Domain-Specific Languages:

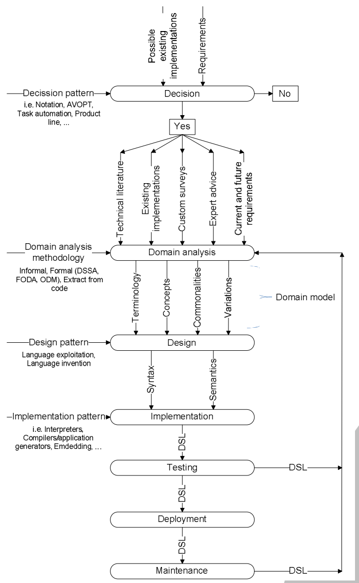
A Systematic Mapping Study

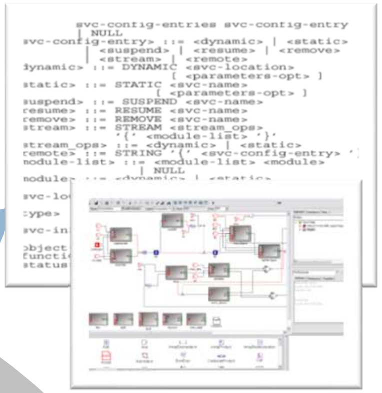
Marjan Mernik

University of Maribor, Slovenia

FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS, FedCSIS
Prague, Czech Republic, 3 - 6 September, 2017

Overview


Possible existing implementations
Requirements
Decision pattern
i.e. Notation, AVOPT, Task automation, Product line, ...
Decision
No
Yes
Technical literature
Existing implementations
Custom surveys
Expert advice
Current and future requirements
Domain analysis methodology
Informal, Formal (DSSA, FODA, ODM), Extract from code
Domain analysis
Terminology
Concepts
Commonalities
Variations
Domain model
Design pattern
Language exploitation, Language invention
Design
Syntax
Semantics
Implementation pattern
i.e. Interpreters, Compilers/application generators, Emdedding, ...
Implementation
DSL
Testing
DSL
Deployment
DSL
Maintenance
DSL



```
        svc-config-entries svc-config-entry
         | NULL
svc-config-entry> ::= <dynamic> | <static>
          | <suspend> | <resume> | <remove>
          | <stream> | <remote>
dynamic> ::= DYNAMIC <svc-location>
                    [ <parameters-opt> ]
static> ::= STATIC <svc-name>
                  [ <parameters-opt> ]
suspend> ::= SUSPEND <svc-name>
resume> ::= RESUME <svc-name>
remove> ::= REMOVE <svc-name>
stream> ::= STREAM <stream_ops>
            '{' <module-list> '}'
stream_ops> ::= <dynamic> | <static>
remote> ::= STRING '{' <svc-config-entry> '
nodule-list> ::= <module-list> <module>
                | NULL
nodule> ::= <dynamic> | <static>

svc-lo

ype>

svc-in

object
functi
status
```

DSLs: Definitions and Examples

DSLs: When & How

DSLs: Systematic Mapping Study

DSLs: Open Problems

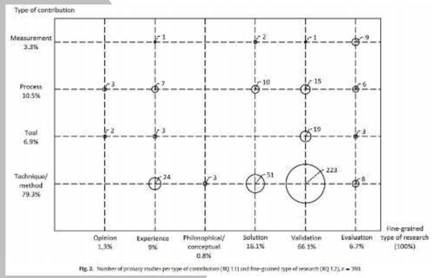
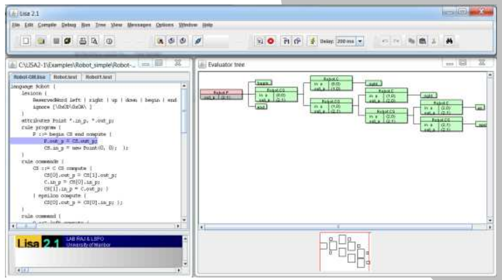
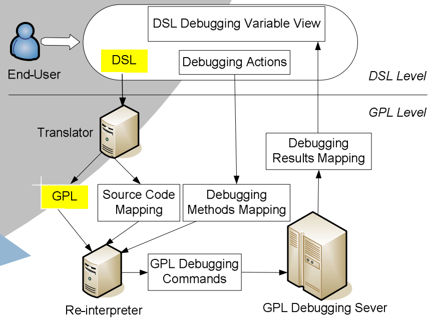


Fig. 2. Number of primary studies per type of contribution (RQ 1.1) and fine-grained type of research (RQ 1.2), n = 390.



DSL Composition

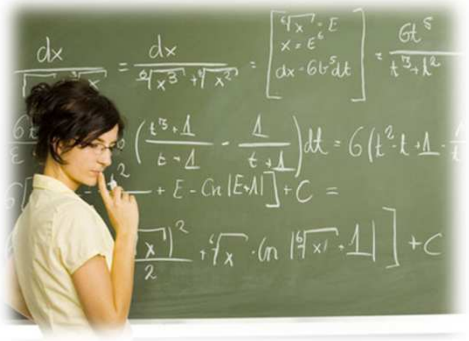


DSL Debugging

DSLs

- Can we envisage how software development will be carried-out in the future?
- Is it possible that end-users will be describing solutions to their problem tasks on their own?
- Appearance of end-user development (EUD) is on horizon.
- The central point of EUD is to make PL easier to adopt and use.
- DSLs have great potential for fulfilling these end-users‘ needs.
- Although that vision for EUD in the future might not be fulfill as predicted, the future for DSLs is bright. This is because nowadays even professional programmers are using DSLs by identifying the benefits of DSLs such as increased flexibility, productivity, reliability, and usability.

Domain-specific languages are increasingly POPULAR



DSLs: Definitions

- “A DSL can be viewed as a **programming language** dedicated to a particular domain or problem. It provides appropriate built-in abstractions and notations; it is usually small, more declarative than imperative, and less expressive than GPL.” – C. Consel

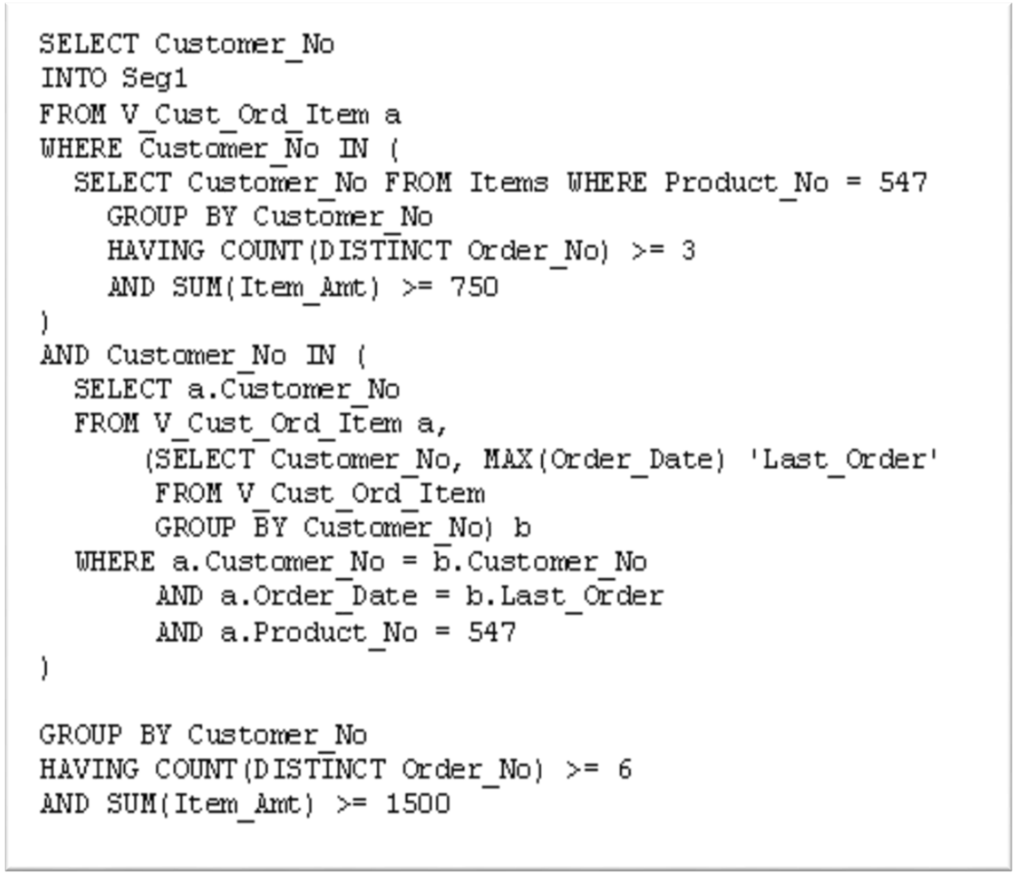
- “A DSL is a **programming language** or **executable specification language** that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain.” – A. van Deursen, P. Klint, J. Visser

DSLs: Definitions

- “DSLs are designed for convenience in a narrow range of applications. They are generally assumed to be used primarily for very small programs and, in many cases, are intended for use by non-programmers (hence, "**end-user languages**"). In most cases, efficiency is not a major concern, and user convenience and conciseness is paramount.” – S. Kamin

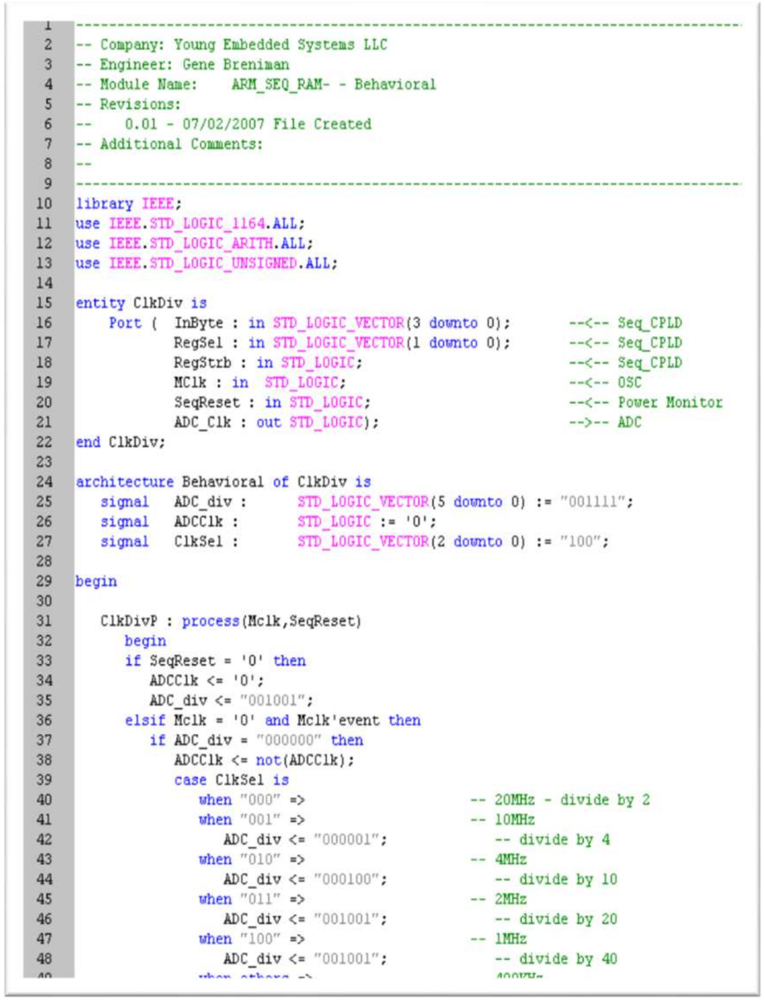
- “DSLs are **computer languages** tailored to a specific application domain. They offer substantial gains in expressiveness and ease of use compared with GPLs in their domain of application.” – M. Mernik, J. Heering, A. Sloane

DSL: Examples



```sql
SELECT Customer_No
INTO Seg1
FROM V_Cust_Ord_Item a
WHERE Customer_No IN (
  SELECT Customer_No FROM Items WHERE Product_No = 547
    GROUP BY Customer_No
    HAVING COUNT(DISTINCT Order_No) >= 3
    AND SUM(Item_Amt) >= 750
)
AND Customer_No IN (
  SELECT a.Customer_No
  FROM V_Cust_Ord_Item a,
       (SELECT Customer_No, MAX(Order_Date) 'Last_Order'
       FROM V_Cust_Ord_Item
       GROUP BY Customer_No) b
  WHERE a.Customer_No = b.Customer_No
       AND a.Order_Date = b.Last_Order
       AND a.Product_No = 547
)

GROUP BY Customer_No
HAVING COUNT(DISTINCT Order_No) >= 6
AND SUM(Item_Amt) >= 1500
```

Structured Query Language (SQL)

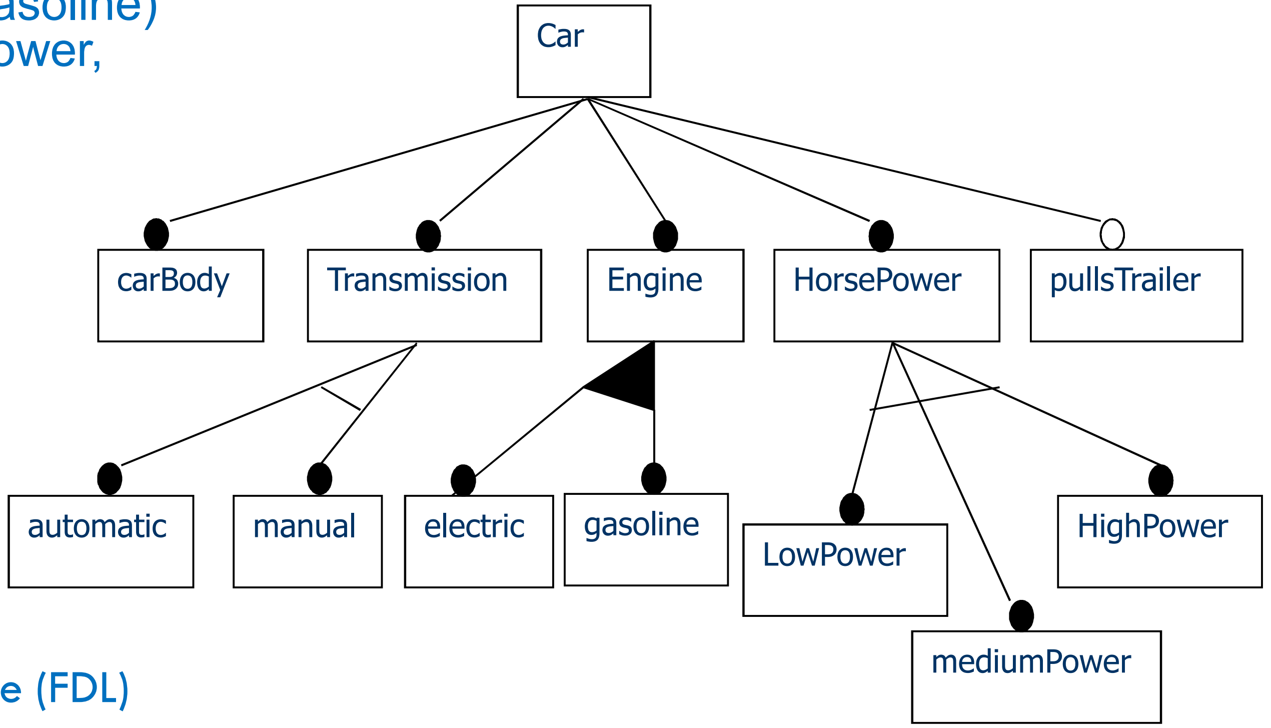


```vhdl
-------------------------------------------------------------------------------
-- Company: Young Embedded Systems LLC
-- Engineer: Gene Breniman
-- Module Name:    ARM_SEQ_RAM- - Behavioral
-- Revisions:
--     0.01 - 07/02/2007 File Created
-- Additional Comments:
--
-------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClkDiv is
    Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);        --<-- Seq_CPLD
            RegSel : in STD_LOGIC_VECTOR(1 downto 0);        --<-- Seq_CPLD
            RegStrb : in STD_LOGIC;                          --<-- Seq_CPLD
            MClk : in  STD_LOGIC;                            --<-- OSC
            SeqReset : in STD_LOGIC;                         --<-- Power Monitor
            ADC_Clk : out STD_LOGIC);                        -->-- ADC
end ClkDiv;

architecture Behavioral of ClkDiv is
   signal   ADC_div :      STD_LOGIC_VECTOR(5 downto 0) := "001111";
   signal   ADCClk :       STD_LOGIC := '0';
   signal   ClkSel :       STD_LOGIC_VECTOR(2 downto 0) := "100";

begin

   ClkDivP : process(Mclk,SeqReset)
      begin
      if SeqReset = '0' then
         ADCClk <= '0';
         ADC_div <= "001001";
      elsif Mclk = '0' and Mclk'event then
         if ADC_div = "000000" then
            ADCClk <= not(ADCClk);
            case ClkSel is
               when "000" =>                           -- 20MHz - divide by 2
               when "001" =>                           -- 10MHz
                  ADC_div <= "000001";                    -- divide by 4
               when "010" =>                           -- 4MHz
                  ADC_div <= "000100";                    -- divide by 10
               when "011" =>                           -- 2MHz
                  ADC_div <= "001001";                    -- divide by 20
               when "100" =>                           -- 1MHz
                  ADC_div <= "001001";                    -- divide by 40
```

