

Abstract Syntax Tree Metamodel Standard

ASTM Tutorial 1.0

Washington, DC
October, 2005

Philip Newcomb
Chief Executive Officer
The Software Revolution, Inc. (TSRI)
11410 NE 122nd Way
Kirkland, Washington 98034
Phone: 425-284-2770

philip.newcomb@softwarerevolution.com

www.softwarerevolution.com



Instructor BIO & Contact Info

Philip H. Newcomb

Chief Executive Officer, Chief Technology Officer
The Software Revolution, Inc
11410 NE 122nd Way, Suite 304
Kirkland, WA 98034
Phone: 425 284 2770

Email: philip.newcomb@softwarerevolution.com

Corporate Web Site: www.softwarerevolution.com

Case Study Web Site: www.legacysystemmodernization.com

Value Proposition Web Site: www.automatedsoftwaremodernization.com

Philip Newcomb is CEO/CTO of The Software Revolution, Inc (TSRI), a company that has completed over 35 automated modernization projects for systems as diverse as satellite command and control, strategic warfare planning, ballistic missile early warning, health care, logistics, engineering operational sequencing, etc. Philip is a prominent contributor to the Object Management Group (OMG) ADM task force that is defining industry-based modeling standards and best practices for Architecture Driven Modernization (ADM) to support Model Driven Architecture (MDA). He leads the joint submission team of ADM TF members that is defining the ASTM standard. Philip was a research scientist at Boeing Artificial Intelligence Laboratory for 12 years before founding TSRI in 1994. He was co-Chair of the Working Conference for Reverse Engineering (WCRE) in 1995. With over 30 technical publications and a wealth of practical knowledge, Philip has contributed at the intersection of the fields of reverse-engineering, automatic programming and formal methods for over 20 years.

The Software Revolution is building OMG ADM compliant services, products and technology under his technical and executive leadership.

Abstract

It has long been observed that although there are many differences between the statements in many programming languages there is a very large set of statements that are common across most languages. The Object Management Group (OMG) Architecture Driven Modernization Task Force has issued an RFP for the Abstract Syntax Tree Meta Modeling standard. The ASTM seeks to establish a single comprehensive set of modeling elements for capturing how many software languages represent the same software language constructs and concepts. Software analysis and transformation tools that use the ASTM will achieve broader applicability through software language independence at the model level. This tutorial will present the vision of the ASTM, its business significance, its relationship to other OMG standards, including the Knowledge Discovery Metamodel (KDM), the OMG Model Driven Architecture (MDA) and Unified Modeling Language (UML). The perspectives of the ASTM submission teams, and their progress towards a unified definition for the ASTM standard will be presented

What is the ADM and its Mission?

- **The OMG chartered the Architecture Driven Modernization Task Force in 2003 to extend MDA practices and standards to existing systems.**
- **The Task Force cochairs are**
 - Bill Ulrich (Tactical Strategy Group)
 - Djenana Campara (Klocwork)
- **The OMG Architecture Driven Modernization Task Force (ADM TF) has defined the following 4 goals:**
 - **The ultimate goal - Revitalization of Existing Applications**
 - **Make existing applications more agile**
 - **Leverage existing OMG modeling standards and the MDA initiative**
 - **Consolidate best practices leading to successful modernization**

(<http://adm.omg.org/>) 4

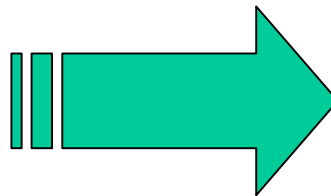
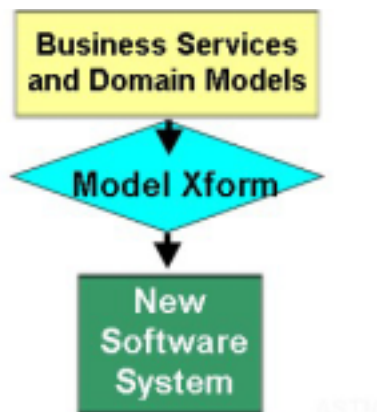
How does ADM support MDA?

MDA is a **top-down** model-driven process for new system development.

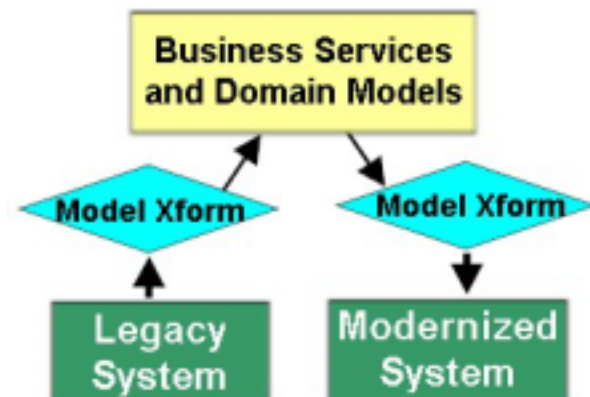
- *Architectural models provide portability, interoperability and reusability through architectural separation of concerns*
- *Architectural models direct the course of understanding, design, construction, deployment, operation, maintenance and modification.*

ADM incorporates **bottom-up** extraction of architectural models followed by top down reuse in MDA processes and scenarios for legacy systems modernization.

THEN (before ADM)

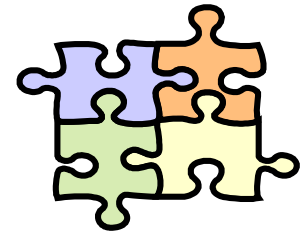


NOW (MDA + ADM)



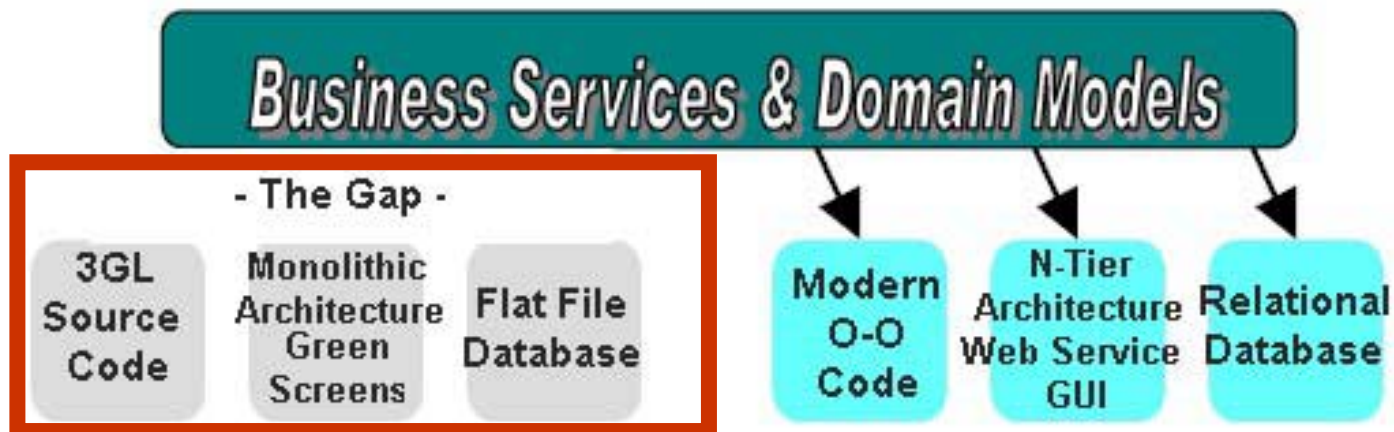
Why does MDA need ADM?

- **ADM closes the gap between methodologies tools available for new and old software**



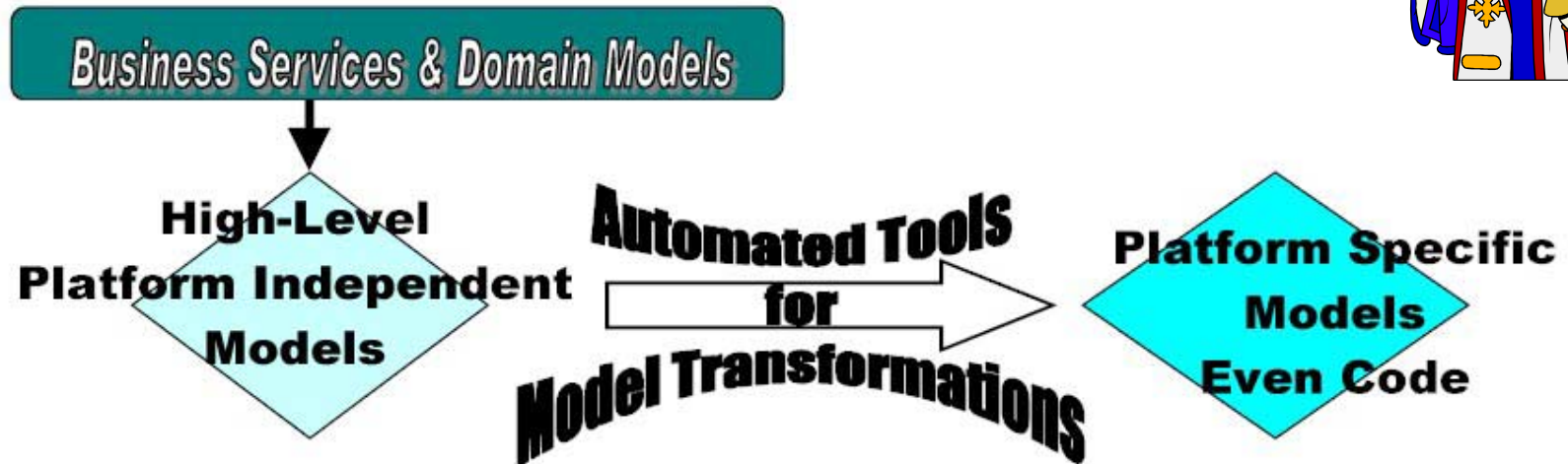
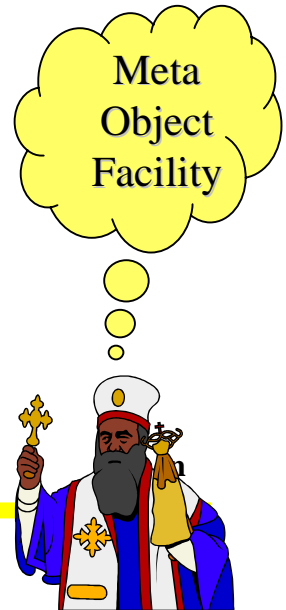
- *“We are addressing a big gap between methodologies and tools that are available for developing new software — for example, the OMG MDA — and the total lack of tools and methodologies that help you to be successful with your existing code.”*

(William Ulrich, 2003)



How do ADM Tools fit into the MDA?

- ADM tools must adhere to the standards for software modeling, as defined by the Object Management Group (OMG).
 - *Technically no ADM tools exist yet ... because the OMG has not yet approved any ADM Task Force Platform Committee standards.*
 - *To be compliant with OMG MDA principles, ADM tools must be compliant with the OMG MOF*



What Are Standards-Based Tools? (and why should you prefer them?)

- **Standard-Based Enterprise**
 - IT Artifacts (such as Data and Applications) are Managed In Repositories As Standards-Based Models
 - IT Artifact Management employs Standards-Based Service Providers, Standards-based Tools, Standards-based Tool Chains, and Standards-Based Tool Chains
- **Standard-Based Service Providers**
 - Offer Services Based Upon Standards-Based Tools, Tool Chains and Tool Suites
 - Collaborate with Customers and Other Services Providers Via Interchange of Industry Standard Models (e.g. Eclipse, MOF™, XMI™, UML™)
- **Standards-Based Tools**
 - Store and Access Models in Industry Standard Formats (XMI™)
 - Interoperate With Other Tools By Interchanging Standards-Based Models
- **Standards-Based Tool Chains**
 - A Series of Tools that Cooperatively Produce A Work Product
 - Cooperate By Interchanging Standards-Based Models.
- **Standards-Based Tool Suites**
 - Collections of Tools that Interoperate Using Standards-based Models

What are the ADM Scenarios?

MODERNIZATION GOAL: Pick One or More (Or Extend the List)

Modernization Scenario	Check Box
I. Application Portfolio Management	
II. Application Improvement	
III. Language-to-Language Conversion	
IV. Platform Migration	
V. Non-Invasive Application Integration	
VI. Services Oriented Architecture Transformation	
VII. Data Architecture Migration	
VIII. Application & Data Architecture Consolidation	
IX. Data Warehouse Deployment	
X. Application Package Selection & Deployment	
XI. Reusable Software Assets / Component Reuse	
XII. Model-Driven Architecture Transformation	

See Handout Bill Ulrich, ADM Scenarios White Paper 04-09-04⁹

Why define ADM Scenarios?

- **Helps envision all potential ADM applications.**
- **Helps a user determine the tasks, tools and use of the ADM.**
- **Provides templates for crafting project objectives, plans and related deliverables.**
- **Defines tasks necessary to complete a given modernization initiative and omits unnecessary tasks that would not apply to such a scenario.**
- **Allows a user to pinpoint the types of tools necessary to perform these tasks.**
- **Identifies the universe of modernization scenarios and tasks and provides a guide as to the role of the ADM within modernization in general.**

Bill Ulrich, ADM Scenarios White Paper 04-09-04

How do you use Modernization Scenarios?

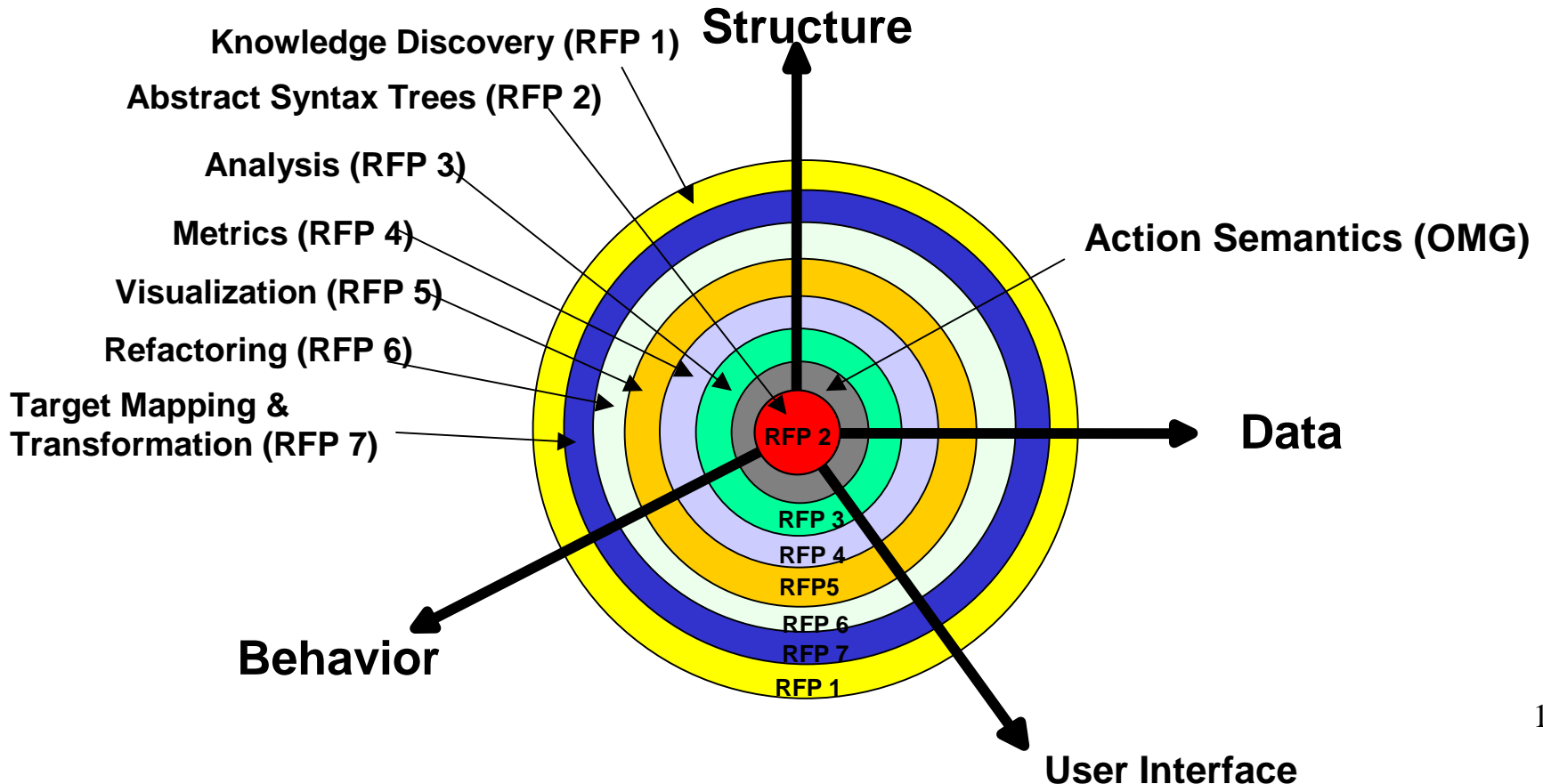
- **Use Scenarios to define approach to various application improvement, migration and redesign initiatives that an organization may pursue.**
- **Use Scenarios to pinpoint the types of tools necessary to perform these tasks.**
- **Mixed and Match Scenarios based upon the organization's Modernization goals.**
- **Language-to-language conversion might be coupled with a platform migration for example.**
- **Other scenarios may be added to this list from time to time...**

Bill Ulrich, ADM Scenarios White Paper 04-09-04

What is the ADM Roadmap?



To structure its work the ADM PTF platform task force is defining seven interrelated standards, starting with the KDM followed by the ASTM.



ADM Roadmap Synopsis

•1. Knowledge Discovery Meta-Model Package (KDM) – (adopt '05)

- establishes an initial meta-model
- allows modernization tools to exchange application meta-data across applications, languages, platforms and environments
- provides a comprehensive (inventory) view of application structure and data, but does not represent software below the procedure level

•2. Abstract Syntax Tree Meta-Model Package (ASTM) – (adopt '06)

- adds representation of software below the procedural level
- allows full representation of applications and facilitates the exchange of granular meta-data at the translation level across multiple languages
- Unifies all syntactical language models into a common abstract syntax meta model.

•3. Analysis Package (AP) – (initiate '05)

- facilitates the examination of structural meta-data with the intent of deriving detailed behavioral and structural meta-data about systems
- may take the form of design patterns, business rules or other aspects of a system that are not an apparent part of the structure of the system, but are rather semantic derivations of that structure and the data

•4. Metrics Package (MP) – (initiate '06)

- derive numeric metrics that describes measurable technical, functional and architectural properties from the structural and the behavioral aspects of the applications of interest and its data.
- supports planning and estimating, ROI analysis and the ability of the enterprise to maintain application and data quality

ADM Roadmap Synopsis (cont.)

