ICSME 2016, Oct 5, 2016

A Taxonomy for Program Metamodels in Program Reverse Engineering



Hironori Washizaki[1], Yann-Gael Gueheneuc[2], Foutse Khomh[2]

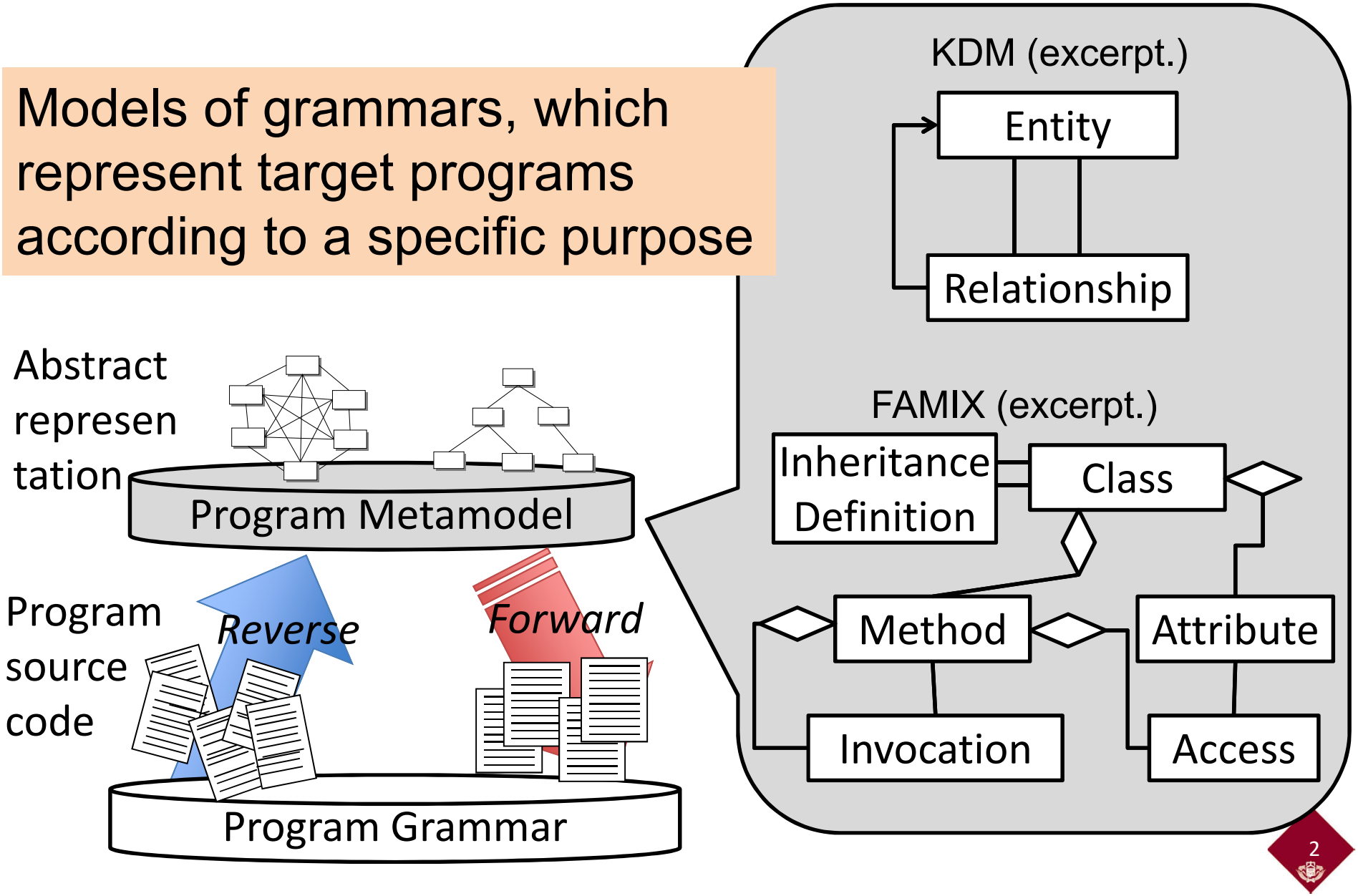
1 WASEDA University
NII National Institute of Informatics
SYSTEM INFORMATION CO., LTD.

2 POLYTECHNIQUE MONTRÉAL

H. Washizaki, Y-G. Gueheneuc, F. Khomh, “A Taxonomy for Program Metamodels in Program Reverse Engineering,” 32nd IEEE International Conference on Software Maintenance and Evolution (ICSME 2016)

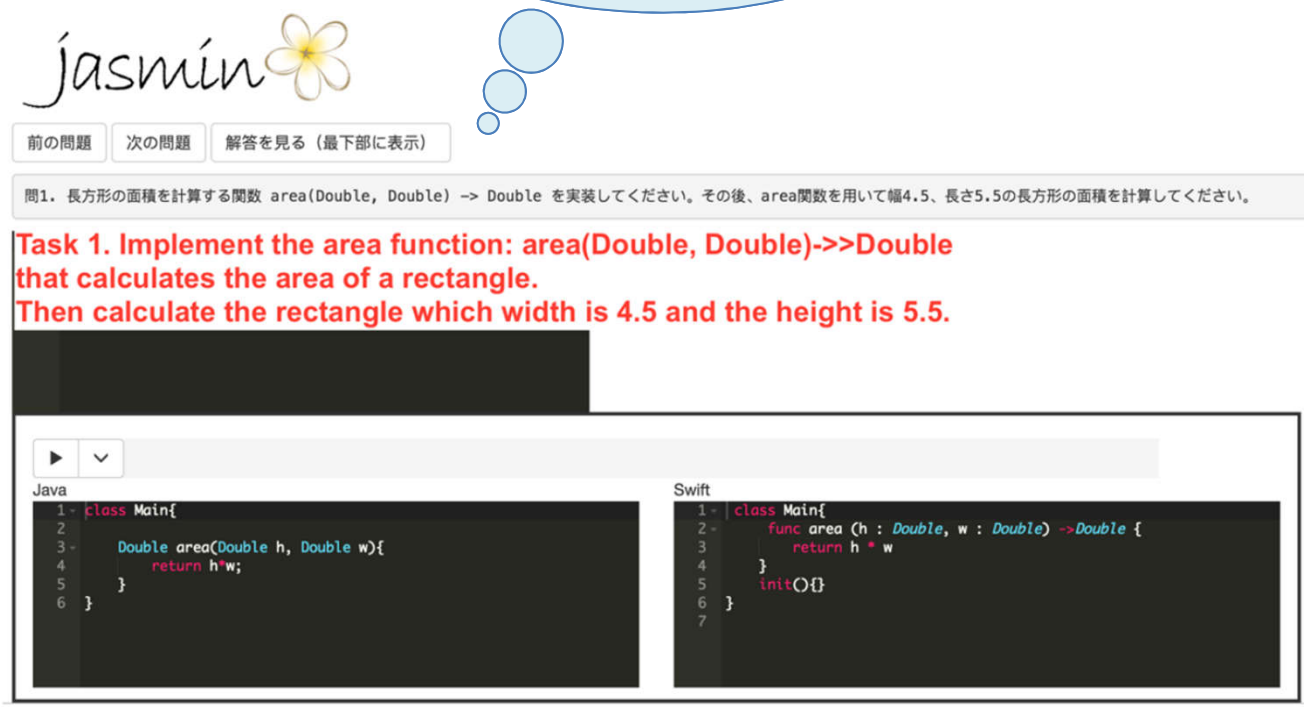
What Are Program Metamodels?