VHSIC hardware description language (VHDL)

DSL: Examples

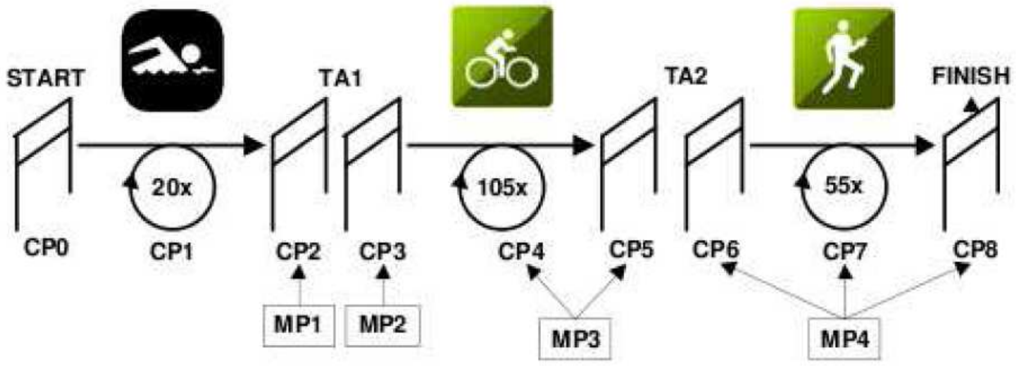
Car: all(carBody, Transmission, Engine, HorsePower, pullsTrailer?)
Transmission: one-of(automatic, manual)
Engine: more-of(electric, gasoline)
HorserPower: one-of(lowPower, mediumPower, highPower)

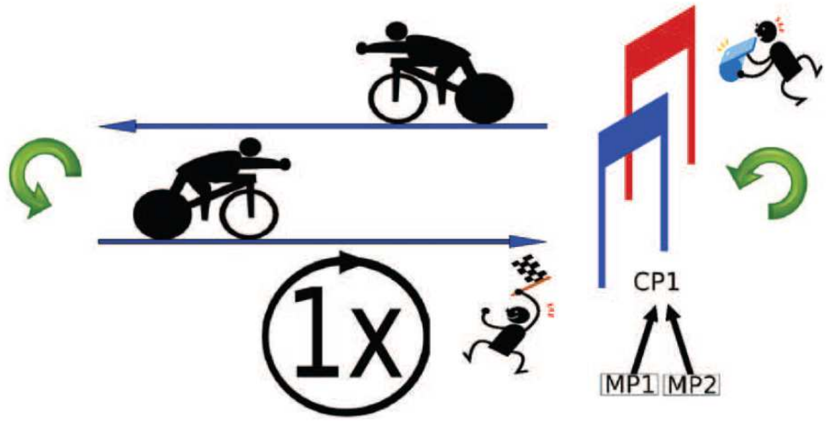


Feature Description Language (FDL)

DSL: Examples

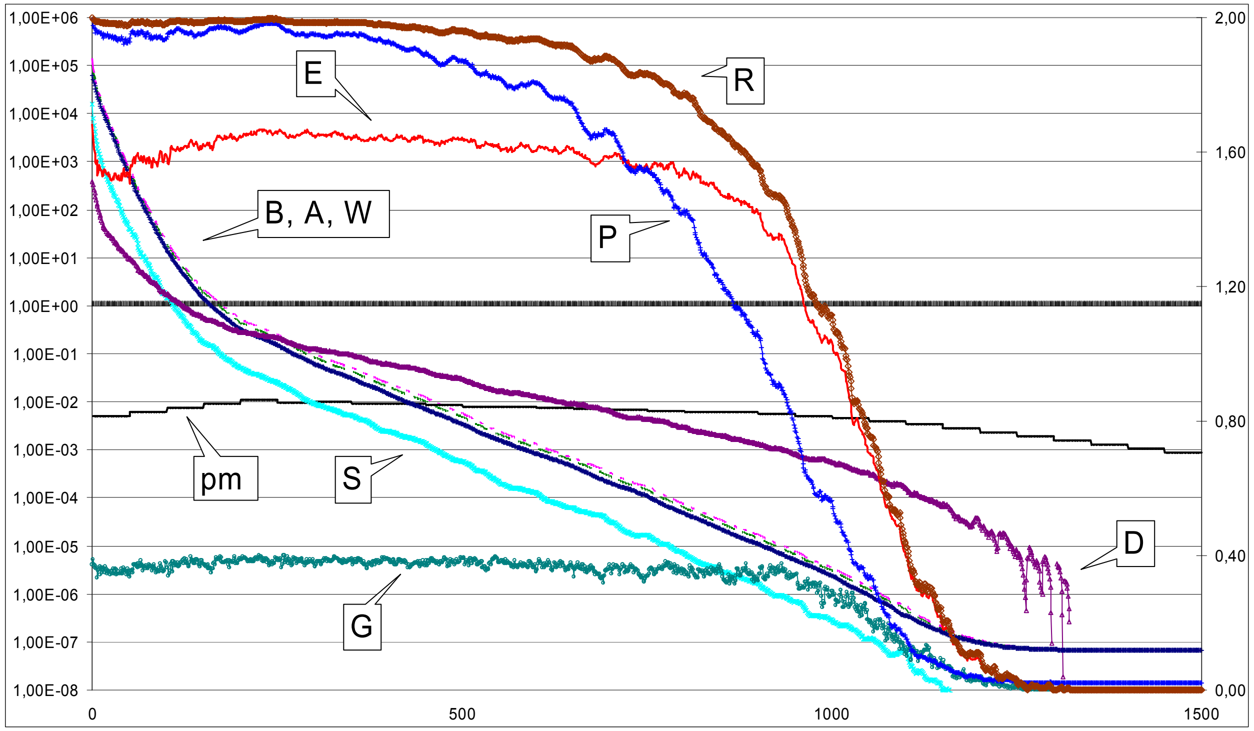
```
1 manual "abc.res";
2 auto 192.168.225.100;

var ROUND1 := 20;
var INTER1 := 0;
var SWIM := 0;
var TRANS1 := 0;
var ROUND2 := 105;
var INTER2 := 0;
var BIKE := 0;
var TRANS2 := 0;
var ROUND3 := 55;
var INTER3 := 0;
var RUN := 0;

mp[1]→agnt[1]{
   (true)→upd SWIM;
   (true)→dec ROUND1;
}
mp[2]→agnt[1]{
   (true)→upd TRANS1;
}
mp[3]→agnt[2]{
   (true)→upd INTER2;
   (true)→dec ROUND2;
   (ROUND2 = = 0)→upd BIKE;
}
```



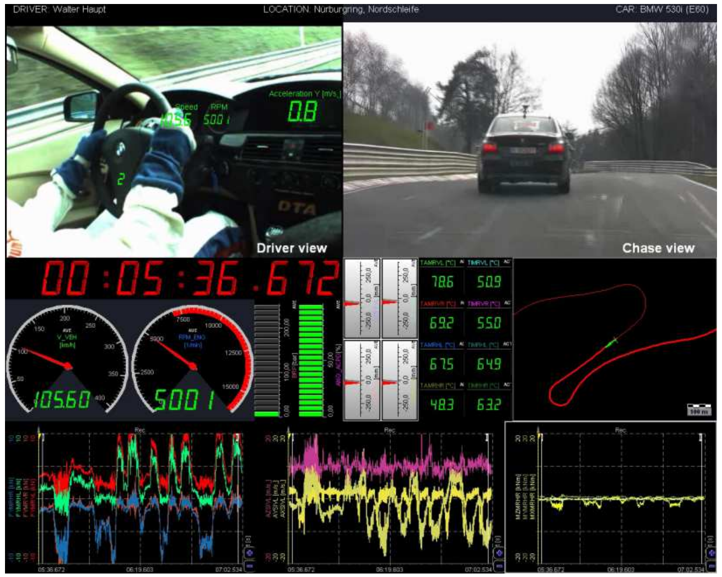


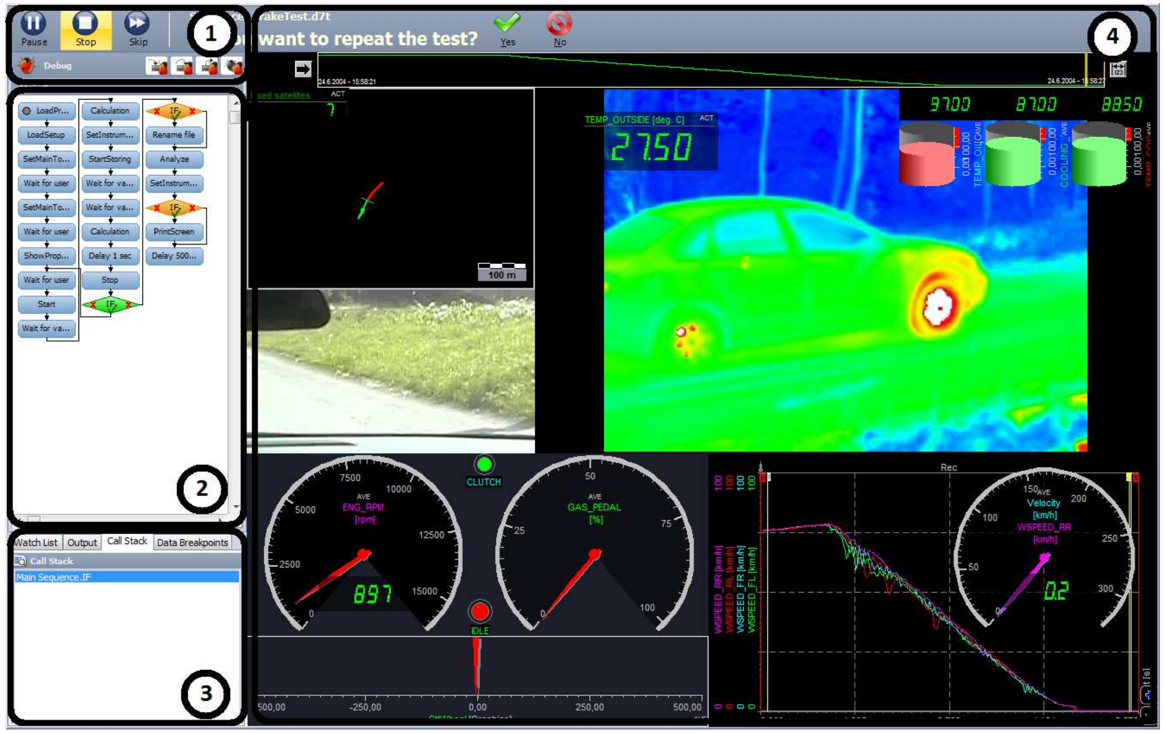
DSL Examples

```
genetic  // 1/5 success rule
  Round := 50;
  g := 0;
  Epoch := 50;
  Maxgen := 1500;
  while ( g < Round ) do
    t := 0;
    pm := 0.005;
    pc := 0.75;
    init;
    while ( t < Maxgen ) do
      callGA;
      if ( RatioM > 0.2 ) then
        pm := pm * 1.22
      fi;
      if ( RatioM < 0.2 ) then
        pm := pm * 0.82
      fi;
      t := t + Epoch
    end;
    g := g + 1
  end
end genetic
```



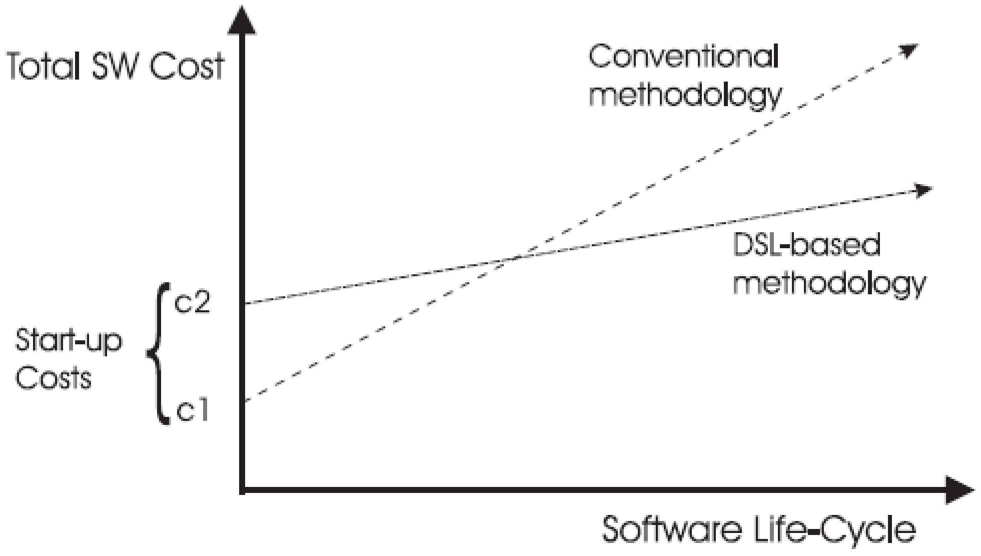
DSL: Examples





DSLs: Costs

- However, the benefits of DSLs are not for free!
- Since the costs of DSL development and maintenance have to be taken into account, one of the main question is "**When and how to develop a DSL?**"

