SLE as an Evolving Body of Knowledge:

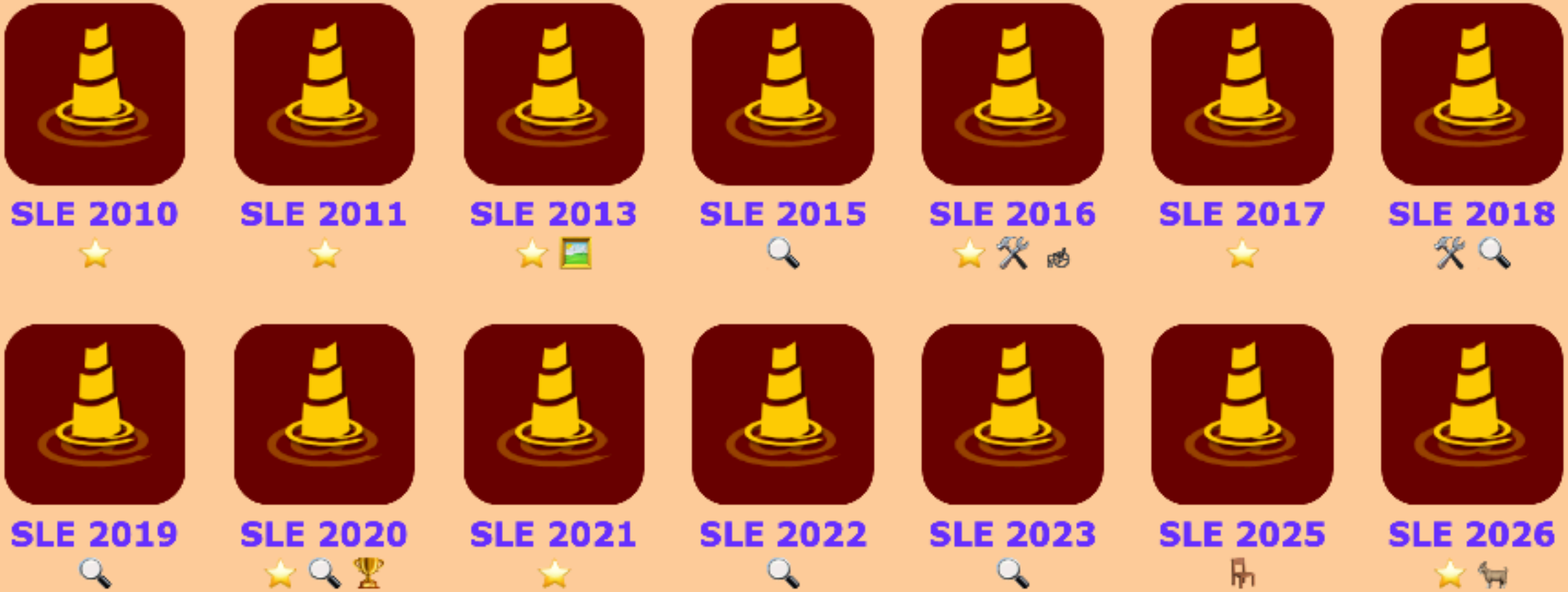
A 19-Year Comparative Analysis of Calls and Proceedings

SLE & STAF 2026, 2 July

dr.ir. Vadim Zaytsev, University of Twente, 🇳🇱

UNIVERSITY OF TWENTE. | Formal Methods & Tools |

Vadim Zaytsev



https://cfpbok.github.io/profile/VadimZaytsev.html

Mark van den Brand


SLE 2008


SLE 2009


SLE 2010


SLE 2011


SLE 2012


SLE 2013


SLE 2017


SLE 2018


SLE 2019


SLE 2021


SLE 2023

https://cfpbok.github.io/profile/MarkvandenBrand.html

Jordi Cabot

https://cfpbok.github.io/profile/JordiCabot.html

Call for Papers (source, 40 taggings)

Scope

The term "software language" comprises all sorts of artificial languages used in software development including general purpose programming languages[T3D], domain-specific languages[T3C], modeling and meta-modeling languages[T3A], data models, and ontologies[T3E]. We use this term in its broadest sense. Thus, for example, modeling languages include UML and UML-based languages, synchronous languages used in safety critical applications, business process modeling languages, and web application modeling languages, to name a few. Perhaps less obviously, the term "software language" also comprises APIs[T3F] and collections of design patterns that are indeed implicitly defined languages.

Software language engineering is the application of a systematic, disciplined, quantifiable approach to the development[T2A], use, and maintenance[T2B] of these languages. Thus, the SLE conference is concerned with all phases of the lifecycle of software languages[T2A]; these include the design[T1A], implementation, documentation[T2E], testing[T5C], deployment[T2D], evolution[T2B], recovery[T4C], and retirement[T2A] of languages. Of special interest are tools, techniques, methods and formalisms[T5D] that support these activities. In particular, tools are often based on or even automatically generated[T4C] from a formal description[T5D] of the language. Hence, of special interest is the treatment of language descriptions as software artifacts[T5B], akin to programs - while paying attention to the special status of language descriptions, subject to tailored engineering principles and methods for modularization[T1D], refactoring[T4B], refinement[T4C], composition[T1D], versioning[T2B], co-evolution[T2B], and analysis.

