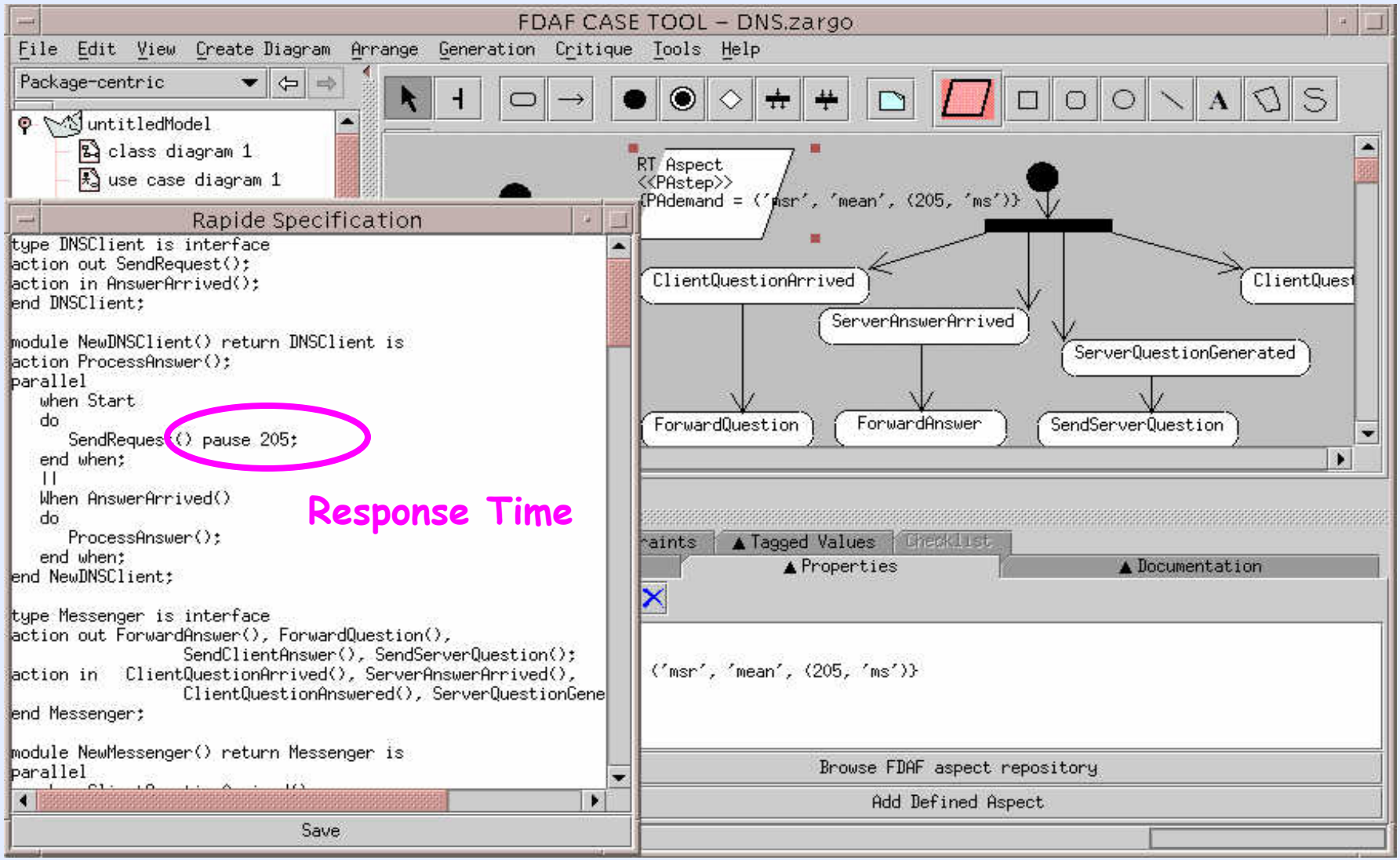
- Journal Papers
  - Cooper K., Dai L., and Deng Y. "Performance Modeling and Analysis of Software Architecture: An Aspect-Oriented UML Based Approach", Special Issue on System and Software Architectures of the Journal of Science of Computer Programming, 2004 (accepted).
  - I-Ling Yen, Lirong Dai, Ing-Ray Chen, and Biao Chen, "A Non-blocking Atomic Transaction Processing Algorithm with Real-Time Property", International Journal of Reliability, Quality and Safety Engineering, Vol. 8, No.4 (2001), pp. 391-408.
  - Dai L., Cooper K., and Wong E, "Modeling and Analysis of Performance Aspects for Software Architectures: a UML Based Approach", International Journal of Software Engineering and Knowledge Engineering: Special Issue on Aspect-Oriented Software Design Models, March 2005. (Submitted)
- Conference/Workshop Papers
  - Dai L., Cooper K., and Wong E., "Modeling Reusable Security Aspects for Software Architectures: a Pattern Driven Approach", Accepted by The Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE'05), July 2005.
  - Cooper K., and Dai L., "Modelling and Performance Cost Analysis for Security Aspects: a UML Based Approach", Accepted by International Conference on Software Engineering Research and Practice (SERP'05): the Fourth International Workshop on Systems and Software Architecting (IWSSA'05), June 2005.
  - Dai L., and Cooper K., "Modeling and Analysis of Non-functional Requirements as Aspects in a UML Based Software Architecture Design", Accepted by the Sixth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2005), May 2005.
  - Dai L., "Formal Design Analysis Framework: An Aspect-Oriented Approach", International Conference on Software Reuse 2004 Doctoral Student Session, July 2004.
  - Cooper K., Dai L., and Deng Y., "Performance Modeling and Analysis of Software Architecture: An Aspect-Oriented UML Based Approach", International Conference on Software Engineering Research and Practice (SERP'04): International Workshop on Systems and Software Architecting (IWSSA'04), June 22, 2004, pp. 111-117.
  - Cooper K., Dai L., and Deng, Y., "Modeling Performance as Aspect: a UML Based Approach", the Sixth International Conference on Unifier Modeling Language: Workshop on Aspect Oriented Modeling in UML, San Francisco, USA, October 2003.
  - Cooper K., Dai L., Deng Y., and Dong J., "Developing a Formal Design Analysis Framework", International Conference on Software Engineering Research and Practice (SERP'03), June 2003, pp.68-73.
  - Cooper K., Dai L., Deng Y., and Dong J., "Towards an Aspect-Oriented Architectural Framework", the Second International Workshop on Aspect-Oriented Requirements Engineering and Architecture Design (Early Aspects), Boston, USA, March 2003.

- Papers in preparation
  - “Survey on Software Architecture Level Security Modeling and Analysis”, International Journal of Network Security
  - “Aspect-Oriented Modeling and Analysis of Security Capabilities in Software Architecture”, IEEE Transaction on Software Engineering
  - “Aspect-Oriented Modeling and Analysis of Performance Properties in Software Architecture”, Journal on System Man Cybernet
  - “UML Extension to Support Aspect-Oriented Modeling of non-functional Properties for Software Architecture”, Journal on System and Software Modeling

2. Automatically generate Rapide formal specification

- Rapide
  - Architecture description language
  - Luckham, Stanford University
  - Evolved from VHDL, ML, and TSL
  - Provides simulation of distributed systems by partial orderings of events
  - Timing model allows designers to describe and analyze time sensitive prototypes

- Mapping from UML class diagram and extended UML swim lane activity diagram to Rapide
  - UML class diagram is used to describe the static structure
    - One UML class is translated into one Rapide type
  - UML swim lane avtivity diagram organizes actions and activities for classes
    - UML action states and the transitions between these action states are translated into Rapide type behaviors
    - UML crossing swimlane transitions are translated into Rapide architectural connections
    - UML action states are extended with response time aspect. Response time aspect is translated to Rapide's timing model


FDAF CASE TOOL – DNS.zargo
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
use case diagram 1
Rapide Specification
type DNSClient is interface
action out SendRequest();
action in AnswerArrived();
end DNSClient;
module NewDNSClient() return DNSClient is
action ProcessAnswer();
parallel
when Start
do
SendReques() pause 205;
end when;
||
When AnswerArrived()
do
ProcessAnswer();
end when;
end NewDNSClient;
type Messenger is interface
action out ForwardAnswer(), ForwardQuestion(),
SendClientAnswer(), SendServerQuestion();
action in ClientQuestionArrived(), ServerAnswerArrived(),
ClientQuestionAnswered(), ServerQuestionGene
end Messenger;
module NewMessenger() return Messenger is
parallel
Save
Response Time
RT Aspect
<<PAstep>>
{PAdemand = ('msr', 'mean', (205, 'ms'))}
ClientQuestionArrived
ServerAnswerArrived
ClientQuest
ServerQuestionGenerated
ForwardQuestion
ForwardAnswer
SendServerQuestion
▲Tagged Values
Checklist
▲Properties
▲Documentation
('msr', 'mean', (205, 'ms'))
Browse FDAF aspect repository
Add Defined Aspect