5. Visualization Package (VP) – (initiate '07)

- builds on KDM, ASTM, AP and MP
 - focuses on ways to depict application meta-data stored within ADM models.
 - May include any variety of views appropriate or useful for planning and managing modernization initiatives
 - Examples include the use of graphs or charts, metric summaries or standardized development models
-

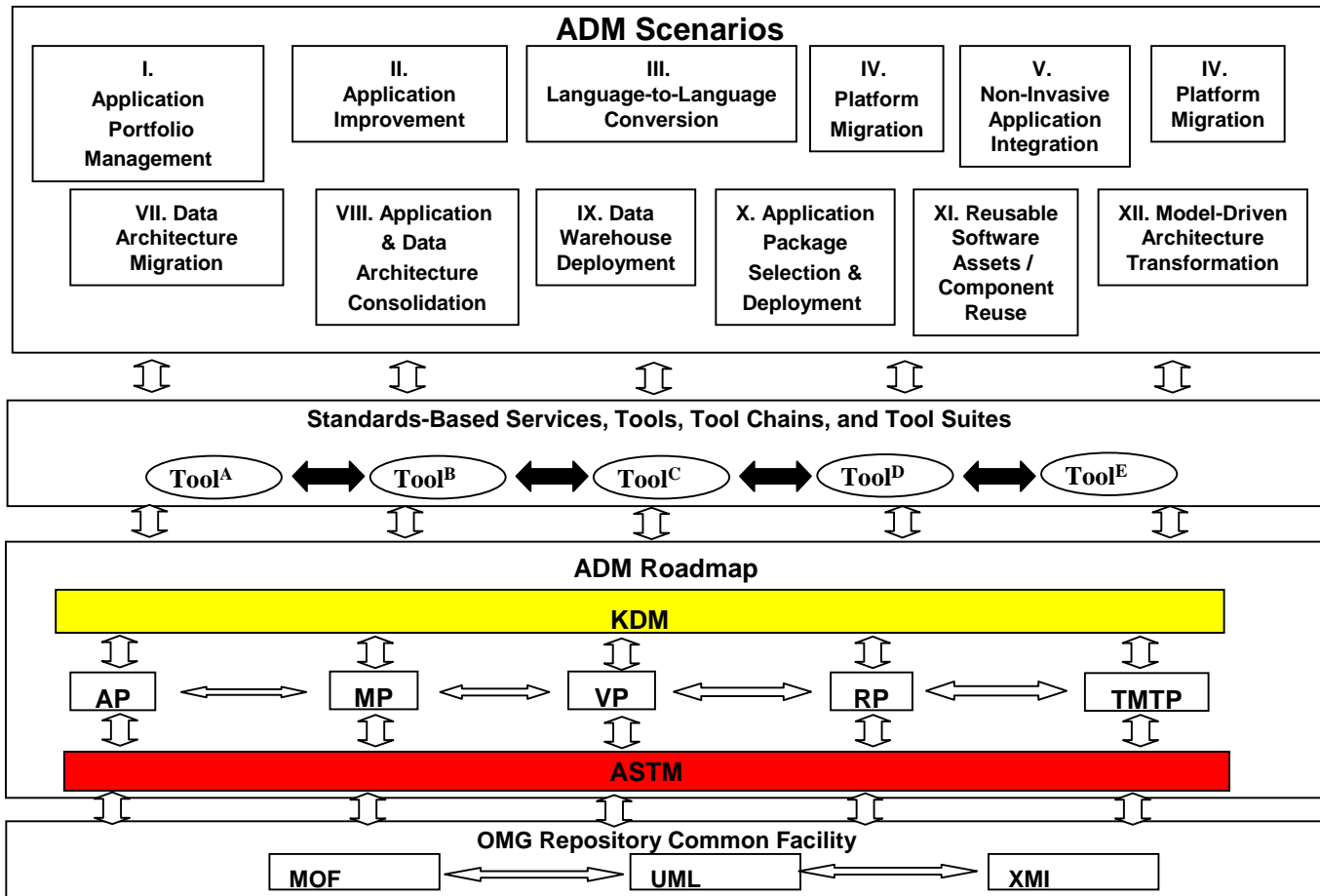
6. Refactoring Package – (initiate '08)

- builds on KDM, ASTM, AP, MP and VP
 - defines ways in which the ADM models be used to re-architect applications
 - Includes structuring, rationalizing, modularizing and in other ways improving existing applications without redesigning those systems or otherwise deriving model-driven views of those systems
-

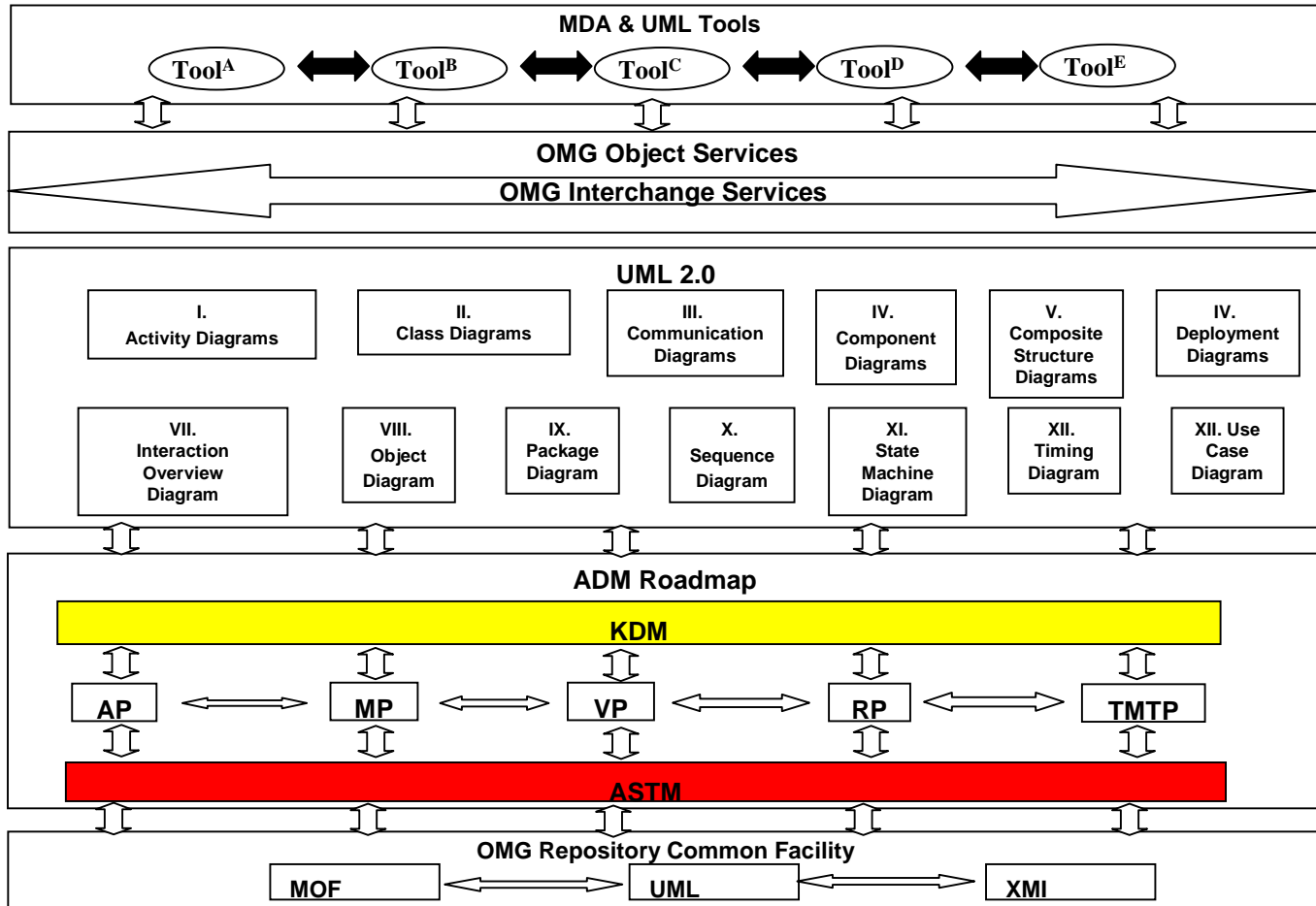
7. Target Mapping & Transformation (TMT) Package – (initiate '09)

- builds on KDM, ASTM, AP, MP, VP and RP
- Defines mappings between the ADM models and target models to enable transformation
- Development paradigms may vary, but will include MDA as a target
- completes the ADM task force efforts in providing a transformational bridge between existing systems and target architectures

How Does the ADM Roadmap Support the ADM Scenarios?



How Does the ADM Roadmap Support UML 2.0 and MDA?



What is the Status of the ASTM RFP?

- Three ASTM Submissions were Received (30 May, 2005)
- Three ASTM Submissions were Joined (23rd June, 2005)
- **ASTM Revised Submission will be Reviewed (February 15th ,2006)**
- **ASTM Adoption expected in 2006**

Duration	Event or Activity	Actual Date
	<i>Preparation of RFP by TF</i>	<i>3 February 2005</i>
	<i>RFP placed on OMG document server</i>	<i>3 February 2005</i>
	<i>Approval of RFP by Architecture Board Review by TC</i>	<i>3 February 2005</i>
0	<i>TC votes to issue RFP</i>	<i>3 February 2005</i>
60	<i>LOI to submit to RFP due</i>	<i>3 April 2005</i>
117	<i>Initial Submissions due and placed on OMG document server ("Three week rule")</i>	<i>30 May 2005</i>
133	<i>Voter registration closes</i>	<i>15 June 2005</i>
139	<i>Initial Submission presentations</i>	<i>21 June 2005</i>
141	<i>Preliminary evaluation by TF</i>	<i>23 June 2005</i>
225	<i>September 15th ADM Task Force Voted To:</i> <ul style="list-style-type: none"> - Change Revised Submission To February 24th - No Time Table Established For Final Submission - No Time Table Established For Adoption 	<i>15 September, 2005</i>
355	<i>Jan 23rd Revised Submission on Server</i>	<i>January 24, 2006</i>
378	<i>Feb 15th ADM TG Review of Revised Submission</i>	<i>February 15, 2006</i>

Who's Who on the ASTM?



ASTM Joint Submission Team

- 7 Submitters
- 9 Supporters
- 26 Voting List
- Lead/Coordinator: TSRI (Philip Newcomb)

• RFP II ASTM

– Submitters

- The Software Revolution (TSRI)
- Klocwork
- IBM
- EDS
- TCS America
- Interactive Objects Software GmbH
- ASG

– Supporters

- Kestrel Institute
- DSTG (Delta Software Technology GmbH)
- Blue Phoenix
- Northrop Grumman
- Tactical Strategy Group
- Adaptive Technologies
- SAIC
- Composable Logic
- 88 Solutions

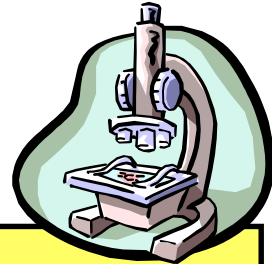
• RFP II ASTM

– Voting List

- ASG
- Adaptive Technologies
- Alcatel
- Blue Phoenix
- DSTC
- Data Access
- EDS
- Ecubesystems
- Fujitsu
- Hewlett Packard
- Interactive Objects Software
- IBM
- Klocwork
- Kestrel Institute
- Lockheed Martin
- Mentor Graphics
- Northrup Grumman
- NIST
- Raytheon
- Relativity
- Rockwell Collins
- SEEC
- TCS America
- Thales
- Unisys
- The Software Revolution

How Does the ASTM Complement The KDM?

The ASTM and KDM are complementary elements of the ADM Roadmap



KDM RFP #1

The Knowledge Discovery Meta-model

Comprehensive high-level view...

Behavior

Structure

Data

Above Procedural Level

ASTM RFP #2

Abstract Syntax Tree Meta-model

Programming Language Constructs

ASTM is information source for KDM

Below Procedural Level

They are complementary but individually useful standalone standards.

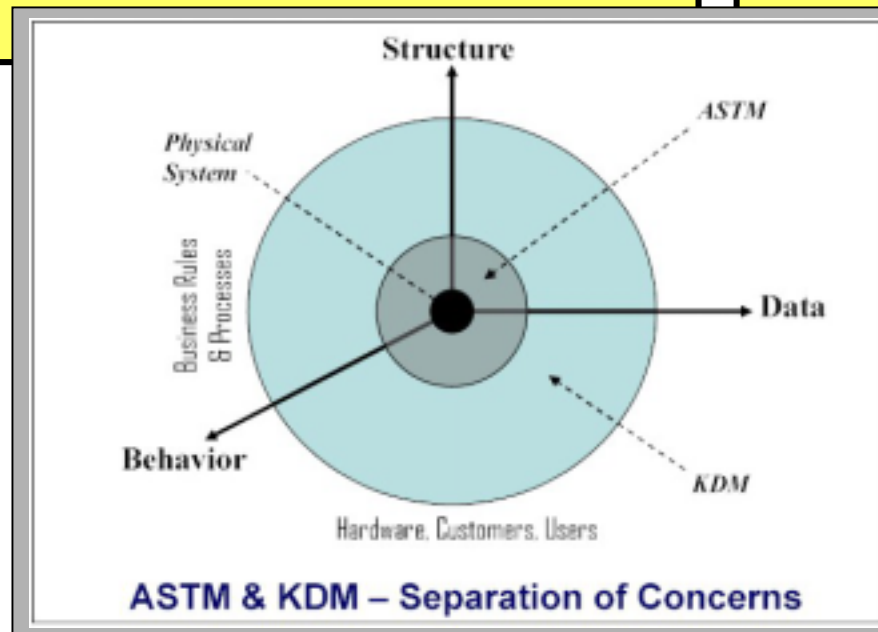
How are the KDM and ASTM independent?

- **From KDM perspective:**

- *The ASTM is one means of populating the KDM.*
- *The ASTM extends the KDM to support comprehensive and detailed modeling of systems.*

- **From ASTM perspective:**

- *The KDM as well as the ADM Roadmap are several MDA MOF models the ASTM populates from its highly detailed and precise models of systems.*
- *The KDM is one source for information about systems which can guide large-grained analysis, metrics, visualization, model mapping, transformations & refactoring which the ASTM supports.*



The ASTM does not directly depend upon or intersect with the KDM for any part of its meta-model definition.

Together the ASTM and the KDM provide high fidelity support for ADM scenarios when effectively combined.

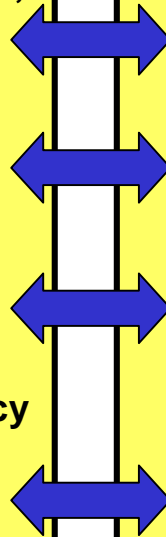
How do KDM and ASTM Regard Transformation?

From KDM perspective:

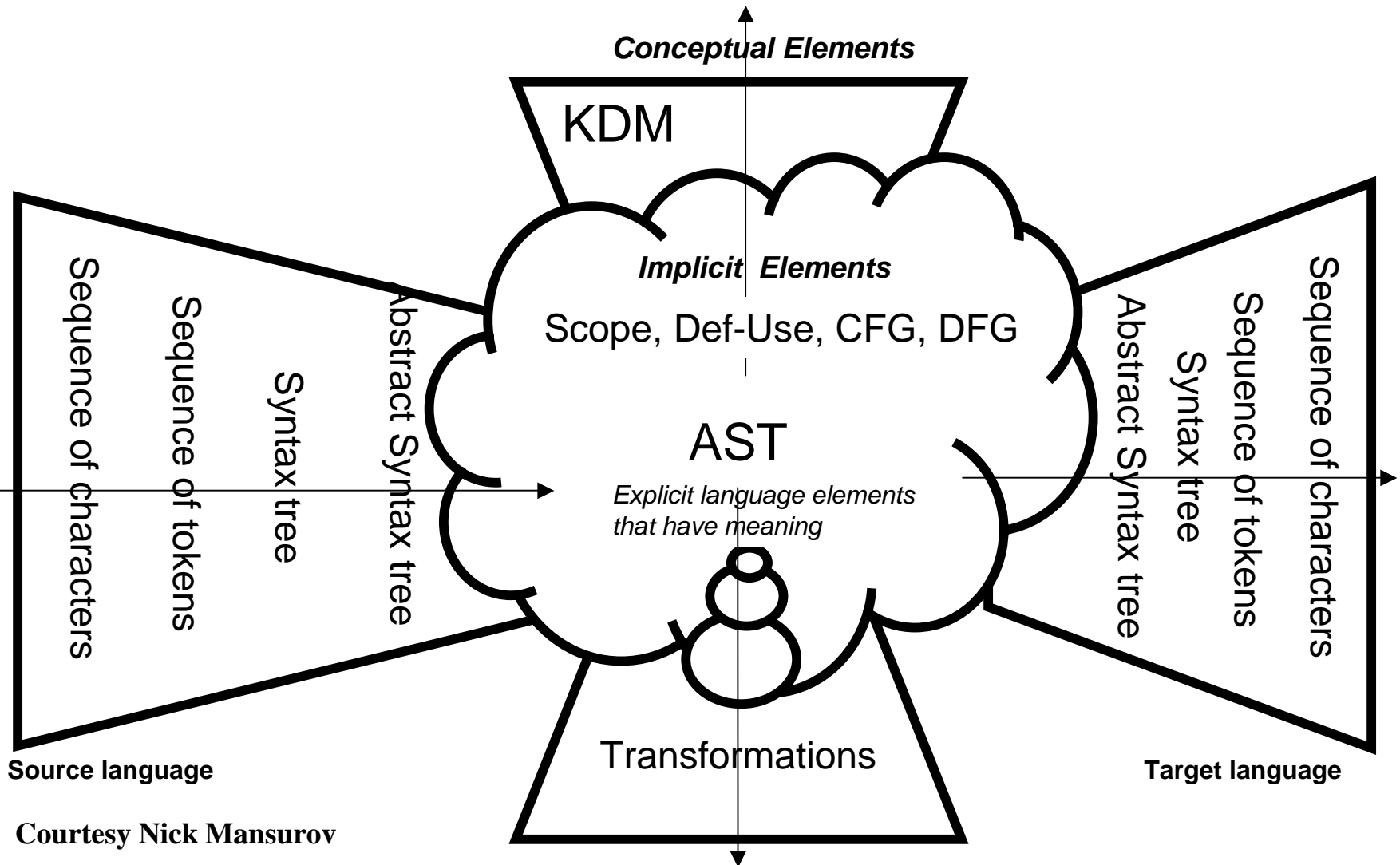
- Transformation is an augmentation strategy that includes:
 - Understanding application data, structure and behavior and architecture
 - Making legacy systems more reliable and adaptable
 - Extracting and rationalizing data definitions, data and business rules
 - Redesigning and reusing legacy rules and data within the context of strategic enterprise architecture.

From ASTM Perspective:

- Transformation is a direct strategy that includes:
 - Model driven mapping of data, structure and behavior between language feature sets (language translation).
 - Model driven restructuring of language feature sets with replacement of undesirable features with reliable and adaptable features (application refactoring)
 - Model driven rationalizing of data definitions, data and business rules by abstraction to MDA data views (models) that support model-driven regeneration of specific language features.
 - Model driven rearchitecting of systems by abstracting design patterns and applying generation, transformation and refactoring to regenerate redesigned and rearchitected enterprise applications.



How Are The ASTM And KDM Models Connected?



Courtesy Nick Mansurov

What is the ASTM Business Value?

Standardizing the format of AST structures, representation and interchange of AST models will ...

- Complement the KDM by completing a comprehensive model for the exchange of application models between application modernization tools.
 - Enable vendors to develop specialized modernization tools.
 - Insure that the aggregate of vendor tools provides a comprehensive modernization capability.
-

A standard KDM complemented by a standard ASTM will enable a user of the technology to bring together a variety of best-of-breed products to analyze, visualize, re-factor and transform legacy applications.

How does ASTM improve MDA?

- **It establishes a standard bottom-most language modeling level for many MDA tools to generate to and derive from.**
- **It allows AST models to be sharable among multiple tools from different vendors accurately support analysis, visualization, refactoring, target mapping and transformations.**
- **Provides high levels of automation for tasks which are highly manual today, such as application rehosting, platform retargeting, legacy system replacement, database upgrade**
- **It provides a GAST with sufficient precision and generality and fineness of granularity to allow its language modeling elements to be used as a common basis for application analysis, metrics, visualization translation, and refactoring.**

What's the Relationship of ASTM to Other OMG Specs?

MOF 2.0 (ptc/04-10-15)	Facilitates ASTM definition and exchange formats (XMI, etc.)
Action Semantics as a part of UML 2.0 – superstructure 2.0 finalization (ptc/2003-08-02)	The ASTM is concerned with syntax, while action semantics is concerned with one form of semantics.
IT Portfolio Management (ptc/04-11-03)	IT Portfolio Mgmt. establishes the universe of non-software aspects of the enterprise, whereas the AST defines the software assets of the enterprise.
UML (ptc/03-08-02; ptc/03-09-15; ptc/03-10-14)	A subsequent ADM roadmap RFP will support the derivation of UML models of software artifacts from the ASTM.
CWM (formal/2003-03-02)	The ASTM provides a language agnostic view of syntactical data declarations while the CWM data model does not.
EAI (includes detailed metamodels for programming languages calls, including COBOL) (ptc/02-02-02)	EAI is concerned with interfaces while the ASTM defines the syntactic structure of the software artifact.
Reusable Asset Specification (RAS) (ptc/04-06-06)	Reusable Asset Specification provides a way to define reusable asset packages for a domain of interest. The software asset modeled by an AST could be a RAS asset. ASTM could be used to populate information in a RAS asset.
MOF/QVT model transformation (ad/2002-04-10)	A subsequent ADM roadmap RFP will support the utilization of MOF/QVT for transformation of ASTM software artifacts.

What is an Abstract Syntax Tree?

AN AST IS...

- A formal representation of software **syntactical structure** of software
 - More amenable to formal analysis than **concrete syntax** or **surface syntax** (*the way the code looks*)
 - A more precise formalism for detailed knowledge discovery than less formal knowledge collectors (e.g. scanners, tokenizers, visual inspection)
-

AN AST MAY ...

- Be an **invertible representation** that allows reconstruction of the “**surface syntax**” of the legacy system from the “**abstract syntax**” (**surface syntax** \Leftrightarrow **abstract syntax**)
 - Be **augmented** with **constraint analysis**, **data-flow analysis**, **control-flow analysis**, **axiomatic analysis** and **denotational analysis** and other views (conceptual, scenario, interaction, collaboration, build, etc)
 - Be used to **derive** software engineering **metrics** and **documentation**
 - Be **mapped** or **transformed** into **other abstract syntax models** using **rewrite rules**.
 - Be **queried** and **manipulated** using **query** and **model manipulation languages** (such as **OMG’s QVT™** and **TSRI’s JTGEN™** and **JRGen™**)
-

An AST IS ALSO...