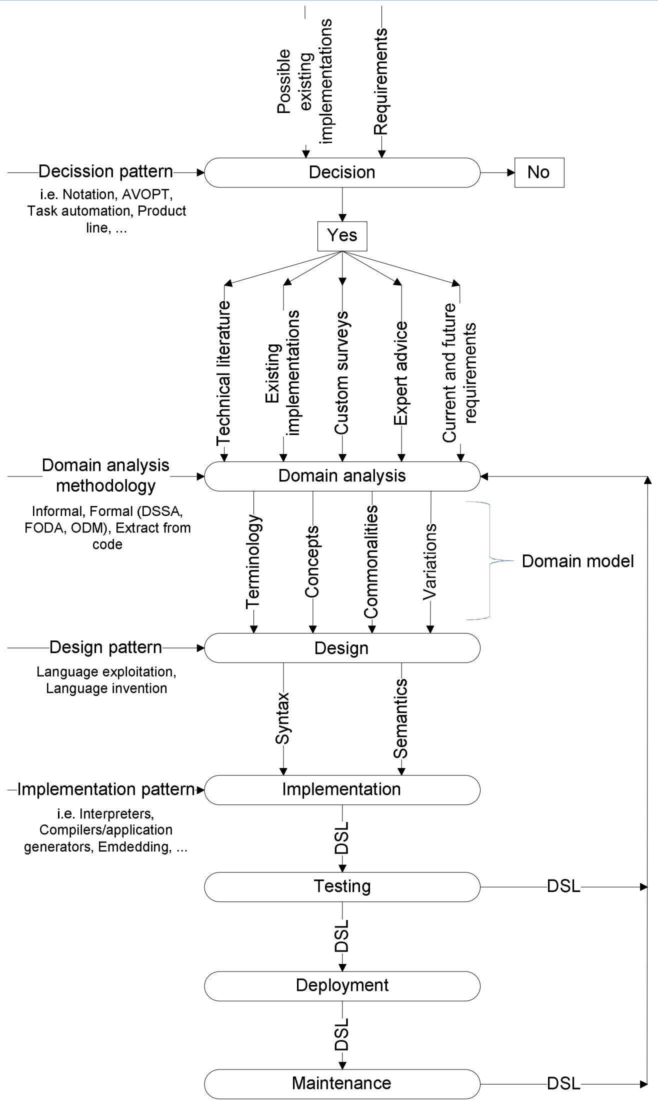


The Payoff of DSL technology
P. Hudak. Modular Domain-Specific Languages and Tools. ICSR'98

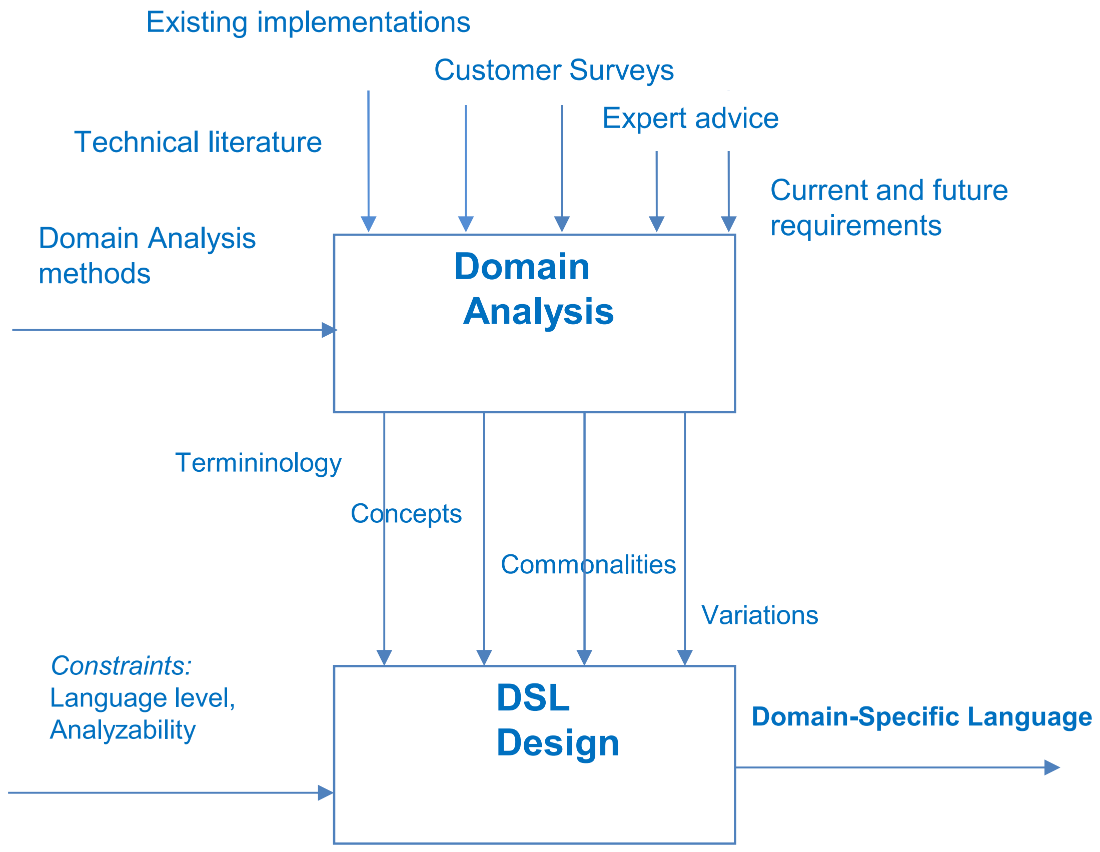
DSLs: When?

- Programmer's productivity and reliability have to be increased
- End-user programming has to be enabled,
- Domain-specific analysis, verifications, optimizations, or transformations are required,
- Problem/program families or software generators are involved.
- …

DSLs: How?


Possible existing implementations
Requirements
Decission pattern
i.e. Notation, AVOPT, Task automation, Product line, ...
Decision
No
Yes
Technical literature
Existing implementations
Custom surveys
Expert advice
Current and future requirements
Domain analysis methodology
Informal, Formal (DSSA, FODA, ODM), Extract from code
Domain analysis
Terminology
Concepts
Commonalities
Variations
Domain model
Design pattern
Language exploitation, Language invention
Design
Syntax
Semantics
Implementation pattern
i.e. Interpreters, Compilers/application generators, Emdedding, ...
Implementation
DSL
Testing
DSL
DSL
Deployment
DSL
Maintenance
DSL

DSLs: Analysis & Design



DSLs: Analysis & Design

- From domain analysis many possible languages can be developed, but all share important information found in feature diagram.
- Variation points are expressed differently.

- Which one is better?

DSLs: Analysis & Design

- From previous experiences with programming language design and criteria for PL evaluation, researchers developed criteria for good language design:
  - Readability (how easy is to read and understand programs)
  - Writeability (how easy is to write programs)
  - Reliability
  - Cost
- Some language design advices:
  - Keep design committee small
  - Simplicity is really the key - avoid complexity. Too many "solutions" make language hard to understand. (KISS principle)
  - Automate mechanical, tedious, or error-prone activities (provide higher level features).
  - Regular rules, without exceptions, are easier to learn, use, describe, and implement.

DSLs: Analysis & Design

- Is a DSL design very much different from GPL design?
- Many DSL researchers believe that DSL design is not very much different from GPL design.
- But, the DSL designer has to keep in mind both the special character of DSLs as well as the fact that users need not be programmers. The latter fact is extremely important and might cause the admirable principles of orthogonality and simplicity to not necessarily be well-applied to DSL design.
- GPLs have been always designed by highly professional programmers. The GPL design process was driven by their personal aesthetics and theoretical judgments resulting in a programming language they would like to use.

DSLs: Analysis & Design

- The DSL for end-users should not be designed by such intuition since DSL designers are not end-users themselves.
- On the contrary, DSL design in this case should be driven by empirical studies, involvement of end-users, or by psychology of programming research.

DSLs: Analysis & Design

- Formal methods for defining syntax and semantics of DSLs shoud be used.

- Several benefits:
  - syntax and the semantics are defined in a precise and unambiguous manner
  - proving properties of programs and the constructs
  - unique possibility for automatic generation of compilers or interpreters
  - as a tool for programming language design (programming languages which have been designed with one of the various formal methods have better syntax and semantics, less exceptions and are easier to learn).

DSLs: Analysis & Design

- Informal language designs can contain imprecisions that cause problems in the implementation phase. They typically focus on syntax, leaving semantic concerns to the imagination.

- Formal specification of both syntax and semantics can bring problems to light before implementation. Formal designs can be implemented automatically by tools, thereby significantly reducing implementation effort.

DSLs: Analysis & Design

- Programming Languages
  - Syntax - BNF is widely accepted for formal syntax description
  - Semantics - no standard method exist; there are many approaches: attribute grammars, axiomatic semantics, operational semantics, denotational semantics, algebraic semantics, action semantics, translational semantics, trace-based semantics

- Modeling Languages
  - Syntax – Metamodels
  - Semantics – not well developed yet

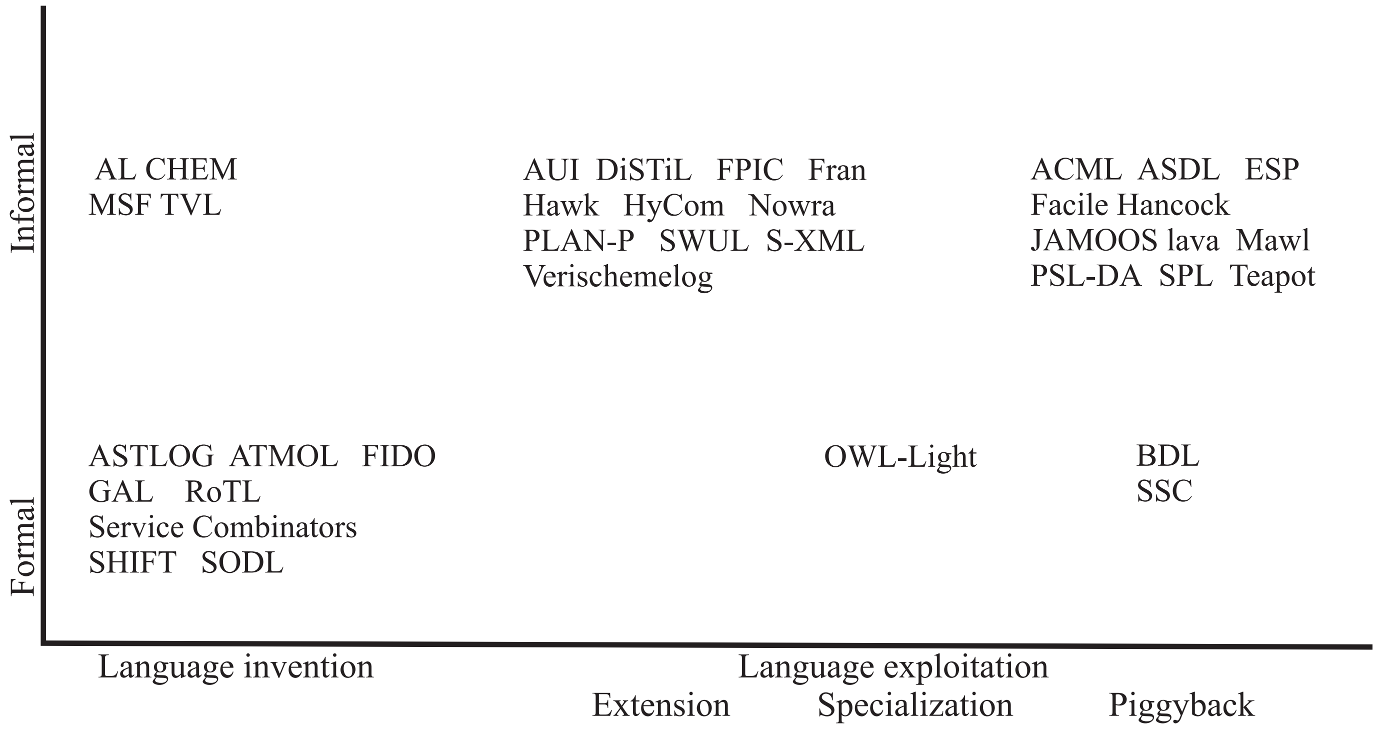
DSLs: Design Patterns

- The easiest way to design a DSL is to base it on an existing language. One possible benefit is familiarity for users, but this only applies if the domain users are also programmers in the existing language, which is often not the case.

- Existing language is partially used (piggyback)
- Existing language is restricted (specialization)
- Existing language is extended (extension)

- If no relationship between DSL and existing language exist than DSL design is very similar to GPL design with additional constraints.

DSLs: Design Patterns

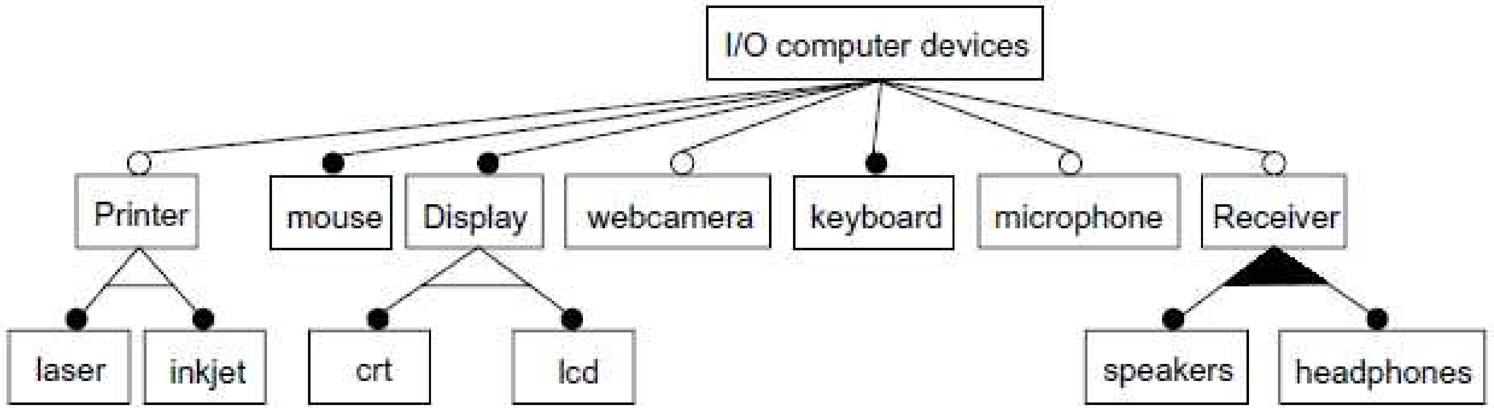


DSLs: Implementation Patterns

- The traditional approaches (API, interpreter/compiler, compiler generators, preprocessing)
- Embedding
- Extensible interpreter/compiler
- COTS (Common Off-The-Shelf) based approach
- Hybrid approach