3. Response time aspect analysis results in Rapide

QBEViewer - Computation 1

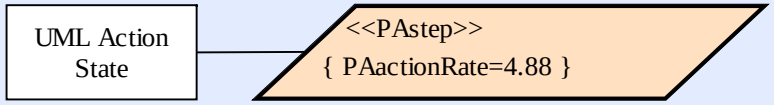
File Select Order Tools Help

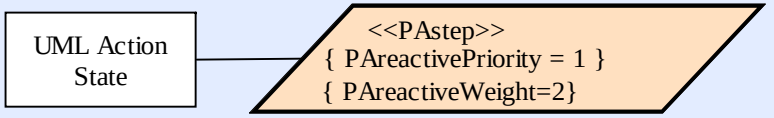
Name	Origine	Destination	Thread	Start	End
.*	.*	.*	.*	.*	.*
Clientquestionanswered	Domainnameserver'29	Newmessenger'31	112	156	156
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	156	156
Sendclientanswer	Newmessenger'31	Newmessenger'31	117	156	156
Answerarrived	Domainnameserver'29	Newdnsclient'30	117	156	156
Processanswer	Newdnsclient'30	Newdnsclient'30	120	156	156
Sendrequest	Newdnsclient'30	Newdnsclient'30	121	156	162
Clientquestionarrived	Domainnameserver'29	Newmessenger'31	121	156	162
Searchanswer	Newqueryprocessor'32	Newqueryprocessor'32	114	156	206
Forwardquestion	Newmessenger'31	Newmessenger'31	119	162	162
Questionpending	Domainnameserver'29	Newqueryprocessor'32	119	162	162
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	162	162
Sendrequest	Newdnsclient'30	Newdnsclient'30	121	162	168
Clientquestionarrived	Domainnameserver'29	Newmessenger'31	121	162	168
Forwardquestion	Newmessenger'31	Newmessenger'31	119	168	168
Questionpending	Domainnameserver'29	Newqueryprocessor'32	119	168	168
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	168	168
Sendrequest	Newdnsclient'30	Newdnsclient'30	121	168	174
Clientquestionarrived	Domainnameserver'29	Newmessenger'31	121	168	174
Forwardquestion	Newmessenger'31	Newmessenger'31	119	174	174
Questionpending	Domainnameserver'29	Newqueryprocessor'32	119	174	174
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	174	174
Sendrequest	Newdnsclient'30	Newdnsclient'30	121	174	180
Clientquestionarrived	Domainnameserver'29	Newmessenger'31	121	174	180
Forwardquestion	Newmessenger'31	Newmessenger'31	119	180	180
Questionpending	Domainnameserver'29	Newqueryprocessor'32	119	180	180
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	180	180
Sendrequest	Newdnsclient'30	Newdnsclient'30	121	180	186
Clientquestionarrived	Domainnameserver'29	Newmessenger'31	121	180	186
Forwardquestion	Newmessenger'31	Newmessenger'31	119	186	186
Questionpending	Domainnameserver'29	Newqueryprocessor'32	119	186	186
Decodemessage	Newqueryprocessor'32	Newqueryprocessor'32	115	186	186

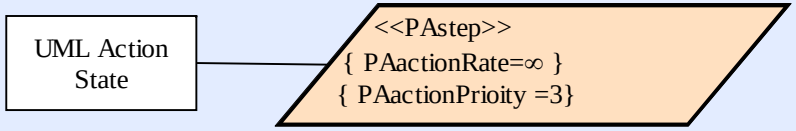
Average response time of DNS server is 50 milliseconds

FDAF Approach Illustration: Throughput Performance Aspect

- Property aspect
- Aspect definition
  - Throughput for active actions
    - Actions cause the change of the system's state
    - <<PAstep>> and
    - Tag PAactionRate, rate of action
    - Tag PAactionPriority, action's priority
    - Tag PAactionWeight, action's execution weight
  - Throughput for passive actions
    - Actions triggered by active actions
    - <<PAstep>> and
    - Tag PAreactivePriority, action's priority
    - Tag PAreactiveWeight, action's execution weight

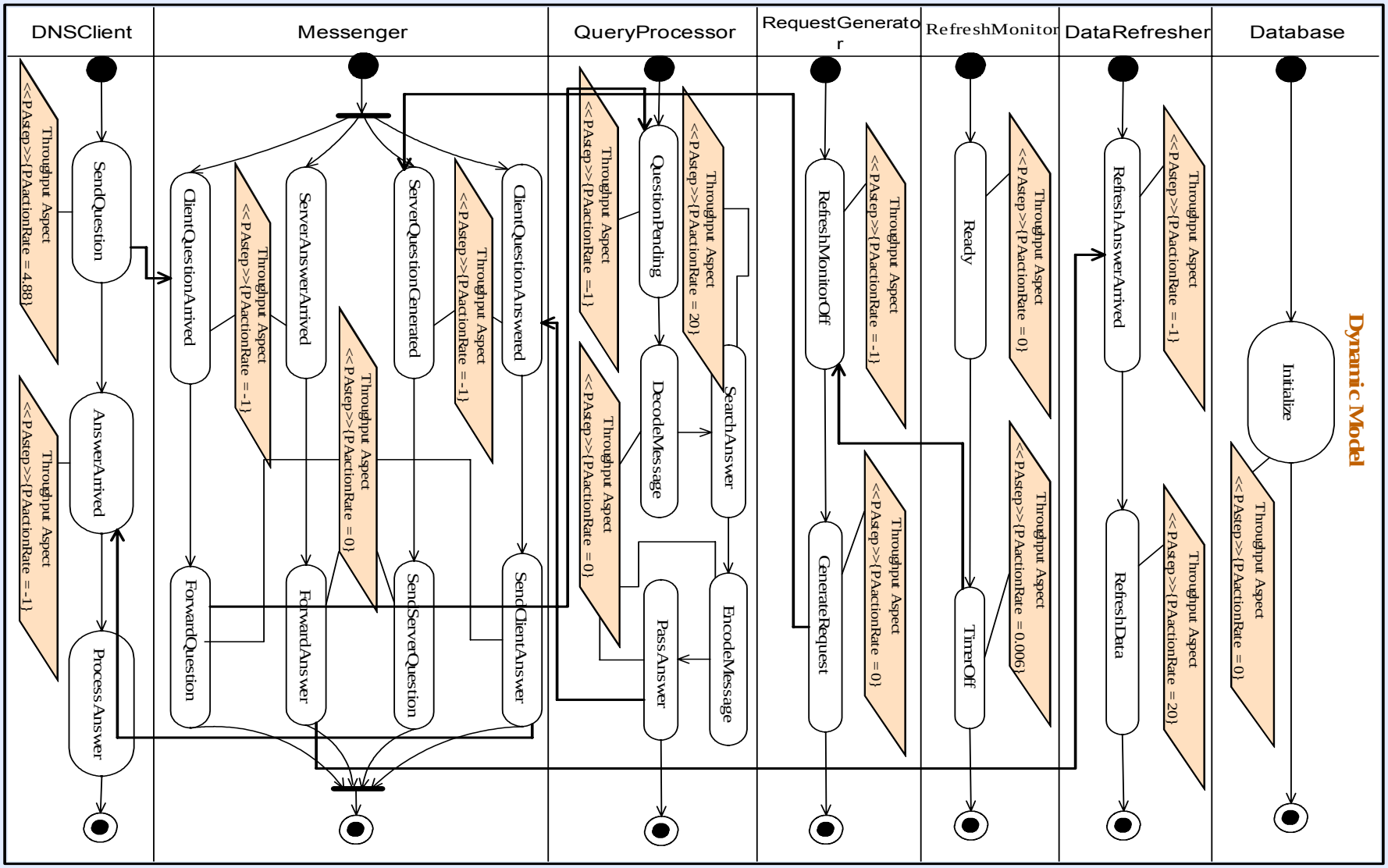






DNS Illustration

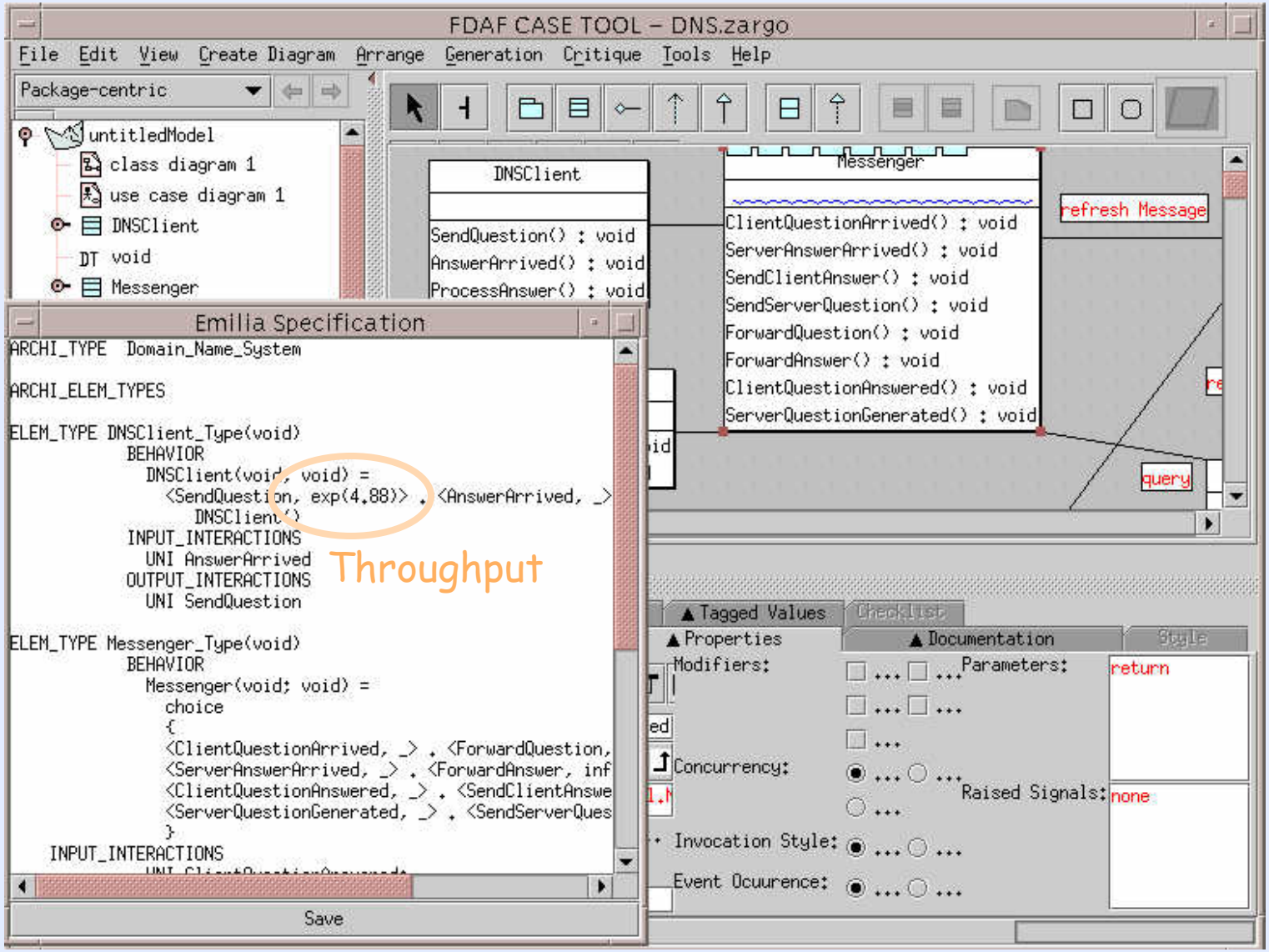
1. Extend DNS architecture design with throughput aspect