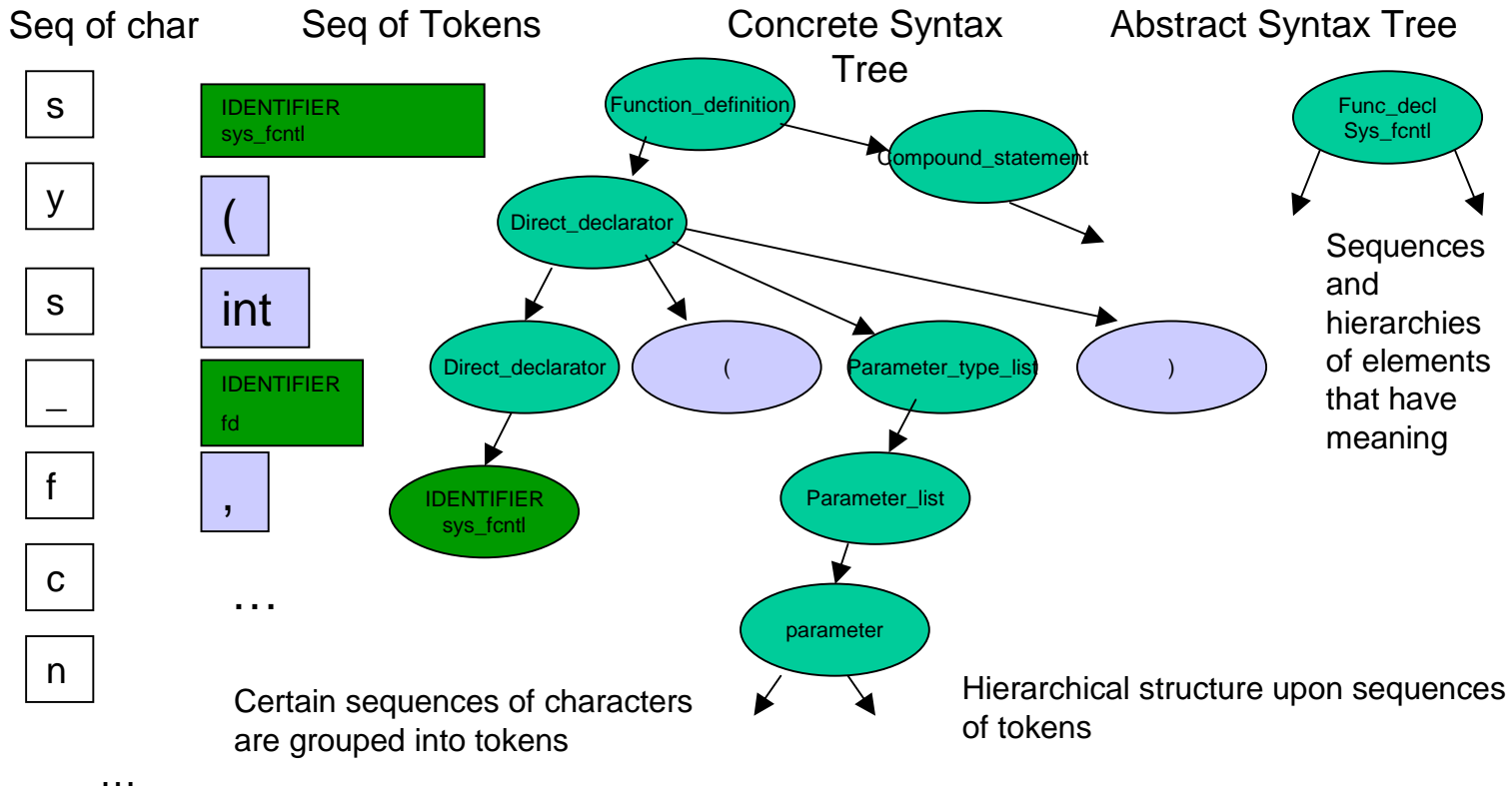
- A well-formed **MOF Model**, **Meta-Model** or **Meta-Meta-Model**

What are Language Theory ASTs?

In programming language parsing theory the Abstract Syntax of a programming language is distinct from the Abstract Syntax Tree of an application.

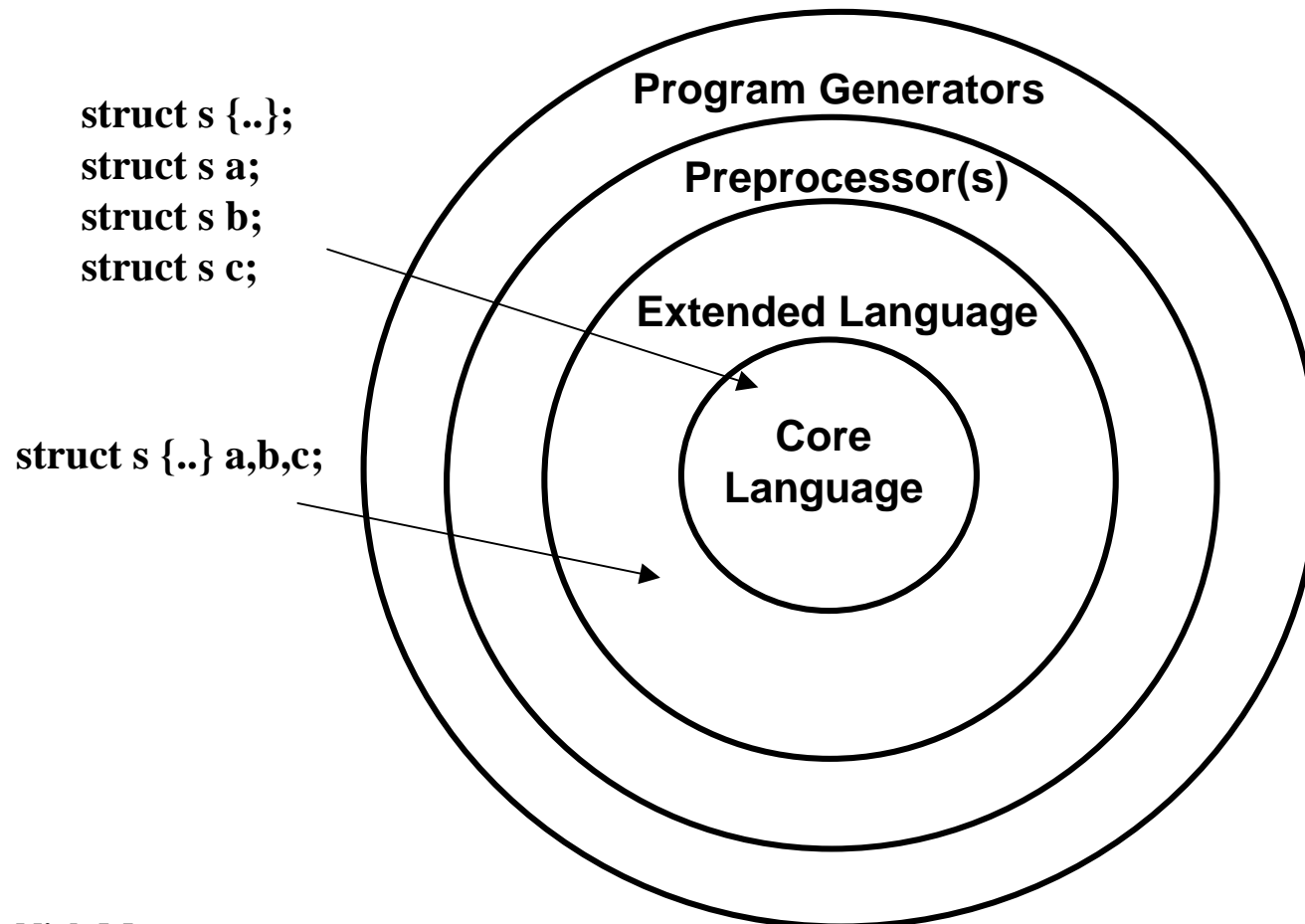
- *Abstract Syntax* is a model consisting of the language constructs from which a language is composed.
- *Abstract Syntax Tree* is a directed acyclic graph (DAG) consisting of instances of Abstract Syntax language constructs.
- A well-formed **BNF** or **EBNF (or grammar specification)** is often used to define the Abstract Syntax of a programming language.
- A **parser generator** is often used to generate a parser which generates the Abstract Syntax Tree by parsing an application in the language defined by the EBNF; but methods other than parsing can be used to construct AST (such as generation from UML or BSBR models).

How Is the AST Related To Other Source Representations?



How Can ASTs Cut Through Syntactic Variation?

- **Many Languages Express The Exact Same Concepts**
 - But In Many Very Different Ways (Highly Varied Syntactical Renderings)
 - Using Nested Languages and Composite Heterogeneous Languages



ASTs Are A Commonly Used Intermediate Representation (IR)

- For example, 3-address IR
- For example, Java bytecode
- For example, Microsoft CLR
- For example, DIANA
- For example, SUIF
- For example, RTL of GCC compiler
- GIMPLE of GCC compiler
- LAST of the McCAT compiler
- TSRI IOM™, JGen™ and JTGEN™
- OMG Meta Object Facility MOF™

The difference is in granularity and amount of implicit information – IRs are further away from the source which makes them more suitable for particular applications

Why Do We Gain By an ASTM Standard?

Why Do We Need an ASTM Standard?

- Makes the many tools that generate ASTs from code more useful.
- Interoperability can be achieved between Tools that produce and Tools that use ASTs.
- Standard Tools can be used to visualize the ASTs produced by Various AST generators.
- QVT, MOF, XMI, OCL and other OMG standards Can be More Easily Extended to the ASTs produced by AST generators.

Why Do We Need A Standard Generic ASTM?

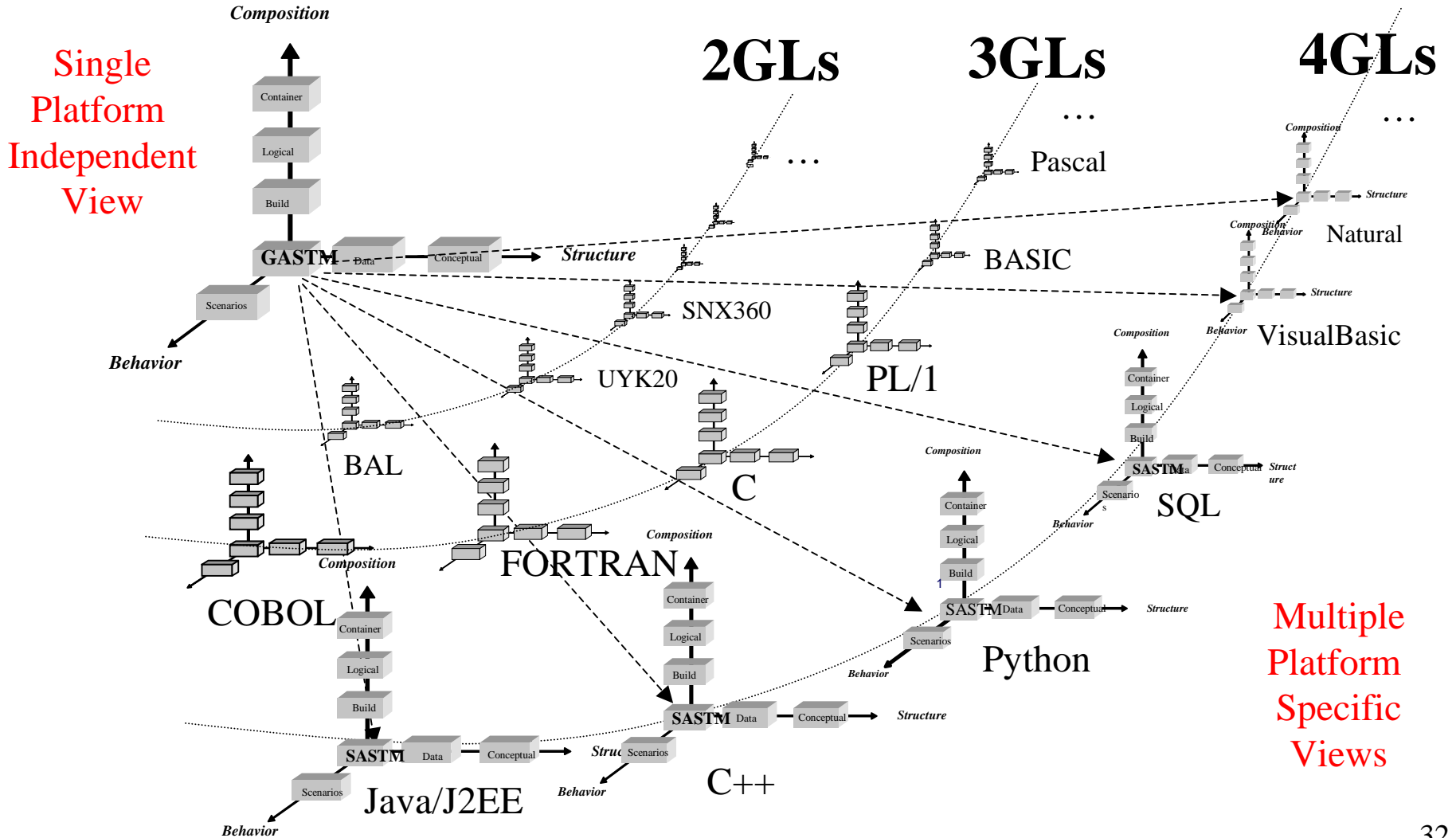
- Object Constraint Language (OCL) can apply constraints to standard ASTs for analysis of code properties.
- Query View Transform (QVT) can be applied to query, view and transform ASTs in standard formats.
- Code generators, such as Eclipse and MDA tools can generate to the generic ASTs, and let code generators produce target code to multiple target ASTs (SASTs) platforms

Why Do We Need An AST Meta-Meta-Model?

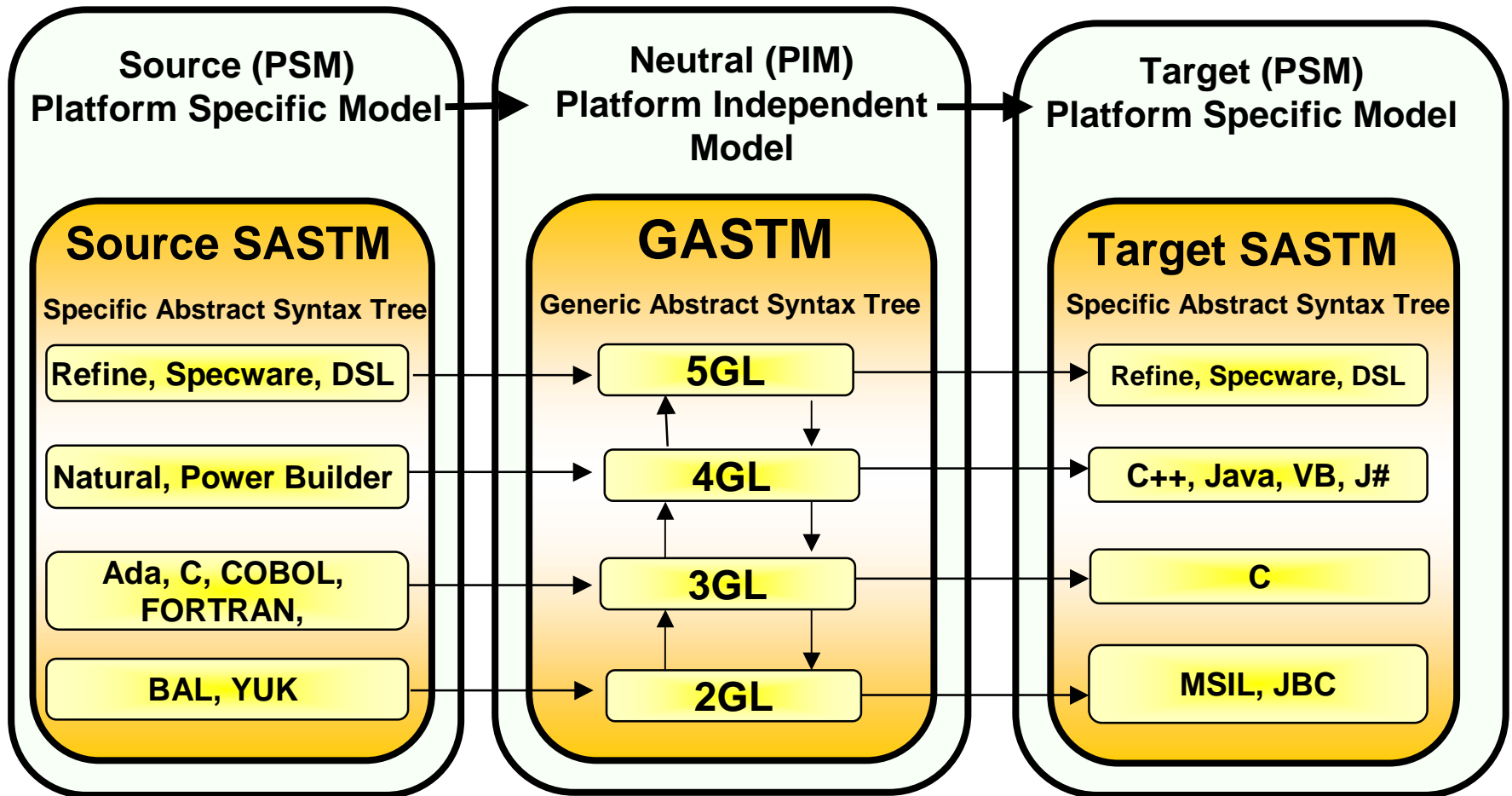
- Supports Application of Visitor Pattern to ASTs expressed as UML Class Diagrams.
- Facilitates Generation of APIs For Front-End tools
- Facilitates Higher-Level Forms of Reuse in tool development
- Facilitates Reasoning About Multi-language systems
- Facilitates Analysis of GASTM and SASTM models vs. GAST and SAST models.

RFP1 + RFP2: KDM + ASTM

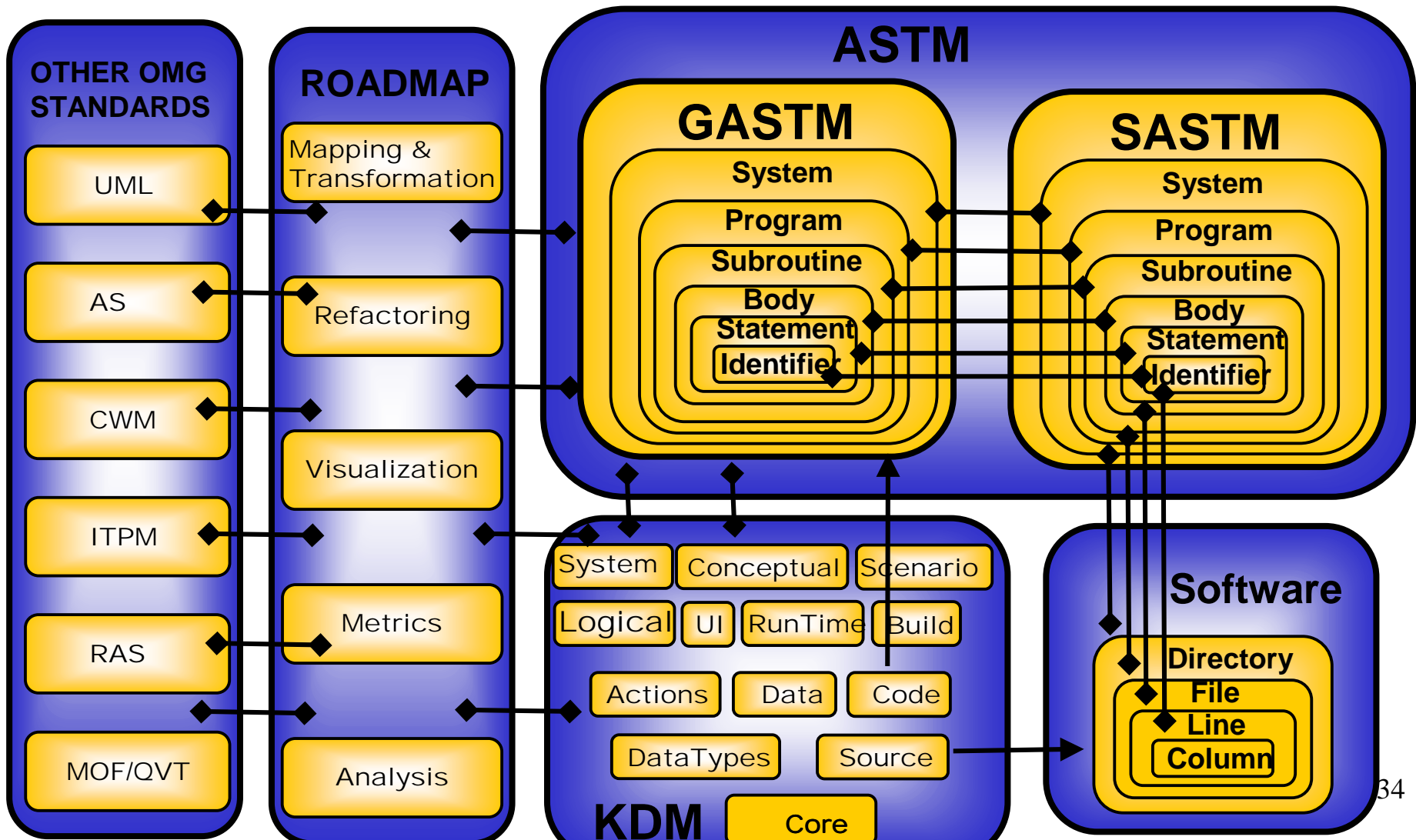
Multiple Dimensional Single Platform Independent View + Multiple Platform Specific Views



MDA (PSM ⇔ PIM ⇔ PSM) ASTM (GASTM ⇔ SASTM)



Integration of the ASTM & KDM



What are Meta Object Facility (MOF) ASTs?