Models of grammars, which represent target programs according to a specific purpose



Case 1: Program Transformation [HICSS'17]

"Program meta model"
Package, class, method, statement, …
Navigation, transformation



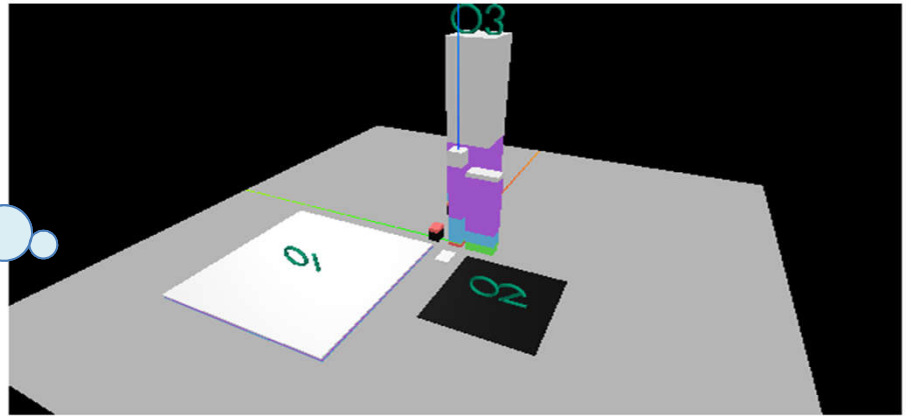
Juhua Li, Kazunori Sakamoto, Hironori Washizaki, Yoshiaki Fukazawa, "Promotion of Educational Effectiveness by Translation-based Programming Language Learning Using Java and Swift," 50th Annual Hawaii International Conference on System Sciences (HICSS-50), Waikoloa, Hawaii, Jan 4-7, 2017

Case 2: Program Visualization [VISSOFT'16]

"Database schema"
Package, class
History, …

Model name:test(2016-06-04)

Metrics:Number of defects



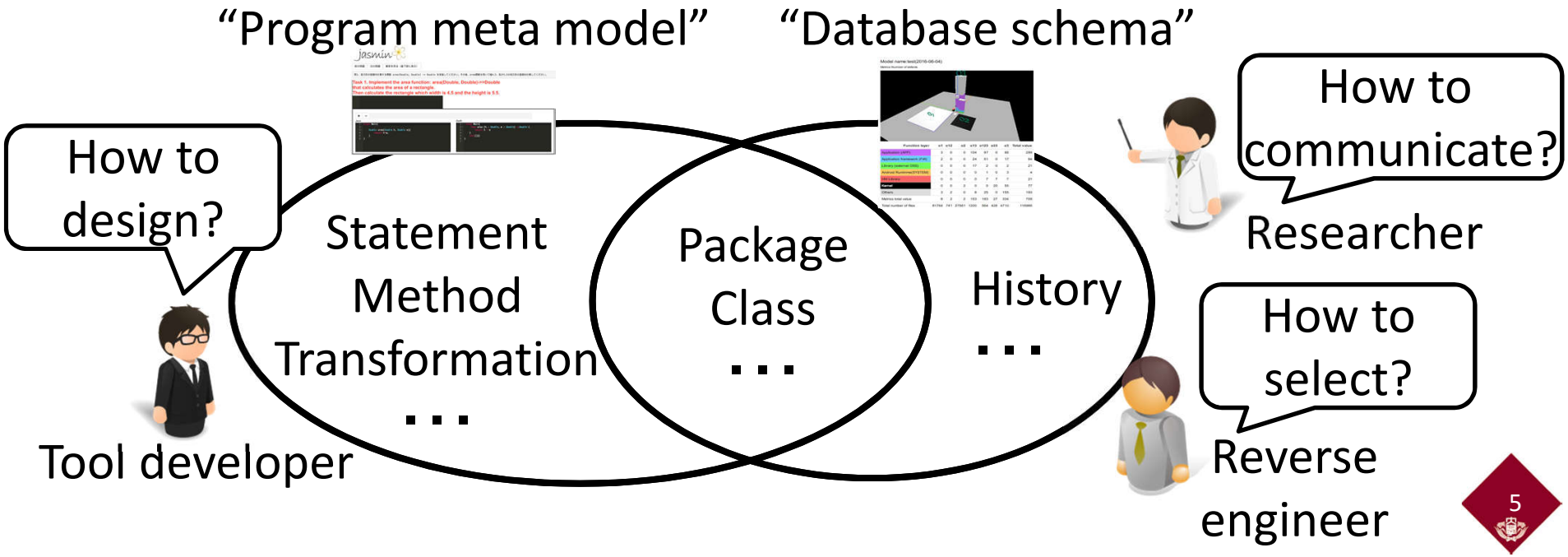
Function layer	o1	o12	o2	o13	o123	o23	o3	Total value
Application (APP)	3	0	0	104	97	0	95	299
Application framework (FW)	2	0	0	24	51	0	17	94
Library (external OSS)	0	0	0	17	2	0	2	21
Android Runtinme(SYSTEM)	0	0	0	0	1	0	3	4
HW Library	0	0	0	0	7	7	7	21
Kernel	0	0	2	0	0	20	55	77
Others	3	2	0	8	25	0	155	193
Metrics total value	8	2	2	153	183	27	334	709
Total number of files	81764	741	27561	1200	564	426	4710	116966

Ryosuke Ishizue, Hironori Washizaki, Yoshiaki Fukazawa, Sakae Inoue, Yoshiiku Hanai, Masanobu Kanazawa and Katsushi Namba, "Metrics visualization technique based on the origins and function layers for OSS-based development," 4th IEEE Working Conference on Software Visualization (VISSOFT 2016)

What's the problem?

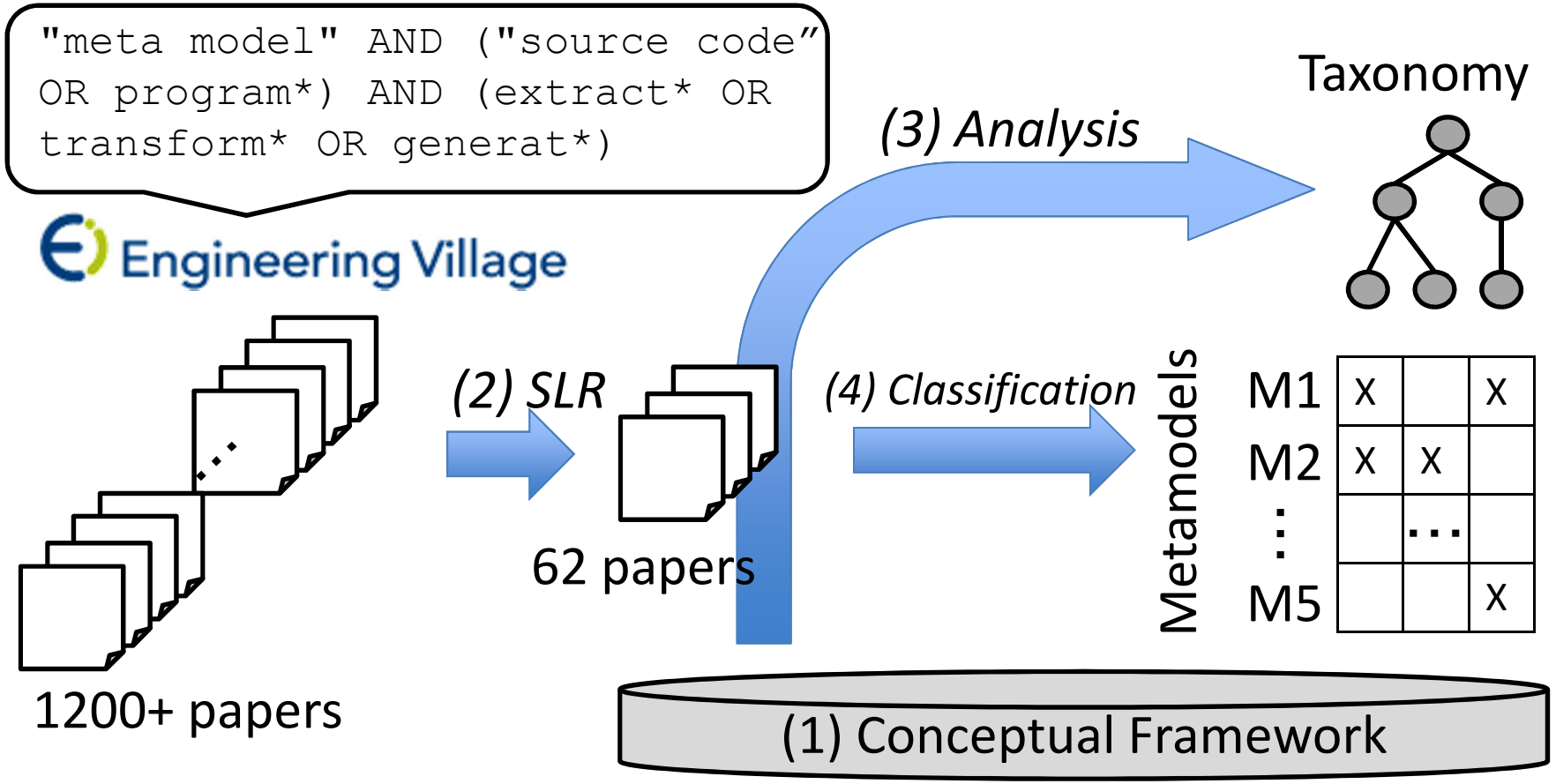
Concepts are not uniformly recognized → Need a common vocabulary!

Hard to select, design or communicate metamodels without common classification → Need a comprehensive taxonomy and classification based on the taxonomy!