Themes and Topics

We solicit high-quality contributions in the area of SLE ranging from theoretical and conceptual contributions to tools, techniques and frameworks that support the aforementioned lifecycle activities. Some examples of tools, techniques, applications, and problems are listed below in order to clarify the types of contributions sought by SLE.

- Formalisms used in designing and specifying languages and tools that analyze such language descriptions[T3A]
- Language implementation tools and techniques[T4A]
- Program and model transformation tools[T4B]
- Composition, integration[T1D], and mapping tools[T4B] for managing different aspects of software languages or different manifestations of a given language
- Language evolution[T2B]
- Approaches to elicitation, specification[T5A], and verification[T5D] of requirements for software languages[T5A]
- Language development frameworks, methodologies, techniques, best practices, and tools for the broader language lifecycle[T2A] covering phases such as analysis[T5B], testing[T5C], and documentation[T2E].
- Design challenges in SLE[T1A]
- Applications of languages including innovative domain-specific languages or "little" languages[T3C]

Do note that this list is not exclusive and many examples of tools, techniques, approaches have not been listed. The program committee chairs encourage potential contributors to contact them with questions about the scope and topics of interest to SLE.

List of Papers (2+16+1+1, source)

- 🔥 *The Field of Software Language Engineering* (Anneke Kleppe)
  - **T6D: Synergies** T1A: Design T4A: Workbenches
- 🔥 *Model-Driven Engineering Meets Generic Language Technology* (Mark van den Brand)
  - **T6D: Synergies** T4A: Workbenches T4B: Horizontal Transformation
- *Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams* (Daniel Laurence Moody, Jos van Hillegersberg)
  - **T5H: Usability** T4F: Visualisation T5E: Empirical Evaluation
- *Neon: A Library for Language Usage Analysis* (Jurriaan Hage, Peter van Keeken)
  - **T4A: Workbenches** T5B: Language Description Analysis T2B: Evolution
- 🏆 *Analyzing Rule-Based Behavioral Semantics of Visual Modeling Languages with Maude* (José Eduardo Rivera, Esther Guerra, Juan de Lara, Antonio Vallecillo)
  - **T1C: Behavioural Semantics** T5D: Formal Methods T4D: Interpretation
- *Parse Table Composition: Separate Compilation and Binary Extensibility of Grammars* (Martin Bravenboer, Eelco Visser)
  - **T1D: Composition** T4C: Vertical Transformation T1E: Reuse
- *Practical Scope Recovery Using Bridge Parsing* (Emma Nilsson-Nyman, Torbjörn Ekman, Görel Hedin)
  - **T1B: Static Semantics** T4A: Workbenches T4C: Vertical Transformation
- *Generating Rewritable Abstract Syntax Trees: A Foundation for the Rapid Development of Source Code Transformation Tools* (Jeffrey Overbey, Ralph E. Johnson)
  - **T4B: Horizontal Transformation** T4A: Workbenches T1E: Reuse
- *Systematic Usage of Embedded Modelling Languages in Automated Model Transformation Chains* (Mathias Fritzsche, Jendrik Johannes, Uwe Aßmann, Simon Mitschke, Wasif Gilani, Ivor Spence, John Brown, Peter Kilpatrick)
  - **T1D: Composition** T4B: Horizontal Transformation T5G: Traceability
- *Engineering a DSL for Software Traceability* (Nikolaos Drivalos, Dimitrios S. Kolovos, Richard F. Paige, Kiran Fernandes)
  - **T5G: Traceability** T3C: DSLs T3A: Meta-languages
- 🦾 *Towards an Incremental Update Approach for Concrete Textual Syntaxes for UUID-Based Model Repositories* (Thomas Goldschmidt)
  - **T2B: Evolution** T4A: Workbenches T1D: Composition
- 🛠️ *A Model Engineering Approach to Tool Interoperability* (Yu Sun, Zekai Demirezen, Frédéric Jouault, Robert Tairas, Jeffrey G. Gray)
  - **T4B: Horizontal Transformation** T4A: Workbenches T6D: Synergies
- *Engineering Languages for Specifying Product-Derivation Processes in Software Product Lines* (Pablo Sánchez, Neil Loughran, Lidia Fuentes, Alessandro Fabricio Garcia)
  - **T2C: Variability** T3C: DSLs T1A: Design
- *Transformation Language Integration Based on Profiles and Higher Order Transformations* (Pieter Van Gorp, Anne Keller, Dirk Janssens)
  - **T1D: Composition** T3B: Transformation Languages T4B: Horizontal Transformation
- *Formalization and Rule-Based Transformation of EMF Ecore-Based Models* (Bernhard Schätz)
  - **T4B: Horizontal Transformation** T5D: Formal Methods T3B: Transformation Languages
- *A Practical Evaluation of Using TXL for Model Transformation* (Hongzhi Liang, Jürgen Dingel)
  - **T5E: Empirical Evaluation** T4B: Horizontal Transformation T3B: Transformation Languages
- *DeFacto: Language-Parametric Fact Extraction from Source Code* (Hendrikus J. S. Basten, Paul Klint)
  - **T5B: Language Description Analysis** T4A: Workbenches T1E: Reuse
- *A Case Study in Grammar Engineering* (Tiago L. Alves, Joost Visser)
  - **T6A: Experience Reports** T2B: Evolution T5C: Testing
- *Sudoku – A Language Description Case Study* (Terje Gjøsæter, Ingelin F. Isfeldt, Andreas Prinz)
  - **T6A: Experience Reports** T5B: Language Description Analysis T3C: DSLs
- *The Java Programmer's Phrase Book* (Einar W. Høst, Bjarte M. Østvold)
  - **T5E: Empirical Evaluation** T5B: Language Description Analysis T2B: Evolution

sle2016/paper10

SLE 2016 paper:

Towards a Universal Code Formatter through Machine Learning

Terence Parr, Jurgen Vinju

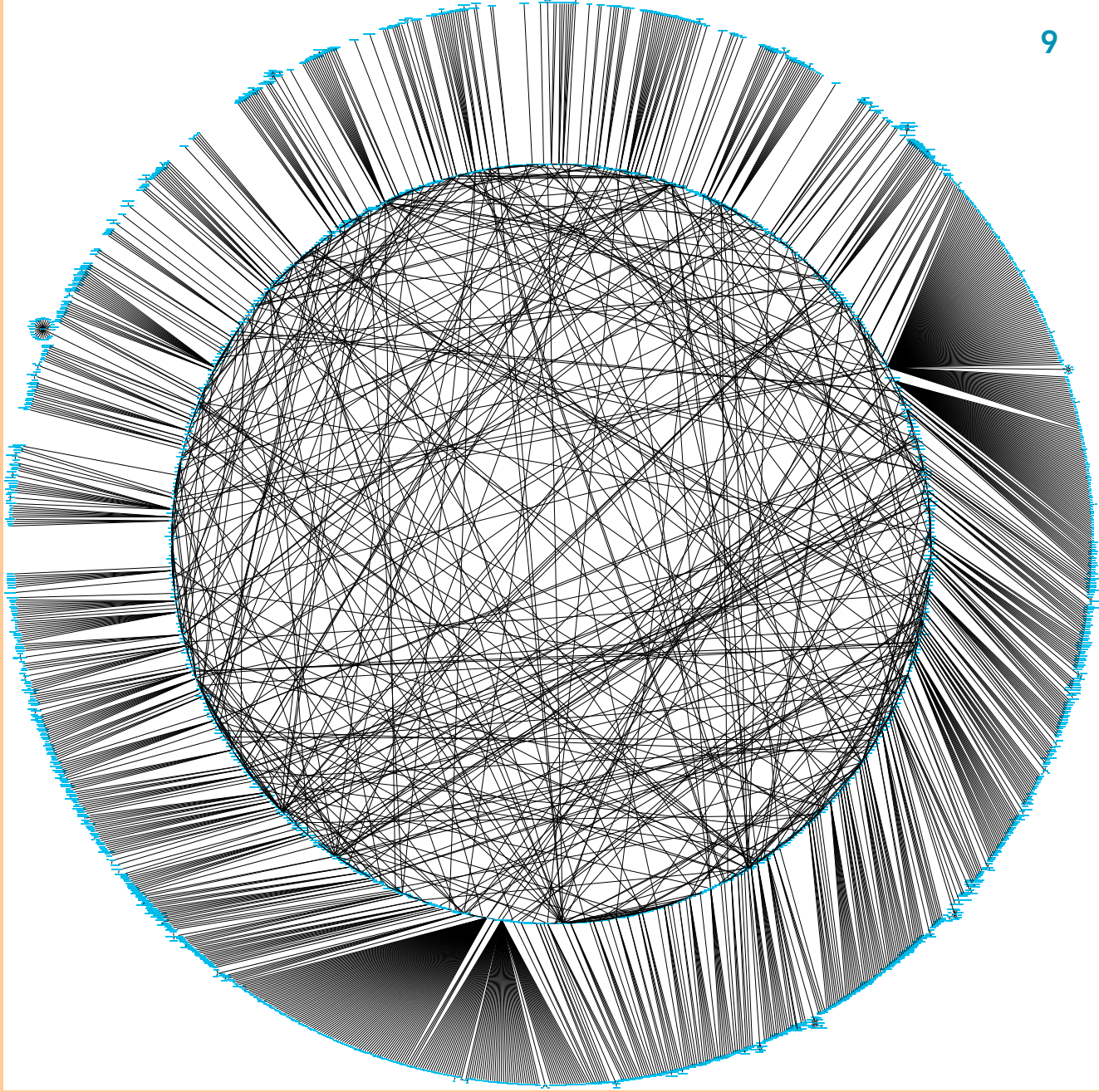
- 🏆 Distinguished Paper Award
- 🏆 Most Influencial Paper Award at SLE 2026
- DOI: 10.1145/2997364.2997383
- **T4G: AI-for-SLE**
  - The key technique is machine learning applied to infer formatting rules for languages.
- T4B: Horizontal Transformation
  - Formatting is a source-to-source transformation that rewrites code while preserving meaning.
- T5H: Usability
  - A main motivation is improving readability and user-specific formatting preferences (usability).
- T3D: GPLs
  - The formatter targets general-purpose programming languages and their concrete syntax.