In the OMG Meta Object Facility (MOF) Abstract Syntaxes are nested meta-modeling levels.

- M3 is the Abstract Syntax of M2 Abstract Syntax Trees
- M2 is the Abstract Syntax of M1 Abstract Syntax Trees
- M1 is the Abstract Syntax of M0 Abstract Syntax Trees.

MOF uses UML Class diagrams to define Abstract Syntax!

MOF platform independence comes from its use of generators that produce software for managing models that conform to meta-models.

- Many perfectly acceptable MOF Abstract Syntax models would be considered malformed by many grammar specification constraint checkers.
- Most well-formed Grammar specifications acceptable to compiler or parser generators should be MOF compatible.

ADM Relationship to MOF

Layer	Description	ADM Examples
Meta-metamodel M3	MOF, I.e. the set of constructs used to define metamodels	MOF Class, MOF Attribute, MOF Association, etc
Metamodel M2	Metamodels, consisting of instances of MOF constructs	KDM UML profiles GASTM UML profile SASMT UML profile
Model M1	Models consisting of instances of M2 metamodel constructs	AS Model of COBOL language. KDM Data Model
Instances (examples) M0	Objects and data, I.e. instances of M1 model constructs	<ul style="list-style-type: none">• AST model instances of source code of real application• KDM Data models instance of data base or data files.

Why Marry MOF and Parsing Technology ASTs

Combining MOF and the Language Parsing Approaches to Abstract Syntax and ASTs is highly powerful!

Abstract Syntaxes defined for Language Parsers can usually be modeled as MOF Models using MOF Tools.

Once the MOF Models for language models exist, MOF has generators to create XML, Java, and CORBA API support for these models.

- This allows the model instances (ASTs) of software application to be modeled and interchanged via MOF Repository technology.
- This allows the exchange of AS (metadata) and ASTs (i.e enterprise applications treated as data) with full machine automation.

Without MOF, the manipulation of applications in Abstract Syntax form will remain proprietary with limited penetration.

With MOF, the manipulation of programs as data (ASTs) will become universal and many hard software problems will be solved efficiently and economically.

How does the ASTM support OMG standards?

The GAST will be used to support other OMG modeling standards by providing a generic set of language modeling elements as the basis of OMG model (AST) derivation and as the basis for OMG language generation.

The GASTM is effectively an Ultra Wide Spectrum Intermediate Language Model which spans 2GLs, 3GLs, 4GLs and 5GL languages.

- Existing OMG generators for MOF models are limited to Java or C++ for *behavior* support, and XML Schema and DTDs for *structure* support.
- Retargeting OMG MOF generators into the GASTM will extend MDA support to a much broader spectrum of target languages than is currently supported by MOF technology.

Application Models and Meta-Models (AST and AS) Will Be ...

- sharable among multiple tools from different vendors with much more uniform support for analysis, visualization, re-factoring, target mapping and transformations across multiple languages.

How is the ASTM Structured ?

It extends MOF modeling to encompass several existing families of languages in a uniform way.

- A generic set of language modeling elements common across numerous languages establishes a common core for language modeling, called the **Generic Abstract Syntax Trees (GAST)**
- Language specific **Specific Abstract Syntax Trees (SAST)** for particular languages such as Ada, C, Fortran, Java, etc must be modeled in MOF or MOF compatible forms.
- **SAST** \Leftrightarrow **GAST** must be demonstrated without loss of meaning even though their abstract syntax model change during transformation between languages.

Separation of concerns dictates the SASTM \Leftrightarrow GASTM separation.

- ASTM reduces the **O (S * T)** transformation problem to an **O(S+T+G)**, where **S** is the number of source languages (or models) and **T** is the number of target languages (or models) + **G** the GASTM (G).

What are the ASTM 10 Mandatory Requirements?

- (1) Include a meta-model compatible current MOF.**
- (2) Use UML to represent ASTM diagrams**
- (3) Define a single unified terminology by complementing the KDM with a low-level detailed discovery model**
- (4.1) Define a set of common concepts that exist in multiple languages**
- (4.2) Define a set of language independent concepts, to be called the generic abstract syntax tree meta-model (GASTM).**
- (5) Specify the means to specialize, differentiate or extend GASTM language independent concepts to represent language dependent constructs, to be called the specialized abstract syntax tree meta-model (SASTM).**

What Are the ASTM Mandatory Requirements? (cont.)

- (6) Provide a list of the languages and platforms that the submitters ASTM proposal is claimed to support.**
- (7) Demonstrate that it is possible to construct a mapping between languages and the ASTM.**
- (8) Represent the original language constructs with sufficient precision to assure preservation of semantic equivalence.**
- (9) Include information on how the ASTM relates to other non-OMG standards pertaining to abstract syntax trees.**
- (10) Define programming constructs used to construct abstract syntax trees to a level of precision and with sufficient detail to allow the regeneration of the code (or surface syntax) from the model.**

What are the ASTM Evaluation Criteria?

- (1) Demonstrate the capability to map the existing software artifacts in a common implementation language, such as C, or C++ or Java or Ada or COBOL into a MOF repository that can be described by the proposed ASTM .**
- (2) Demonstrate the ability of the proposed ASTM to support ASTs for programs in more than one implementation language.**
- (3) Demonstrate the usability of the proposed ASTM for the purpose of representing information about the existing system.**
- (4) Demonstrate the capability of the ASTM to represent a broad range of languages and language types including 2GL, 3GL, 4GL and 5GL languages such as Ada, Assembler, C, C#, COBOL, FORTRAN, Java, Natural, Power Builder, Refine, SQL, etc.**

What Is the Status of the Joint Submission?

- **Three Submissions and Three Complementary Perspectives**
 - 3 Views of GASTM & SASTM
 - TSRI: Discrete Model
 - TCS: Continuous Model
 - Klocwork: Interface Module Model
 - 2 Views of ASTM Meta Models
 - TSRI & TCS : Plain UML without profiling.
 - Simple Class, Associations and Class Members
 - Klocwork : UML with Profiling
 - Associations are Classes
 - Types
 - Sets
 - Attribute Hierarchy
- **The Three Teams are Working to Achieve A Joint Revised Submission (due January 24th)**

Related Perspectives

- **ASTM Modeling Framework (TSRI, TCS)**
- **ASTM Meta-Data Repository (TSRI, TCS)**
- **Task Complexity Descriptions (TSRI)**
- **AST Support for Road Map (TSRI)**
- **ASTM Support for Scenarios (TSRI)**
- **ASTM Partitioning into Logical groups (TCS)**
- **ASTM Hierarchy (TSRI & TCS)**
- **Explicit separation of GASTM, SASTM (TSRI & TCS)**
- **GASTM, SASTM Union models (TCS)**
- **ASTM Meta-Meta Model (KW)**
- **ASTM Language Transformation Combinatorics (TSRI)**

TSRI Discrete View of GASTM and SASTMs

- The GASTM is a broad subset of lowest common denominator language elements found in many languages.
- Each SASTM model is a distinct set of language elements for a specific language for which it defines the abstract syntax.
- There are many discrete SASTM models, one for each language, or vendor defined SASTM.
- A SASTM \Leftrightarrow GASTM mapping preserves functional equivalence.
- A SASTM is considered a PSM and the GASTM is considered a PIM ala MDA/MDD.
- Analysis is done entirely against the GASTM without reference to the SASTM and without loss of meaning.
- Multi-language and language-neutral mapping, analysis and transformation is supported by GASTM and SASTM \Leftrightarrow GASTM mapping.

TSRI ASTM Perspective

Submitted By: TSRI

Philip H. Newcomb	Chief Executive Officer	425 284 2770	philip.newcomb@softwarerevolution.com
-----------------------------------	-------------------------	--------------	--

Supported By: Kestrel Institute

Douglas Smith	Assoc. Director	650 493 6871	smith@kestrel.edu
Cordell Green	Director	650 493 6871	green@kestrel.edu

Blue Phoenix

Chris Caputo	Software Architect	919-380-5412	ccaputo@bphx.com
--------------	--------------------	--------------	--

Northrop Grumman

Roy E. Kimbrell	Fellow, Technical Director	402 682 4330	roy.kimbrell@ngc.com
-----------------	----------------------------	--------------	--

Tactical Strategy Group

William M. Ulrich	President	831-464-5346	WMMULRICH@cs.com
-------------------	-----------	--------------	--

Adaptive Technologies

Peter Rivett	Chief Technology Officer, Adaptive Inc.	441 202449419	pete.rivett@adaptive.com
--------------	---	---------------	--

SAIC

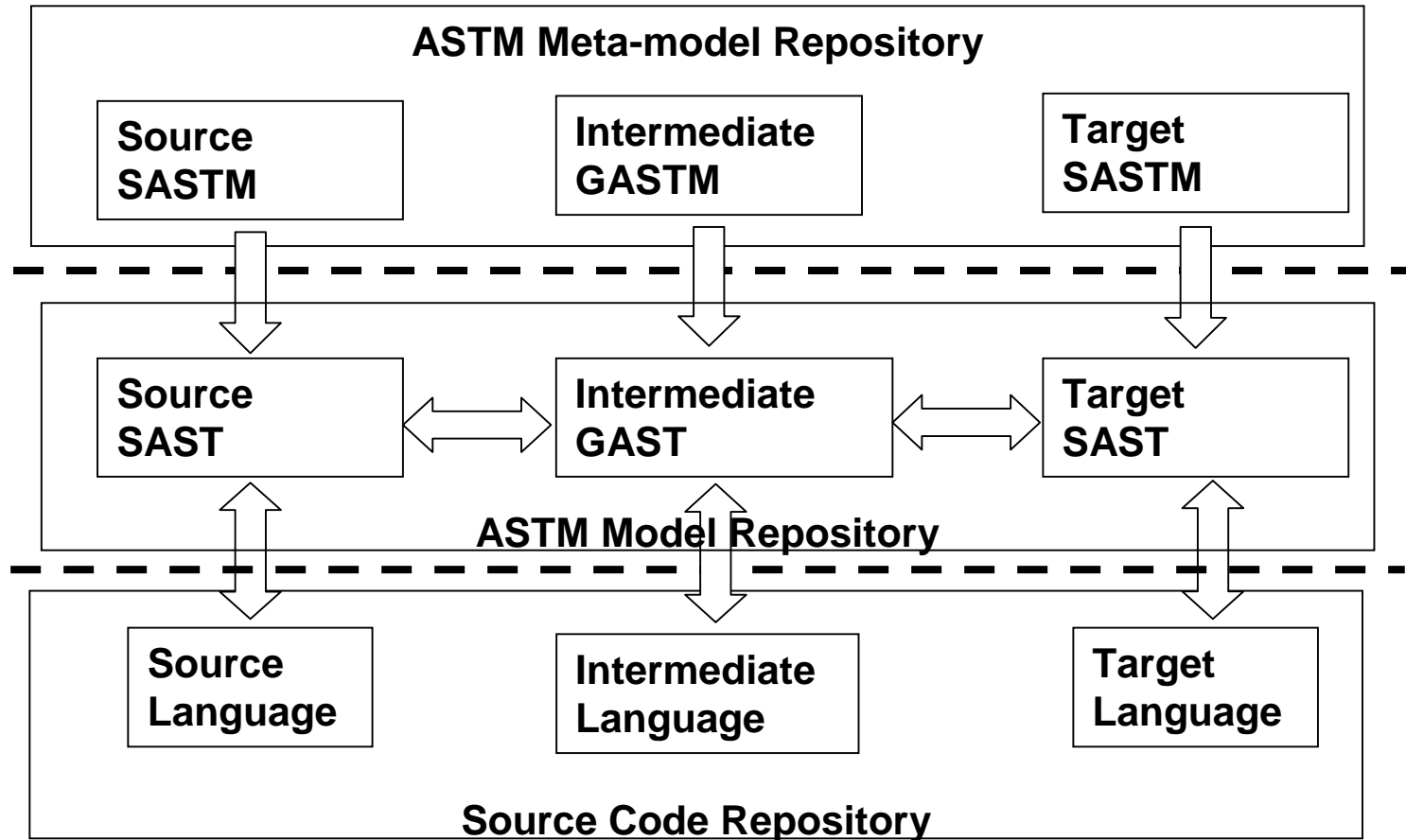
George Hou	Account Manager	703-824-5450	houg@saic.com
----------------------------	-----------------	--------------	--

Composable Logic

Jeff Smith	President	603-566-0124.	jesmith@ComposableLogic.com
------------	-----------	---------------	--

88 Solutions

TSRI View of Meta-model and Metadata Repository



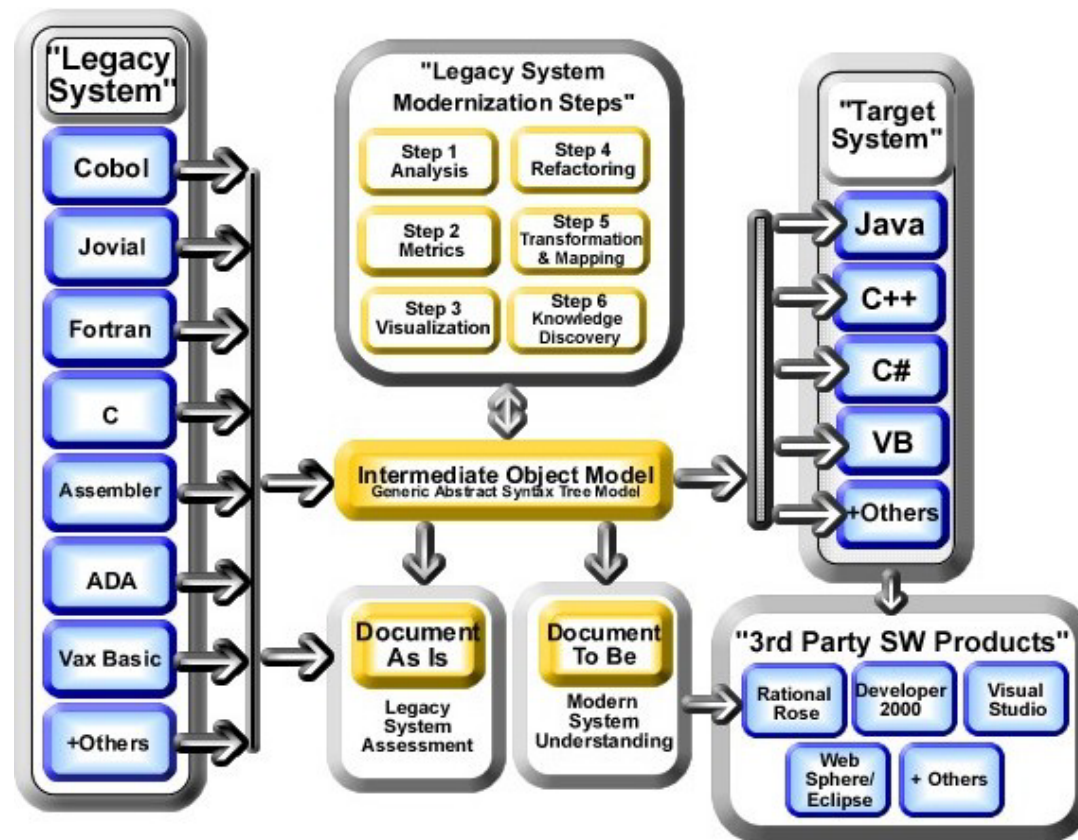
TSRI SASTM/GASTM

Modular Packaging with Strong Separation of Concerns

- GASTM is the **Complete** Common Denominator of All SASTMs
- Each SASTM is in a Separate Language Distinct Package.
- GASTM is in its Own Separate Package
- SASTM and GASTM Modeling Element Naming Consistency Observed But Not Required.
- Model-Level Mapping Between SASTM and GASTM Defines Relationships Between Elements
- Concrete Syntax Mappings Separately Defined For Each SASTM

TSRI Product and Services Implementation of GASTM / SASTM

TSRI Product and Services Strong Pragmatic Orientation Towards Fully Automated:
(1) Analysis, (2) Metrics, (3) Visualization, (4) Refactoring (5) Transformation & Mapping (6) Knowledge Discovery



TSRI GASTM/SASTM Framework Applications

TSRI GASTM / SASTM Framework Has Been Applied To 37 Architecture Driven Modernization Projects For Our Nation's Most Important Defense Systems

Raytheon

SAIC

DSR

DynCorp

LOCKHEED MARTIN

BOEING

U.S. AIR FORCE
CROSS INTO THE BLUE

NATIONAL ENDOWMENT FOR THE ARTS

Customer/Project	Source - Target Pair	"As Is"	Transform	Re-Factor	"To Be"	Web Enable
NEA	COBOL to C++	X	X	X		X
Northrop Grumman (REMIS)	COBOL to C++	X	X	X	X	X
CSC (FBISHED)	COBOL to C++	X	X	X	X	
STG (DODAAD)	COBOL to C++		X	X	X	
Premiera	COBOL to C++	X	X	X	X	
DynCorp (VSCRS-I)	COBOL to C++		X			
One Star (VSCRS-II)	COBOL to C++		X	X	X	
STG (VSMIS)	COBOL to C++	X	X	X	X	
Telos (DURS)	COBOL to Java		X			X
USAF (CAMS)	COBOL	X				
Oregon (OPERS)	COBOL	X		X	X	
Northrop Grumman (JMPS)	Ada to C++	X	X	X		
Raytheon (MCS)	Ada to C++		X	X		
DSR (E-2C)	Ada to C++		X			
Boeing (ALCA)	Jovial to C++		X			
CSC (CCS-C)	Jovial to C++	X	X			
TRV (MILSTAR)	Jovial to C++	X	X			
TRV (DMEVS)	Jovial to C++		X			
USAF (F-16 DECIS)	Jovial to C++					
Litton/PRC (USSTRATCOM)	Jovial to C++		X			
Raytheon (TCS)	Fortran to C++		X	X		
ITT (Assessment)	Fortran	X				
Lockheed Martin (I-SPAN)	Fortran to C++		X			
Litton/PRC (USSTRATCOM)	Fortran to C++		X			
SAIC (EOSS)	Vax-Basic to Java	X	X	X	X	X
Litton/PRC (USSTRATCOM)	Assembler to C++		X			
ITT (Assessment)	C	X				
CSC (R2UPLD)	C to C++	X	X	X	X	
LMCO (P-3C)	Ada to C++		X	X	X	
ITT (SENSOR)	Ada to Java		X	X	X	
SAIC (VHA)	MUMPS to Java	X	X	X		
CSC (ETMS)	Assessment	X				
Thales-ATM (ETMS)	Ada to C++ & Java		X		X	
NGIT (MPEC)	Fortran to Java	X	X	X	X	
EDS (Pilot)	PIL 1 to Java		X	X	X	
Raytheon (Patriot Missile)	Fortran to C++	X	X	X	X	
Raytheon (VDAC)	Fortran to Java		X		X	