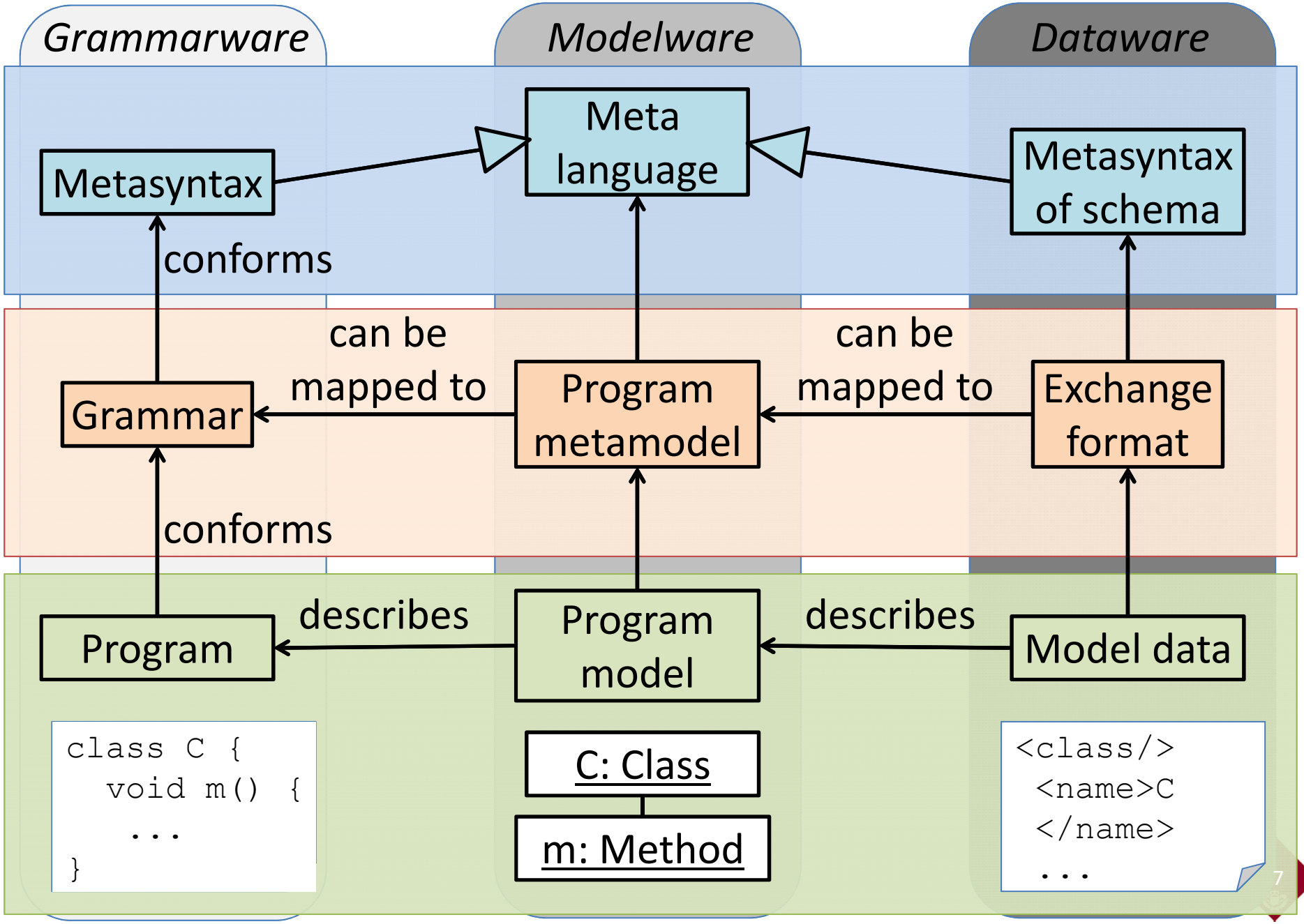
Research Goal and Method

To provide a comprehensive taxonomy and use this taxonomy to classify some popular metamodels

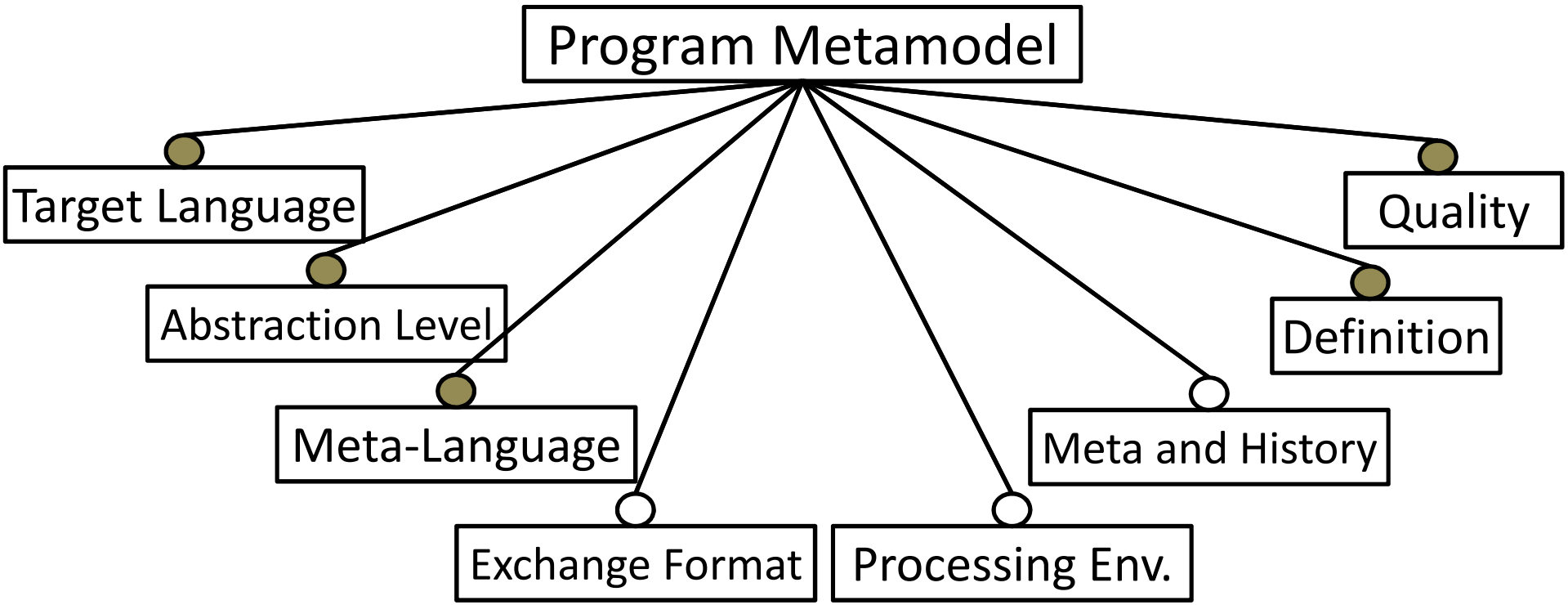


Engineering Village covers 12 trusted databases incl. Ei Compendex and Inspec http://www.engineeringvillage.com/