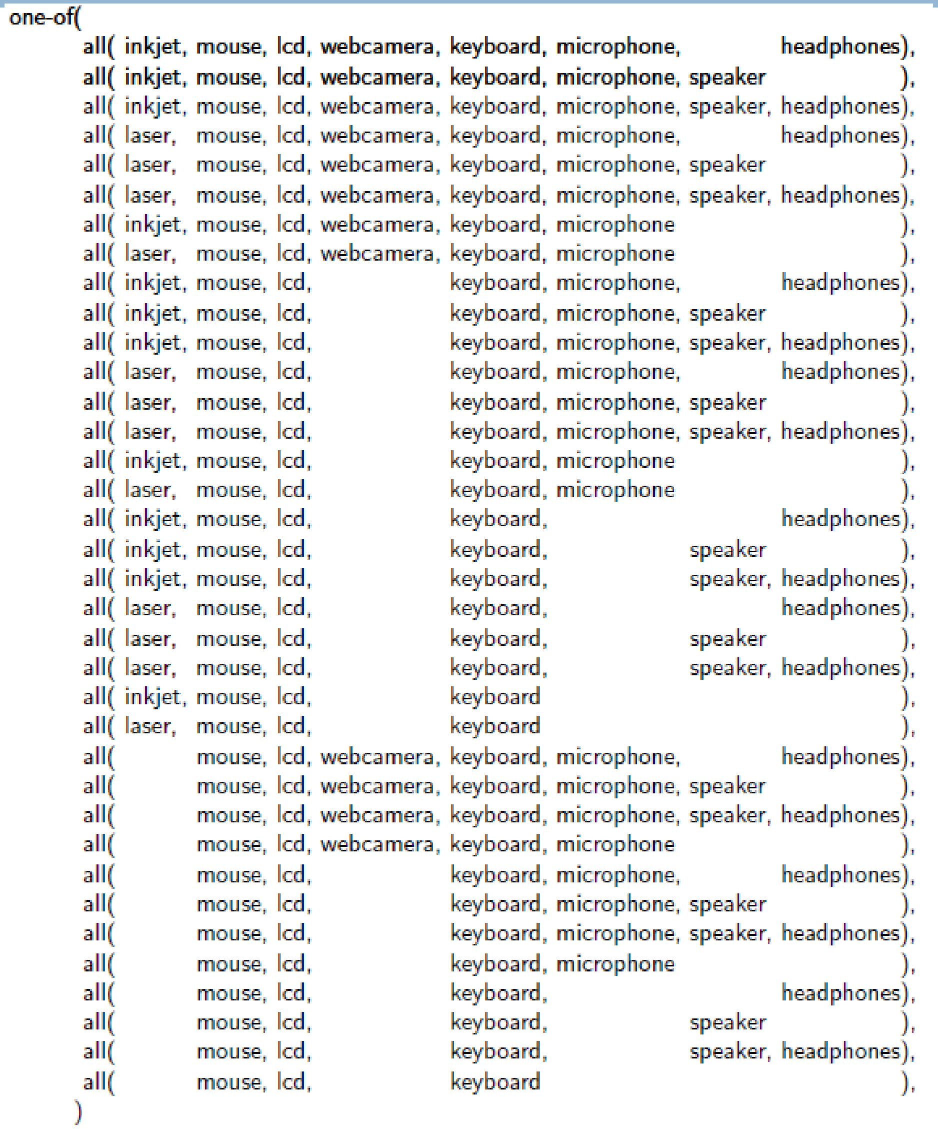
DSLs: Implementation Patterns

FDL – a DSL for describing feature diagrams



```
IOdevice : all    (opt(Printer), mouse, Display , opt(webcamera),
                   keyboard    , opt(microphone), opt(Receiver) )
Printer  : one-of (laser       , inkjet                         )
Display  : one-of (crt         , lcd                            )
Receiver : more-of(speaker     , headphones                     )
webcamera requires microphone
include   lcd
```

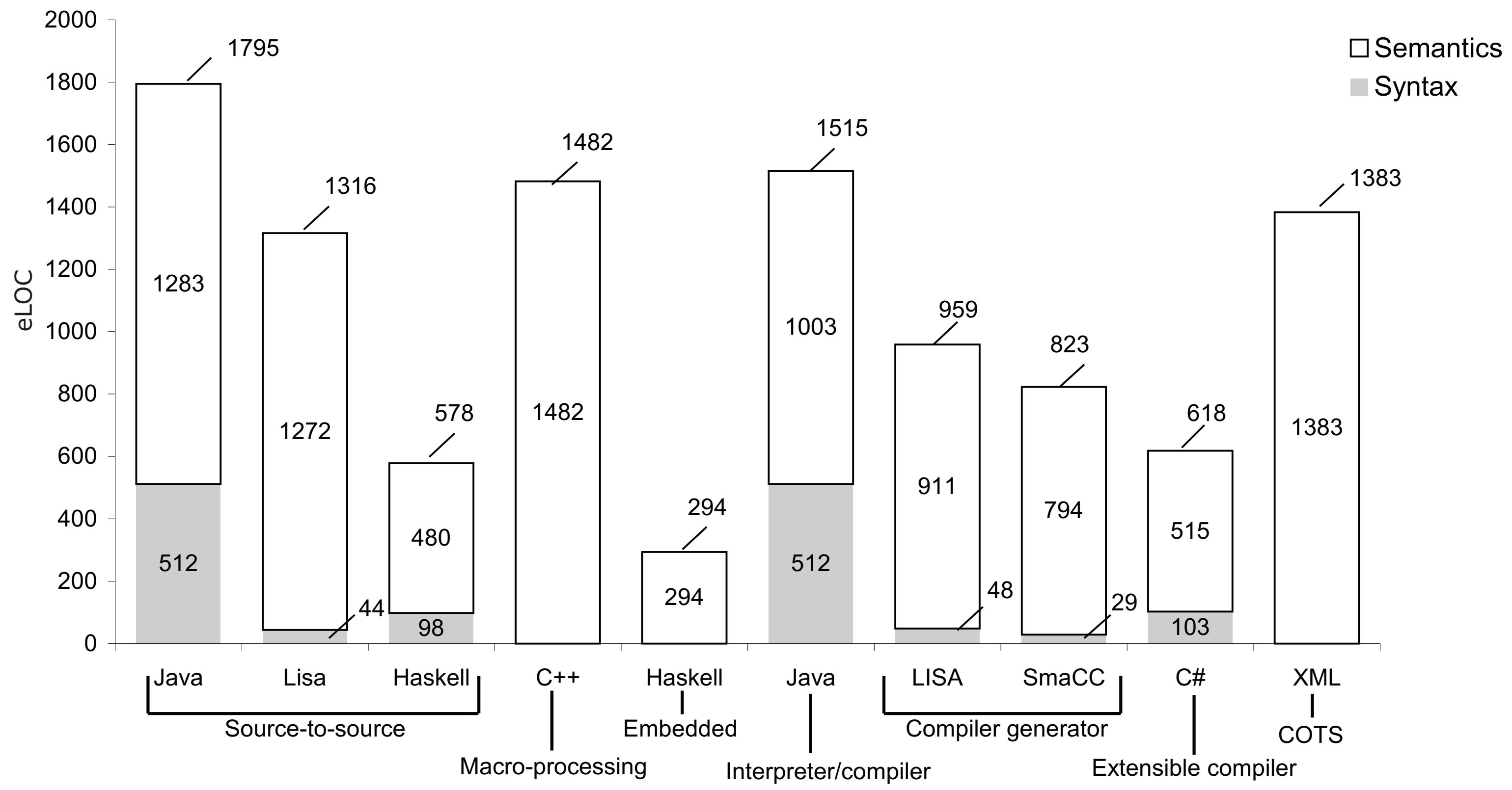
DSLs: Implementation Patterns



```
one-of(
    all( inkjet, mouse, lcd, webcamera, keyboard, microphone,          headphones),
    all( inkjet, mouse, lcd, webcamera, keyboard, microphone, speaker            ),
    all( inkjet, mouse, lcd, webcamera, keyboard, microphone, speaker, headphones),
    all( laser,  mouse, lcd, webcamera, keyboard, microphone,          headphones),
    all( laser,  mouse, lcd, webcamera, keyboard, microphone, speaker            ),
    all( laser,  mouse, lcd, webcamera, keyboard, microphone, speaker, headphones),
    all( inkjet, mouse, lcd, webcamera, keyboard, microphone                     ),
    all( laser,  mouse, lcd, webcamera, keyboard, microphone                     ),
    all( inkjet, mouse, lcd,            keyboard, microphone,          headphones),
    all( inkjet, mouse, lcd,            keyboard, microphone, speaker            ),
    all( inkjet, mouse, lcd,            keyboard, microphone, speaker, headphones),
    all( laser,  mouse, lcd,            keyboard, microphone,          headphones),
    all( laser,  mouse, lcd,            keyboard, microphone, speaker            ),
    all( laser,  mouse, lcd,            keyboard, microphone, speaker, headphones),
    all( inkjet, mouse, lcd,            keyboard, microphone                     ),
    all( laser,  mouse, lcd,            keyboard, microphone                     ),
    all( inkjet, mouse, lcd,            keyboard,                      headphones),
    all( inkjet, mouse, lcd,            keyboard,             speaker            ),
    all( inkjet, mouse, lcd,            keyboard,             speaker, headphones),
    all( laser,  mouse, lcd,            keyboard,                      headphones),
    all( laser,  mouse, lcd,            keyboard,             speaker            ),
    all( laser,  mouse, lcd,            keyboard,             speaker, headphones),
    all( inkjet, mouse, lcd,            keyboard                                 ),
    all( laser,  mouse, lcd,            keyboard                                 ),
    all(         mouse, lcd, webcamera, keyboard, microphone,          headphones),
    all(         mouse, lcd, webcamera, keyboard, microphone, speaker            ),
    all(         mouse, lcd, webcamera, keyboard, microphone, speaker, headphones),
    all(         mouse, lcd, webcamera, keyboard, microphone                     ),
    all(         mouse, lcd,            keyboard, microphone,          headphones),
    all(         mouse, lcd,            keyboard, microphone, speaker            ),
    all(         mouse, lcd,            keyboard, microphone, speaker, headphones),
    all(         mouse, lcd,            keyboard, microphone                     ),
    all(         mouse, lcd,            keyboard,                      headphones),
    all(         mouse, lcd,            keyboard,             speaker            ),
    all(         mouse, lcd,            keyboard,             speaker, headphones),
    all(         mouse, lcd,            keyboard                                 ),
)
```

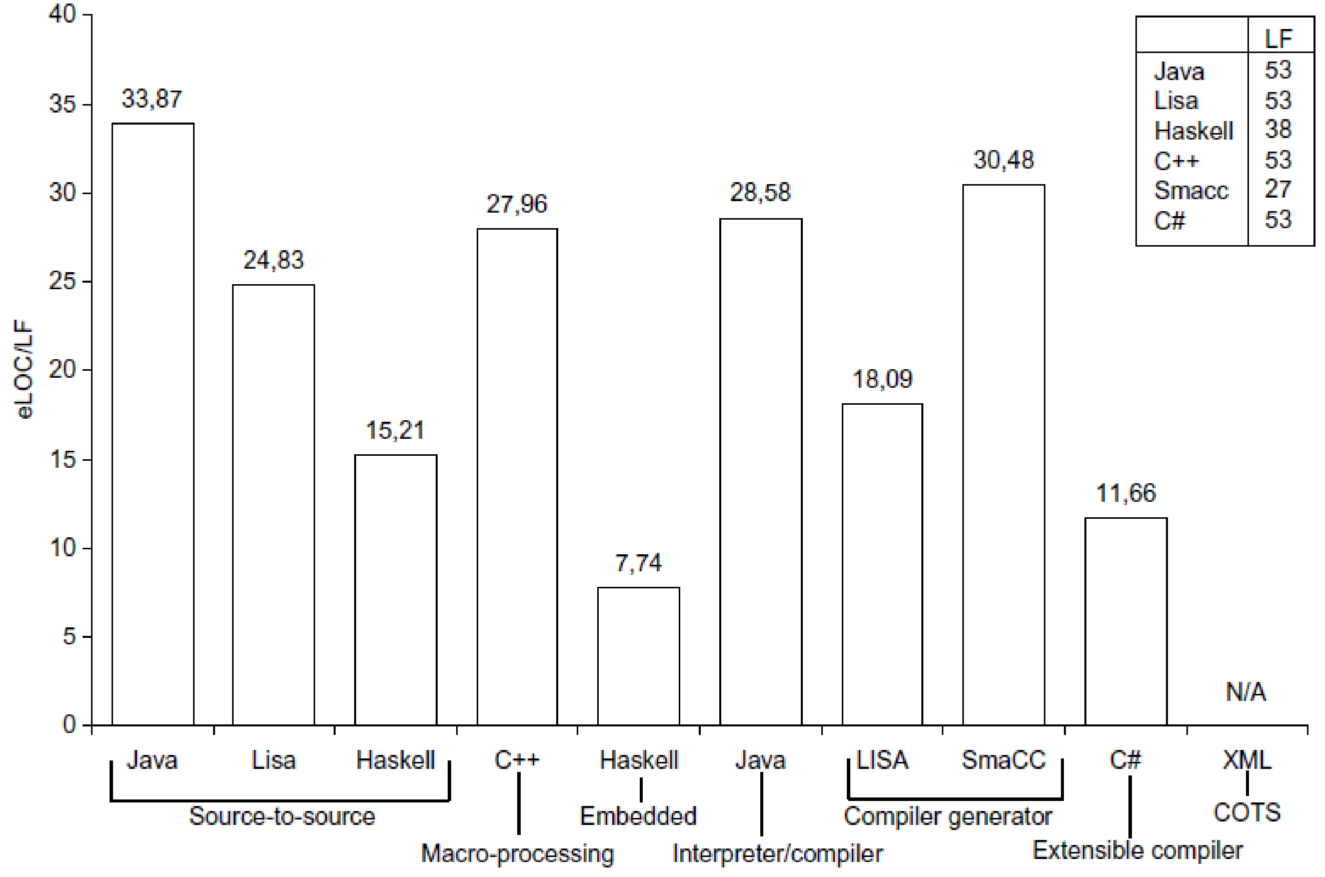
DSLs: Implementation Patterns

Implementation effort - eLOC (effective lines of code)