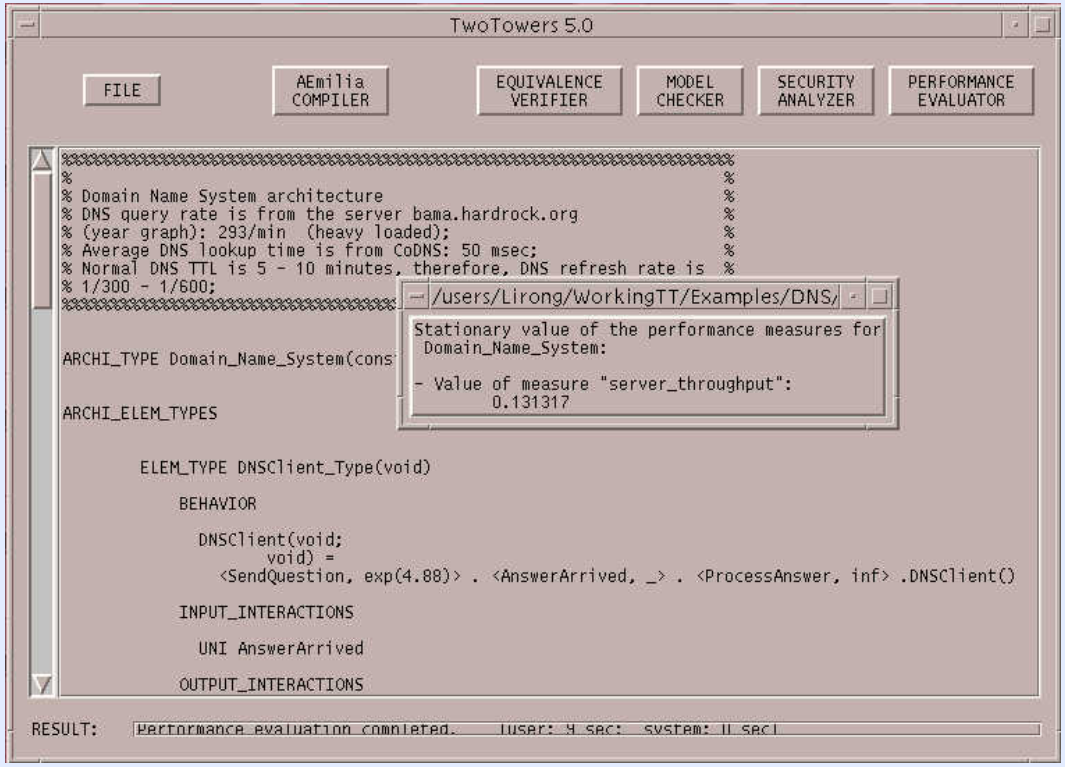
2. Automatically generate Æmilia formal specification

- Æmilia
  - Architecture description language
  - Balsamo S.
  - Process Algebra Architecture Description Language (PADL)
  - underlying performance analysis theory: Markov Chain

- Mapping from UML class diagram and UML swim lane activity diagram to Æmilia
  - UML class diagram is used to describe the static structure
    - One UML class is translated into one Æmilia architecural element type (AET)
  - UML swim lane avtivity diagram organizes actions and activities for classes
    - UML action states and transitions are translated into Æmilia AET behaviors
    - UML action states may be extended with throughput aspect, translated into Æmilia's throughput information
    - UML crossing swimlane transitions are translated into Æmilia architectural attachments among AETs


FDAF CASE TOOL – DNS.zargo
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
use case diagram 1
DNSClient
DT void
Messenger
DNSClient
SendQuestion() : void
AnswerArrived() : void
ProcessAnswer() : void
Messenger
ClientQuestionArrived() : void
ServerAnswerArrived() : void
SendClientAnswer() : void
SendServerQuestion() : void
ForwardQuestion() : void
ForwardAnswer() : void
ClientQuestionAnswered() : void
ServerQuestionGenerated() : void
refresh Message
query
Emilia Specification
ARCHI_TYPE Domain_Name_System
ARCHI_ELEM_TYPES
ELEM_TYPE DNSClient_Type(void)
BEHAVIOR
DNSClient(void; void) =
<SendQuestion, exp(4.88)> . <AnswerArrived, _>
DNSClient()
INPUT_INTERACTIONS
UNI AnswerArrived
OUTPUT_INTERACTIONS
UNI SendQuestion
ELEM_TYPE Messenger_Type(void)
BEHAVIOR
Messenger(void; void) =
choice
{
<ClientQuestionArrived, _> . <ForwardQuestion,
<ServerAnswerArrived, _> . <ForwardAnswer, inf
<ClientQuestionAnswered, _> . <SendClientAnswe
<ServerQuestionGenerated, _> . <SendServerQues
}
INPUT_INTERACTIONS
Throughput
Save
Tagged Values
Checklist
Properties
Documentation
Style
Modifiers:
Parameters:
return
Concurrency:
Raised Signals:
none
Invocation Style:
Event Ocuurence:

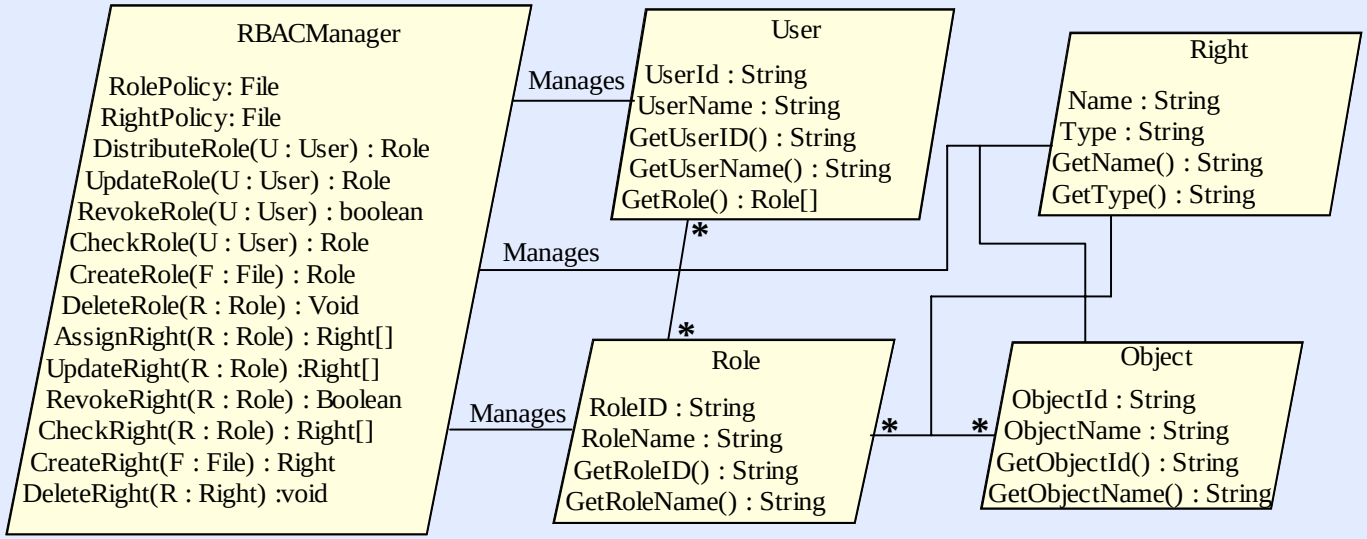
3. Throughput aspect analysis results in Æmilia



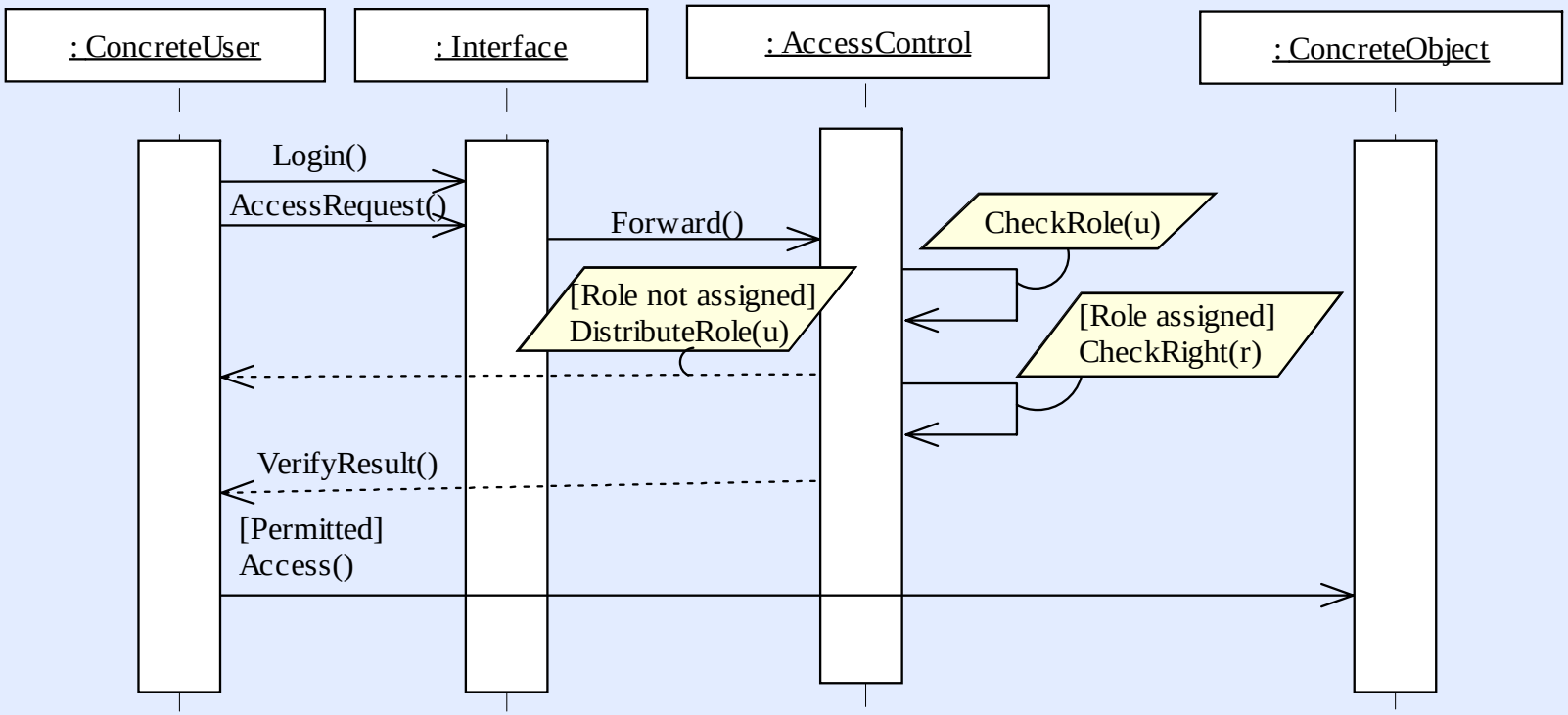
# of Clients	States	Transitions	Throughput
1	5	5	3.92283
2	11	13	2.86861
3	19	24	1.59453
4	29	38	1.15995
...	...	...	...
36	1405	2070	0.113107
37	1481	2183	0.109997
38	1559	2299	0.107054
39	1639	2418	0.104264
40	1721	2540	0.101617
41	1805	2665	0.0991002