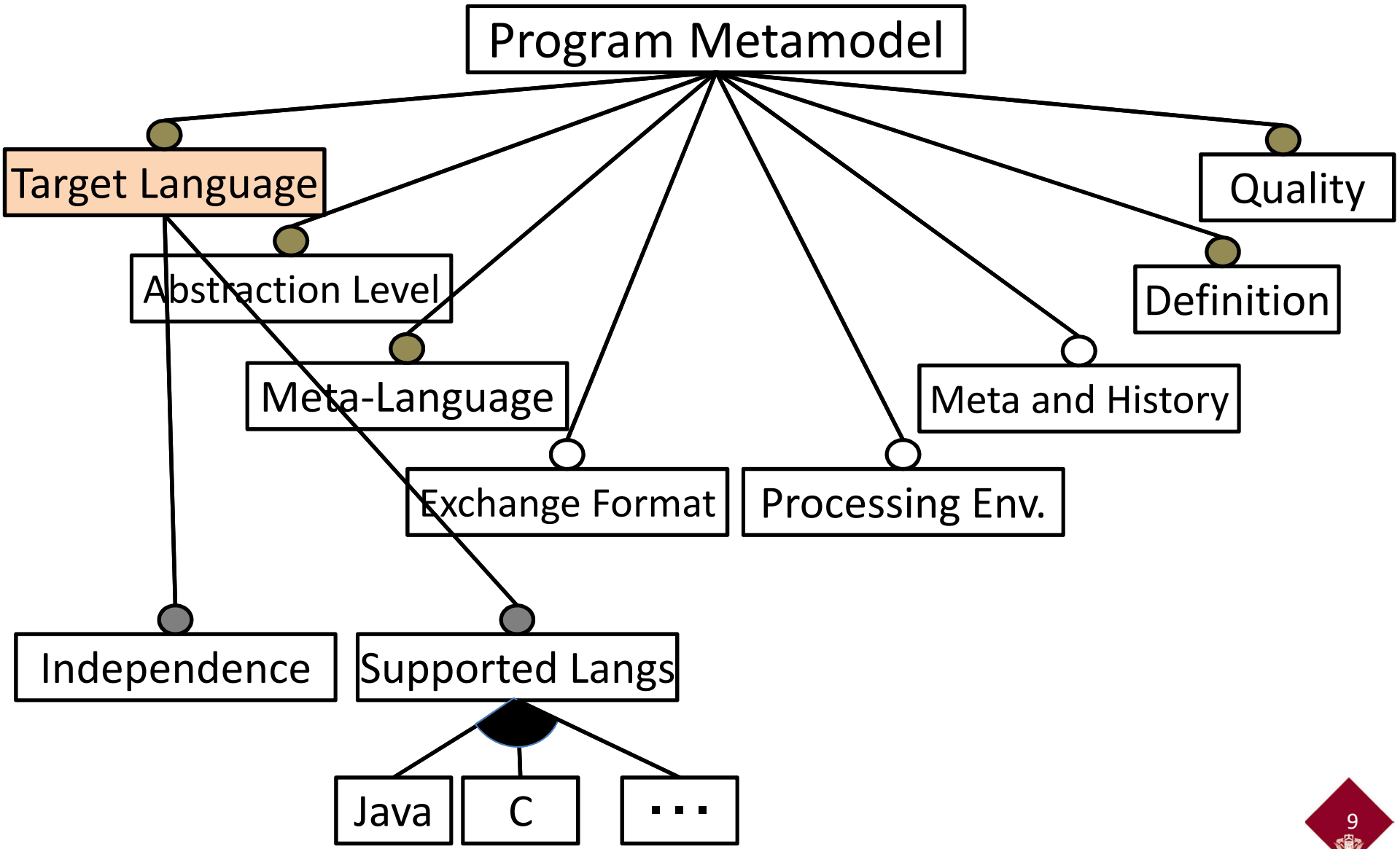
(1) Conceptual framework



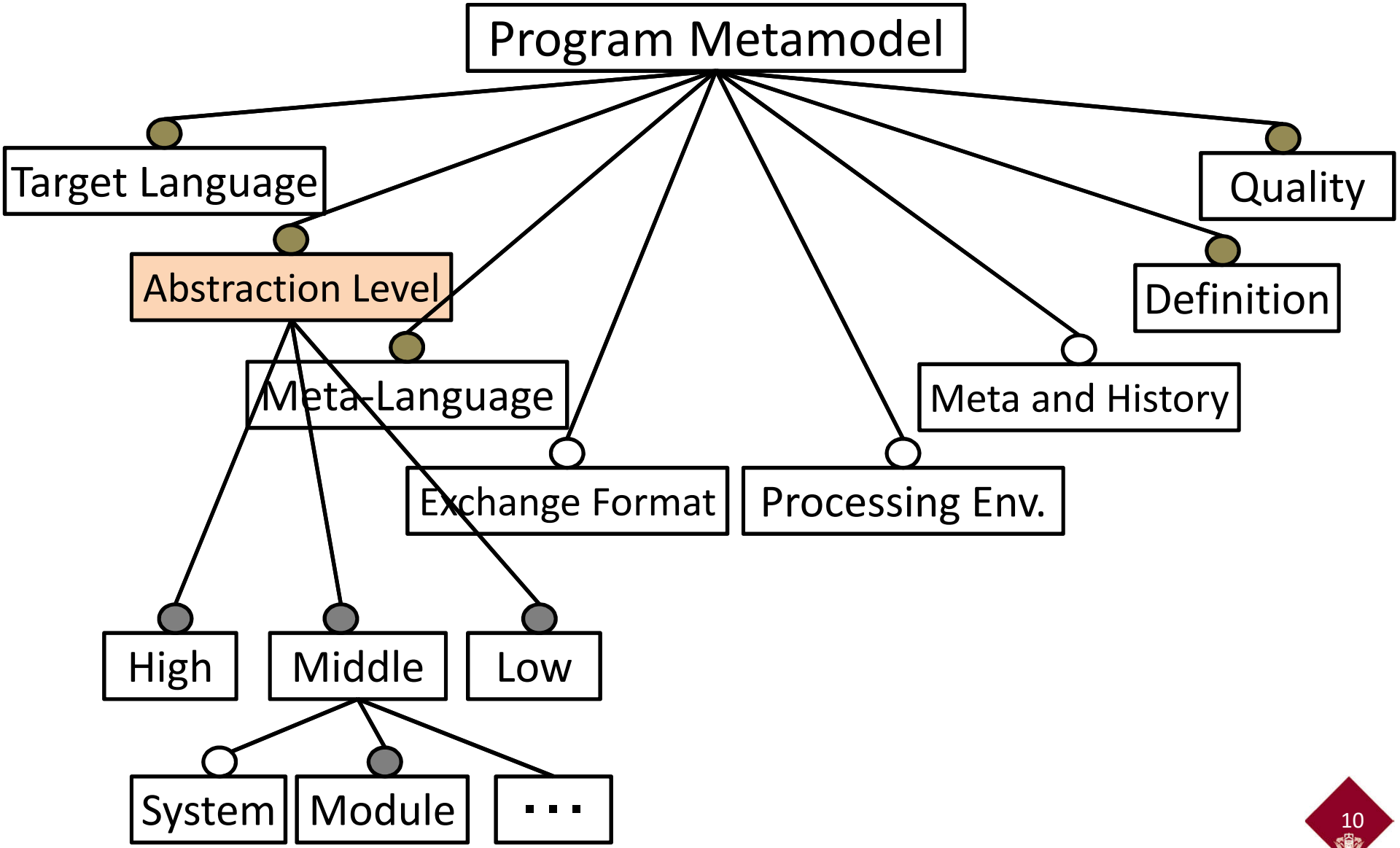
(2-3) ProMeTA: Program Metamodel Taxonomy