NORTHROP GRUMMAN

Litton PRC

TRW

CSC

ITT Industries

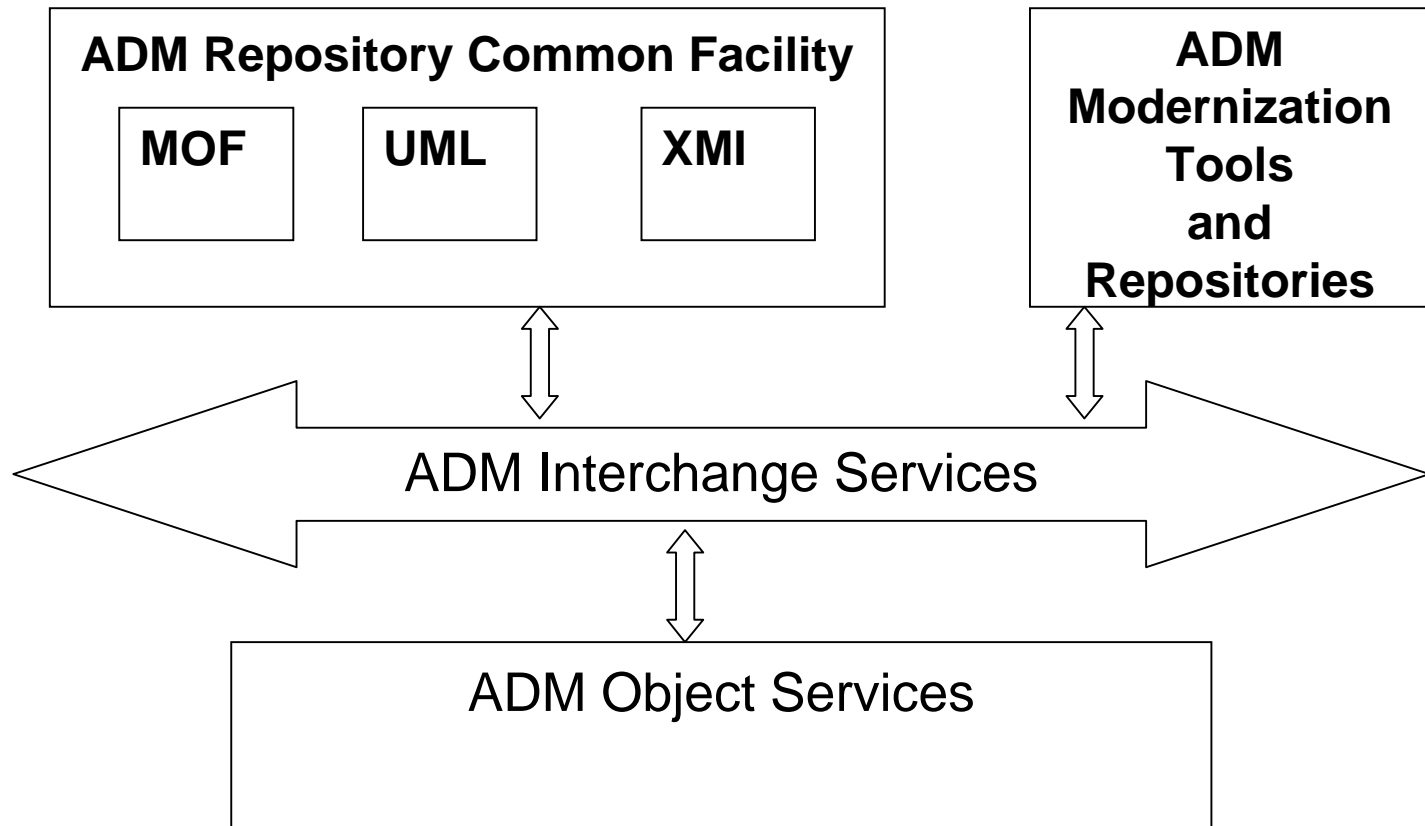
TELOS

STG

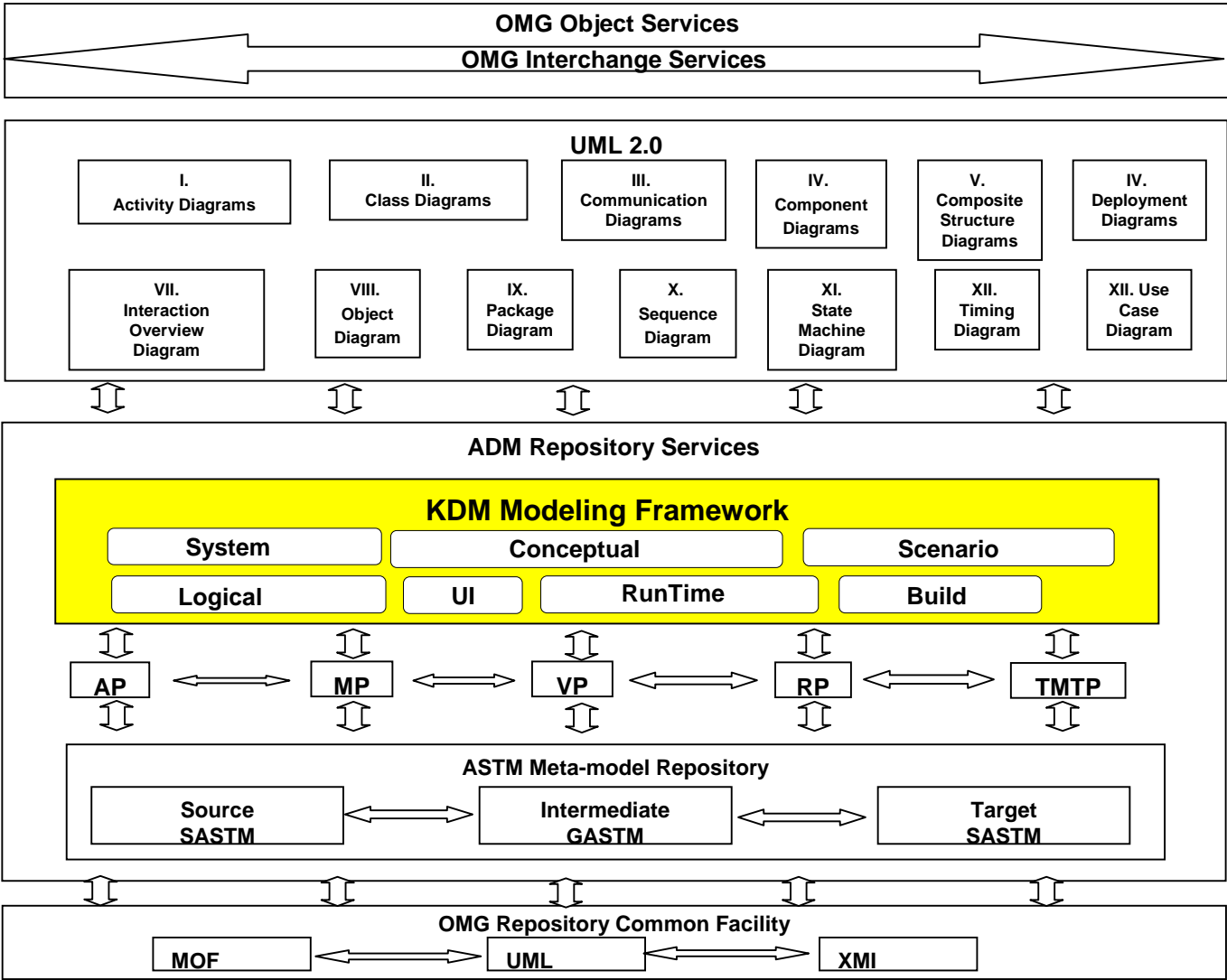
ADM Modeling Framework

(proposed by TSRI, supported by TCS)

ADM Repository and Object Services will Support AST Persistence, Distribution, Interchange and Object Services.



ASTM Meta-Data Repository (proposed by TSRI, supported by TCS)



ASTM Support for Roadmap (TSRI)

Matrix from submission describes how ASTM supports KDM Models

- KDM consists of the System, Conceptual, Logical, UI, Runtime, Actions, Data, Code, DataType, Source, Core models.
 - Perform a one-time mapping of Ada to the GASTM: Ada2GASTM.rul

KDM Mapping	GASTM	SASTM							
		Ada	C	Cobol	Fortran	JCL	Jovial	Java	C++
System									
Conceptual									
Scenario									
Logical									
UI									
Runtime									
Actions									
Data									
Code									
DataType									
Source									
Core									

ASTM Support for Roadmap (TSRI)

- The KDM consists of a set of Models which can be partially derived from the GASTM.
i.e. KDM := System, Conceptual, Logical, UI, Runtime, Actions, Data, Code, DataType, Source, Core models.
- The Mapping from an SAST model to the KDM consists of a three step process:
 - A one-time mapping for each language to the GASTM:
Ada2GASTM, C2GASTM, Cobol2GASTM, Fortran2GASTM, JCL2GASTM, etc.
 - A one-time mapping from the GASTM to each KDM Model:
GASTM2Behavioral, GASTM2Conceptual, GASTM2Scenario, etc.
 - Specialized Mappings to the KDM to the extent the GASTM is not *quasimorphic* with the SASTM.
Ada2Behavioral, Ada2Conceptual, ..., C2Behavioral, C2Conceptual, ...

The key design principle of the GASTM is achieve a 100% mapping from the SASTMs into the GASTM so as To eliminate the need for Step 3 above. The proliferation of language specific mappings (as shown below) Is BAD and should be avoided.

KDM Mapping	GASTM	SASTM								
		Ada	C	Cobol	Fortran	JCL	Jovial	Java	C++	...
System	GASTM2System	Ada2System	C2System	Cobol2System	Cobol2System	JCL2System	Jovial2system	Javal2system	Cpp2system	...2system
Conceptual	GASTM2Conceptual	Ada2Conceptual	C2Conceptual	Cobol2Concpetual	Cobol2Conceptual	JCL2Conceptual	Jovial2Concpetual	Javal2Concpetual	Cpp2Concpetual	...2Concpetual
Scenario	GASTM2Scenario
Logical	GASTM2Logical
UI	GASTM2UI
Runtime	GASTM2Runtime
Actions	GASTM2Actions
Data	GASTM2Data
Code	GASTM2Code
DataType	GASTM2DataType
Source	GASTM2Source
Core	GASTM2Core

What Are Task Complexity Descriptions?

Task Complexity is a formula that describes the approximate task complexity and effort associated with defining a mapping between models.

This is the Task Complexity Statement for Mapping to the KDM from the ASTM.

ASTM establishes a task complexity, O , for defining KDM support for a set of languages S to be $O(M(S) + KDM(G) + KDM(S))$ where $M(S)$ is the effort to Map each language into the GASTM and $KDM(G)$ is the effort to provide a set of Reusable KDM functions based upon the GASTM for the set of languages, and $KDM(S)$ is the effort to provide language specific KDM functionality for each SASTM, language specific specializations.

Here is a Task Complexity Statement for The Automated Portfolio Management Scenario.

Task Complexity can be used to define the complexity of effort associated with supporting ADM Scenarios.

This is the Task Complexity Statement for Supporting ADM Automated Portfolio Management Scenarios Tool Support from the ASTM.

This ASTM supports the ADM Automated Portfolio Management (APM) scenario by providing a language neutral framework upon which APM Tools can provide a uniform and high level of automated support. The use of the ASTM establishes a task complexity, O , for defining APM support for a set of languages S to be $O(M(S) + (APM(G) + APM(S)))$ where $M(S)$ is the effort to Map each language into the ASTM, $APM(G)$ is the effort to provide a set of Reusable APM functions based upon the GASTM for each language, and $APM(S)$ is the effort to provide language specific APM functionality for language unique features for specific languages.

GASTM Language Transformation Combinatorics

- **Conversion Between Language Families Entails *Complex Mappings* (N to 1, and 1 to N, M to N) Between Modeling Elements.**

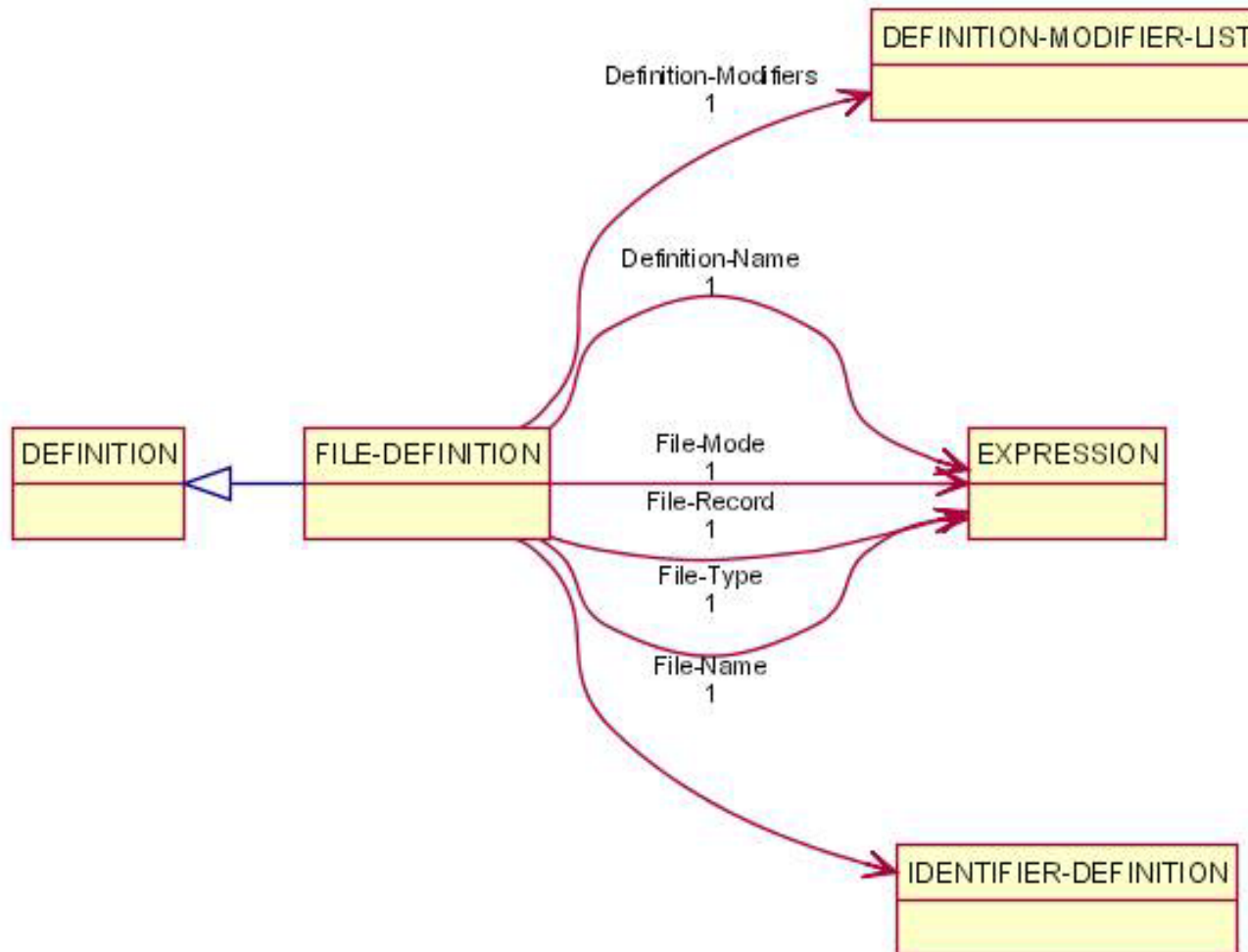
Top Down Language To Language (L2L) Conversion Scenarios		
5GL to 4GL		
5GL to 4GL to 3GL	4GL to 3GL	
5GL to 4GL to 3GL to 2GL	4GL to 3GL to 2GL	3GL to 2GL

Bottom Up Language To Language (L2L) Conversion Scenarios		
2GL to 3GL		
2GL to 3GL to 4GL	3GL to 4GL	
2GL to 3GL to 4GL to 5GL	3GL to 4GL to 5GL	4GL to 5GL

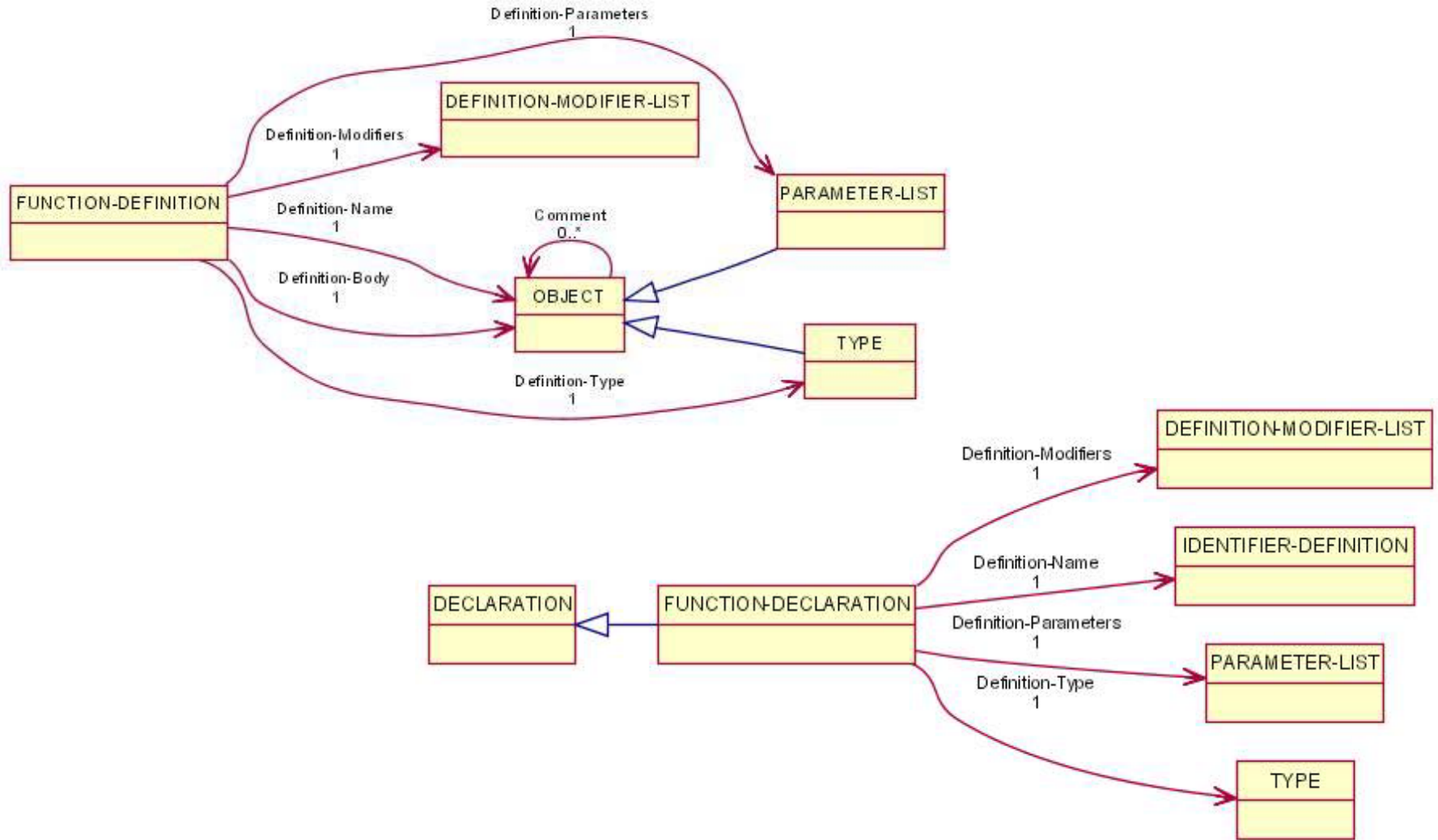
Five Examples of TSRI GASTM Model

- File Definition
- Function Definition and Function Declaration
- Block, If Statement, While Statement
- For Statement, Switch Statement
- Try Statement, Catch Statement