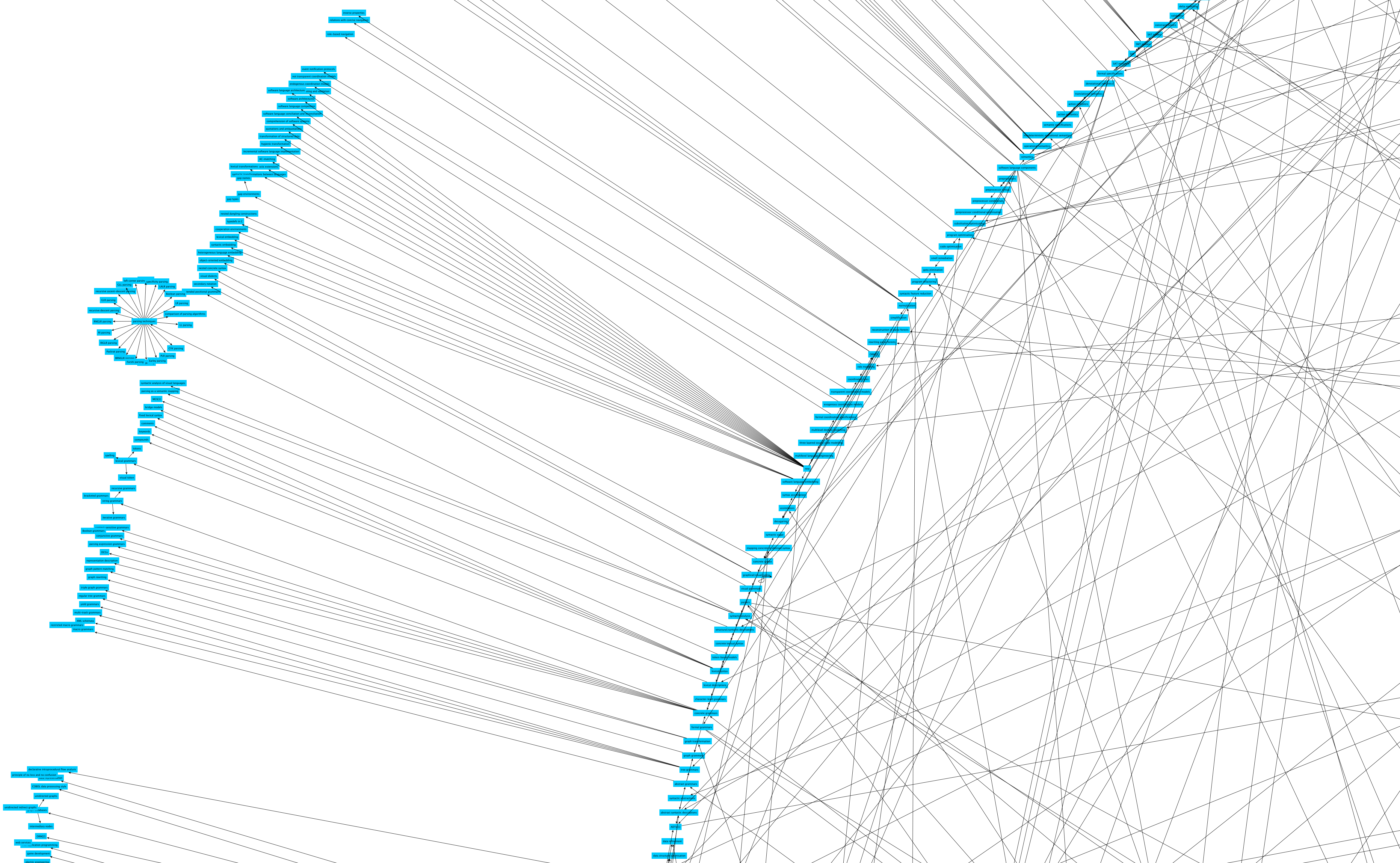
	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	Subtotal	Total
DBLP	20	25	26	22	22+9	20	19	17	24	25	23	20	21	15	24	20	20	18	390	390
Doctoral Symposium					-9														-9	381
Keynotes					+1		+1	+1				+1		+1		+2			+7	388
Posters						+2													+2	390
Special issues	+6				+7		+6			+2	+3						+11		+35	425
ATEM	+7																		+7	432