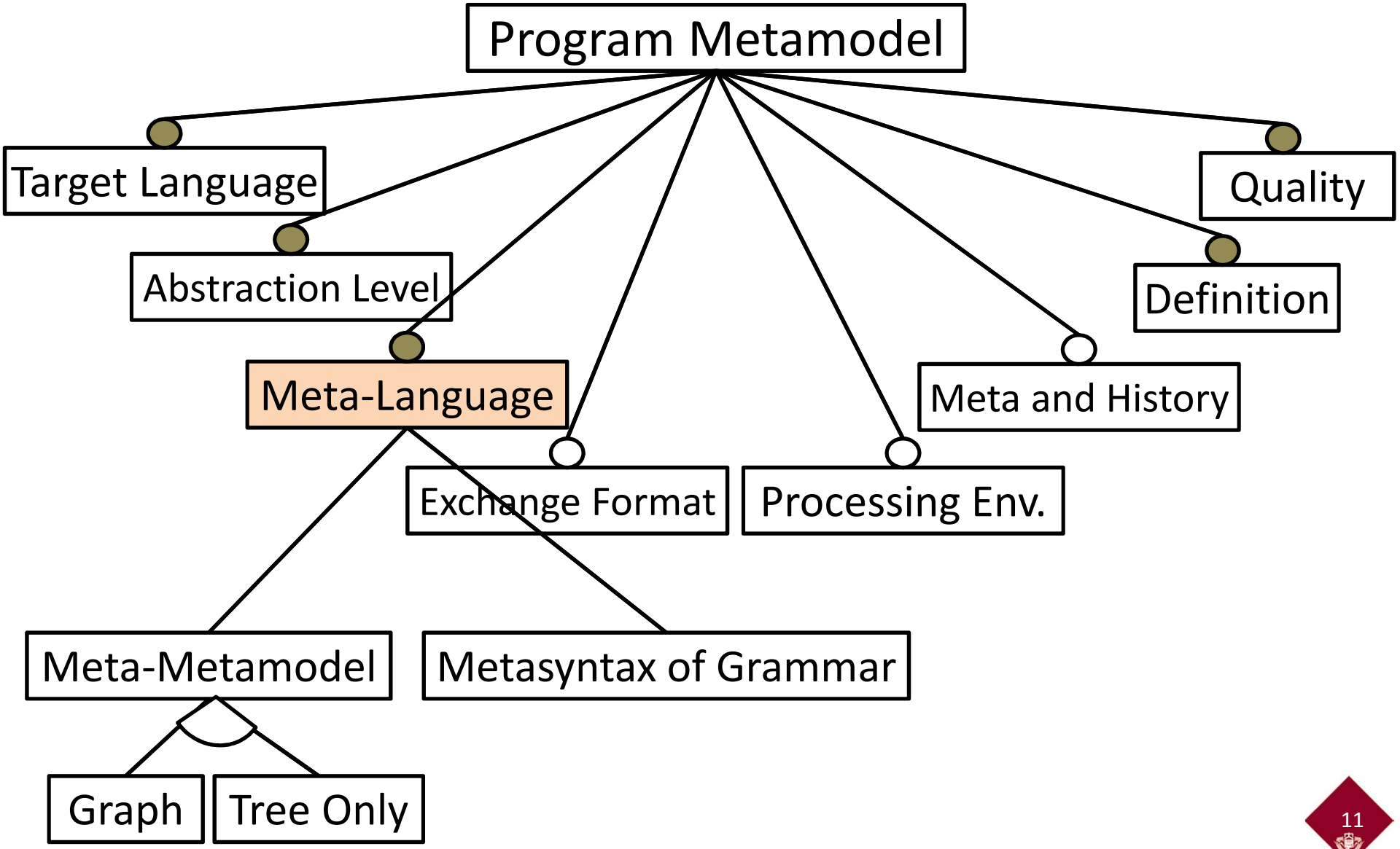
(2-3) ProMeTA – Target Language



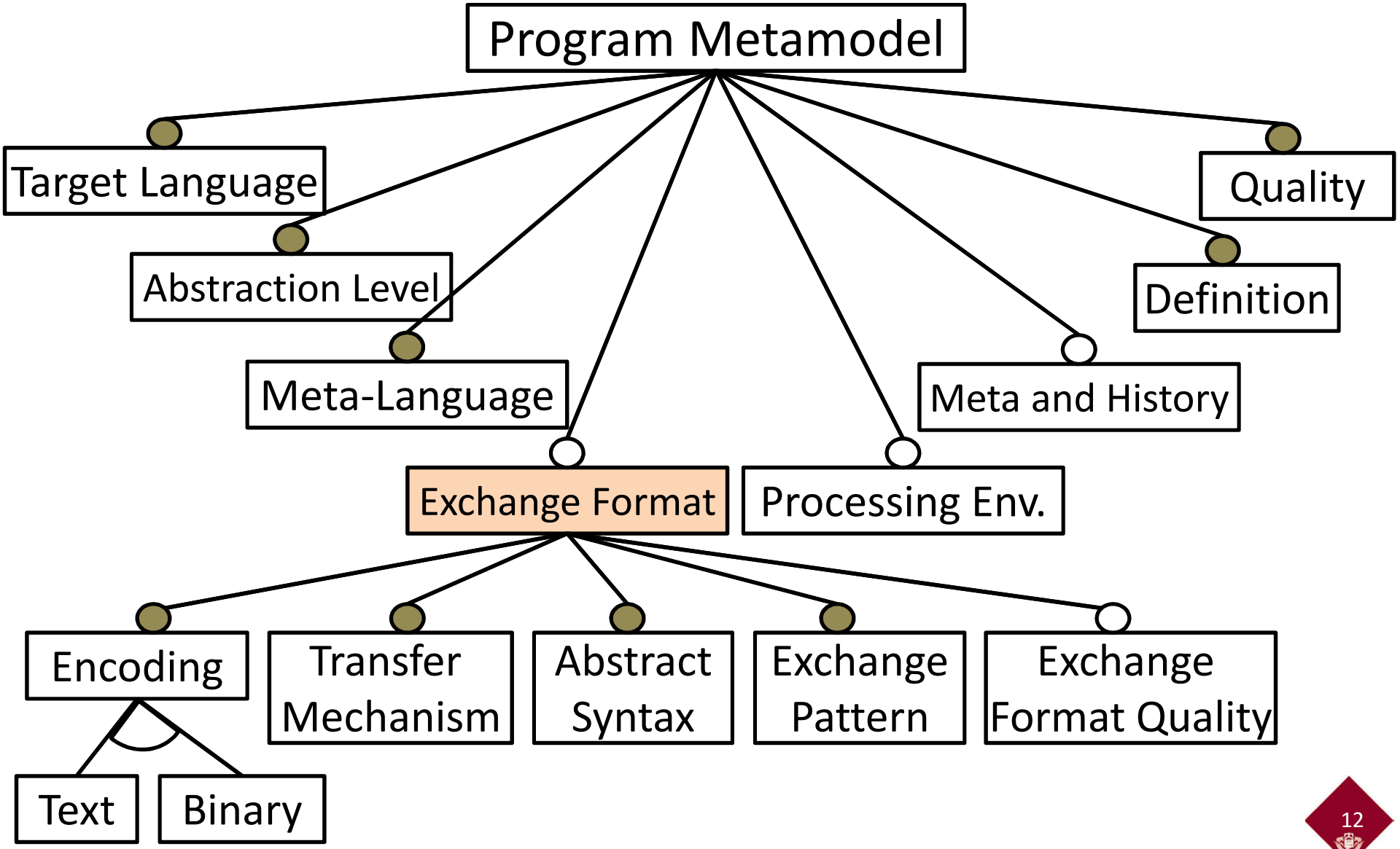
(2-3) ProMeTA – Abstraction Level



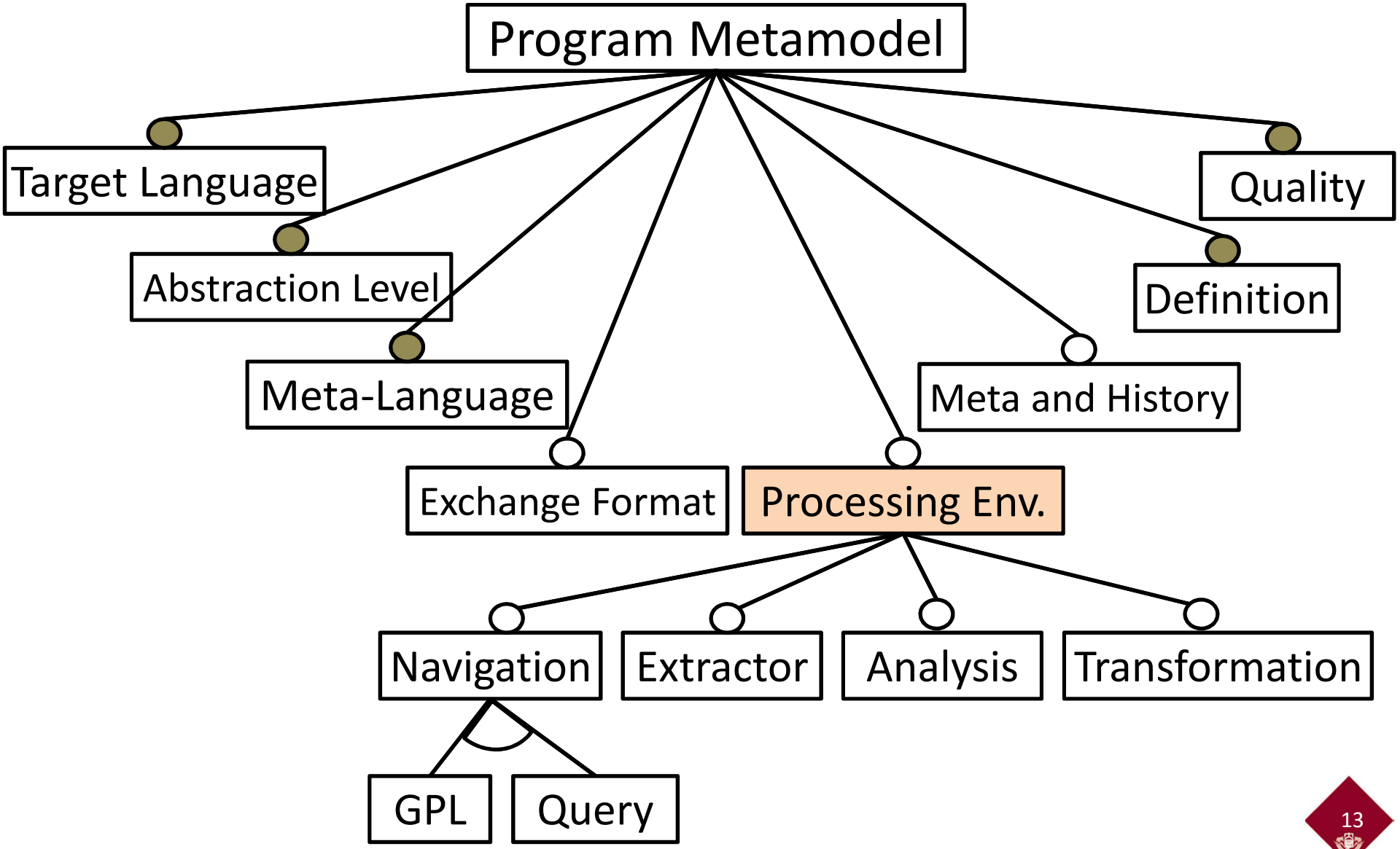
(2-3) ProMeTA – Meta-Language



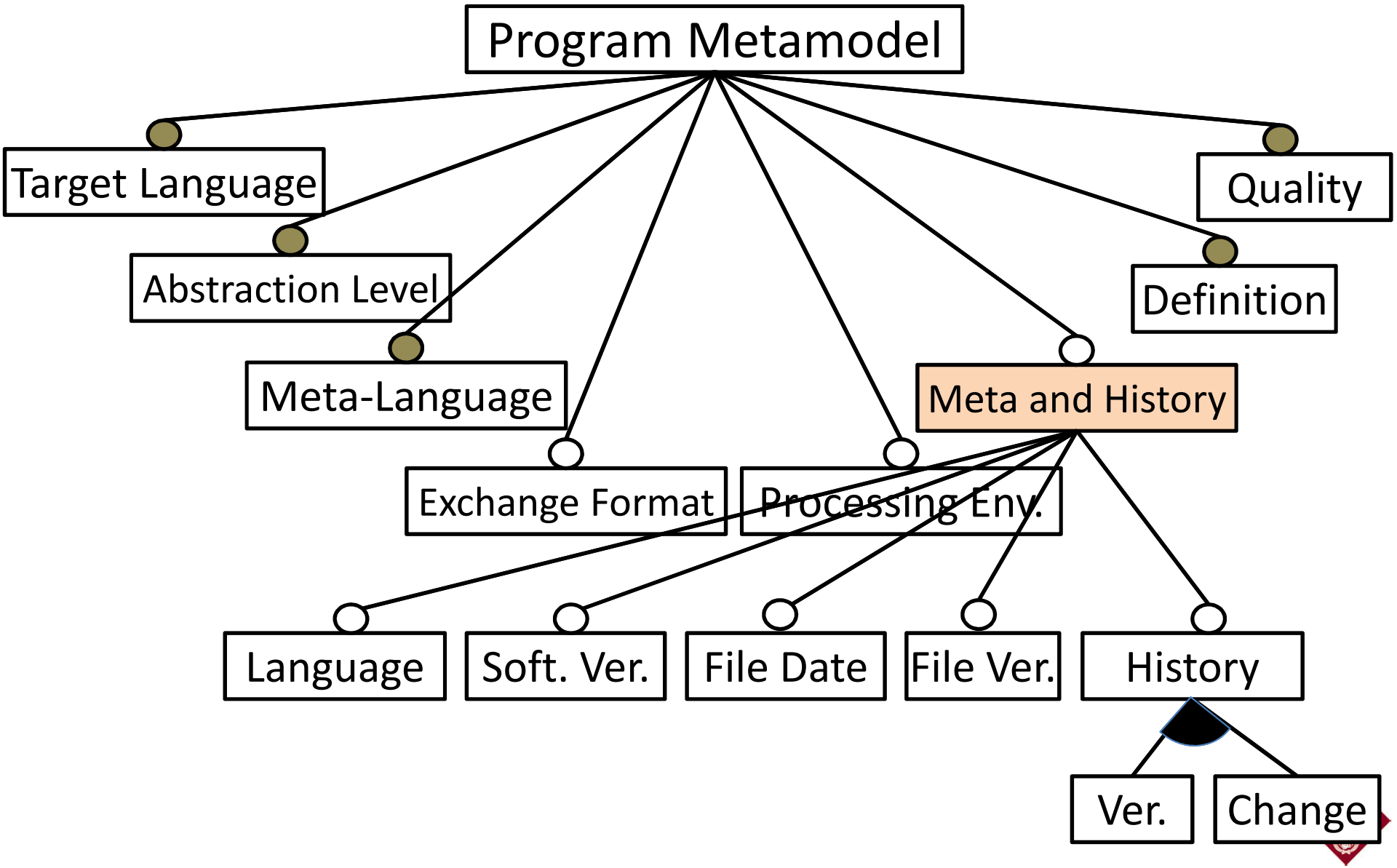
(2-3) ProMeTA – Exchange Format



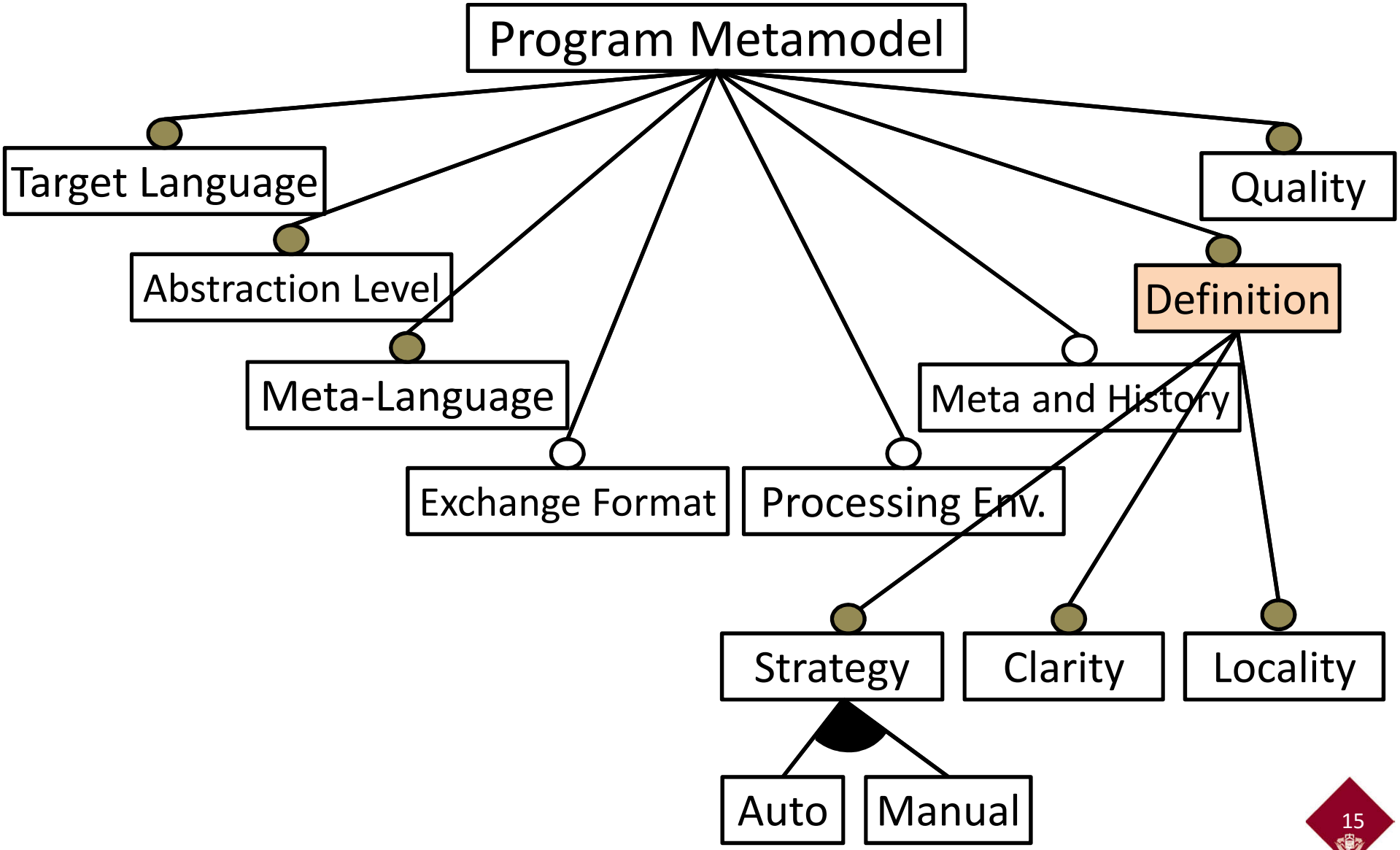
(2-3) ProMeTA – Processing Environment



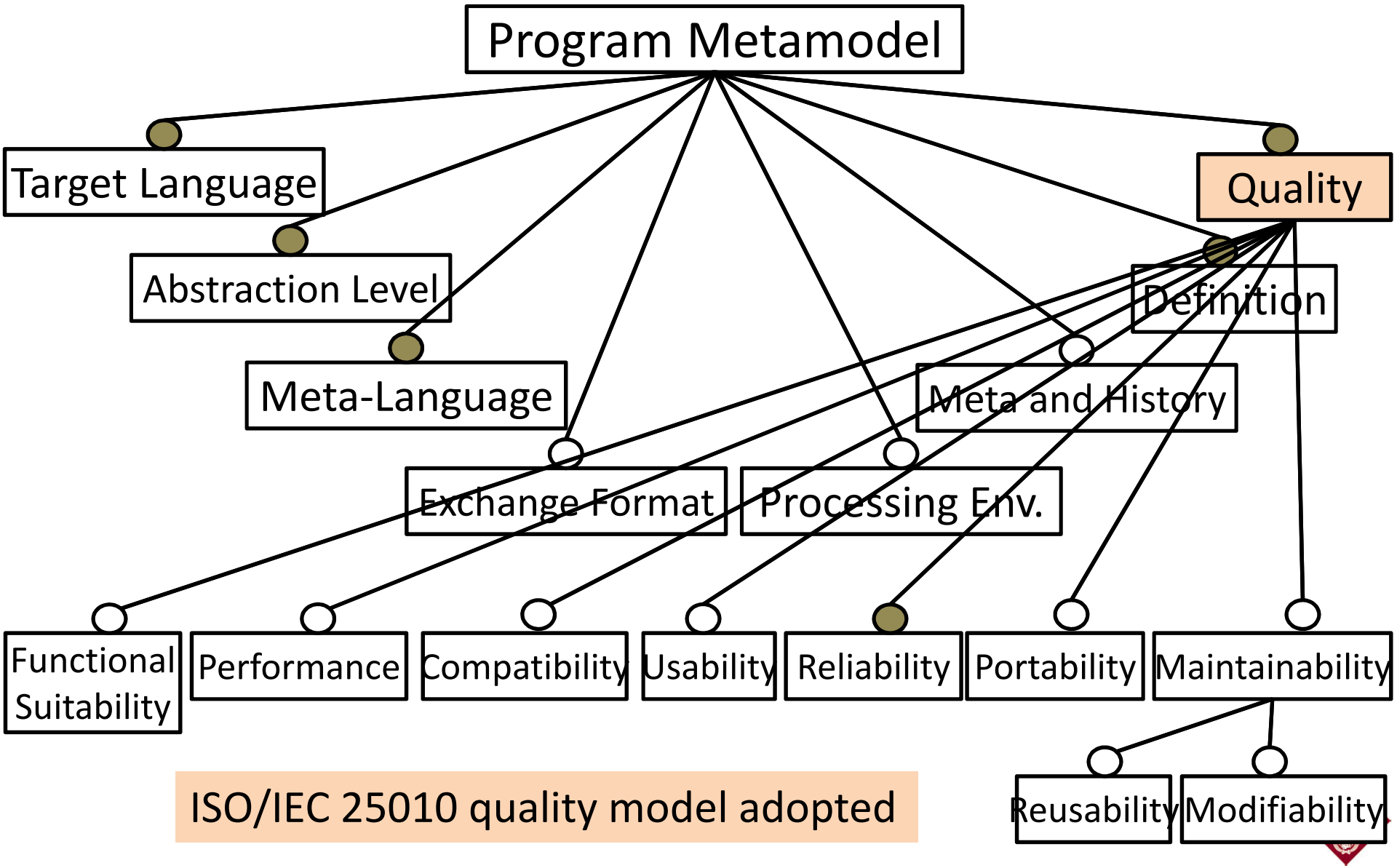
(2-3) ProMeTA – Program Meta and History



(2-3) ProMeTA – Definition



(2-3) ProMeTA – Quality



ISO/IEC 25010 quality model adopted

M	Target Language		High		Middle						Lexical Structure					Syntax				Semantics		Dialects
	T1	T2	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
M1	Independent	Java, Delphi			X	X	X	X	X					X	X	X	X	X	X	X	X	
M2	Independent	Java, PL/SQL	X	X	X	X		X	X	X				X	X	X	X		X	X	X	
M3	Object-Oriented	Java, C++, Ada, Smalltalk			X	X		X	X							X	X					
M4	Object-Oriented	Java, C++			X	X			X							X	X		X			
M5	Independent	Java, C++, C			X	X		X	X							X	X			X	X	

M	Meta-Language				Exchange Format																				
	L1	L2	L3	L4	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	F21	F22
M1	MOF				Text	File Transfer	XMI, XSD			Exp	Ext		+	+	++	++	++	++	+	+	++	+	+		Exp	Ext
M2	MOF				Text	File Transfer	XMI			Exp	Ext		+	+	++	++	++	++	+	+	++	+	+		Exp	Ext
M3	UML				Text	Text Stream	XMI, CDIF			Exp	Ext		+	+	++	++	++	++	+	+	++	+	+		Exp	Ext
M4	UML				Text	File Transfer	XMI			Exp	Ext		+	+	++	++	++	++			++	+	+		Exp	Ext