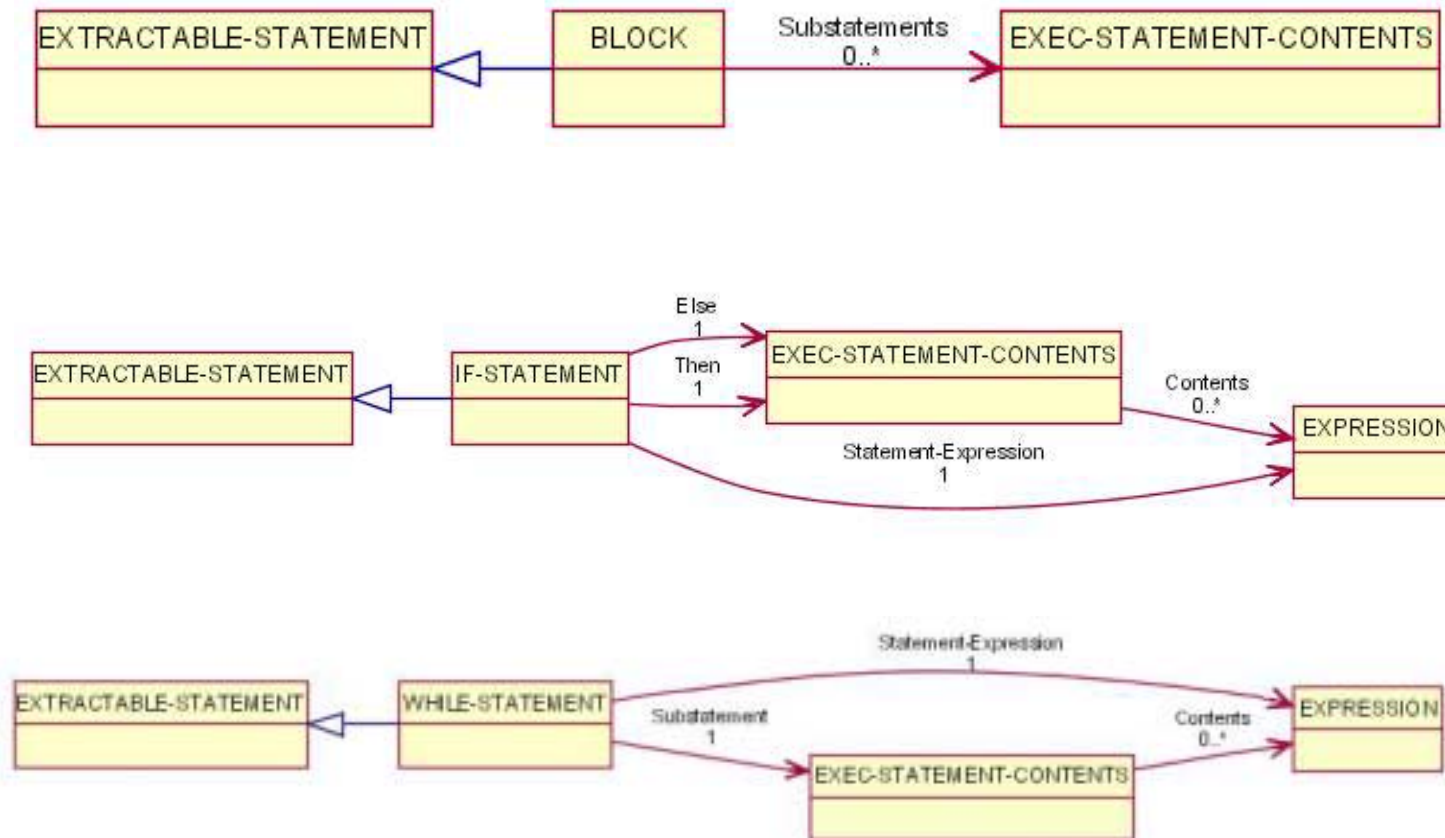
FILE-DEFINITION



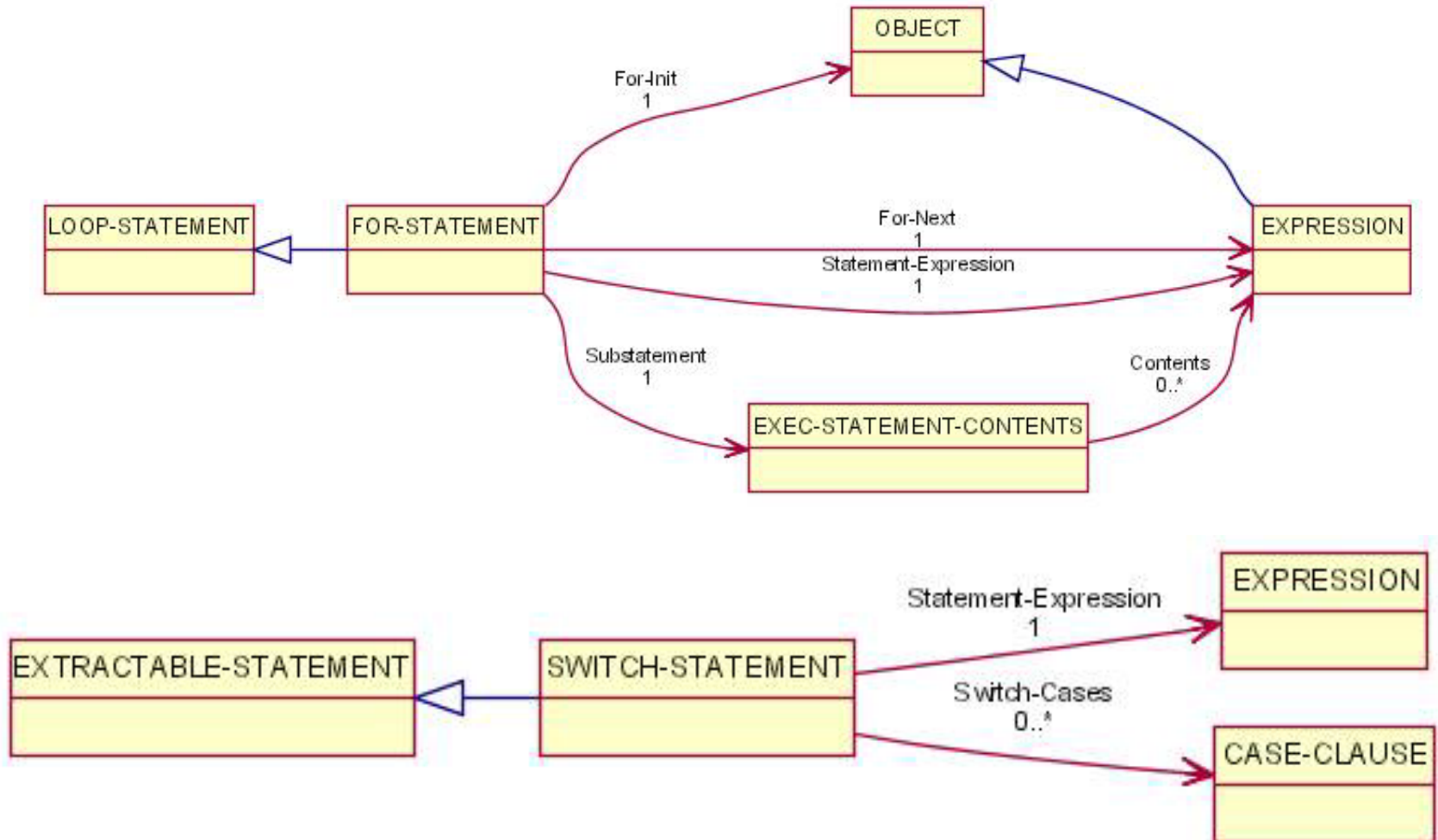
FUNCTION-DECLARATION & FUNCTION-DEFINITION



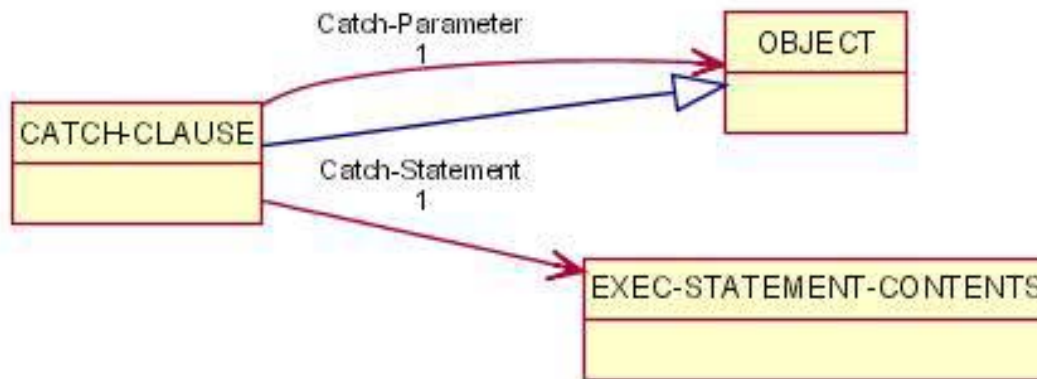
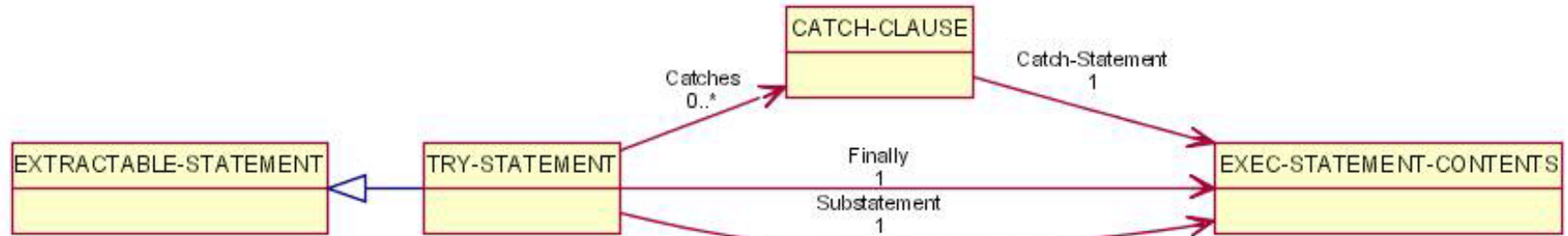
BLOCK, IF-STATEMENT, WHILE-STATEMENT



FOR-STATEMENT, SWITCH-STATEMENT



TRY-STATEMENT, CATCH-CLAUSE



TSRI GASTM Conventions

- TSRI Model Composition and Naming Conventions
 - Upper Case used for all CLASSES
 - Hyphen separates words
 - No Embedded Terms Abbreviations
 - Mixed Case for all Attributes including Associations
 - Multiplicity of all Associations
 - Textual Literal only at leaf-most level

TCS/IOS ASTM Perspective

Submitted By: TCS

Sharwan Kumar	Scientist, Software R & D : Program Analysis	91 20 5608 6315	shrawan.kumar@tcs.com
Ravindra D. Naik	Scientist, Software R & D : Re-engineering	91 20 5608 6336	rd.naik@tcs.com

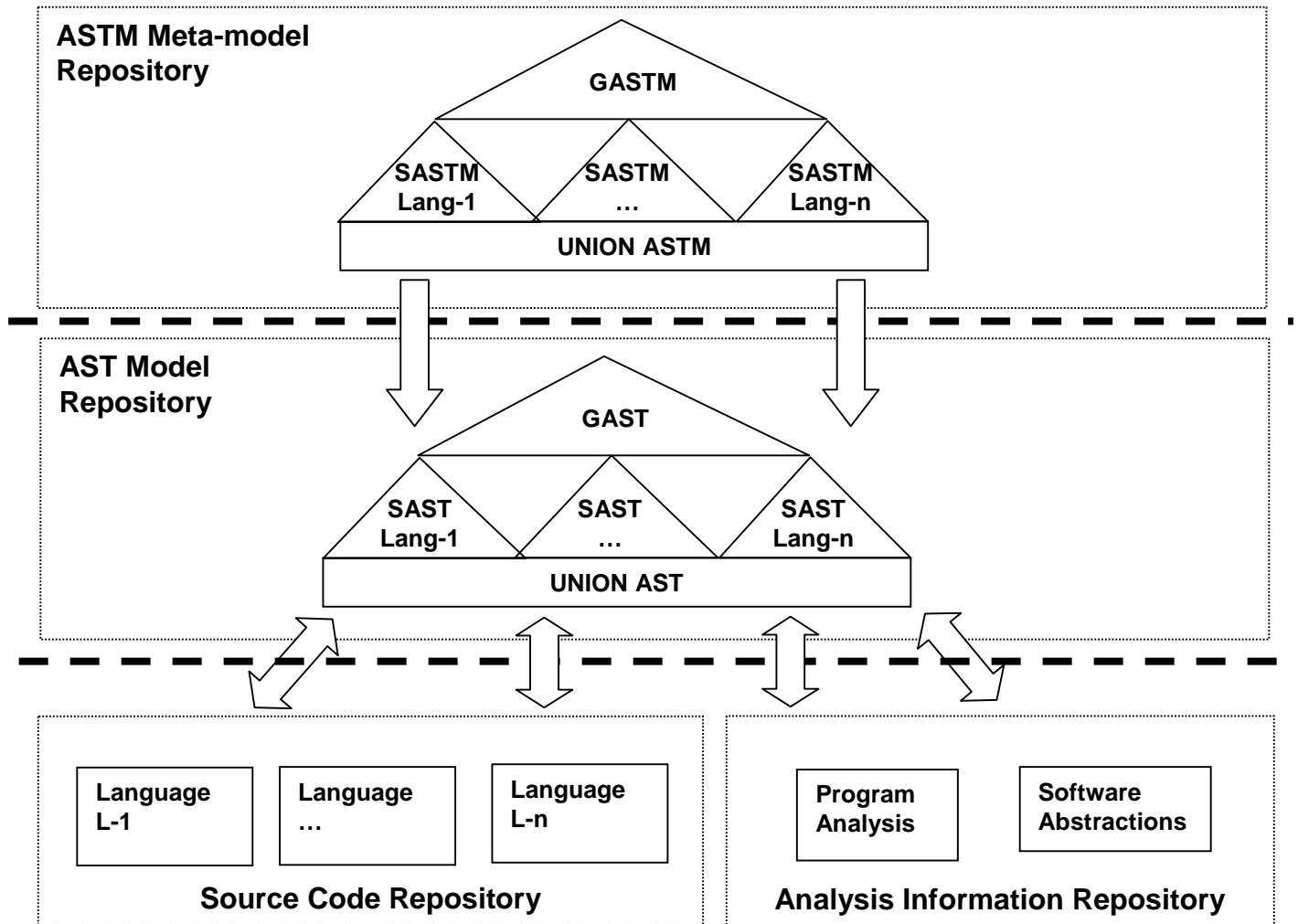
Interactive Objects

Simon Helsen	Software Architect	"+49 761 400 73 0"	simon.helsen@interactive-objects.com
Jens Rommel	Software Architect	"+49 761 400 73 0"	jens.rommel@interactive-objects.com

TCS Continuous View of GASTM and SASTMs

- The GASTM is a broad subset of lowest common denominator language elements found in many languages.
- Each SASTM model is an extension or specialization of the GASTM model.
- There are SASTM model extensions, one for each language. If required, vendors can create their own vendor-specific SASTM.
- The SASTM in combination with the GASTM completes the AST model of each language.
- Analysis is done entirely against the GASTM + SASTM without without loss of meaning.
- Multi-language and language-neutral mapping, Program analysis and Transformation is supported by a UNION model, which is a merge of GASTM and language specific SASTMs.

TCS View of Meta-model and Metadata Repository



TCS GASTM Conventions

- Classes and Attributes use standard abbreviations
- First letter of embedded terms in Upper Case; subsequent letters of term in lowercase.
- Common Primitive Expressed with Class Member Values of string, integer, float types rather than associations to Class of that Designation
- Explicit expression of all associations cardinalities.

TCS GASTM Partitioning

Explicit Packaging Of Elements

GASTM-RDBMS-Symbol
GASTM-RDBMS-Expr
GASTM-RDBMS-Stmt
GASTM-Symbol
GASTM-Expr
GASTM-Stmt
GASTM-Core

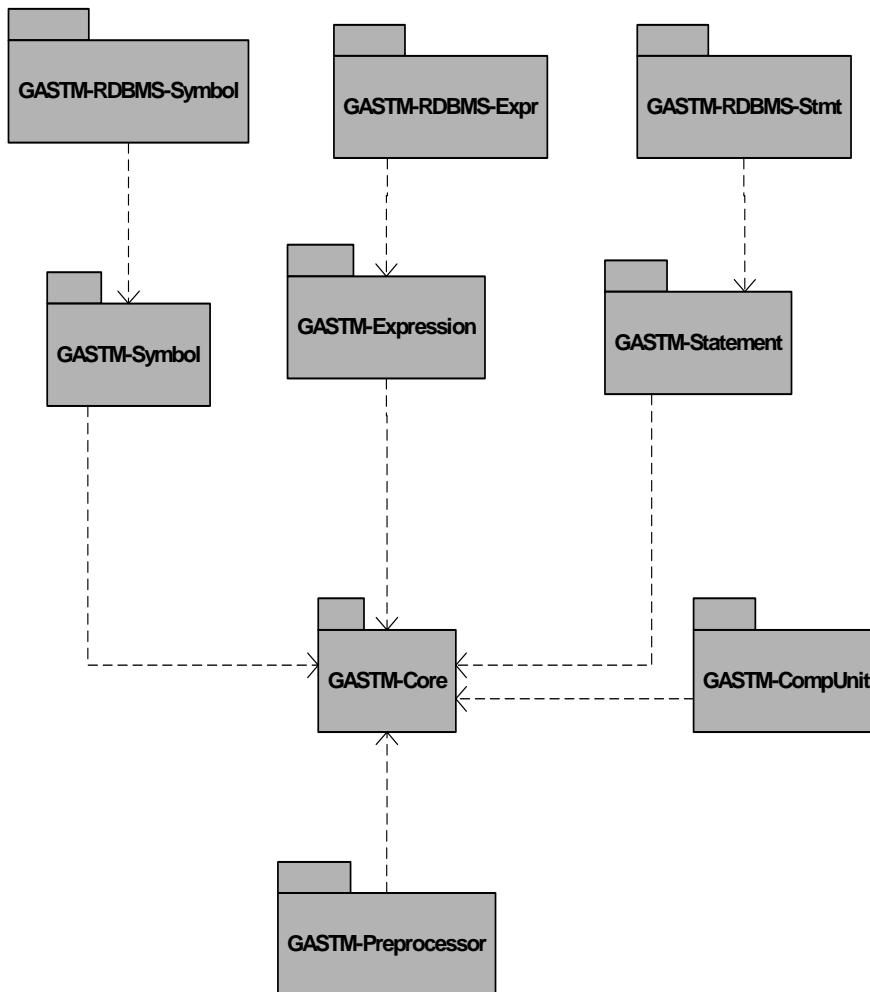
Naming of GASTM Language Specific Elements

TCS : 4GL-STATEMENTS
TSRI : ADA-STATEMENTS
FORTRAN-STATEMENTS
ETC.

Five TCS Modeling Construct Examples

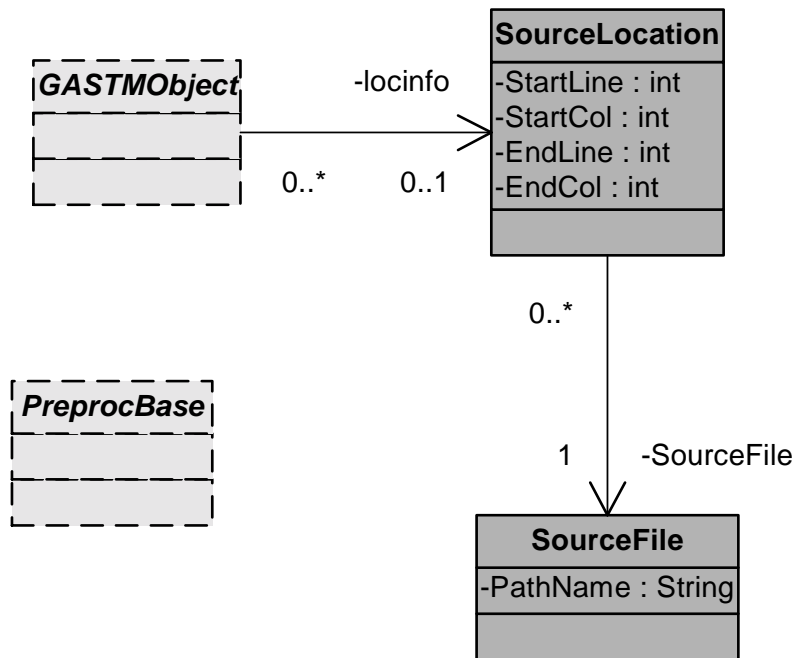
- Inheritance between Packages
- Core Package
- Statements
- Iterative, Conditional and Guarded Statements
- Symbol and Datatypes

TCS Inheritance Between Packages



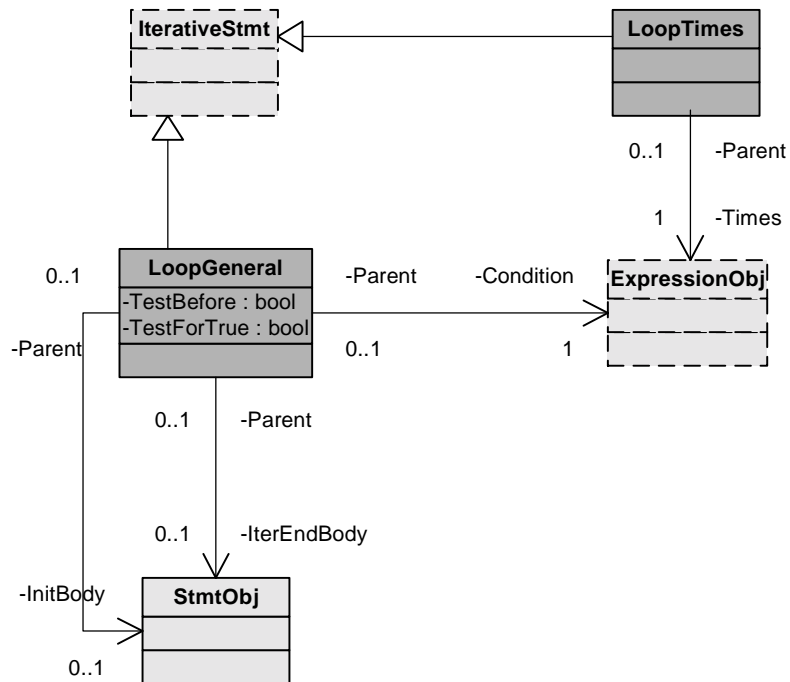
- GASTM is partitioned into logical packages
- Common objects and properties are modelled in GASTM-Core
- Common elements of general purpose Programming Languages are modelled in GASTM-Symbol, GASTM-Statement and GASTM-Expression packages
- Relational Database Query elements are modelled in the respective GASTM packages
- SASTM packages are created by inheritance from the related package