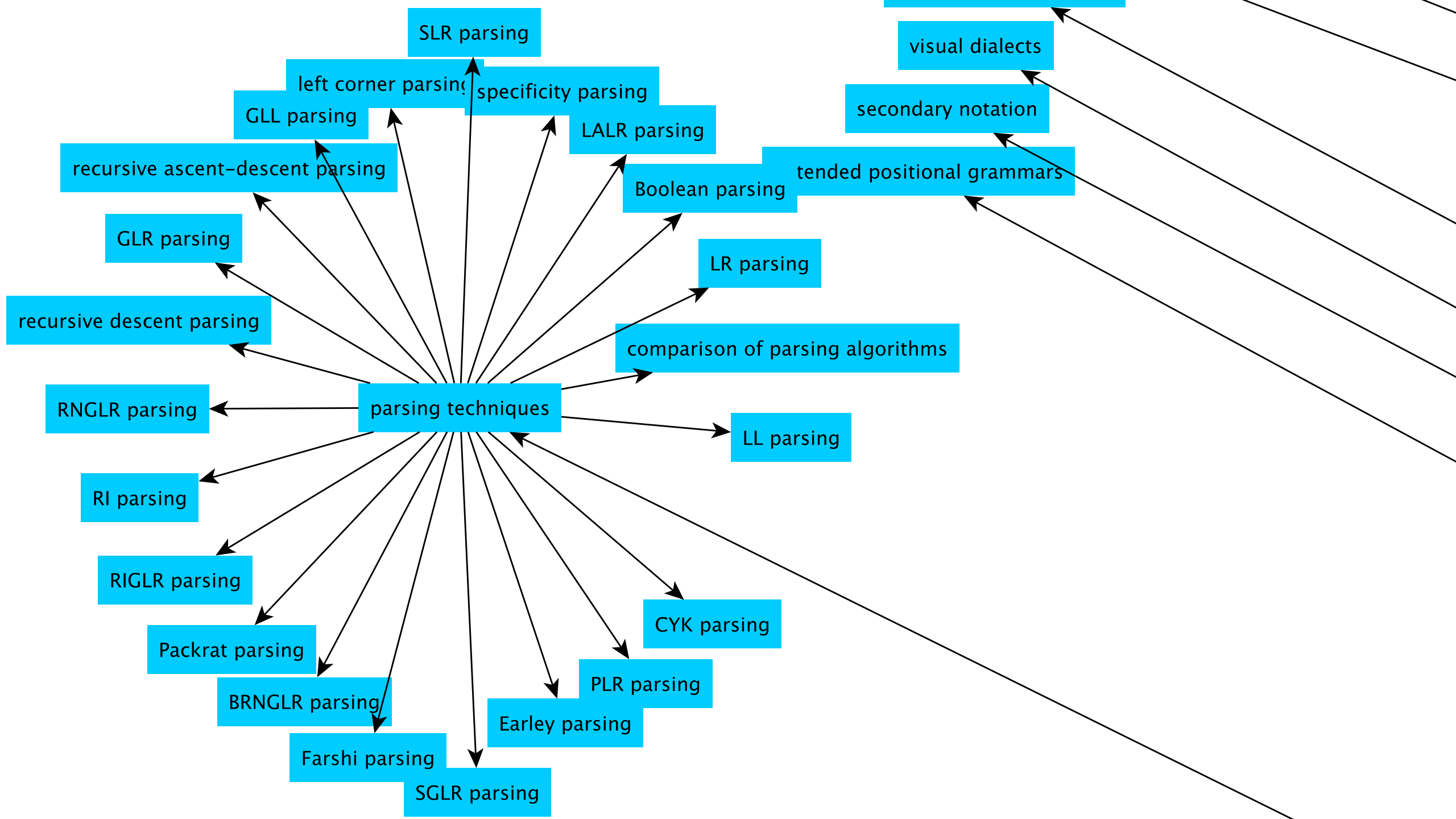
Year	Proc.		Keyn.	Special issue		Co-location
(2007)				7	IET Software	
2008	20	LNCS	●●	6	IEEE TSE	MoDELS
2009	25	LNCS	●●			MoDELS, GPCE
2010	26	LNCS	●●			FOSD, GPCE
2011	22	LNCS	●●			GTTSE
2012	23	LNCS	●●	7	SCP	FOSD, GPCE
2013	22	LNCS	●	6	COMLAN	SPLASH
2014	20	LNCS	●			ASE, GPCE
2015	18	ACM	●			SPLASH
2016	24	ACM	●			SPLASH
2017	25	ACM	●	2	JLVC/COLA	SPLASH
2018	23	ACM	●	3	COLA	SPLASH
2019	21	ACM	●			SPLASH
2020	21	ACM	–			SPLASH
2021	16	ACM	●●			SPLASH
2022	24	ACM	●			SPLASH
2023	22	ACM	●●			SPLASH
2024	20	ACM	●	11	JSS	SPLASH
2025	18	ACM	●●			STAF
2026	1	ACM	●			STAF

Body of Papers

“Planet SLE”




inverse properties
relations with concise navigation
role-based navigation
event notification protocols
not transparent coordination models
endogenous coordination models
software language architecture
software architectures
software language competition
software language conciliation and reconciliation
comprehension of software systems
quotations and antiquotations
transformation of structured data
hygienic transformation
incremental software language implementation
AC-matching
lexical transformations
syntactic transformations between languages
gap names
gap environments
gap types
nested dangling constructions
typedefs in C
cooperation environments
lexical embedding
syntactic embedding
heterogeneous language embedding
object-oriented embedding
nested concrete syntax
visual dialects
secondary notation
parsing techniques
left corner parsing
specificity parsing
GLL parsing
LALR parsing
recursive ascent-descent parsing
Boolean parsing
GLR parsing
LR parsing
recursive descent parsing
comparison of parsing algorithms
RNGLR parsing
LL parsing
RI parsing
RIGLR parsing
CYK parsing
Packrat parsing
PLR parsing
BRNGLR parsing
Farshi parsing
Earley parsing
syntactic analysis of visual languages
parsing as a semantic mapping
MGS[1]
bridge models
fixed lexical syntax
comments
keywords
compounds
tokens
spelling
lexical grammars
visual token
recursive grammars
bracketed grammars
string grammars
iterative grammars
Boolean grammars
conjunctive grammars
parsing expression grammars
DCGs
representation description
graph pattern matching
graph rewriting
triple graph grammars
regular tree grammars
yield grammars
multi-track grammars
XML schemata
restricted macro grammars
macro grammars
delta modelling
validators
constraint solvers
SAT solvers
SMT provers
SAT
SAT encoders
formal specifications
denotational semantics
translational semantics
action compilers
action semantics
semantic specifications
nondeterministic operational semantics
operational semantics
semantics
software language components
preprocessors
preprocessor syntax
preprocessor conditionals
preprocessor conditional optimisation
substitution optimisation
program optimisation
code optimisation
smell remediation
goto elimination
program refactoring
syntactic feature reduction
normalisation
simplification
reconstruction of parse forests
rewriting parse forests
ORM[1]
role modelling
coordinated roles
transparent coordination models
exogenous coordination models
formal coordination specifications
multilevel domain modelling
three layered source code modelling
multilevel language engineering
new
software language embedding
syntax assimilation
assimilators
desugaring
syntactic sugar
mapping concrete to abstract syntax
concrete syntax
graphical/visual syntax
visual grammars
parsers
syntactic analysis
structural/syntactic descriptions
concrete textual syntax
token-based models
lexical syntax
lexical descriptions
character-level grammars
concrete grammars
formal grammars
graph transformation
graph grammars
tree grammars
abstract grammars
syntactic abstractions
abstract syntactic descriptions
ADTs[1]
data refinement
data structure optimisation
declarative intraprocedural flow analysis
principle of no loss and no confusion
COBOL data processing style
unidirected graphs
unidirected indirect graphs
intermediary nodes
ORM[2]
web services
game development
electric engineering