M5				X	Binary	Direct			RDB	Imp	Int		−	−	−	−	−	−			−	−	−		Imp	Int

M	Processing Environment							
	P1	P2	P3	P4	P5	P6	P7	P8	P9
M1		OCL, KDM Analysis Package	MoDisco (dedicated parsers), Gra2MoL	KDM Target Mapping & Transformation Package				X	
M2		OCL, Modisco Java Model Query	MoDisco (KDM Source Discovery, Java Discoverer)		ADM tools			X	X
M3		MOOSE Navigation and Querying Engine	MOOSE	MOOSE Refactoring Engine				X	X
M4	X		Datrix				X	X	
M5		SQL	SPOOL (dedicated extractors)						X

M	Definition			Program Meta and History Data							Functionality						
	D1	D2	D3	H1	H2	H3	H4	H5	H6	H7	Q1	Q2	Q3	Q4	Q5	Q6	Q7
M1	Manually	Exp	Ext	X							+	Embedded	Manual	+		+	
M2	Manually	Exp	Ext						X		+	Embedded	Manual	+		+	
M3	Manually	Exp	Ext													+	
M4	Manually	Exp	Int											+		+	
M5	Manually	Imp	Int														Dependency analysis

M	Non-Functionality																	
	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25
M1	−	Doc, Sample, Community	++		++	+	+	Free	Fully		Inheritance, Composition	+	++	Fully	++	+	+
M2		Doc, Sample, Community	++		++	+	+	Free	Fully	Package	Inheritance, Composition	+	++	Fully	++	++	+
M3		Doc, Sample, Community	++		++	+		Free	Fully		Inheritance, Composition	+	++	Fully	++	++	+
M4			++		−	−		Free	Unavailable		Inheritance, Composition	+	−	Partially	+		+
M5			−		−	−		Free	Unavailable			−	−	Partially	−		−

(4) Classification Results and Findings

- Metamodels can be reused for major languages (Java, C++)
- Better to choose/create metamodels defined by explicitly-externally defined major metalanguages/exchange formats