DSLs: Implementation Patterns

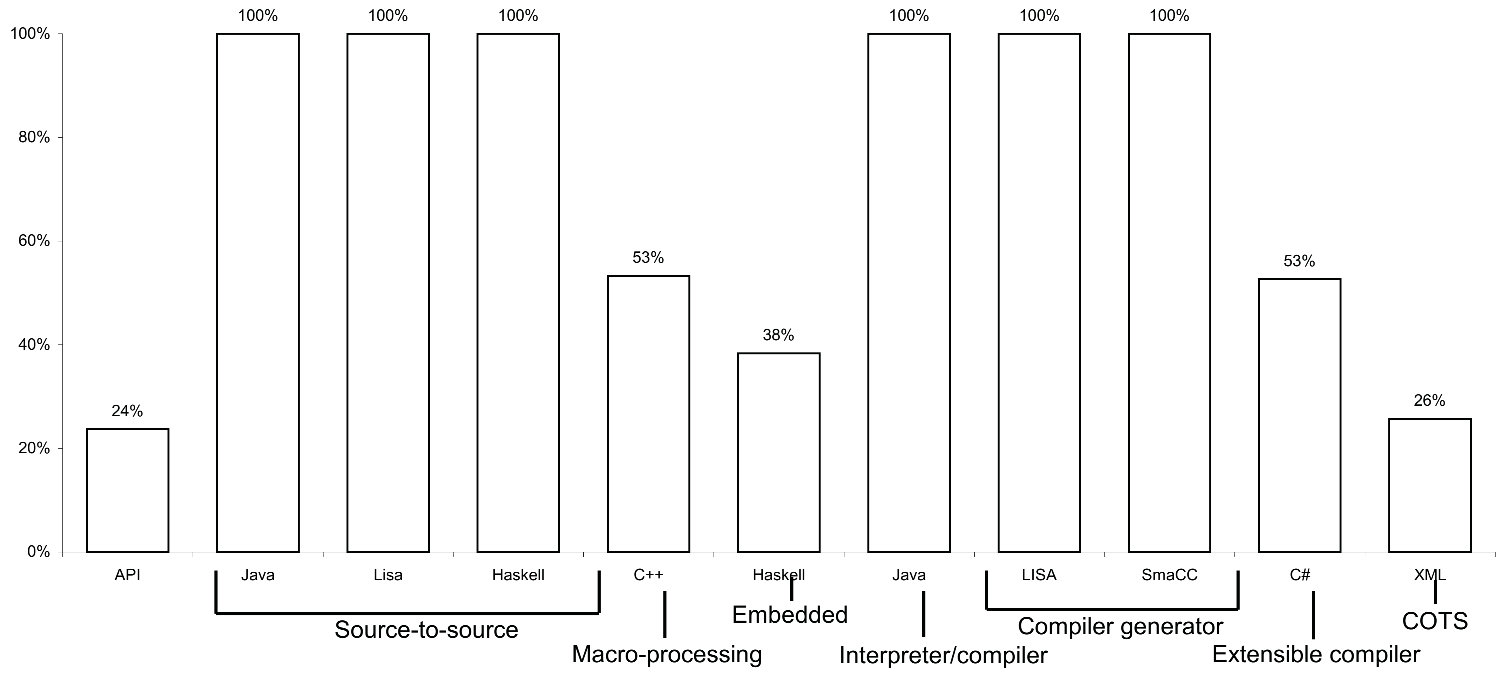
Implementation effort - eLOC/LF (LF – language factor)



	LF
Java	53
Lisa	53
Haskell	38
C++	53
Smacc	27
C#	53

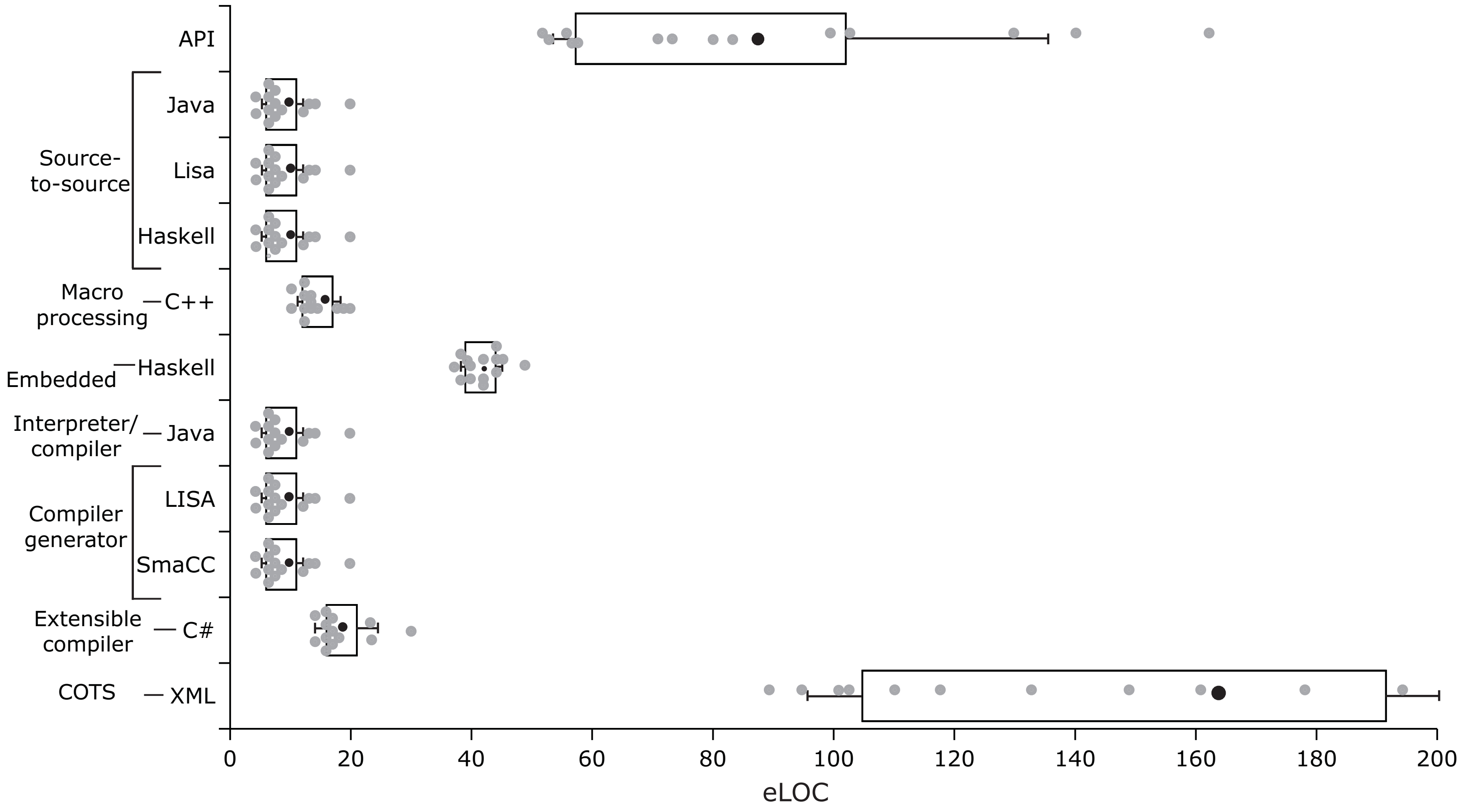
DSLs: Implementation Patterns

End-user effort - similarity factor



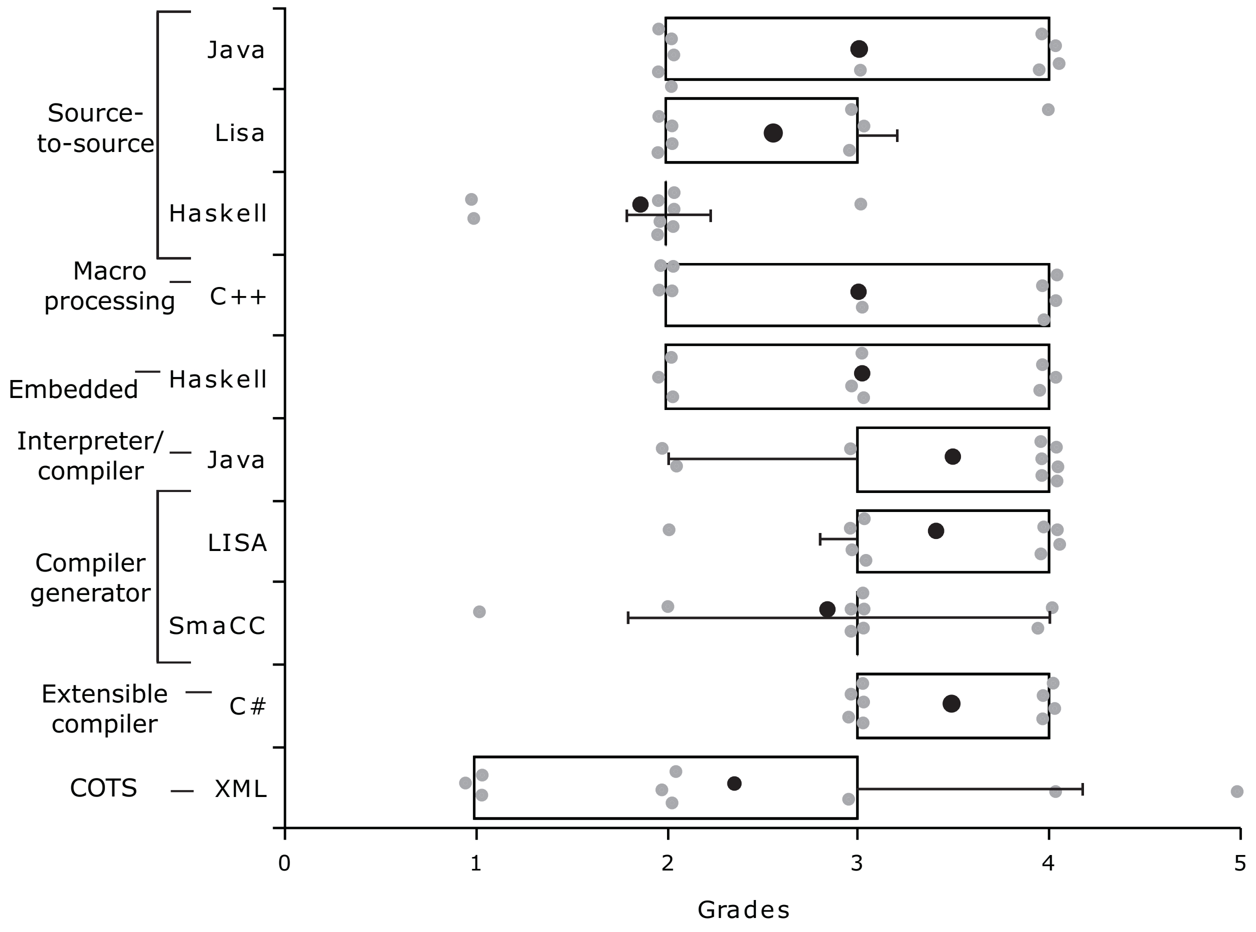
DSLs: Implementation Patterns

End-user effort - program length



DSLs: Implementation Patterns

End-user effort – error reporting



DSLs: Implementation Patterns

- Key findings were:
  - Standard metrics were used (eLOC, eLOC/LF) to achieve fair comparisons among different implementation approaches. These standard metrics show that the most efficient way, in terms of lines of code to implement a DSL, is the embedded approach. Normalizing with the language factor did not change this fact.
  - All approaches, where original notation was not achieved (COTS, embedded, extensible compiler, macro-processing), resulted in worse end-user productivity.

DSLs: A Systematic Mapping Study

- A systematic review (SR) is a secondary study that reviews primary studies with the aim of synthesising evidence related to a specific research question. Several forms of SRs exists:
  - Systematic literature review (SLR): A form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and repeatable.
  - Systematic mapping study (SMS): A broad review of primary studies in a specific topic area that aims to identify what evidence is available on the topic.
  - Tertiary review (TR): which is a systematic review of systematic reviews.

DSLs: A Systematic Mapping Study

- A Systematic Mapping Study (SMS) has been performed to better understand the DSL research field, identify research trends, and any possible open issues.
- Automatic search including primary studies from journals, conferences, and workshops during the period from 2006 until 2012.
- A SMS was conducted over 5 stages: defining research questions, conducting the search, screening, classifying, and data extraction.

DSLs: A Systematic Mapping Study

- RQ1: What has been the research space of the literature within the field of DSLs since the survey paper on DSLs was published in 2005?
- RQ2: What have been the trends and demographics of the literature within the field of DSLs after the survey on DSLs was published in 2005?

- In a very broad sense we were interested in DSL studies from three different prospectives: type of contribution, type of research, and focus area.

DSLs: A Systematic Mapping Study

- RQ1.1Type of contribution: What is the main contribution of DSL studies with respect to techniques/methods, tools, processes, and measurements?
- A particular study would fall into:
  - the 'DSL development techniques/methods' category if the study's main contribution were to be a technique/method of any DSL development phase: domain analysis, design, implementation (e.g., DSL compiler), validation, and maintenance (e.g., DSL evolution).

DSLs: A Systematic Mapping Study

- the 'DSL development tools' category if the study's main contribution were to be a tool that supported one or more phases of DSL development (domain analysis, design, implementation, validation, and maintenance).
- the 'DSL processes' category if the study's main contribution were DSLs within a wider context of software engineering (e.g., integration within a larger project), or a particular DSL process were to be described (DSL debugging, DSL testing, DSL usability test).
- the 'DSL measurement' category if the study's main contribution were to be a proposal or application of metrics regarding the effectiveness of DSL approaches (e.g., measuring comprehensibilities of DSL programs, measuring the productivities of DSL users).

DSLs: A Systematic Mapping Study

- RQ1.2 Type of research: What types of research methods have been used in DSL studies?
  - Opinion paper, which reports on authors' opinions as to whether a certain technique/tool/process/measurement is good or bad.
  - Experience paper, which reports on authors' experiences of certain technique/tools/processes/measurements, as used in practice.
  - Philosophical/conceptual paper, which provides the taxonomy of a research field or a conceptual framework for structuring phenomena under investigation/design.

DSLs: A Systematic Mapping Study

- Solution proposal, which proposes a certain technique/tool/process/measurement as a solution to a particular problem and explains it as small example or a good line of argumentation. However, technique/tool/process/measurement has not yet been implemented.
- Validation research, where a certain technique/tool/process/measurement is implemented as a solution to a problem and validated by simulations, prototyping, experiments, mathematical systematical analysis, mathematical proof of properties, etc. However, the implementation has not yet been evaluated in practice.

DSLs: A Systematic Mapping Study

- Evaluation research, where a certain technique/tool/process/measurement as a solution to a problem is implemented and validated in practice. Its benefits and drawbacks are evaluated by controlled experiments, observational studies, or case studies.

DSLs: A Systematic Mapping Study

- RQ1.3 Focus area: Which research topics have been investigated in DSL studies?
- The following DSL development phases based on a DSL survey paper will be included: domain analysis, design phase, implementation phase, validation phase, and maintenance phase.

DSLs: A Systematic Mapping Study

- Search string: ("domain-specific language" OR "DSL")

AND (year>2005 AND year<2013)

- Inclusion criteria:
  - study must have addressed DSL research,
  - peer reviewed studies had been published in journals, conferences, and workshops,
  - study must be written in English,
  - study must be accessible electronically, and
  - computer science literature.

DSLs: A Systematic Mapping Study

- Exclusion criteria:
  - irrelevant publications that lay outside the core DSL research field, which also excluded DSMLs, modelware, and MDE publications, visual/graphical languages (based on graph-grammars or other formalisms) or those mentioning DSL as future work;
  - non-peer reviewed studies (abstracts, tutorials, editorials, slides, talks, tool demonstrations, posters, panels, keynotes, technical reports);
  - peer-reviewed but not published in journals, conferences, workshops (e.g., PhD thesis, books, patents);
  - publications not in English;
  - electronically non-accessible; and
  - non-computer science literature.

DSLs: A Systematic Mapping Study

- The inclusion and exclusion criteria were applied to the titles, keywords, and abstracts. In those case where it wasn't completely clear from the title, keywords, and abstract that a publication really addressed the DSL research then such publications were temporarily included but might be excluded during the next phase (classification phase) when the whole publication (not only the abstract) had been read.