SLR parsing
left corner parsing
specificity parsing
GLL parsing
LALR parsing
recursive ascent-descent parsing
Boolean parsing
GLR parsing
LR parsing
recursive descent parsing
comparison of parsing algorithms
RNGLR parsing
parsing techniques
LL parsing
RI parsing
RIGLR parsing
CYK parsing
Packrat parsing
PLR parsing
BRNGLR parsing
Earley parsing
Farshi parsing
SGLR parsing
visual dialects
secondary notation
tended positional grammars

T1: Defining a Software Language

- **T1A Design** 🟢🟠
  - CfP: 88%|100% BoP: 84%|94%
- **T1B Static Semantics** ⚫🔴
  - 68|84 72|89
- **T1C Behavioural Semantics** ⚫⚫
  - 56|58 76|83
- **T1D Composition** 🟢🟢
  - 88|100 92|100
- **T1E Reuse** ⚫⚫
  - 60|68 60|56

T2: Software Languages over Time

- T2A Lifecycle ⚫⚫ 40|32 32|33
- T2B Evolution 🟢⚫ 96|100 76|83
- T2C Variability ⚫⚫ 56|68 60|72
- T2D Deployment 🔴⚫ 80|89 28|39
- T2E Documentation ⚫⚫ 32|32 20|22

T3: Software Language Kinds

- T3A Meta-languages 🔴🟠 72|89 92|89
- T3B Transformation Languages 🔴🔴 72|89 76|89
- T3C DSLs 🟢🟢 96|100 96|100
- T3D GPLs ⚫⚫ 44|32 48|61
- T3E Ontologies ⚫⚫ 52|42 32|33
- T3F API ⚫⚫ 24|21 36|39

T4: Software Language Tooling

- T4A Workbenches 🟢🟢 84|100 100|100
- T4B Horizontal Transformation 🟢🟢 96|100 80|100
- T4C Vertical Transformation 🔴🟢 80|89 84|100
- T4D Interpretation ⚫🟠 16|5 80|94
- T4E Simulation ⚫⚫ 68|79 12|17
- T4F Visualisation ⚫⚫ 12|16 60|78
- T4G AI-for-SLE ⚫⚫ 8|11 32|39