FDAF Approach Illustrations : Role-Based Access Control Security Aspect

- Substantive aspect
- Aspect definition
  - Adapted from role-based access control security pattern
  - Advanced access control because it reduces the complexity and cost of security administration
  - Concepts: user, role, right, object
  - Roles assigned to users and rights assigned to roles
  - Static view

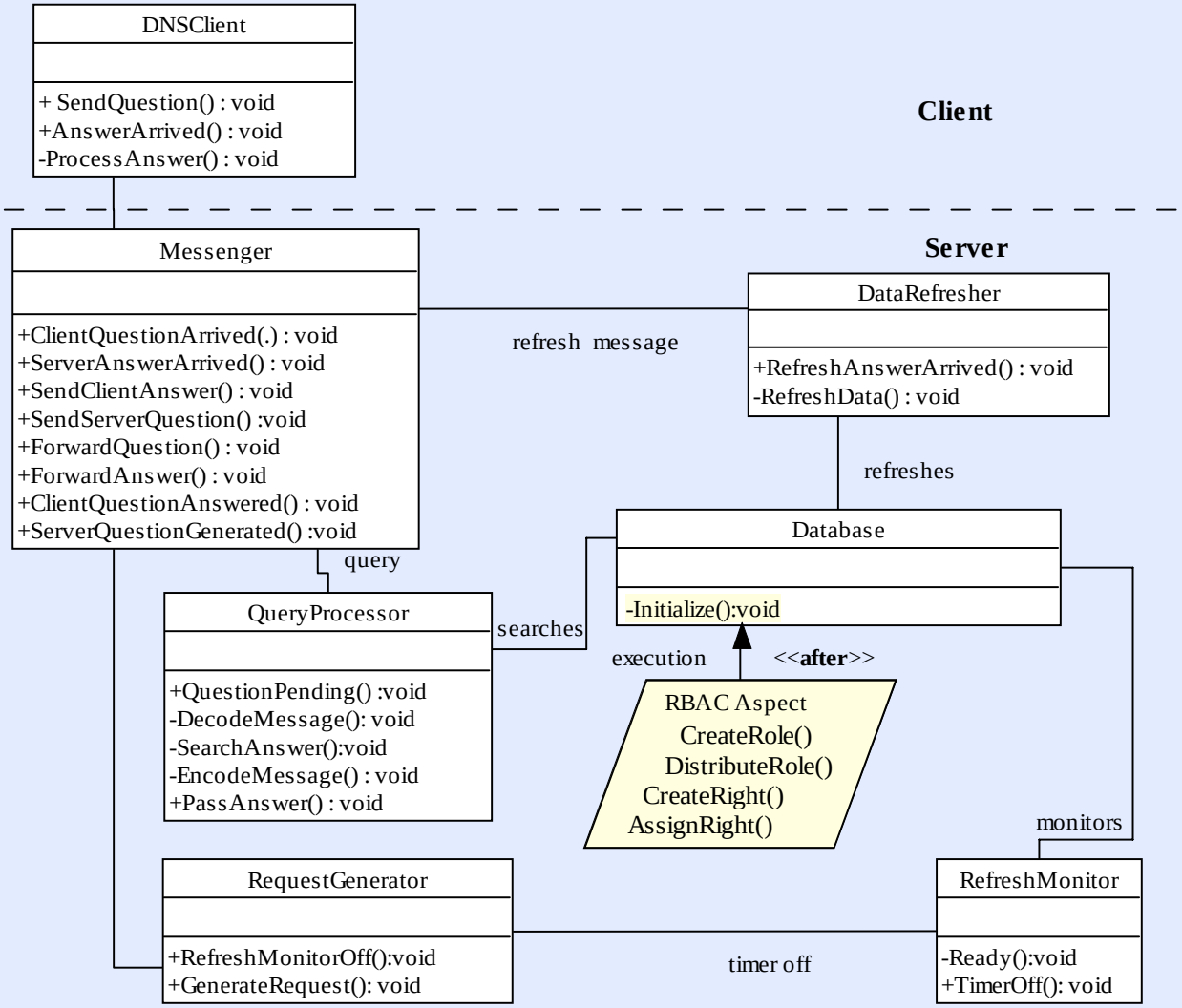


- Dynamic view



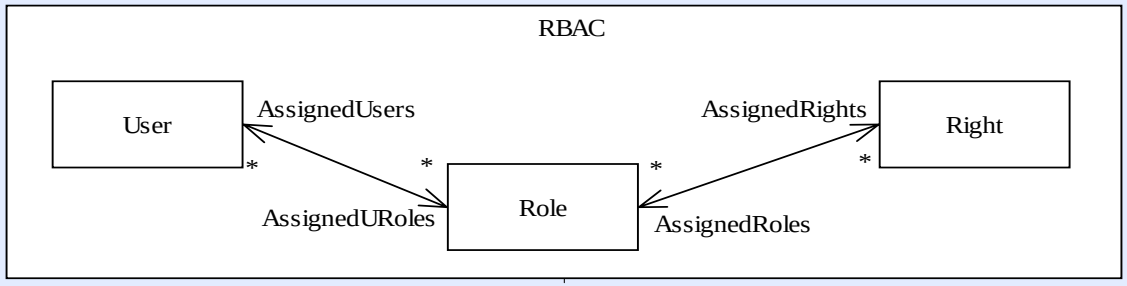
DNS Illustration

1. Extend DNS architecture design with role-based access control aspect



2. Automatically generate Alloy formal specification

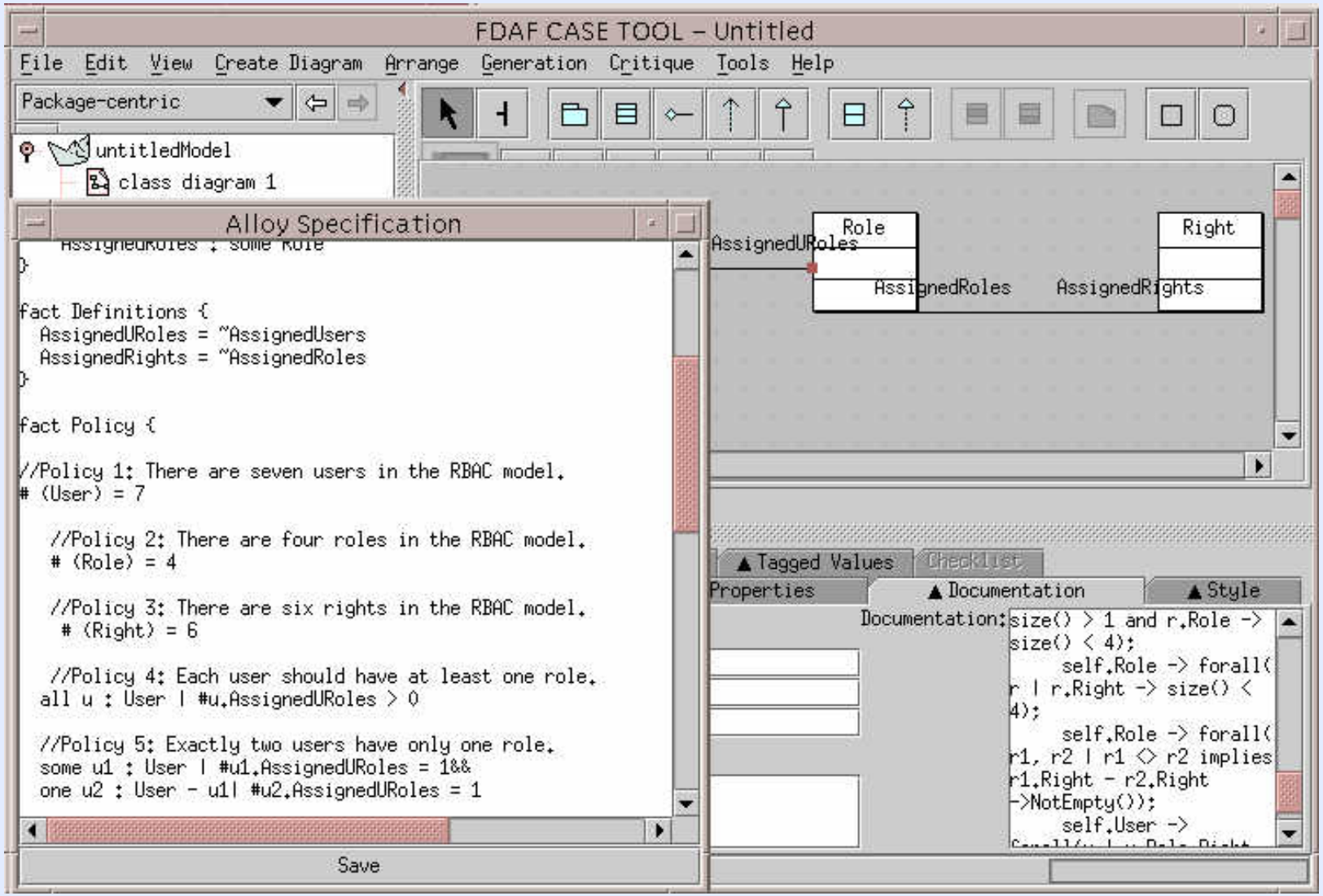
- Why formal analysis?
  - Prevent inconsistency of role-based access control policies
- How?
  - Access control policies defined in OCL for the DNS role-based access control model
  - Create Alloy formal specification for the design
  - Use Alloy's analysis tool to discover design errors