- Most are suitable for transformations and program analysis
- Few supports to describe meta and history data

	Lang	Abst	Meta	Exch	Env.	Hist	Defi	Func	Qual
ASTM	any	M L	MOF	XMI	OCL, MoDisco	Lang	Ext	General	++
KDM	any	H M L	MOF	XMI	OCL, MoDisco	Ver.	Ext	General	++
FAMIX	OOP	M	UML	MSE	MOOSE		Ext	General	++
SPOOL	OOP	M	UML	XMI	Datrix		Int	General	+
UNIQ	Any	M L	EBNF	RDB	SPOOL, SQL		Int	Dependency	

Abstract Syntax Tree Metamodel (ASTM), Knowledge Discovery Meta-Model (KDM)

FAMOOS Information Exchange Model (FAMIX)

Related Work and Conclusion

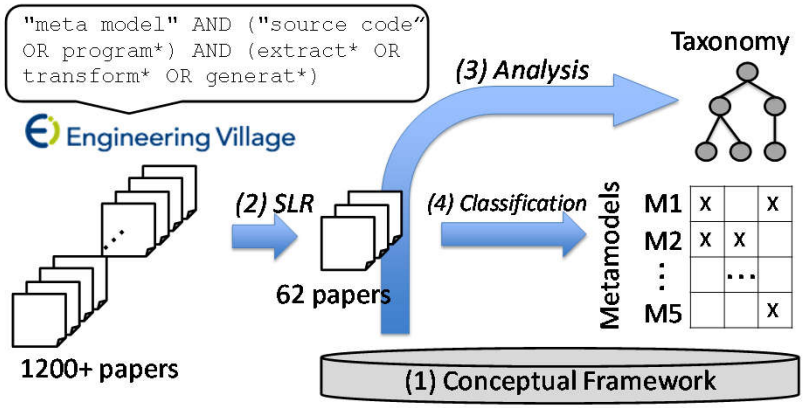
- Existing comparisons and evaluations (e.g., [Jin06][Izq14]) were conducted independently
  - Do not provide a comprehensive guide of characteristics and limitations of metamodels.
- Contribution
  - A conceptual framework
  - A comprehensive taxonomy, named ProMeTA
  - A classification of existing popular program metamodels
- Future work
  - Validate ProMeTA by conducting experiments
  - Make ProMeTA available and modifiable to the community

[Jin06] D. Jin and J. R. Cordy, "Integrating reverse engineering tools using a service-sharing methodology," in 14th IEEE International Conference on Program Comprehension (ICPC'06). IEEE Computer Society, 2006, pp. 94–99.
[Izq14] J. L. C. Izquierdo and J. G. Molina, "Extracting models from source code in software modernization," Software and Systems Modeling, vol. 13, no. 2, pp. 713–734, 2014.