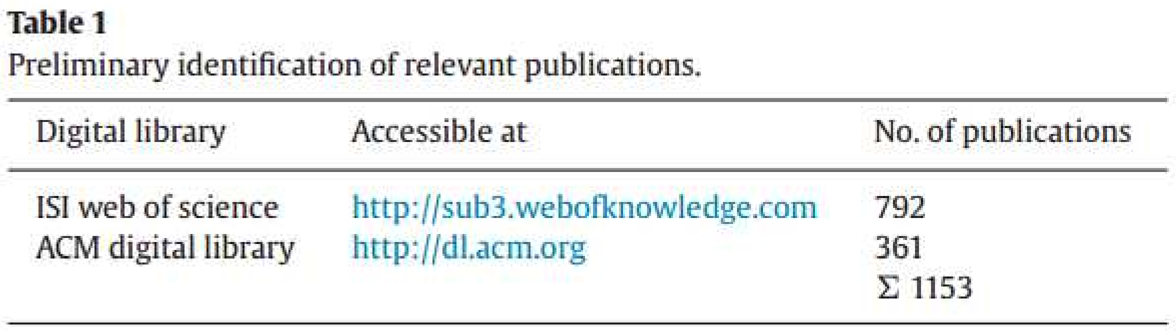


Table 1
Preliminary identification of relevant publications.

Digital library	Accessible at	No. of publications
ISI web of science	http://sub3.webofknowledge.com	792
ACM digital library	http://dl.acm.org	361
		Σ 1153

DSLs: A Systematic Mapping Study

- After the screening of 1153 publications 713 publications satisfied the aforementioned criteria and entered into the next phase – classification, where an additional 323 publications were then excluded. Hence, altogether 390 primary studies were classified.

DSLs: A Systematic Mapping Study

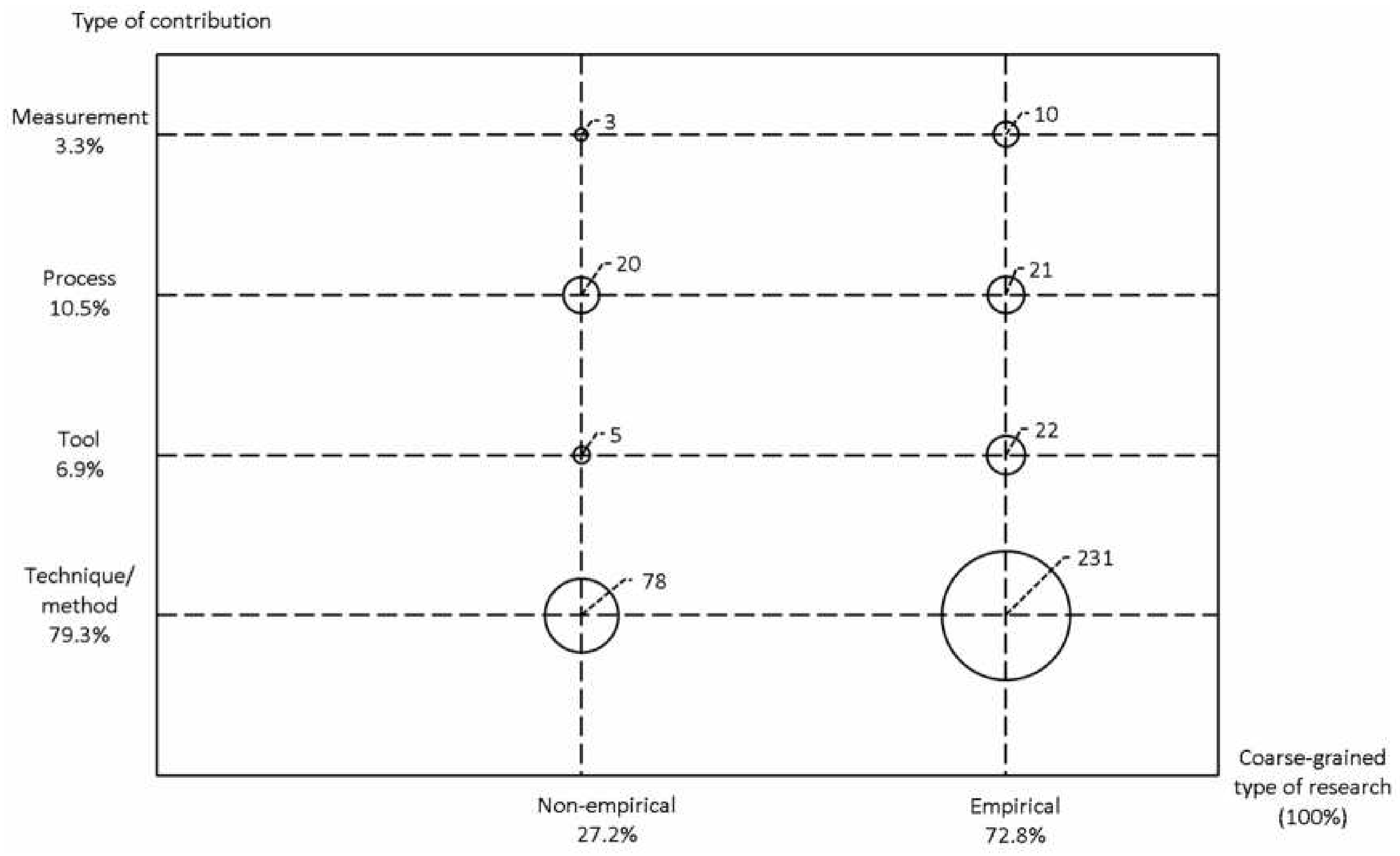


Fig. 1. Number of primary studies per type of contribution (RQ 1.1) and coarse-grained type of research (non-empirical vs. empirical) (RQ 1.2), $n = 390$.

DSLs: A Systematic Mapping Study

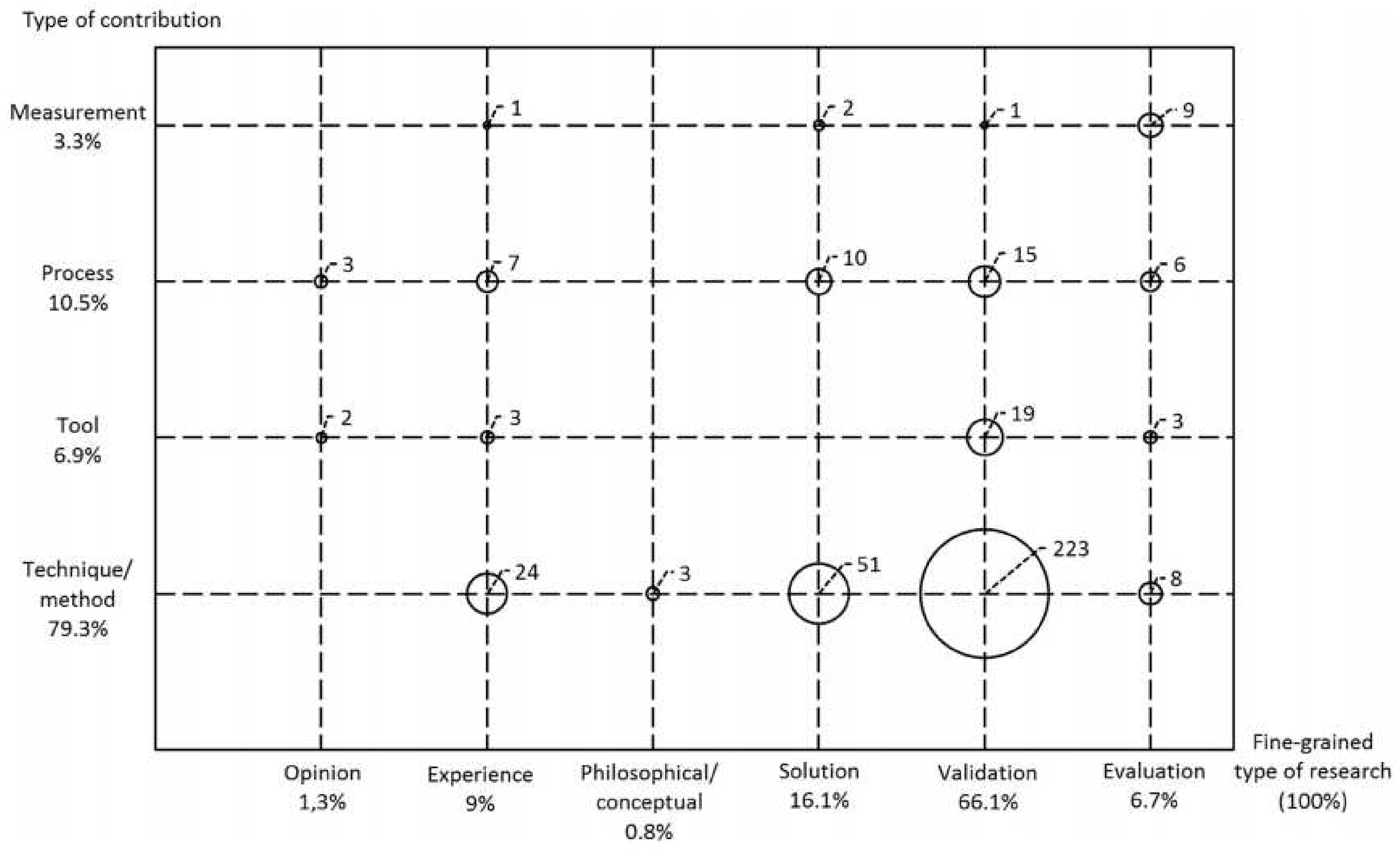


Fig. 2. Number of primary studies per type of contribution (RQ 1.1) and fine-grained type of research (RQ 1.2), $n = 390$.

DSLs: A Systematic Mapping Study

- The primary studies usually discussed the following three DSL development phases: domain analysis, design and implementation, whilst validation and maintenance have been rarely presented.

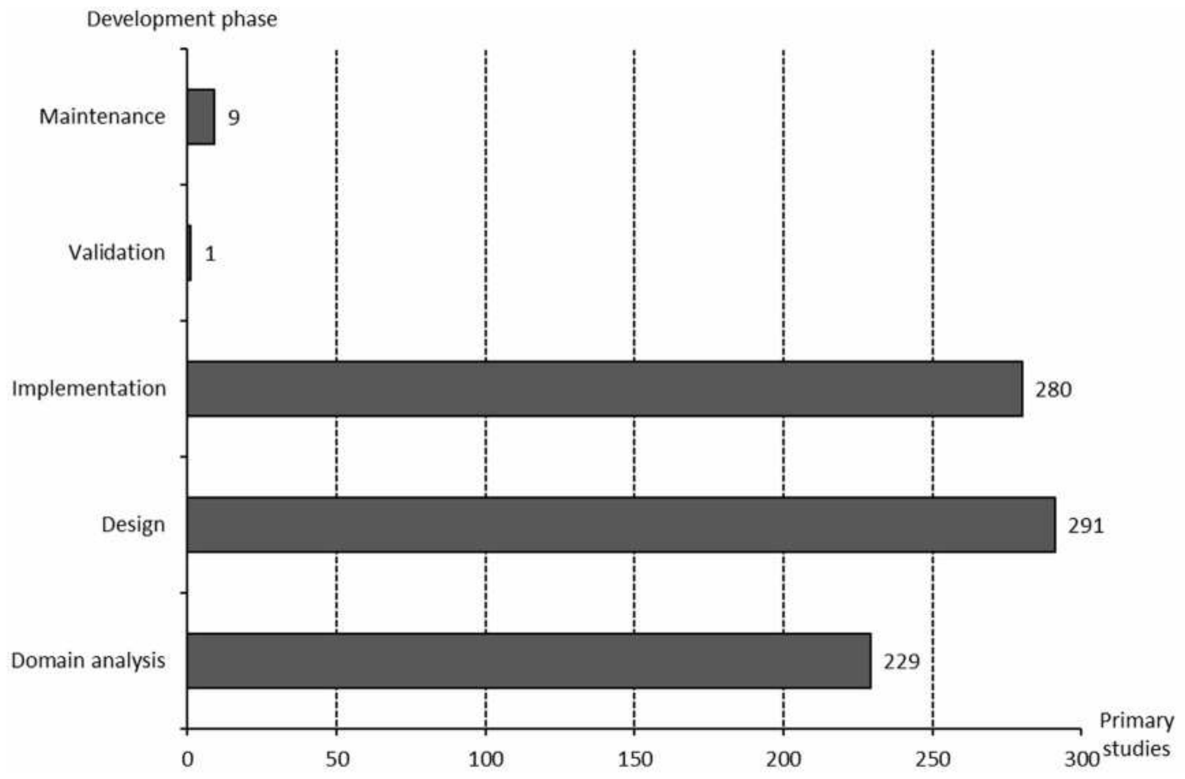


Fig. 3. Research distribution in DSL development phases (RQ 1.3).

DSLs: A Systematic Mapping Study

- Only 5.7% of primary studies that included domain analysis had used a formal domain analysis approach. Hence, formal domain analysis methods had been rarely used in DSL development and domain analysis had usually been done informally and probably in an incomplete manner.

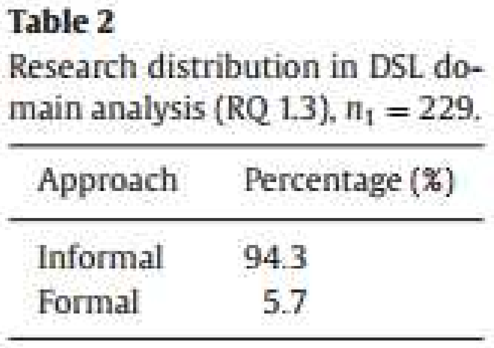


Table 2
Research distribution in DSL domain analysis (RQ 1.3), $n_1 = 229$.

Approach	Percentage (%)
Informal	94.3
Formal	5.7

DSLs: A Systematic Mapping Study

- Only 16.8% of primary studies that included the design phase used formal approaches for describing syntax and semantics. Although internal DSLs, which rarely require formal description as they rely on (formal) description of existing language, comprised 47.8% out of 83.2% informal cases, the number of DSLs using formal syntax and especially semantic description had still been low.

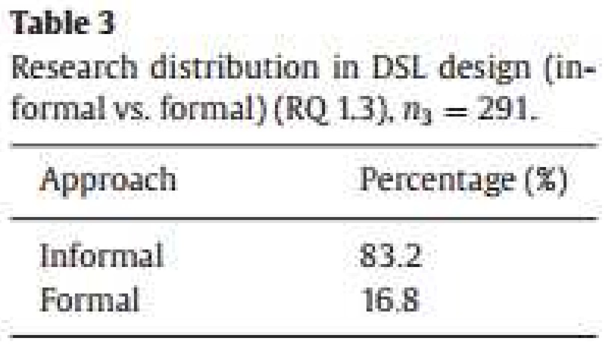


Table 3
Research distribution in DSL design (informal vs. formal) (RQ 1.3), $n_3 = 291$.

Approach	Percentage (%)
Informal	83.2
Formal	16.8

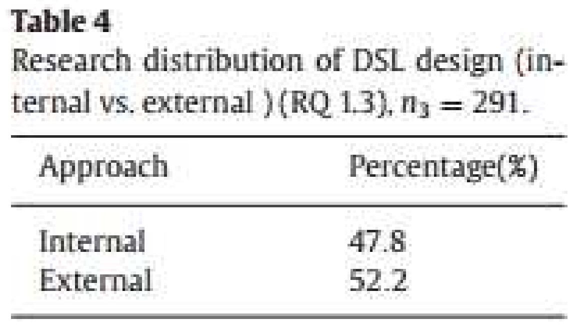


Table 4
Research distribution of DSL design (internal vs. external) (RQ 1.3), $n_3 = 291$.