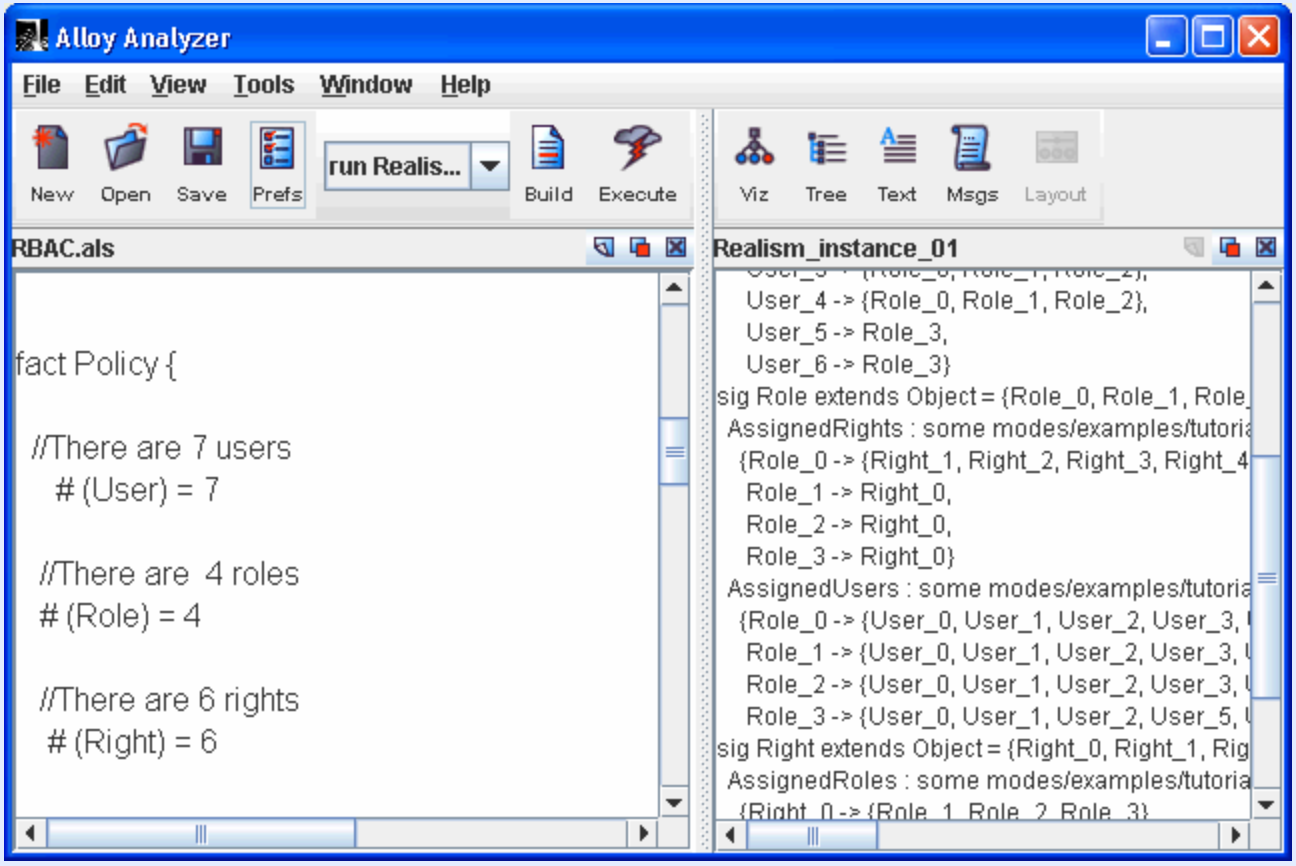

```
Context RBAC inv
    self.User -> size() = 7;
    self.Role -> size() = 4;
    self.Right -> size() = 6;
    self.User -> forall(u | u.Role -> size() > 0);
    self.User -> select(u | u.Role -> size() = 1) -> size() = 2;
    self.User -> forall(u | u.Role -> size() < 3);
    self.Role -> forall(r | r.User -> size() > 0);
    self.Role -> forall(r | r.User -> size() < 4);
    self.Role -> one(r | r.User -> size() = 1);
    self.Role -> forall( r | r.Right -> size() = 1);
    self.Right -> forall( r | r.Role -> size() > 0);
    self.Rigth -> forall( r | r.Role -> size() > 1 and r.Role -> size() < 4);
    self.Role -> forall( r | r.Right -> size() < 4);
    self.Role -> forall( r1, r2 | r1 <> r2 implies r1.Right - r2.Right ->NotEmpty());
    self.User -> forall(u | u.Role.Right -> size() < 3);
```

- Alloy
  - Software Design Group
  - Based on set theory and first order logic
  - Analysis tool is a constraint solver that provides automatic simulation and checking
  - Moder finder
- Mapping form extended UML class diagram with OCL into Alloy
  - UML class diagram is used to describe the static structure
    - UML classes and interfaces are translated to Alloy signatures
    - A UML association is translated into a relationship (or a filed) for the Alloy signature
  - OCL invariants are translated to equivalent Alloy formulae UML

OCL	Equivalent Alloy Expression
X-> size() Op y	#X Op y or #(X) Op y
X-> forall(x | P(x))	all x : X | P(x)
X-> exists(x | P(x))	some x : X | P(x)
X-> one(x | P(x))	one x : X | P(x)
X-> notEmpty()	some X
X-> isEmpty()	no X
X-> select(x | P(x))-> size() = 1	one x : X | P(x)
X-> select(x | P(x))-> size() = y (y>1)	some x_1 : X | P(x_1) && some x_2 : X-x_1 | P(x_2) && …some x_{y-1} : X- (x_1 + x_2 + … x_{y-2}) | P(x_{y-1}) && one x_y : X- (x_1 + x_2 + … x_{y-1}) | P(x_y)
X-> select(x | P(x))-> size() > y (y>0)	some x_1 : X | P(x_1) && some x_2 : X-x_1 | P(x_2) && …some x_y : X- (x_1 + x_2 +… x_{y-1}) | P(x_y)
X-> select(x | P(x)) -> notEmpty()	some x : X | P(x)
X-> select(x | P(x)) -> isEmpty()	no x : X | P(x)
X-> union(Y)	X + Y
X->intersection(Y)	X & Y
X-Y	X-Y
P and Q	P && Q
P or Q	P || Q
P implies Q	P => Q
a <> b	a != b
a = b	a = b


FDAF CASE TOOL – Untitled
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
Alloy Specification
}
fact Definitions {
AssignedURoles = ~AssignedUsers
AssignedRights = ~AssignedRoles
}
fact Policy {
//Policy 1: There are seven users in the RBAC model.
(User) = 7
//Policy 2: There are four roles in the RBAC model.
(Role) = 4
//Policy 3: There are six rights in the RBAC model.
(Right) = 6
//Policy 4: Each user should have at least one role.
all u : User | #u.AssignedURoles > 0
//Policy 5: Exactly two users have only one role.
some u1 : User | #u1.AssignedURoles = 1&&
one u2 : User - u1| #u2.AssignedURoles = 1
Save
Role
AssignedURoles
AssignedRoles
Right
AssignedRights
Tagged Values
Checklist
Properties
Documentation
Style
Documentation:
size() > 1 and r.Role ->
size() < 4);
self.Role -> forall(
r | r.Right -> size() <
4);
self.Role -> forall(
r1, r2 | r1 <> r2 implies
r1.Right - r2.Right
->NotEmpty());
self.User ->

3. Role-based access control aspect analysis results in Alloy



Access Policies

#1: 7 users
#2: 4 roles
#3: 6 rights
#4: Each user should have at least one role

#5: Exactly two users have only one role

#6: No user has more than two roles
#7: All roles have been assigned
#8: No role can be assigned to more than half the users

Solutions

```
……
 AssignedURoles : some modes/examples/tutorials/RBAC/Role =
   {User_0 -> Role_0,  User_1 -> Role_1,
    User_2 -> {Role_2, Role_3},  User_3 -> {Role_2, Role_3},
    User_4 -> {Role_2, Role_3},  User_5 -> {Role_2, Role_3},
    User_6 -> {Role_2, Role_3}}
……
```

```
……..
 AssignedURoles : some modes/examples/tutorials/RBAC/Role =
   {User_0 -> Role_0, User_1 -> {Role_0, Role_1, Role_2, Role_3},
    User_2 -> {Role_0, Role_1, Role_2, Role_3},
    User_3 -> {Role_0, Role_1, Role_2, Role_3},
    User_4 -> {Role_0, Role_1, Role_2},
    User_5 -> {Role_0, Role_1, Role_2},  User_6 -> Role_1}
……
```

```
……
 AssignedURoles : some modes/examples/tutorials/RBAC/Role =
   {User_0 -> Role_0, User_1 -> {Role_0, Role_1},
    User_2 -> {Role_0, Role_2}, User_3 -> {Role_1, Role_3},
    User_4 -> {Role_1, Role_3}, User_5 -> {Role_2, Role_3},
    User_6 -> Role_2}
AssignedUsers : some modes/examples/tutorials/RBAC/User =
   {Role_0 -> {User_0, User_1, User_2},
    Role_1 -> {User_1, User_3, User_4},
    Role_2 -> {User_2, User_5, User_6},
    Role_3 -> {User_3, User_4, User_5}}
```

Access Policies

#9: There is one role that can only be assigned to one user

Inconsistency: Policy #5 and Policy #9 together over-constraint the model

#5 Updated from “Exactly two users have only one role” to “At least two users only has one role “