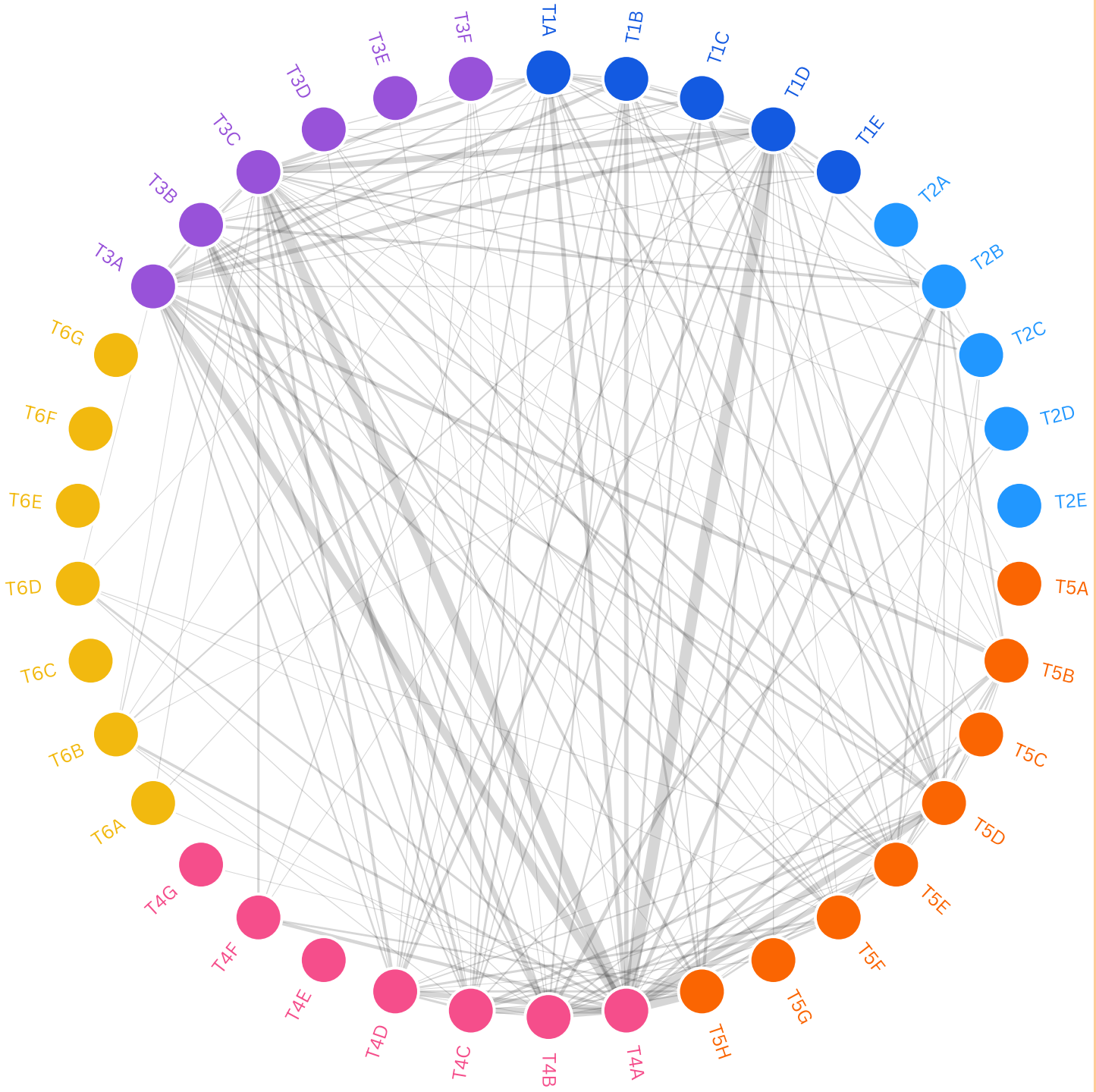
T5: Quality Assurance

- T5A Requirements ⚫⚫ 44|58 28|28
- T5B Language Description Analysis ⚫⚫ 48|47 68|67
- T5C Testing 🟠⚫ 84|89 52|61
- T5D Formal Methods 🟢🟢 96|100 88|100
- T5E Empirical Evaluation 🔴⚫ 80|89 76|83
- T5F Performance ⚫⚫ 56|58 60|61
- T5G Traceability ⚫⚫ 48|58 28|33
- T5H Usability ⚫🔴 60|68 76|89

T6: Software Language Context

- T6A Experience Reports ●● 68|79 28|39
- T6B Industrial ●● 56|74 48|67
- T6C SLE-for-AI ●● 20|26 4|6
- T6D Synergies ●● 64|63 64|78
- T6E Quantum ●● 20|26 0|0
- T6F CPS/IoT/DT ●● 20|26 20|22
- T6G Socio-technical ●● 20|26 12|17

- T1A Design
- T1B Static Semantics
- T1C Behavioural Semantics
- T1D Composition
- T1E Reuse
- T2A Lifecycle
- T2B Evolution
- T2C Variability
- T2D Deployment
- T2E Documentation
- T3A Meta-languages
- T3B Transformation Languages
- T3C DSLs
- T3D GPLs
- T3E Ontologies
- T3F API
- T4A Workbenches
- T4B Horizontal Transformation
- T4C Vertical Transformation
- T4D Interpretation
- T4E Simulation
- T4F Visualisation
- T4G AI-for-SLE
- T5A Requirements
- T5B Language Description Analysis
- T5C Testing
- T5D Formal Methods
- T5E Empirical Evaluation
- T5F Performance
- T5G Traceability
- T5H Usability
- T6A Experience Reports
- T6B Industrial
- T6C SLE-for-AI
- T6D Synergies
- T6E Quantum
- T6F CPS/IoT/DT
- T6G Socio-technical