Approach	Percentage(%)
Internal	47.8
External	52.2

DSLs: A Systematic Mapping Study

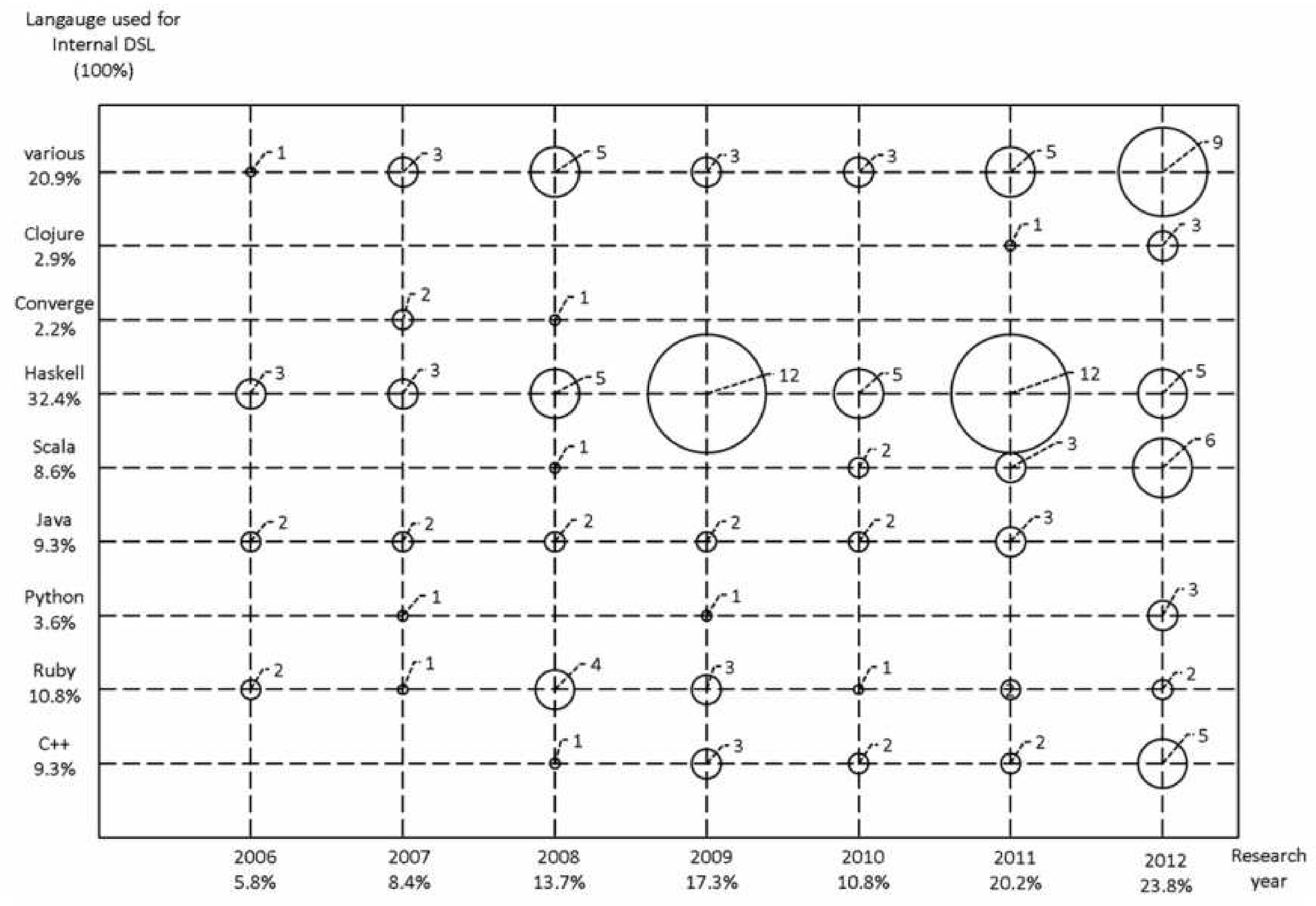


Fig. 5. Distribution of GPLs used for DSL embedding per years (RQ 1.3), $n_4 = 139$.

DSLs: A Systematic Mapping Study

- Amongst the more frequently used implementation patterns have been the embedding approach (34.3%) and the compiler approach (28.1%). Other implementation approaches had been less frequently used: preprocessing (15%), COTS (7.9%), interpreter (7.9%), hybrid (3.9%), and the extensible compiler/interpreter approach (2.9%). Hence, this study doesn't support some claims that the embedding approach has prevailed over DSL implementation approaches.

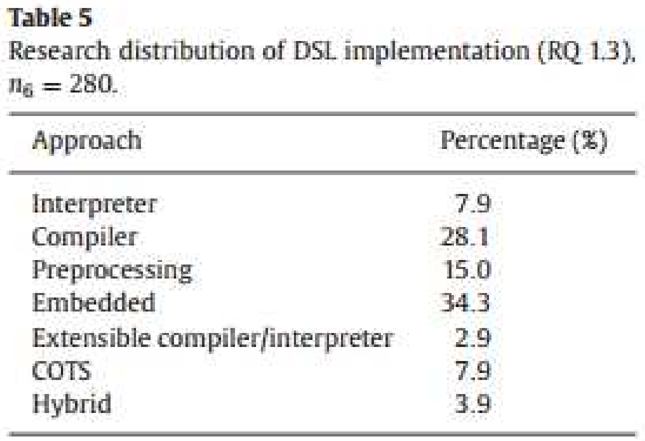


Table 5
Research distribution of DSL implementation (RQ 1.3), $n_6 = 280$.

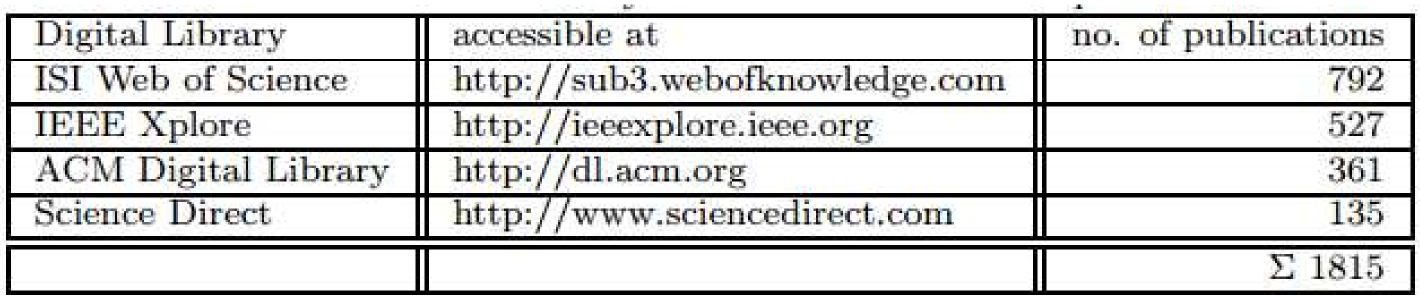
Approach	Percentage (%)
Interpreter	7.9
Compiler	28.1
Preprocessing	15.0
Embedded	34.3
Extensible compiler/interpreter	2.9
COTS	7.9
Hybrid	3.9

DSLs: A Systematic Mapping Study

- The main findings of our SMS on DSLs are:
  - Research about DSL integration with other SE process is lacking, as well as measuring the effectiveness of DSL approaches.
  - Clear lack of evaluation research, in particular controlled experiments.
  - Amongst different DSL development phases the following phases have been insufficiently investigated: domain analysis, validation and maintenance.
  - Lack of use regarding formal methods within domain analysis and in the semantic description of DSLs.

DSLs: A Systematic Mapping Study

- Our previous SMS for Domain-Specific Languages (DSLs) has been extended by the inclusion of two additional DLs. It is shown that the original SMS for DSLs with 390 primary studies classified produces similar and reliable results when compared to extended SMS for DSLs presented with 477 primary studies classified.



Digital Library	accessible at	no. of publications
ISI Web of Science	http://sub3.webofknowledge.com	792
IEEE Xplore	http://ieeexplore.ieee.org	527
ACM Digital Library	http://dl.acm.org	361
Science Direct	http://www.sciencedirect.com	135
		Σ 1815

DSLs: A Systematic Mapping Study

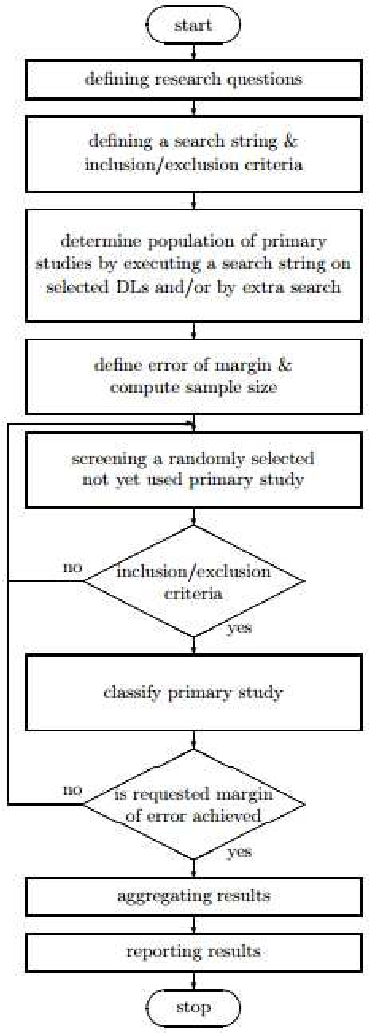
- Performing SMS is time consuming tasks.
- On average to read and classify one primary study took us one hour.
- Can this cost be minimized and still producing reliable results?

DSLs: A Systematic Mapping Study

- Systematic Mapping Study Driven by Margin of Error
- By defining the acceptable error margin, a sample size is computed from Eqs. 1 and 2. The main idea is to randomly select candidate primary studies from the population until the requested margin of error is surpassed.

$$n = \frac{z^2_{\alpha/2} * p * (1-p)}{margin\ of\ error^2}$$

$$n' = \frac{n}{1 + \frac{n-1}{N}}$$



start
defining research questions
defining a search string & inclusion/exclusion criteria
determine population of primary studies by executing a search string on selected DLs and/or by extra search
define error of margin & compute sample size
screening a randomly selected not yet used primary study
no
inclusion/exclusion criteria
yes
classify primary study
no
is requested margin of error achieved
yes
aggregating results
reporting results
stop

DSLs: A Systematic Mapping Study

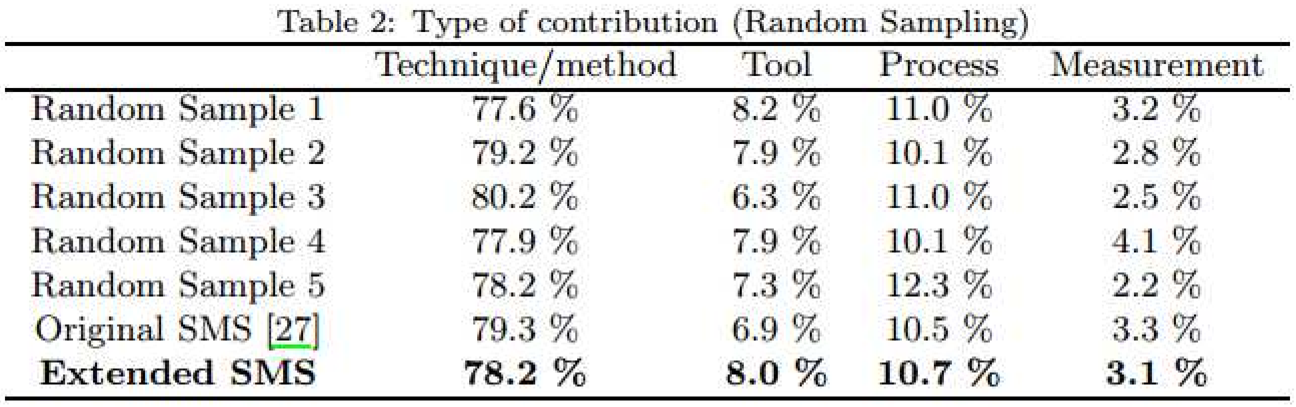


Table 2: Type of contribution (Random Sampling)

	Technique/method	Tool	Process	Measurement
Random Sample 1	77.6 %	8.2 %	11.0 %	3.2 %
Random Sample 2	79.2 %	7.9 %	10.1 %	2.8 %
Random Sample 3	80.2 %	6.3 %	11.0 %	2.5 %
Random Sample 4	77.9 %	7.9 %	10.1 %	4.1 %
Random Sample 5	78.2 %	7.3 %	12.3 %	2.2 %
Original SMS [27]	79.3 %	6.9 %	10.5 %	3.3 %
Extended SMS	**78.2 %**	**8.0 %**	**10.7 %**	**3.1 %**

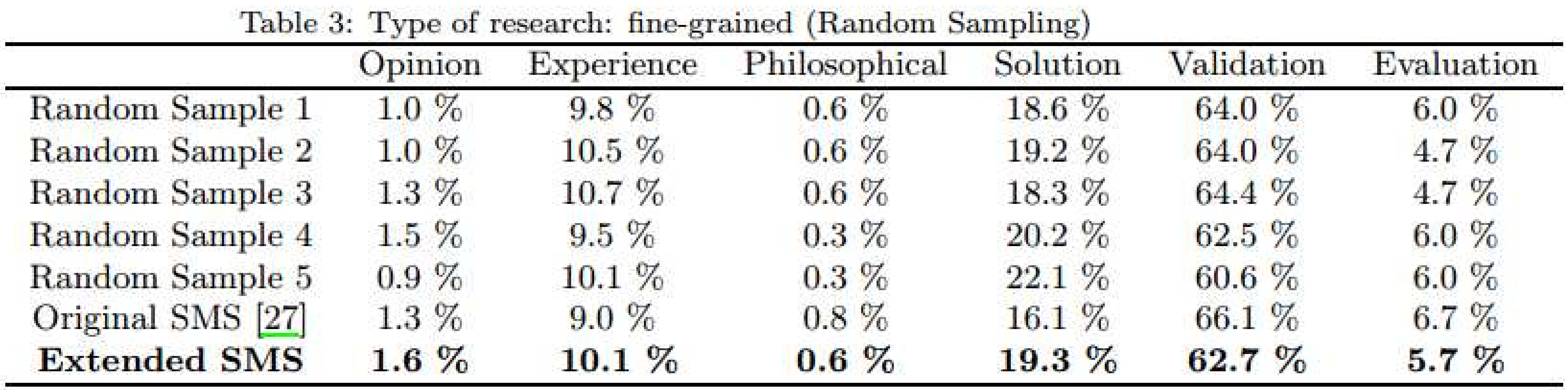


Table 3: Type of research: fine-grained (Random Sampling)

	Opinion	Experience	Philosophical	Solution	Validation	Evaluation
Random Sample 1	1.0 %	9.8 %	0.6 %	18.6 %	64.0 %	6.0 %
Random Sample 2	1.0 %	10.5 %	0.6 %	19.2 %	64.0 %	4.7 %
Random Sample 3	1.3 %	10.7 %	0.6 %	18.3 %	64.4 %	4.7 %
Random Sample 4	1.5 %	9.5 %	0.3 %	20.2 %	62.5 %	6.0 %
Random Sample 5	0.9 %	10.1 %	0.3 %	22.1 %	60.6 %	6.0 %
Original SMS [27]	1.3 %	9.0 %	0.8 %	16.1 %	66.1 %	6.7 %
Extended SMS	**1.6 %**	**10.1 %**	**0.6 %**	**19.3 %**	**62.7 %**	**5.7 %**