TCS Core Package



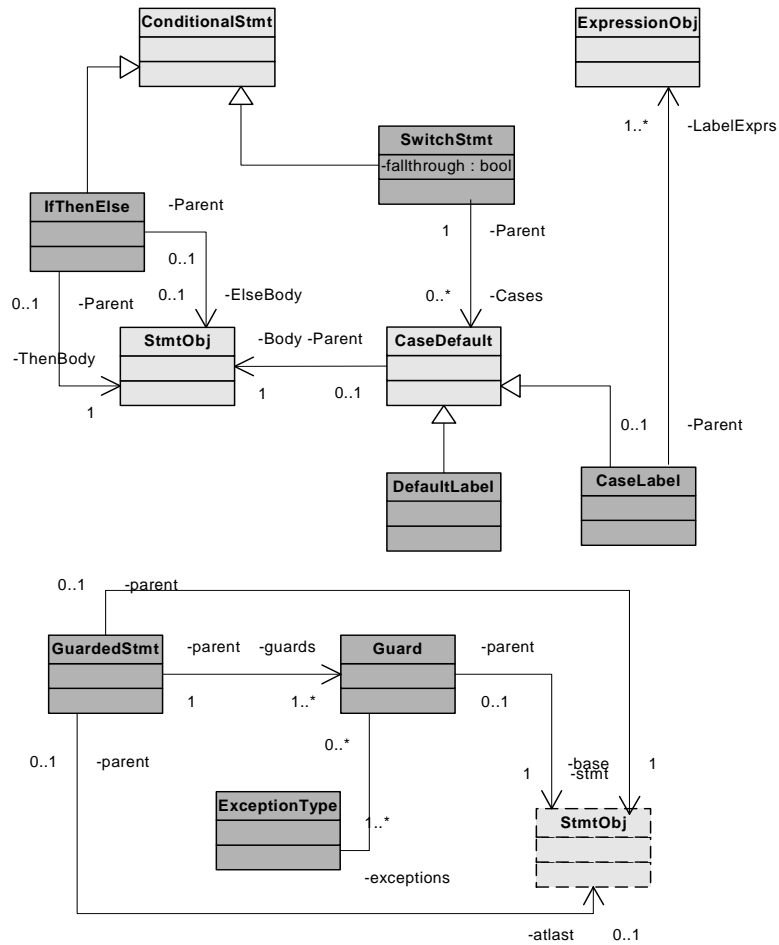
- GASTMObject is the base-most object used to capture common properties, like source code location
- SourceFile object represents the source file name of construct
- PreprocBase is the base-most object for preprocessing constructs

TCS Iterative Statement



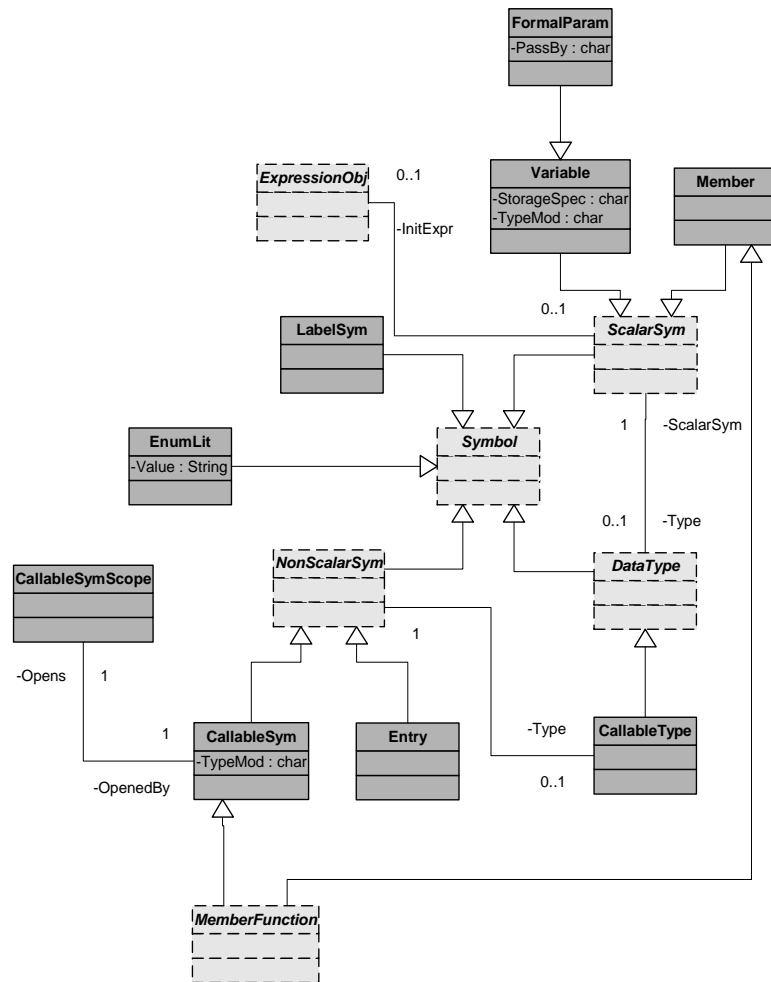
- Generic representation for loops
 - While ...
 - For ...
 - Perform ... Until ...
 - Do ...

TCS Conditional, Guarded Stmt



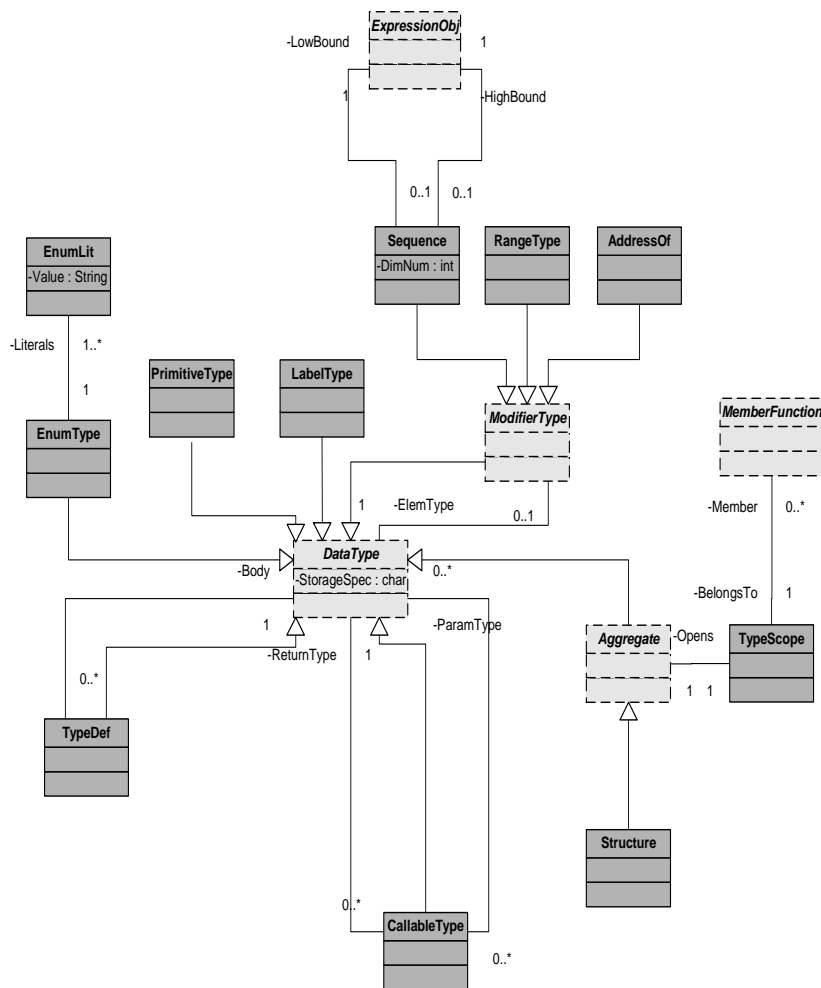
- Conditional Statements represent If-Then-Else, Switch-Case, Evaluate
- Guarded Statement represents Exceptional handling constructs
 - On Exception
 - Try ... Catch ...

TCS Symbol Package



- Variable Names
- Types of Variables
- Scope of Variables
- Subroutine Names
- Subroutine parameters
- Enumeration literals

TCS Datatype Package



- Primitive Types
- Aggregate (Structure, Record, Union)
- Modifier Types
 - Pointer, Sequence (Array), Range
- Enumeration Type
- UserDefined types (TypeDefs)

Klocwork, EDS, IBM ASTM Submission Perspective

Submitted By: klocwork

Djenana Campara	CTO, Klocwork	(613) 224-2277	djenana@klocwork.com
-----------------	---------------	----------------	--

EDS

Barbara Errickson	Portfolio Manager, Application Modernization Services	972-605-6122	barbara.errickson@eds.com
-------------------	---	--------------	--

IBM

Sara Porat, Ph.D.	Mgr, Software Assets Management Group	(972) 4-829-6309	porat@il.ibm.com
-------------------	---------------------------------------	------------------	--

Design of the ASTM Specification

- Architecture similar to CWM and KDM
 - Three level specification
 - Core package describes ASTM
 - Generic package describes common AST nodes
 - Explicit elements
 - » Scope package
 - » Statement package
 - » Expression package
 - » Identifier package
 - Implicit elements
 - » Def-use
 - » Control flow
 - » Data flow
 - Light-weight extension for language-specific definitions
- Alignment with KDM on CodeUnit, Module and Action
- Tokens are explicit part of the metamodel
- Preprocessor is part of the metamodel

How Do We Address Proprietary ASTs?

- Multitude of “AST” tools in industry.
- Each with proprietary AST definitions.
- Many of these “AST” tools generate
 - the physical code representation,
 - APIs
 - large number of useful utility functions.
- The “AST” tool
 - Code generation patterns are proprietary and closed.
 - Implementation language is “hardcoded”, and can not be changed.
 - New utility functions can not be added.
- Tools are often part of a larger Compiler Construction toolkit,
 - the generated API is used by other tools of the toolkit (for example, the AST builder module of the parser).

How Do We Address Proprietary ASTs?

- Standardizing a meta-meta-model for ASTs will enable interoperability between various AST tools.
 - Export AST definitions from one proprietary tool to the standard representation,
 - use open MOF-based code generation tools to extend functionality of the proprietary code generator
 - Achieve standardization of AST utility functions, e.g. factories, visitors, copy, match, etc.
 - Achieve standardized “reflectivity” mechanism for AST navigation and traversal .
- **Several Issues Still Remain:**
 - **No Way To Standardize on a specific AST for a given language.**
 - **No Way To Reuse Proprietary AST Implementations for Multi-Language Static Analysis tools.**

How Can We Achieve Reusable Analysis?

- **We Need A Reusable AST Implementation To Support Tools That Can Work with Multiple Languages:**
 - Analysis
 - Metrics
 - Flowcharting
 - Defect detection
 - Visualization
- This proposal includes two mechanisms:
 - Generic Interfaces to Specific ASTs
 - Visitor Pattern

In modernization of legacy
it is the problem that is a constant
and the language is a variable

What is A Generic Interface?

- **What is a Generic Interface?**

- Proprietary or Language Specific ASTs should implement generic interfaces, like “Definition”, “Declaration”, “Statement”, “Condition”, etc.
- Generic interfaces:
 - accessible either through “reflexive” traversal
 - query a certain base AST node, if it implements a certain generic interface
 - If so, through the visitor pattern (set a handler of a certain generic node), or it should be possible to do generic navigation entirely on generic nodes
 - Specific ASTs are still the main physical structure,
 - generic interfaces are a sort of “virtual” nodes, an API.
 - The physical structure of the proprietary AST does not need to be changed to support generic interfaces

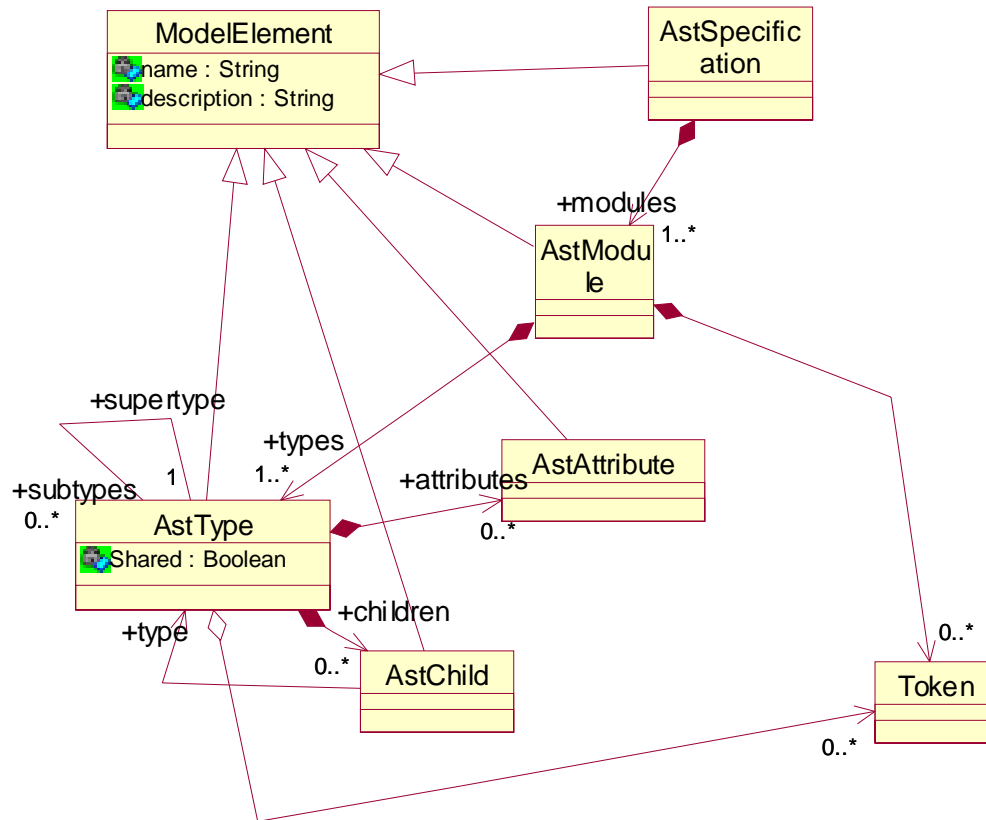
What is the Status of Generic Interfaces To ASTs?

- Static analysis tool are only now coming of age;
 - industry is becoming aware on the need for static analysis,
 - Few tools work on multiple languages;
 - static analysis were point solutions
 - Lack of motivation between tool vendors and researchers to agree on common APIs
- Static analysis technology is still rooted in compiler construction,
 - Which deals with a single language at a time.
 - Is highly complexity for defining the API to one language, let alone many.
 - Only recently have implementation technologies, generative approaches and metamodeling approaches begun to mature.
- Generic language-independent concepts are obscured by language-specific details
 - e.g. multiple variations on types, condition, assignment
 - AST traversals complicated by intricacies of AST
 - (in order to get to a condition one needs to traverse a very language-specific path).
 - Visitor pattern can mitigate this.
- Why KDM is immune to this problem (which faces the ASTM)?
 - KDM is language-independent partly because it does not involve the notion of a traversal, rather one simply selects necessary entities by performing queries

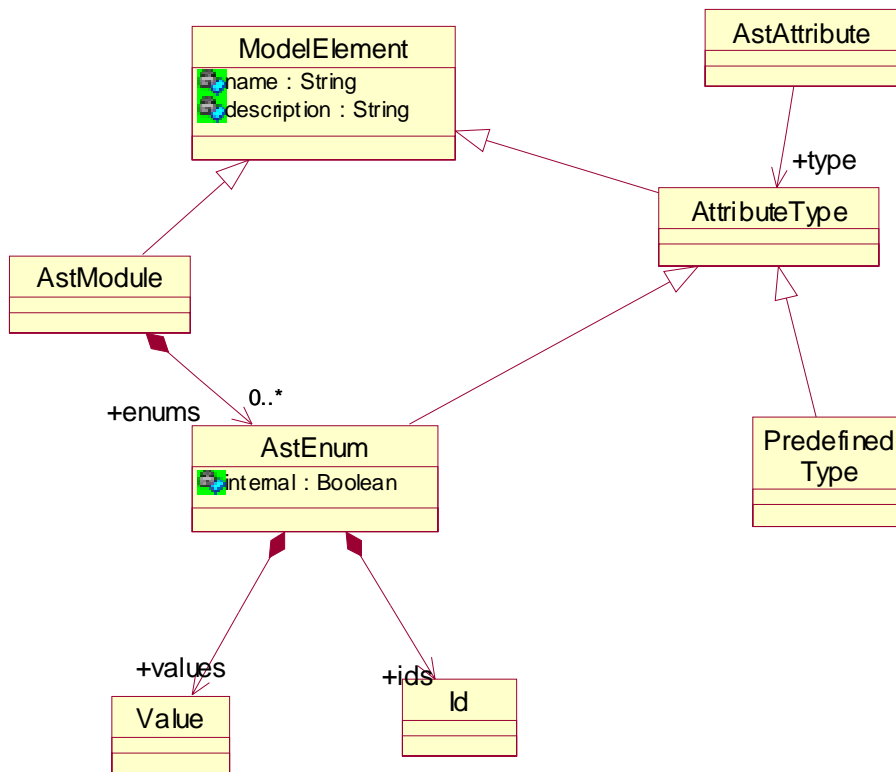
How Can We Address Language and Tool Specific AST Variation?

- Normalization
 - How to deal with “syntactic sugar” in a given language?
 - How fine-grained is AST for core language constructs ?
- Need to preserve high-level user defined information
 - information about declarations, types, type casting
 - Array and structure references
 - Templates
- Preprocessor and program generators
- Generation of source from AST
 - Optical image?
 - Whitespace ?
 - Comments ?
 - Normalization ?
- How to represent semantics of AST elements
 - None ?
 - Is a 3-address IR an AST ?
 - Is Java bytecode an AST for Java ?
 - UML action semantics ?
- Error handling policy