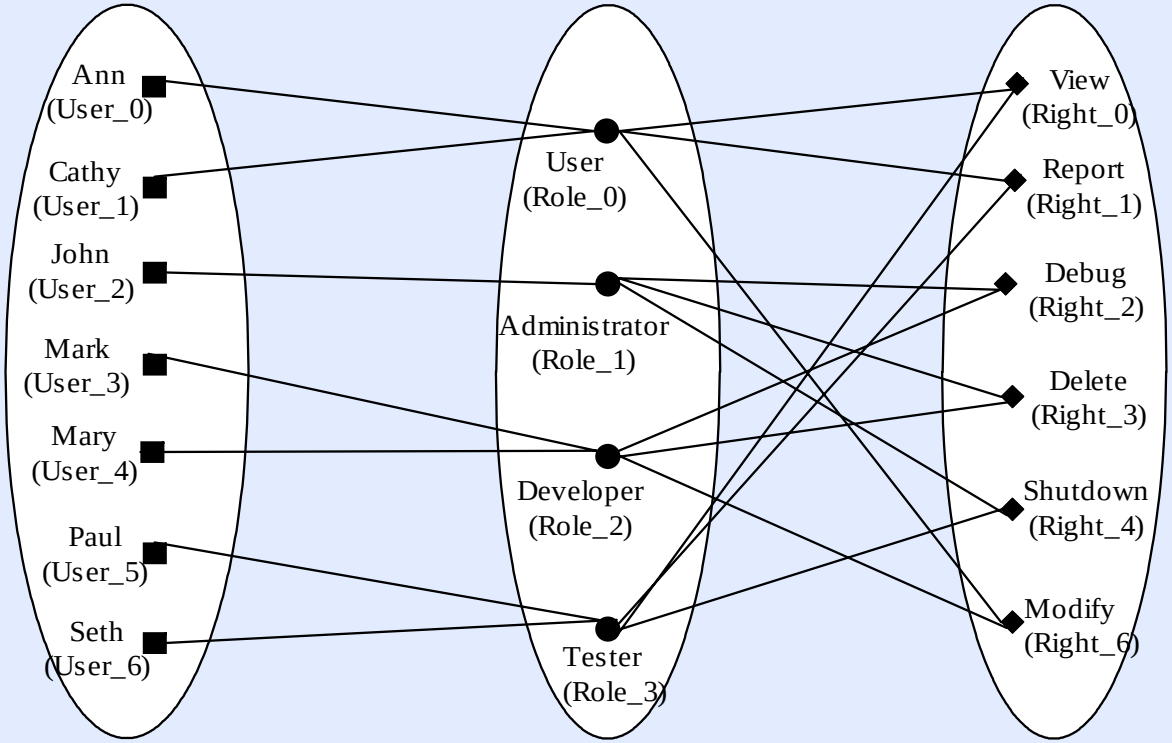
#10: All roles should have at least one right
#11: All rights have been assigned
#12: One right should be assigned to at least two roles, but at most three roles
#13: No role should have more than half the rights
#14: Different roles should not have the same set of rights
#15: No user can have more than half the rights

Solutions

Compiling RBAC.als ...
Compilation successful.
Executing command run Realism for 2 but 17 modes/examples/tutorials/RBAC/Object...
Instance found: Realism is consistent. (00:50)
Compiling RBAC.als ...
Compilation successful.
Executing command run Realism for 2 but 17 modes/examples/tutorials/RBAC/Object...
No instance found: Realism may be inconsistent. (00:49)

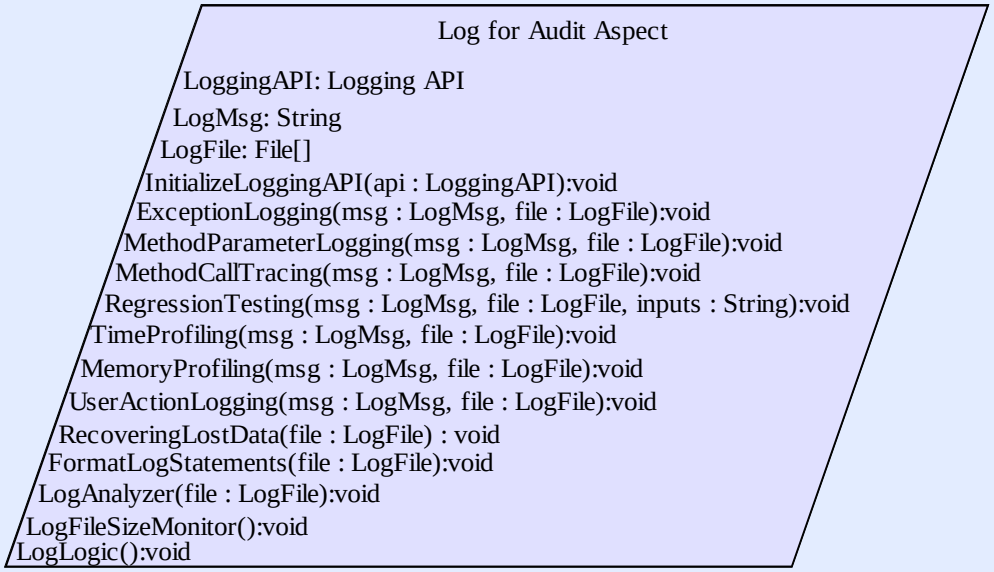
……
{User_0 -> Role_0, User_1 -> Role_0, User_2 -> Role_1, User_3 -> Role_2, User_4 -> Role_2, User_5 -> Role_3, User_6 -> Role_3}
……
{Role_0 -> {Right_0, Right_1, Right_5}, Role_1 -> {Right_2, Right_3, Right_4}, Role_2 -> {Right_2, Right_3, Right_5}, Role_3 -> {Right_0, Right_1, Right_4}}
……
{Role_0 -> {User_0, User_1}, Role_1 -> User_2, Role_2 -> {User_3, User_4}, Role_3 -> {User_5, User_6}}
……
{Right_0 -> {Role_0, Role_3}, Right_1 -> {Role_0, Role_3}, Right_2 -> {Role_1, Role_2}, Right_3 -> {Role_1, Role_2}, Right_4 -> {Role_1, Role_3}, Right_5 -> {Role_0, Role_2}}

Role assignment and right assignment for DNS RBAC Model



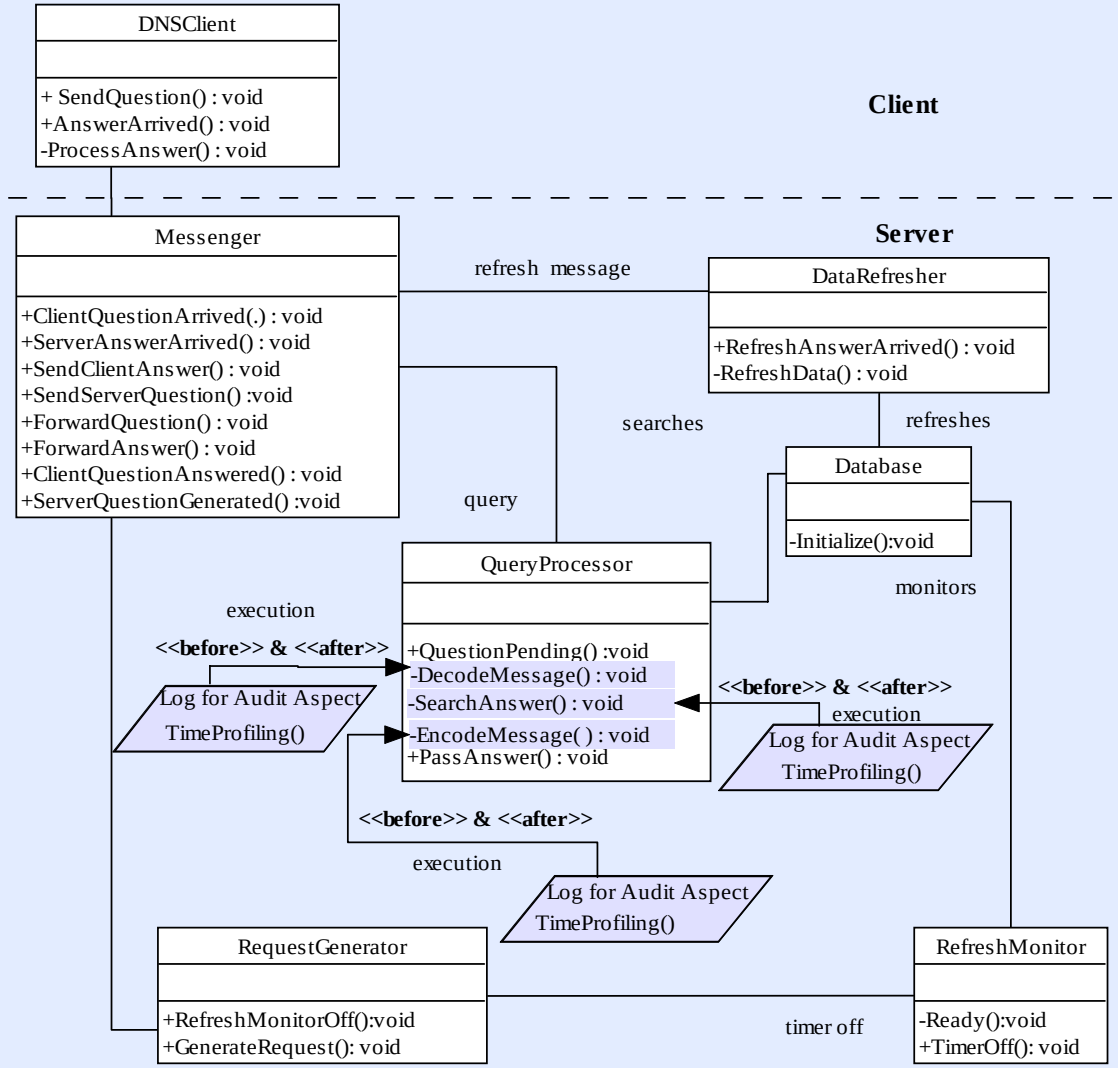
FDAF Approach Illustrations : Log for Audit Security Aspect

- Substantive aspect
- Aspect definition
  - Adapted from log for audit security pattern
  - Systematic evaluation of company information security measuring to established criteria
  - Ties logging to auditing, to ensure that logging is configured with audit in mind and auditing is understood to be integral to effective logging
  - Static view