Topic T4C: Vertical Transformation

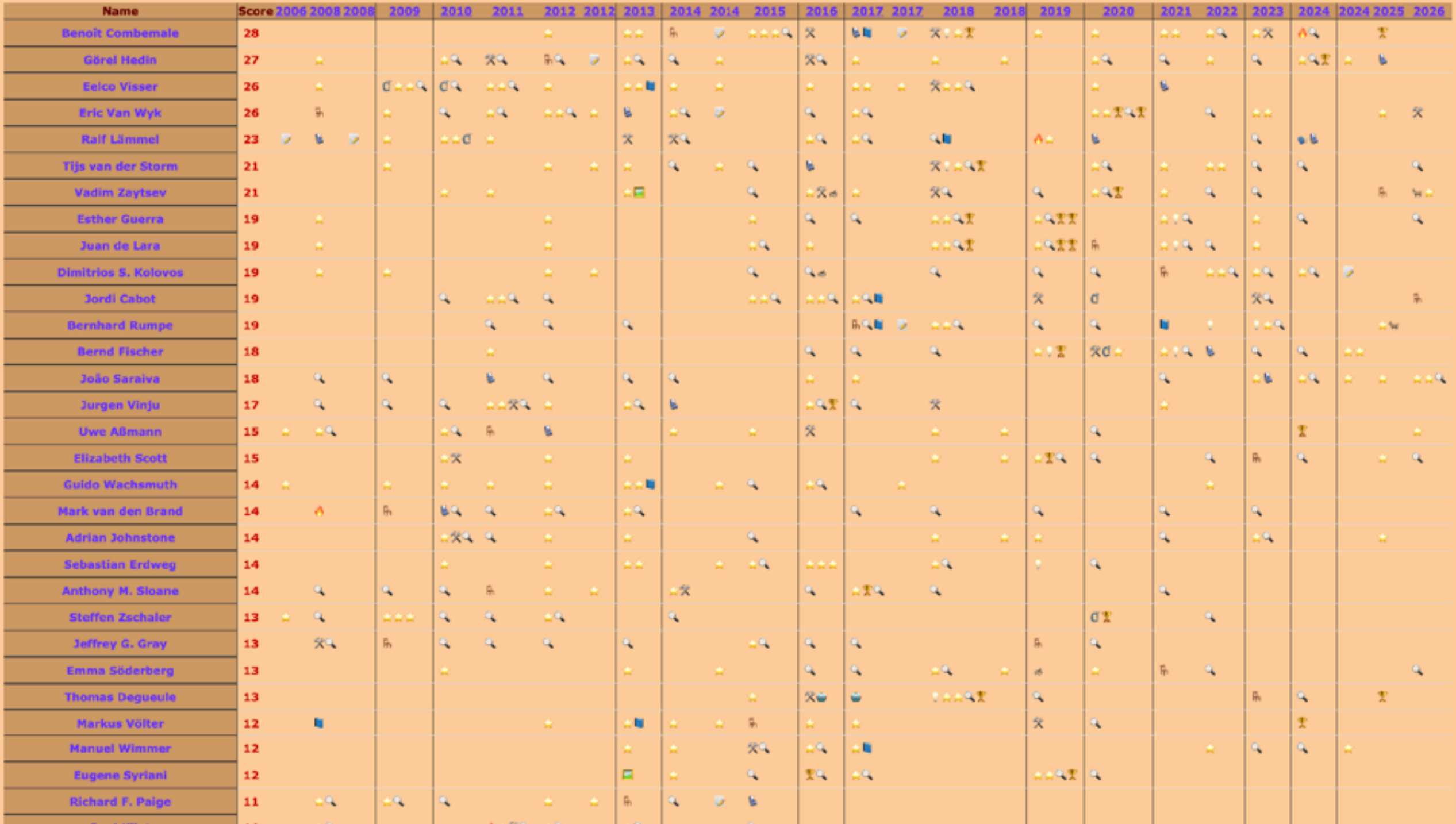
Generative + Compilation + Refinement + Recovery. Work about shifting abstraction levels downwards: compilation, code generation, model-to-text transformations, refinement, synthesis, lowering, as well as extraction, recovery, reverse engineering techniques that raise abstraction. If the key novelty is the transformation language itself rather than the pipeline or the implementation, we consider T3B.

Summary

- Requested in **20/26** calls
- Requested in **17/19** pure calls (2008–2026 with the longest streak of **15** years)
- Primary tagged in **11/26** paper bundles
- Primary tagged in **9/19** pure proceedings (2010–2026 with the longest streak of **5** years)
- Secondary tagged in **19/26** paper bundles
- Secondary tagged in **18/19** pure proceedings (2008–2025 with the longest streak of **18** years)
- Top 3 co-occurring topics:
  - T4A: Workbenches (**26** times)
  - T3C: DSLs (**13** times)
  - T5D: Formal Methods (**13** times)
- Top 3 contributors:
  - Bernd Fischer (**5** times)
  - Adrian Johnstone (**4** times)
  - Elizabeth Scott (**4** times)

List of papers (63)

https://cfpbok.github.io/tag/t4c.html


Name
Score
2006
2008
2008
2009
2010
2011
2012
2012
2013
2014
2014
2015
2016
2017
2017
2018
2018
2019
2020
2021
2022
2023
2024
2024
2025
2026
Benoit Combemale 28
Görel Hedin 27
Eelco Visser 26
Eric Van Wyk 26
Ralf Lämmel 23
Tijs van der Storm 21
Vadim Zaytsev 21
Esther Guerra 19
Juan de Lara 19
Dimitrios S. Kolovos 19
Jordi Cabot 19
Bernhard Rumpe 19
Bernd Fischer 18
João Saraiva 18
Jurgen Vinju 17
Uwe Aßmann 15
Elizabeth Scott 15
Guido Wachsmuth 14
Mark van den Brand 14
Adrian Johnstone 14
Sebastian Erdweg 14
Anthony M. Sloane 14
Steffen Zschaler 13
Jeffrey G. Gray 13
Emma Söderberg 13
Thomas Degueule 13
Markus Völter 12
Manuel Wimmer 12
Eugene Syriani 12
Richard F. Paige 11

What now?

- Stable identity core for **SLEBoK**
  - https://slebok.github.io/cfpbok
- Confirmation that SLE is cross-* and inter-*
- Broader CfPs ≠ broader paper coverage
- **Predecessors**? (LDTA, ASF+SDF, ATEM, WAGA, GTTSE)
- **Neighbours**? (GPCE, PLDI, MoDELS, OOPSLA, ICSE)
- Forward/backward references?
- Questions?

UNIVERSITY OF TWENTE. | Formal Methods & Tools