DSLs: A Systematic Mapping Study

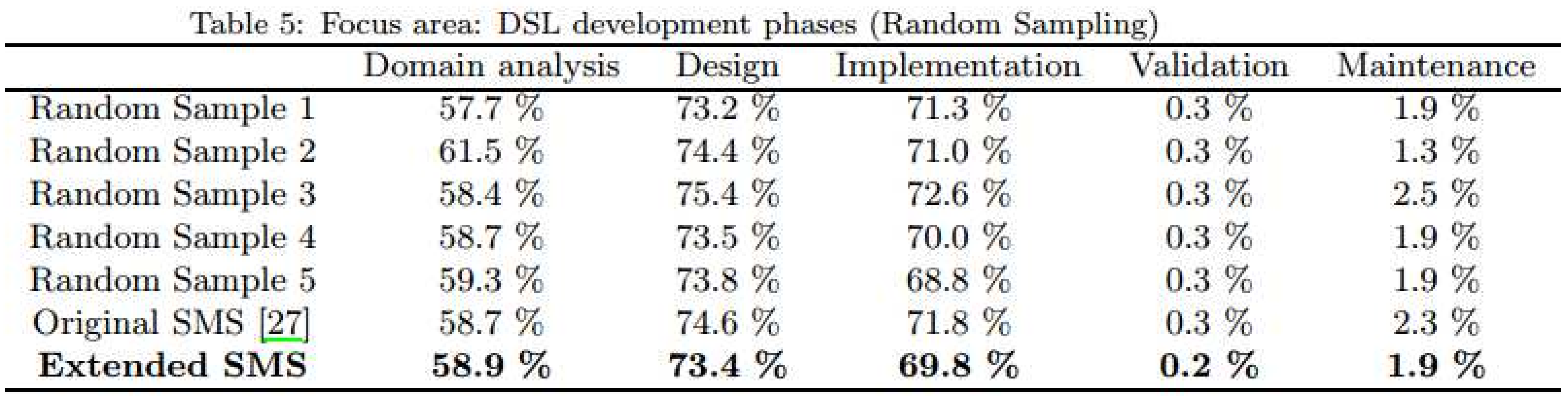


Table 5: Focus area: DSL development phases (Random Sampling)

	Domain analysis	Design	Implementation	Validation	Maintenance
Random Sample 1	57.7 %	73.2 %	71.3 %	0.3 %	1.9 %
Random Sample 2	61.5 %	74.4 %	71.0 %	0.3 %	1.3 %
Random Sample 3	58.4 %	75.4 %	72.6 %	0.3 %	2.5 %
Random Sample 4	58.7 %	73.5 %	70.0 %	0.3 %	1.9 %
Random Sample 5	59.3 %	73.8 %	68.8 %	0.3 %	1.9 %
Original SMS [27]	58.7 %	74.6 %	71.8 %	0.3 %	2.3 %
Extended SMS	**58.9 %**	**73.4 %**	**69.8 %**	**0.2 %**	**1.9 %**

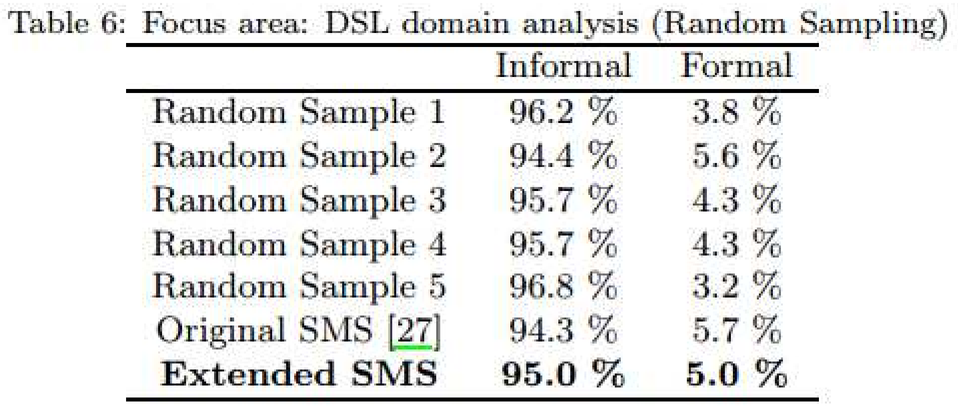


Table 6: Focus area: DSL domain analysis (Random Sampling)

	Informal	Formal
Random Sample 1	96.2 %	3.8 %
Random Sample 2	94.4 %	5.6 %
Random Sample 3	95.7 %	4.3 %
Random Sample 4	95.7 %	4.3 %
Random Sample 5	96.8 %	3.2 %
Original SMS [27]	94.3 %	5.7 %
Extended SMS	**95.0 %**	**5.0 %**

DSLs: A Systematic Mapping Study

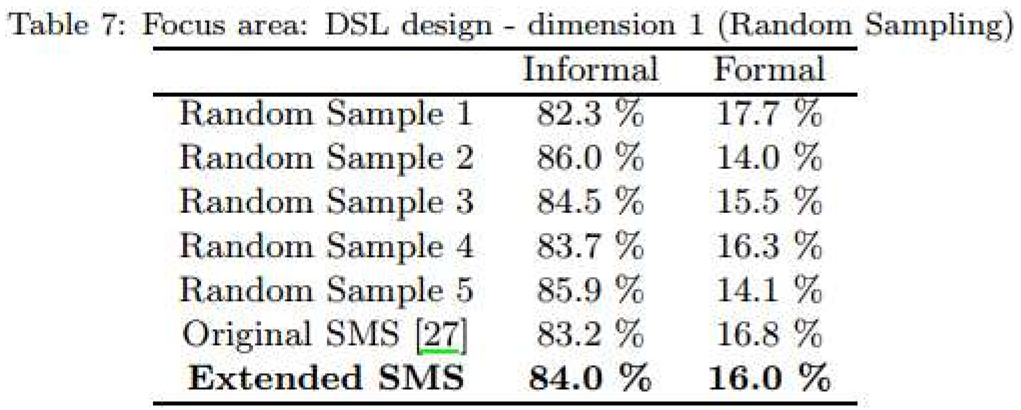


Table 7: Focus area: DSL design - dimension 1 (Random Sampling)

	Informal	Formal
Random Sample 1	82.3 %	17.7 %
Random Sample 2	86.0 %	14.0 %
Random Sample 3	84.5 %	15.5 %
Random Sample 4	83.7 %	16.3 %
Random Sample 5	85.9 %	14.1 %
Original SMS [27]	83.2 %	16.8 %
Extended SMS	**84.0 %**	**16.0 %**

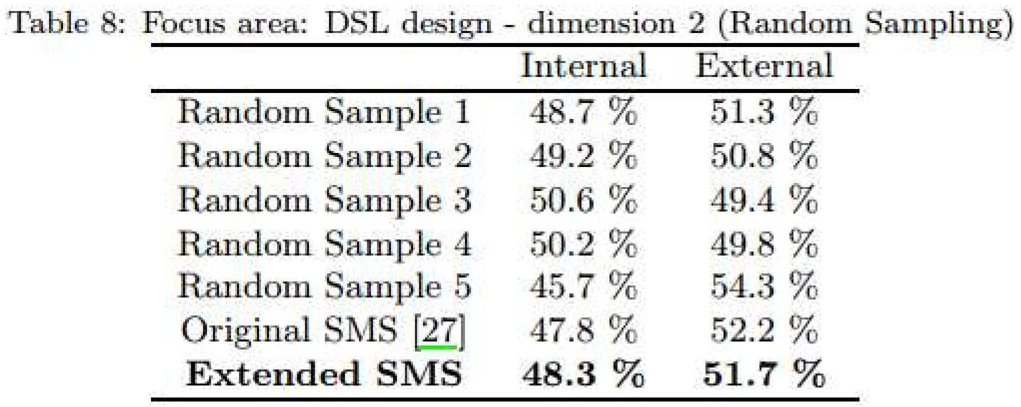


Table 8: Focus area: DSL design - dimension 2 (Random Sampling)

	Internal	External
Random Sample 1	48.7 %	51.3 %
Random Sample 2	49.2 %	50.8 %
Random Sample 3	50.6 %	49.4 %
Random Sample 4	50.2 %	49.8 %
Random Sample 5	45.7 %	54.3 %
Original SMS [27]	47.8 %	52.2 %
Extended SMS	**48.3 %**	**51.7 %**

DSLs: A Systematic Mapping Study

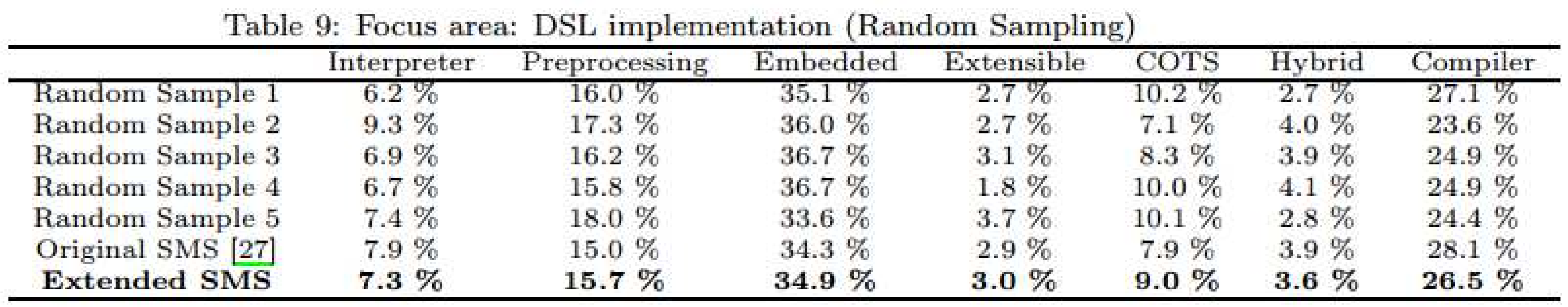


Table 9: Focus area: DSL implementation (Random Sampling)

	Interpreter	Preprocessing	Embedded	Extensible	COTS	Hybrid	Compiler
Random Sample 1	6.2 %	16.0 %	35.1 %	2.7 %	10.2 %	2.7 %	27.1 %
Random Sample 2	9.3 %	17.3 %	36.0 %	2.7 %	7.1 %	4.0 %	23.6 %
Random Sample 3	6.9 %	16.2 %	36.7 %	3.1 %	8.3 %	3.9 %	24.9 %
Random Sample 4	6.7 %	15.8 %	36.7 %	1.8 %	10.0 %	4.1 %	24.9 %
Random Sample 5	7.4 %	18.0 %	33.6 %	3.7 %	10.1 %	2.8 %	24.4 %
Original SMS [27]	7.9 %	15.0 %	34.3 %	2.9 %	7.9 %	3.9 %	28.1 %
Extended SMS	**7.3 %**	**15.7 %**	**34.9 %**	**3.0 %**	**9.0 %**	**3.6 %**	**26.5 %**

- The results from original SMS for DSLs with 390 and extended SMS with 477 classified primary studies were not much different than results from SMS driven by the margin of error where 317 primary studies have been identified.

DSLs: Other Open Problems

- How to obtain fully modular, extensible, and reusable language description?
- How to obtain good integration of DSLs with other software development tools?
- How DSL development can be made easier for domain experts not versed in a programming language design?
- How to define semantics of DSML?

Conclusion

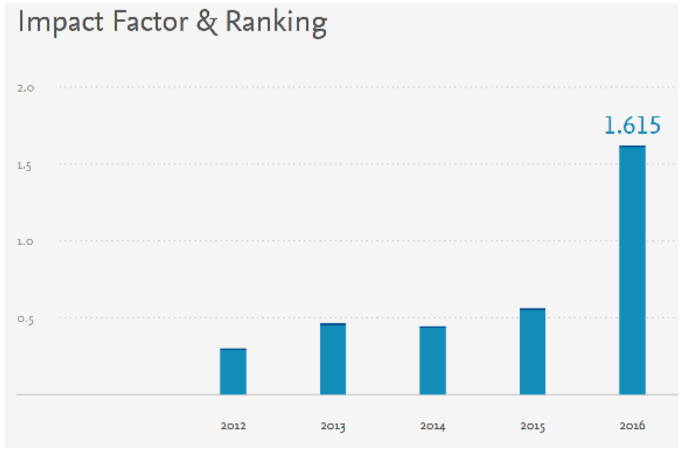
- DSLs are computer languages of the forthcoming years.
- Many DSLs problems still need to be solved!

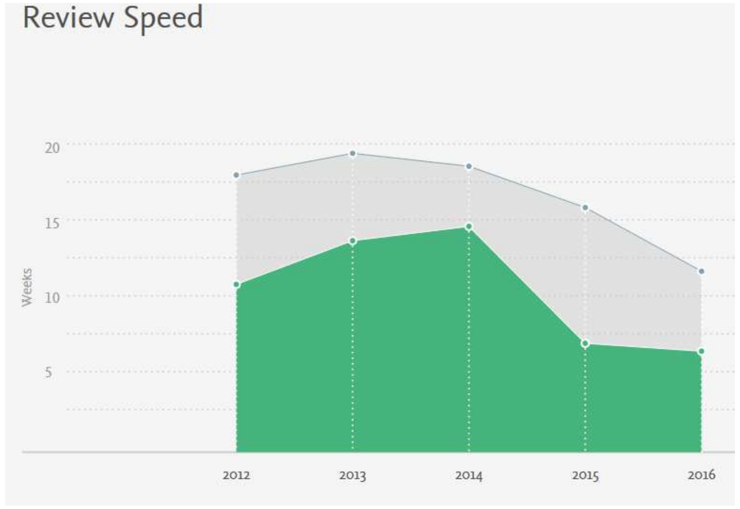
My 5 most important DSL publications:

- MERNIK, Marjan, HEERING, Jan, SLOANE, Anthony M. When and how to develop domain-specific languages. *ACM comput. surv.*, 2005, vol. 37, no. 4, pp. 316-344. JCR IF: 7.4
- KOSAR, Tomaž, MARTÍNEZ LÓPEZ, Pablo E., BARRIENTOS, Pablo A., MERNIK, Marjan. A preliminary study on various implementation approaches of domain-specific language. *Inf. softw. technol.*, 2008, vol. 50, iss. 5, pp. 390-405. JCR IF: 1.2
- KOSAR, Tomaž, MERNIK, Marjan, CARVER, Jeffrey C. Program comprehension of domain-specific and general-purpose languages : comparison using a family of experiments. *Empirical software engineering*, ISSN 1382-3256, 2012, vol. 17, no. 3, pp. 276-304, JCR IF: 1.18
- MERNIK, Marjan. An object-oriented approach to language compositions for software language engineering. *The Journal of Systems and Software*, ISSN 0164-1212, 2013, vol. 86, iss. 9, pp. 2451-2464, JCR IF: 1.245
- KOSAR, Tomaž, BOHRA, Sudev, MERNIK, Marjan. Domain-specific languages: a systematic mapping study. *Information and software technology*, ISSN 0950-5849, 2016, vol. 71, pp. 77-9, JCR IF: 1.569

Conclusion

Where to submit your work on DSLs?





LEGEND

○ Submission to final decision

○ Submission to first decision

Questions