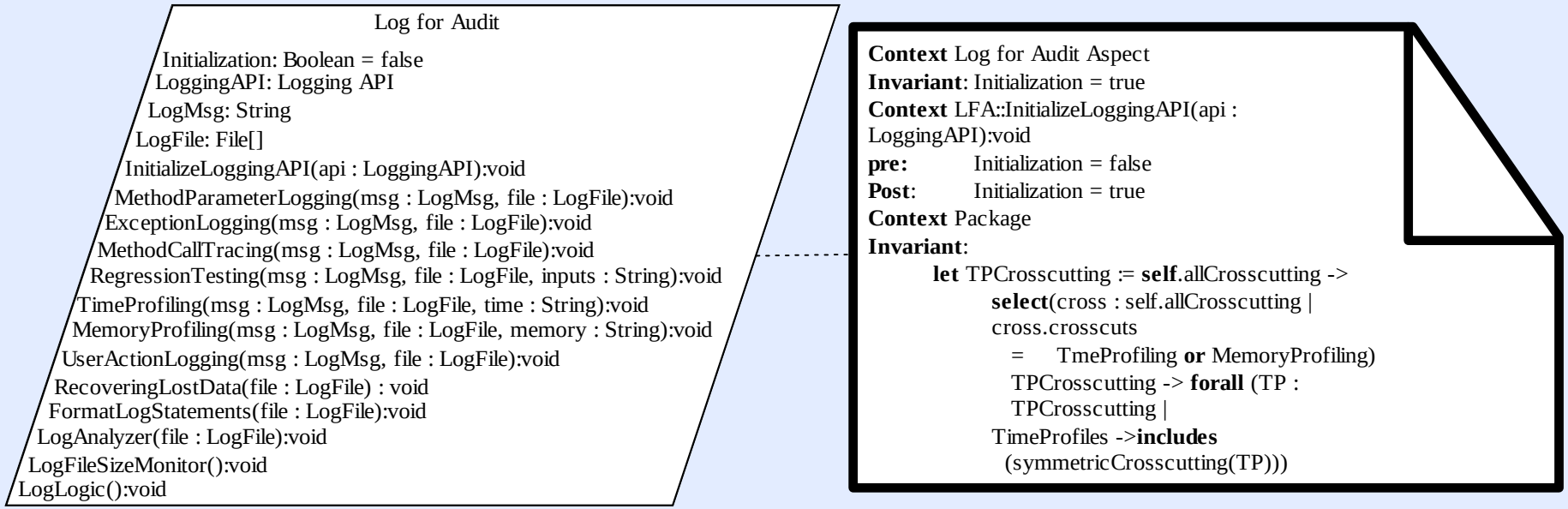
DNS Illustration

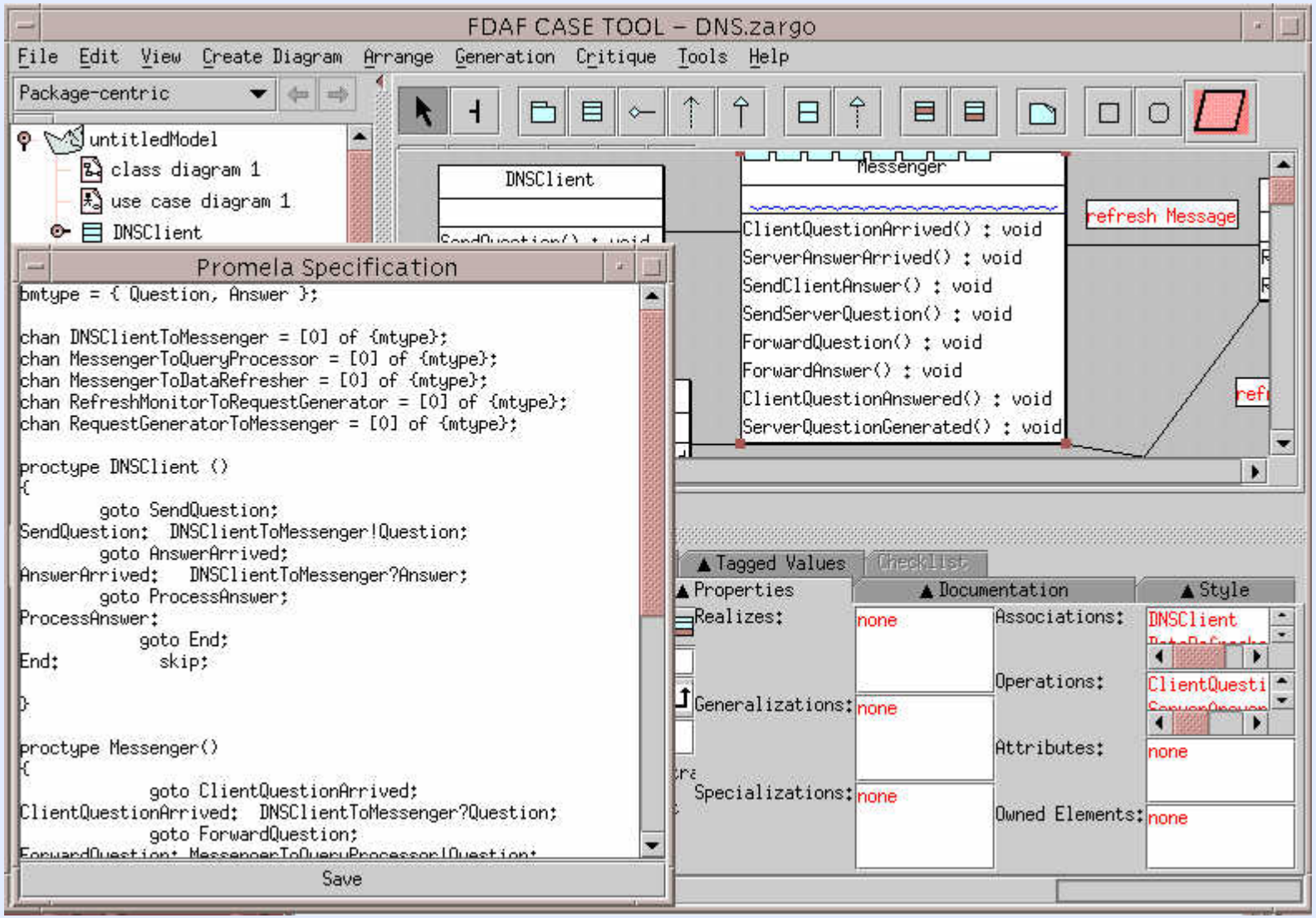
1. Extend DNS architecture design with log for audit aspect



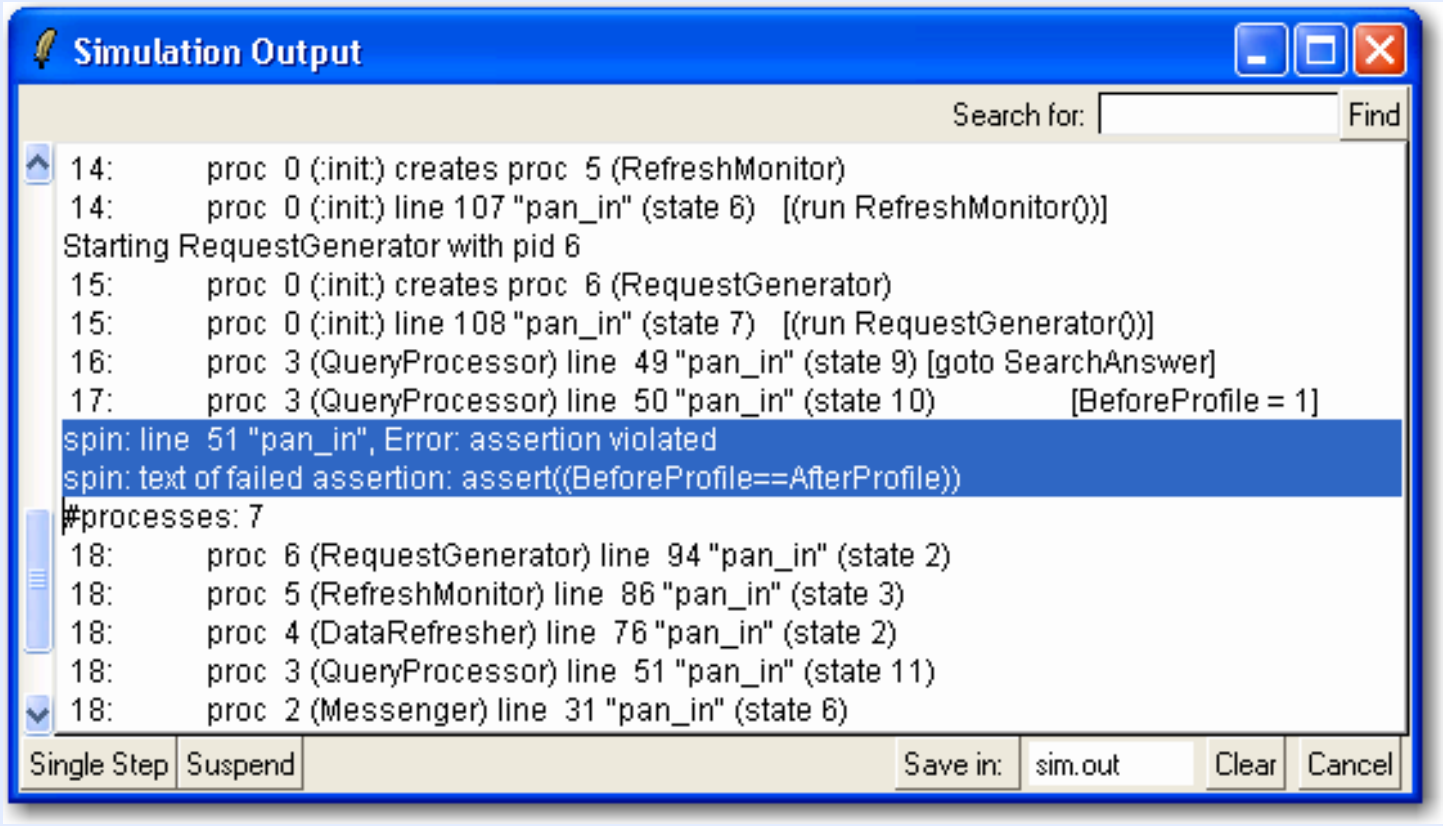
2. Automatically generate Promela formal specification

- Why formal analysis?
  - Prevent inconsistency of an aspect-oriented design
    - "TimeProfiling" and "MemoryProfiling" should crosscut a UML operation twice (before and after)
- How?
  - OCL constraints defined for the log for audit aspect
  - Create Promela formal specification for the design
  - Use model checker Spin to discover design errors