ASTM Core Package: Nodes

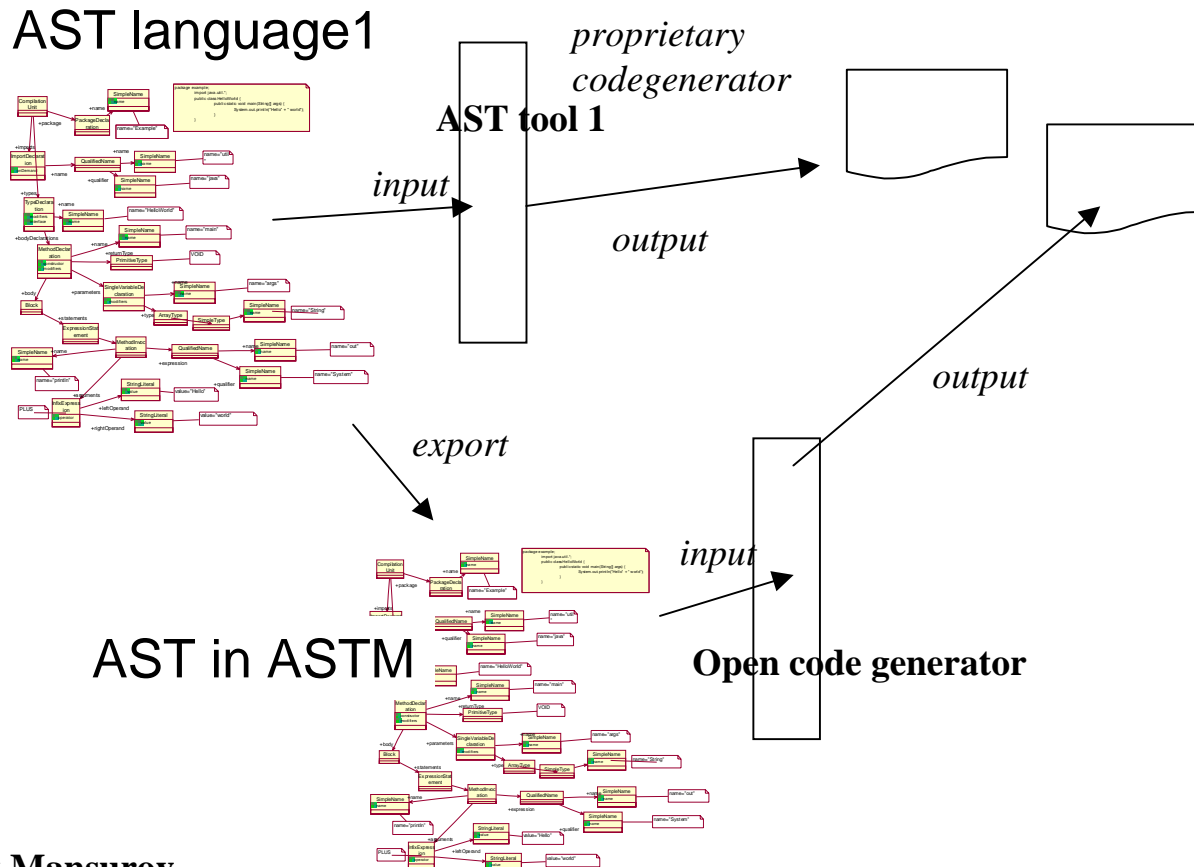


ASTM Core Package: Attributes



Generic Interface Module Opens Access to Proprietary AST Representations

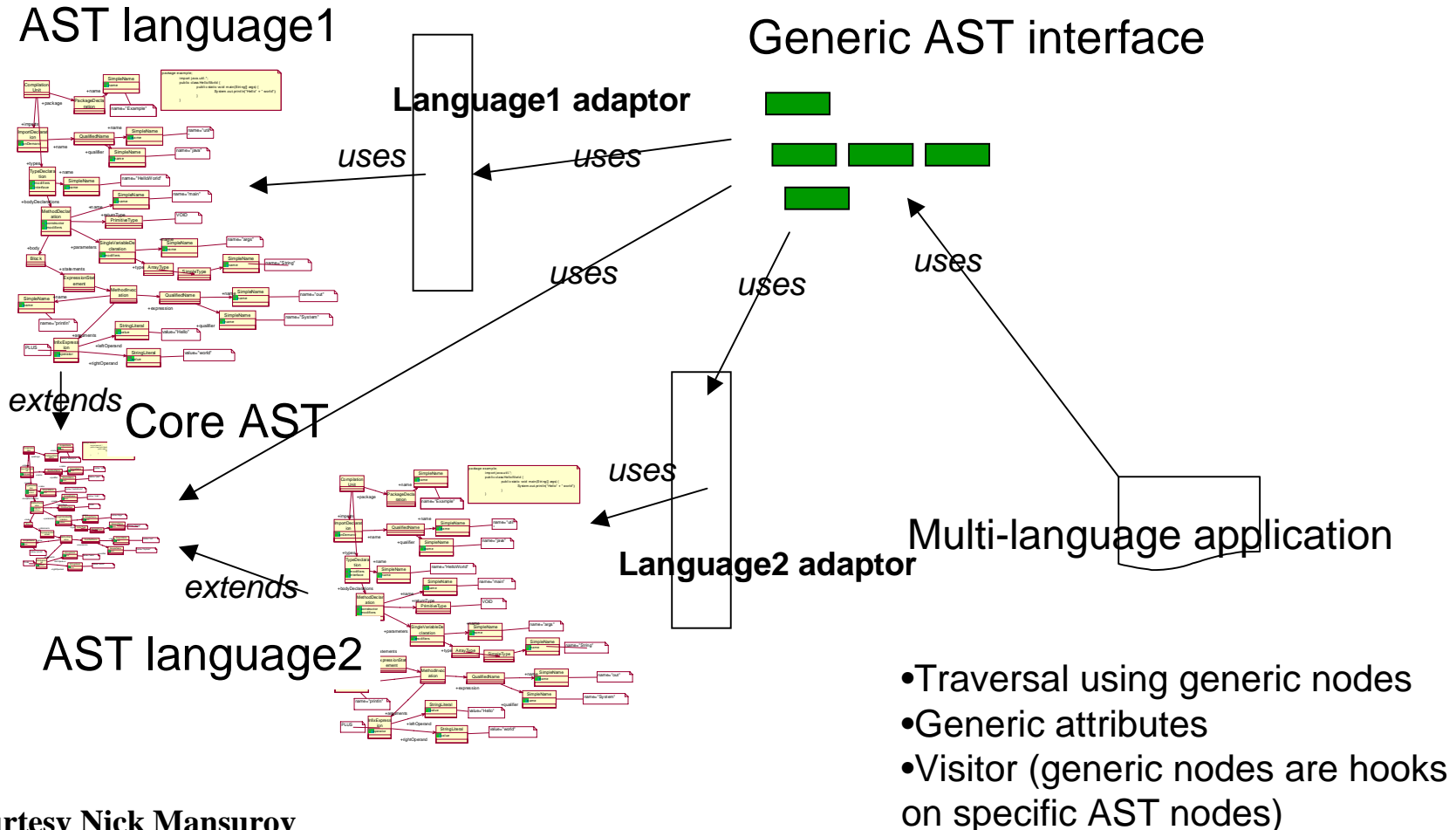
Each AST vendor develops Interface Module that maps to ASTM



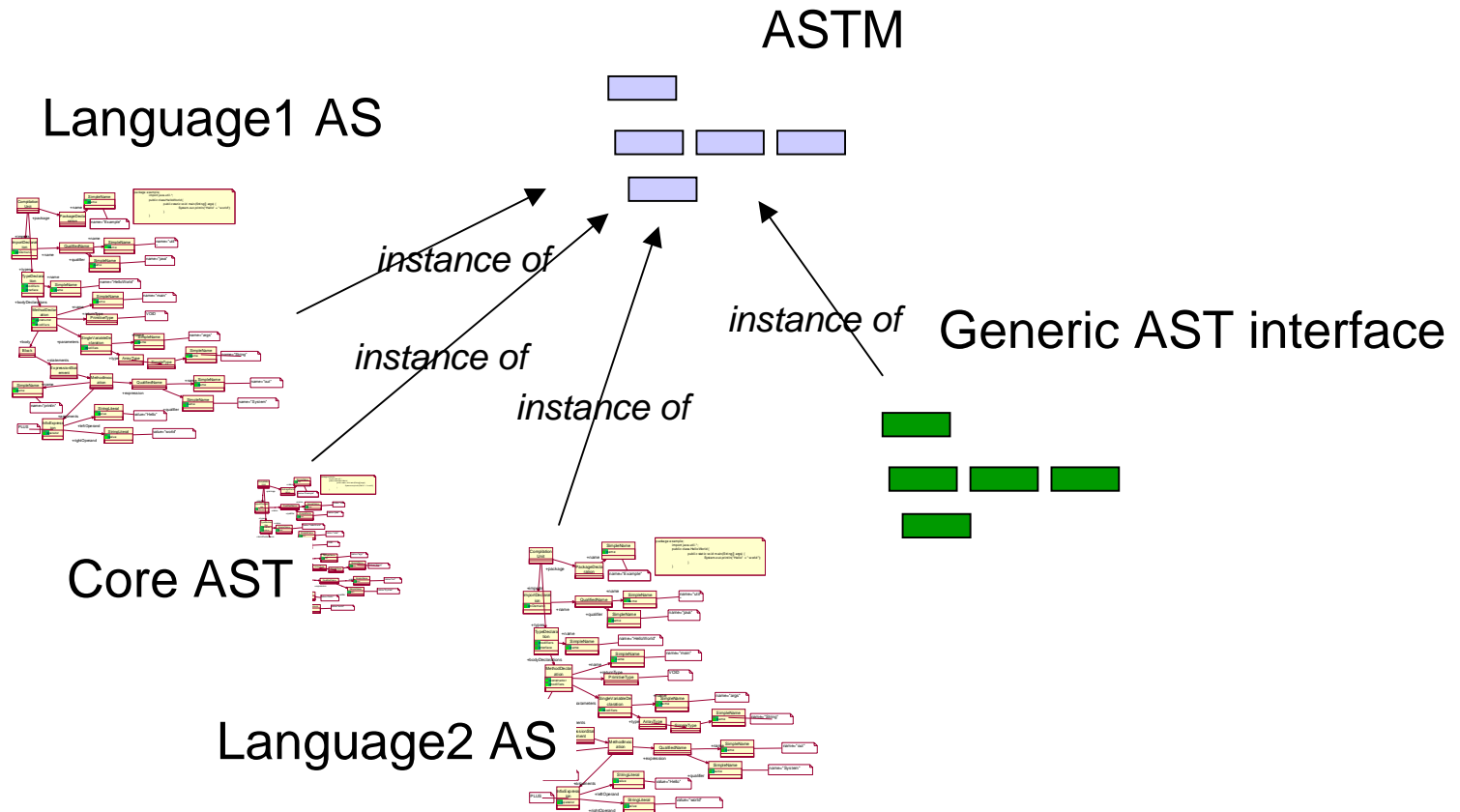
Courtesy Nick Mansurov

Use Visitor Pattern For AST Access

Each Vendor's ASTs use the Generic AST Interface (meta-meta-mode) and Extends a Minimal CORE AST.



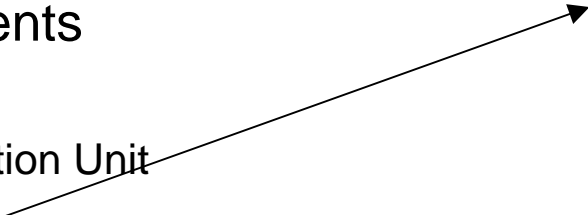
Proprietary ASTs Become instances of Extended Core AST



Interface Module Implementation via the Dynamic Visitor Pattern

- **Vendors can achieve compatibility with ASTM**
 - By Defining Proprietary AST models as a SASTM extensions of the GASTM meta-meta-model.
- **To achieve interchangeability and interoperability**
 - the SASTM model and objects must be expressed in a language which can use the DVP (e.g. Java or C++)
- **DVP Services From Implementers of the Standard**
 - will provide operations for visiting, describing, copying, mapping, etc. elements of the proprietary ASTs.

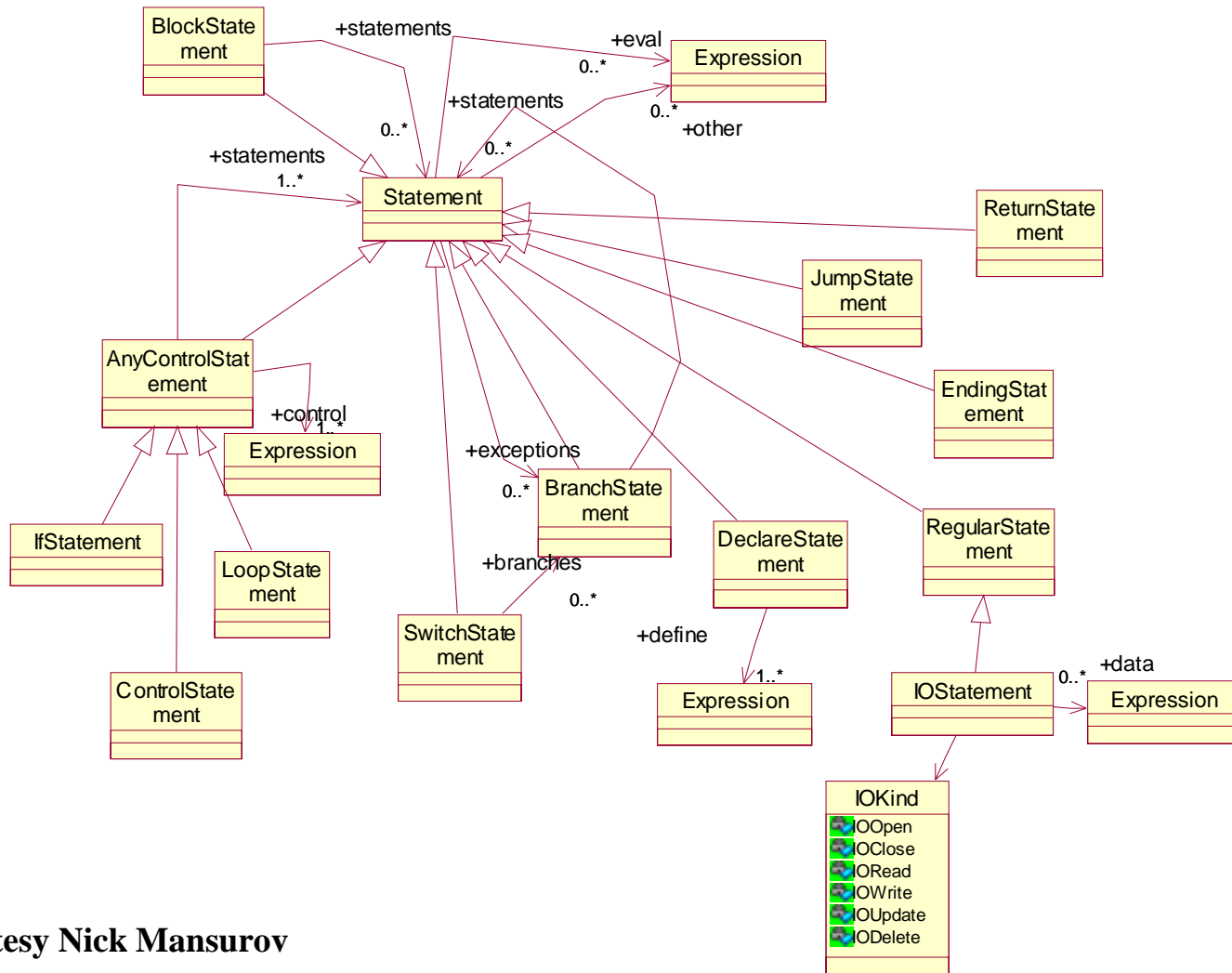
Generic AST

- Explicit elements
 - Structure
 - Translation Unit
 - CodeUnit 
 - Action
 - Statement
 - Expression
 - Identifier
 - Implicit elements
 - Scope
 - Binding
 - Control flow
 - Data flow
- CodeUnit
 - CallableUnit
 - ClassUnit
 - MethodUnit
 - MemberUnit
 - DeclarationUnit
 - DataTypeUnit
 - GlobalData
 - LocalData
 - ParameterData
 - MacroUnit

Key GASTM Modeling Elements

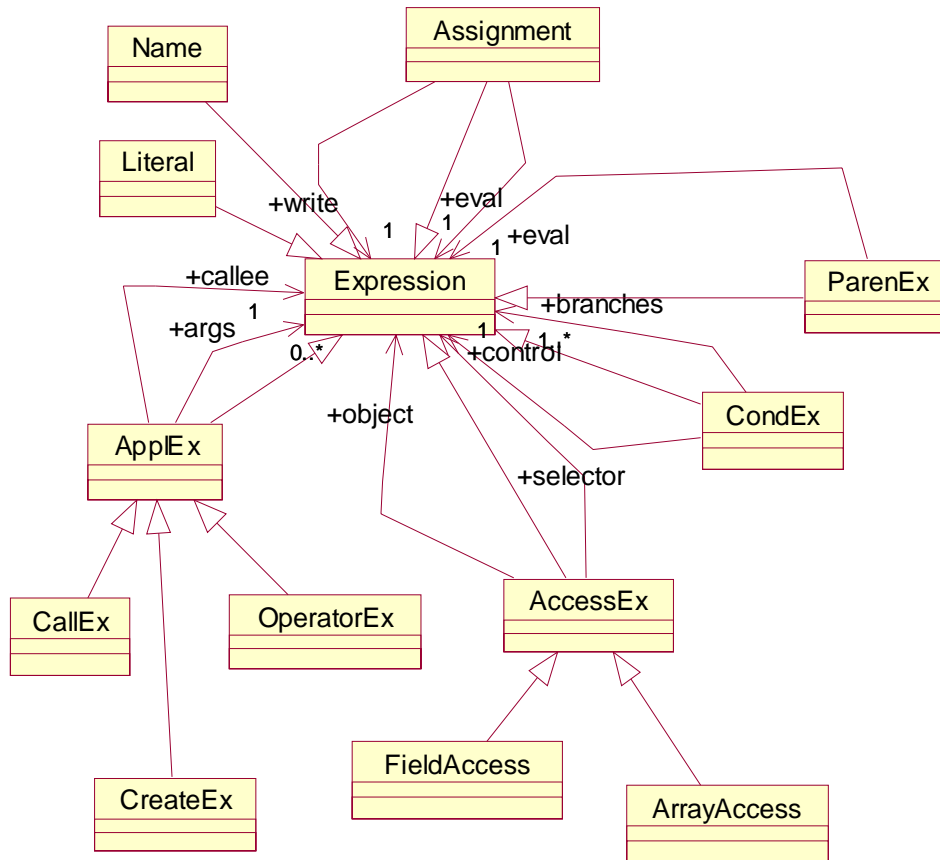
- Lists (types or untyped)
- Binding and binding resolution
- Compilation unit
- Declarations (nested)
- Types and instances, instant types
- Block
- Statement
- Expression
- name
- exceptions

GASTM Statement Package

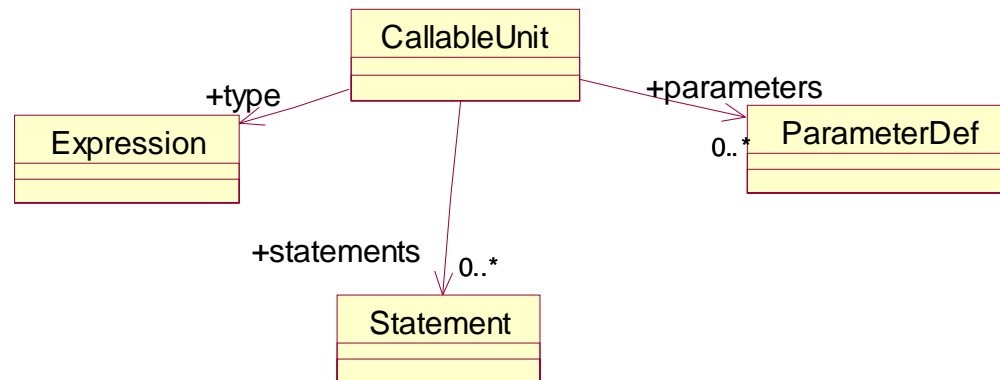


Courtesy Nick Mansurov

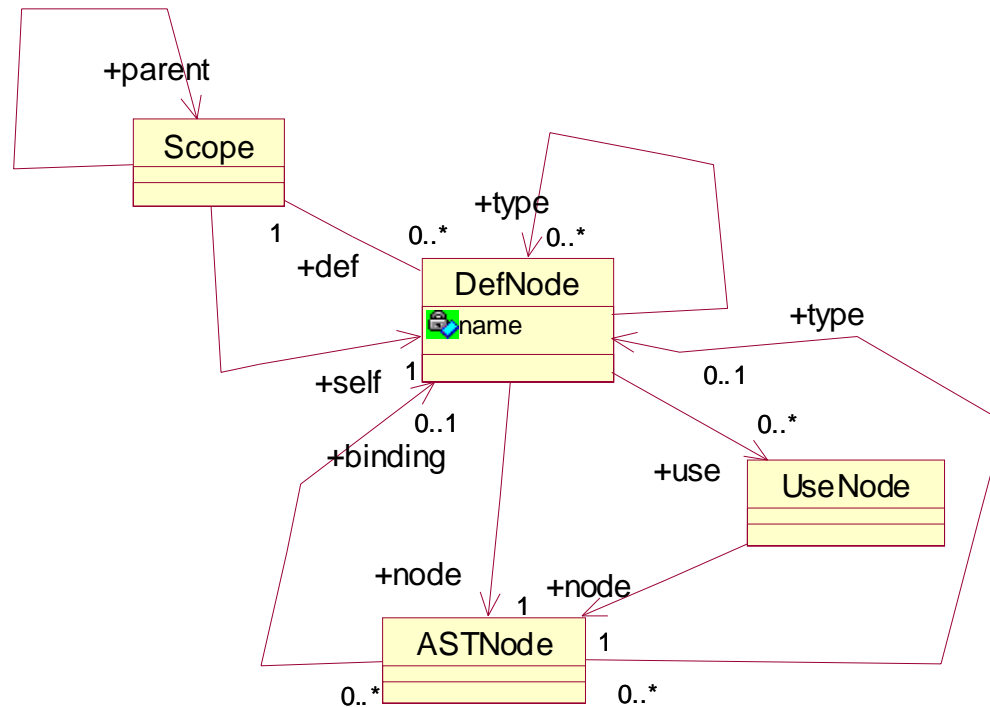
GASTM Expression Package



GASTM CodeUnit Package



GASTM Binding Package



Gaps and Issues to be Addressed

- **GAPS BETWEEN RFP REQUIREMENTS AND SUBMISSIONS**
 - Absence of support for 5GL and 2GL In The Three Submissions.
 - Basic Semantic Not Required In the RFP, But Needed In The Standard.
 - Demonstration Needed Of How the ASTM Supports the KDM.
 - A Common Glossary of Terms Is Needed To Commence Reconciliation Of GAST classes, associations, attributes In the Three Core Models.
 - A List of References Is Needed to Other non-OMG standards
 - The Standard Should include an Agreed Upon Method for Reconstituting Source Code from the AST Models.
 - The Three Models should be Reconciled with Respect to Subgrouping GASTM Elements for Various Languages and Language Families.
 - Demonstration Needed Of Mappings Between ASTM And KDM
 - Demonstration Needed Of Mappings Between ASTM And OMG models.
 - Demonstration Of How Unusual Constructs be Captured Within the CORE GASTM.

Additional RFP Evaluation Criteria For The Demonstration.

- A mapping of the GASTM and SASTM models into a MOF repository (I.e. eclipse or .xmi) is required.
 - Eclipse regards CVS or SOURCE as repository
 - No MOF repository is known adequate for storing ASTs
- A Demonstration is required to Show How Each Of The Three Perspectives Support Operations On Multiple Languages. (analysis,metrics, visualization, mapping, etc)
- A Demonstration is required using the GASTM to represent information about the existing system using all three perspectives.
- A Demonstration is required showing applicability of the perspective to many languages.

Time Table For Joint Submission Team

- Sept 30th
 - Complete ASTM Tutorial 1.0
- Dec 5th Address all gaps and issues
 - Agreed upon GASTM model
 - Capture each perspective in written specification
 - Initial Interface Module defined
- Jan 23rd Revised Submission on Server.

OMG ADM Task Force Industry Participation

<http://adm.omg.org/>