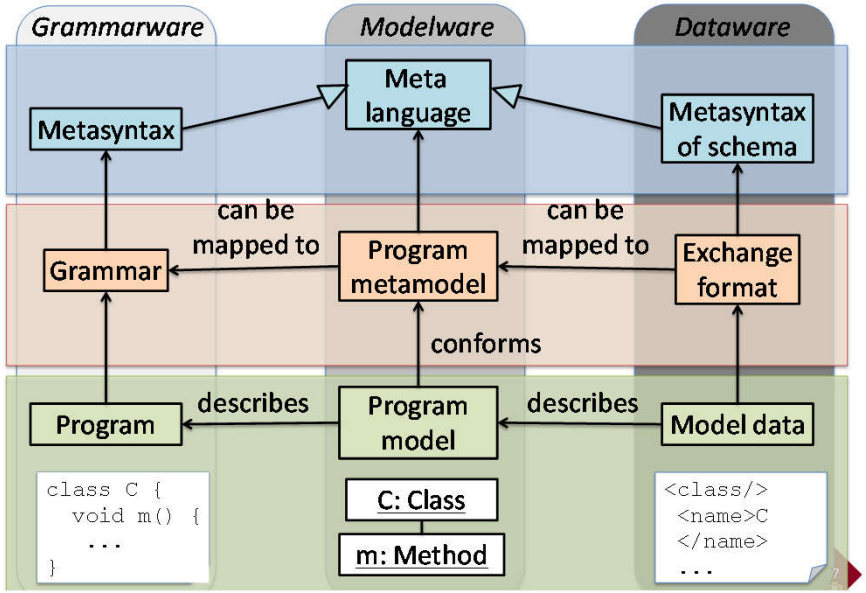
Thanks! Questions?

Research Goal and Method

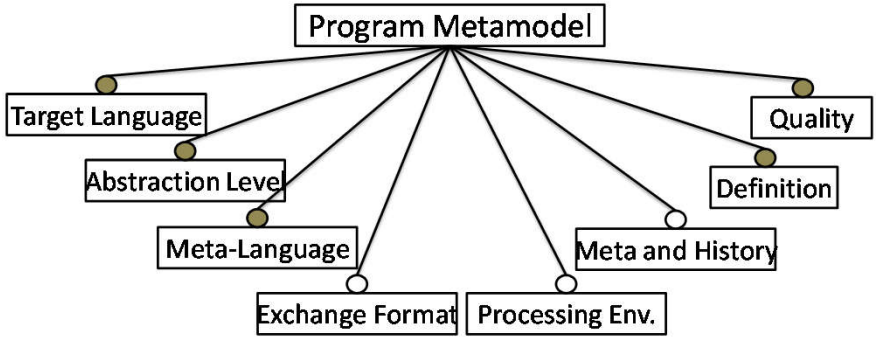
To provide a comprehensive taxonomy and use this taxonomy to classify some popular metamodels



(1) Conceptual framework



(2-3) ProMeTA: Program Metamodel Taxonomy



(4) Classification Results and Findings

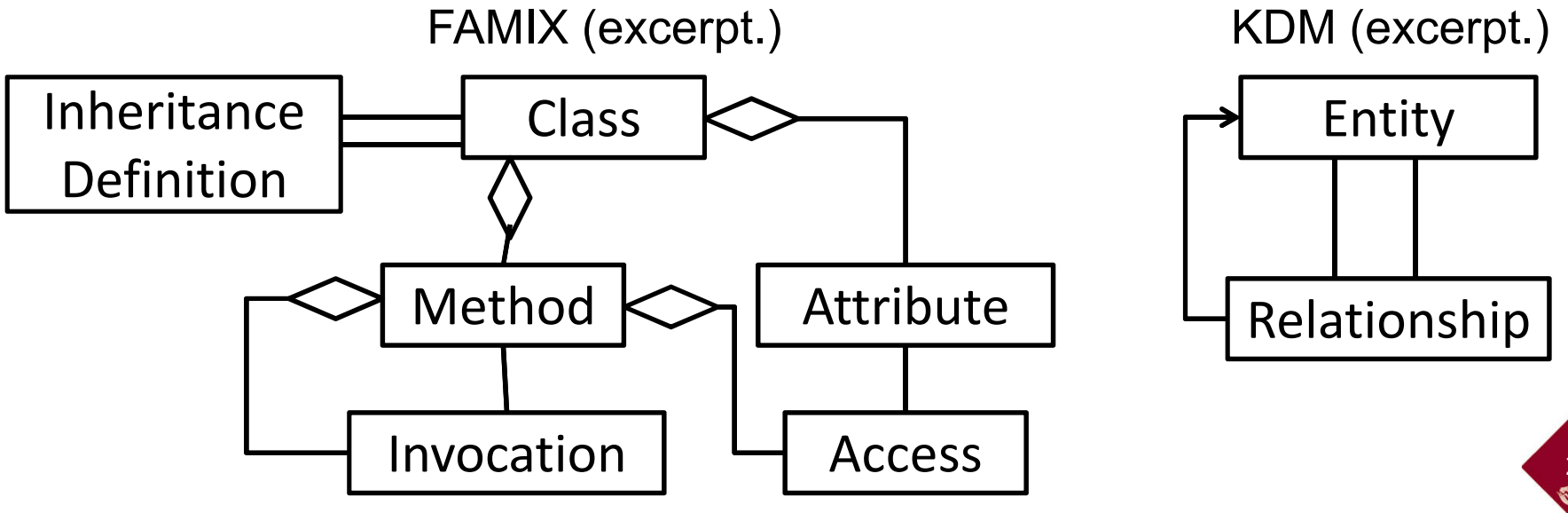
- If the target language is a major one like Java or C++, existing metamodels and tools may be reused.
- Better to choose/create metamodels defined by widely accepted, explicitly-externally defined metalanguages/formats
- Most are suitable for transformations and program analysis.
- Few supports to describe meta and history data

	Lang	Abst	Meta	Exch	Env.	Hist	Defi	Func	Qual
ASTM	any	M L	MOF	XMI	OCL, MoDisco	Lang	Ext	General	++
KDM	any	H M L	MOF	XMI	OCL, MoDisco	Ver.	Ext	General	++
FAMIX	OOP	M	UML	MSE	MOOSE		Ext	General	++
SPOOL	OOP	M	UML	XMI	Datrix		Int	General	+
UNIQ	Any	M L	EBNF	RDB	SPOOL, SQL		Int	Dependency	

Abstract Syntax Tree Metamodel (ASTM), Knowledge Discovery Meta-Model (KDM), FAMOOS Information Exchange Model (FAMIX), SPOOL, UNIQ-ART

What are metamodels?

- Reverse engineering: analysis process to identify elements and create target's representations in another or at a higher level of abstraction
- Program metamodel: a model of a programming language grammar, which represents target programs according to a specific purpose



Key Findings

Need to revise

- Target language: If the target is a major one like Java or C++, existing metamodels and tools may be reused.
- Abstraction level: None of the existing metamodels supports all of the required features at certain abstraction levels.
- Metalanguage: Better to choose or create metamodels defined by widely accepted, explicitly-externally defined metalanguages like MOF and UML, for long-term usage.
- Exchange format: Better to choose of create metamodels which support the widely accepted, explicitly-externally defined SEFs like XMI, for long-term usage.
- Processing environment: Most of the metamodels are suitable for transformations and program analysis.
- Definition: Better to select or create explicitly-externally defined metamodels, for long-term usage.
- Program meta and history data: There are few supports to describe meta and history data in metamodels.
- Functionality: Better to select a general metamodel for various reverse engineering purposes.
- Non-functionality: Should select fully available and formalized metamodels.