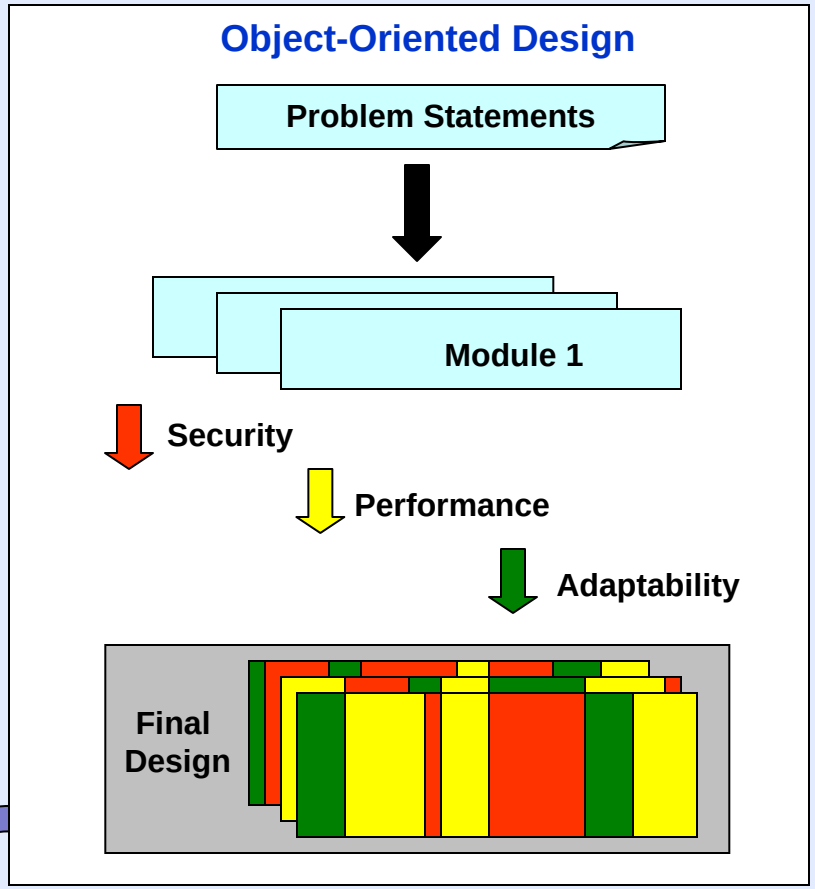


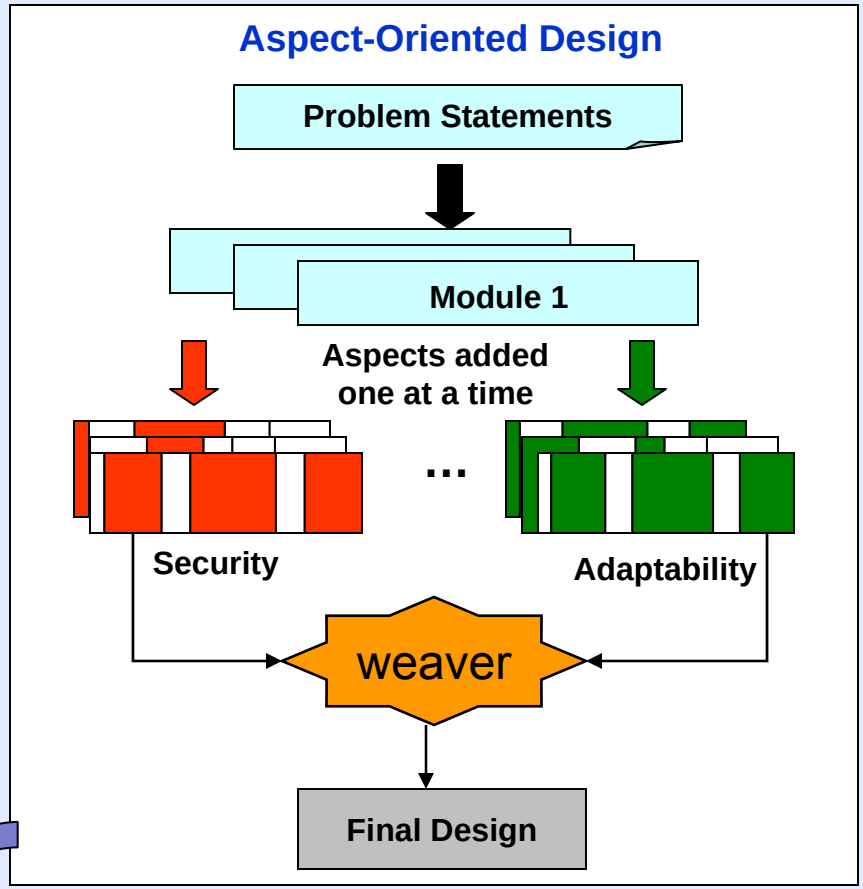

FDAF CASE TOOL – DNS.zargo
File Edit View Create Diagram Arrange Generation Critique Tools Help
Package-centric
untitledModel
class diagram 1
use case diagram 1
DNSClient
Promela Specification
mtype = { Question, Answer };
chan DNSClientToMessenger = [0] of {mtype};
chan MessengerToQueryProcessor = [0] of {mtype};
chan MessengerToDataRefresher = [0] of {mtype};
chan RefreshMonitorToRequestGenerator = [0] of {mtype};
chan RequestGeneratorToMessenger = [0] of {mtype};
proctype DNSClient ()
{
goto SendQuestion;
SendQuestion: DNSClientToMessenger!Question;
goto AnswerArrived;
AnswerArrived: DNSClientToMessenger?Answer;
goto ProcessAnswer;
ProcessAnswer:
goto End;
End: skip;
}
proctype Messenger()
{
goto ClientQuestionArrived;
ClientQuestionArrived: DNSClientToMessenger?Question;
goto ForwardQuestion;
Save
DNSClient
Messenger
ClientQuestionArrived() : void
ServerAnswerArrived() : void
SendClientAnswer() : void
SendServerQuestion() : void
ForwardQuestion() : void
ForwardAnswer() : void
ClientQuestionAnswered() : void
ServerQuestionGenerated() : void
refresh Message
Tagged Values
Checklist
Properties
Documentation
Style
Realizes:
none
Generalizations:
none
Specializations:
none
Associations:
DNSClient
Operations:
ClientQuesti
Attributes:
none
Owned Elements:
none

3. Log for audit aspect analysis results in Promela



Indicates a defect in the design, a mismatch in using profiling operations. Architects need to go back to revise their design.





Problem: lower cohesion, higher coupling, maybe significant re-work when facing changes

Advantages: Tangling aspects handled separately, Dramatically reduce the complexity of understanding, change, and analysis, promoting reusability

Rapide Timing Model

- A clock is a monotonically increasing counter
- The rate at which it increases is not related to any physical unit but rather is controlled by a computation
- An increase in time as a tick. Clocks tick when there are no more events to be generated at the current time
- An event may be timed with respect to one or more clocks.
- Events have start and finish times modeling event duration

Armani Performance Analysis Theory

- Queuing network theory
- Little's law
  - *<<sNetworkPop>>.value = <<sArrivalRate>>.value × <<sNetworkResponse>>.value*
  - *M/M/1 queue*
    - *<<sUtilization>>.value = <<sArrivalRate>>.value × <<sVisits>>.value × <<sServiceTime>>.value*
    - *<<sResponseTime>>.value = <<sServiceTime>>.value / (1 - <<sUtilization>>.value)*
    - *<<sLength>>.value = <<sArrivalRate>>.value × <<sVisits>>.value × <<sResponseTime>>.value*
- Adaptation at the software architecture level
  - *<<sNetworkRespone>>.value =*

$$\sum_{i=1}^{m} Component_i.<<sResponseTime>>.value \times Component_i.<<sVisits>>.value +$$

$$\sum_{j=1}^{n} Connector_j.<<sVisitis>>.value \times Connector_j.<<sDelayTime>>.value$$

  - *<<sNetworkPop>>.value = <<sArrivalRate>>.value × <<sNetworkResponse>>.value*

Æmilia Performance Analysis Theory

- Markov chain

$$P\left[\chi(t) = x \mid \chi(t_n) = x_n, \chi(t_{n-1}) = x_{n-1}, \ldots, \chi(t_0) = x_0\right] = P\left[\chi(t) = x \mid \chi(t_n) = x_n\right]$$

- Reward structures
  - A yield function $y_{i,j}(t)$ expressing the rate at which reward is accumulated at state *i* *t* time units after *i* was entered when the successor state is *j*
  - A bonus function $b_{i,j}(t)$ expressing the reward awarded upon exit from state *i* and subsequent entry into state *j* given that the holding time in state *i* was *t* time units

VHDL example (Video)

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;

Entity CONTROL Is
        Port (
                Reset: in std_logic;
                Clk: in std_logic;
                Mode: in std_logic;
            ……);
End CONTROL;

Architecture CONTROL_A of CONTROL Is
        constant FRAMESIZE: integer := 253243;
        constant TESTADDR: integer := 253000;

        signal ENDFR: std_logic;
        signal INCAD: std_logic;
        signal VS: std_logic;
    ……
Begin
        ADDRCTR: process(Clk)
        variable cnt: unsigned (17 downto 0);
        begin
            if rising_edge(Clk) then
                if TestLoad = '1' then cnt := to_unsigned(TESTADDR,18); ENDFR <= '0';
                else
        ……
End CONTROL_A;
```

Standard ML Example (Mergesort)

```
-fun merge([],y) = y
= | merge(x,[]) = x
= | merge(a::x,b::y) =
= if a < b then a::merge(x,b::y)
= else b::merge(a::x,y);
val merge = fn : int list * int list -> int list - merge([1,4,5],[2,3,4]);
val it = [1,2,3,4,4,5] : int list
```

Process Algebra

$$
\begin{aligned}
Sender &:= \mu\alpha.m!.a?.\alpha \\
Receiver &:= \mu\beta.m?.a!.\beta \\
System &:= (\nu m)(\nu a)(Sender \mid Receiver)
\end{aligned}
$$

CSP Example (Consumer & Producer)

```
[ Buffer::
    buf (0 .. bufsize-1)  : buffer-element;
    first, last           : integer;    -- queue pointers
    j, k                  : integer;    -- process counters
    first := 0;
    last  := 0;
    *[  (j: 1..numbprod)                    -- For each of the numbprod
                                               producers,
        (last+1) mod bufsize ≠ first;       -- if there is room in the buffer,
            Producer(j) ? buf(last) →       -- read an element.
                last := (last + 1) mod bufsize
     □
       (k: 1..numbcons)                     -- For each of the numbcons
                                               consumers,
        first ≠ last;                       -- if there is something in the
                                               buffer
            Consumer(k) ? more( ) →         -- and a consumer signals a
                                               desire to consume,
                Consumer(k) ! buf(first);   -- send that consumer an
                                               element.
                first := (first + 1) mod bufsize ]
  -- The buffer runs concurrently with the producers and consumers.
     PRODUCER is the text of the producer processes; CONSUMER, of the
     consumer processes.
|| (i: 1..numbprod) PRODUCER
|| (i: 1..numbcons) CONSUMER ]
```