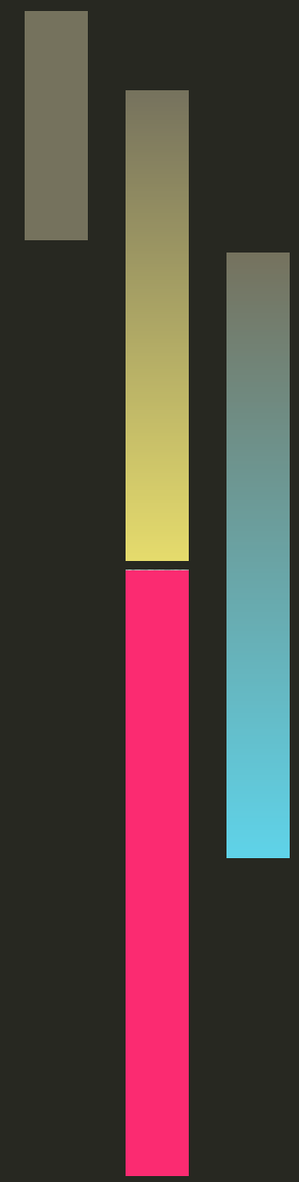
Reverse Engineering

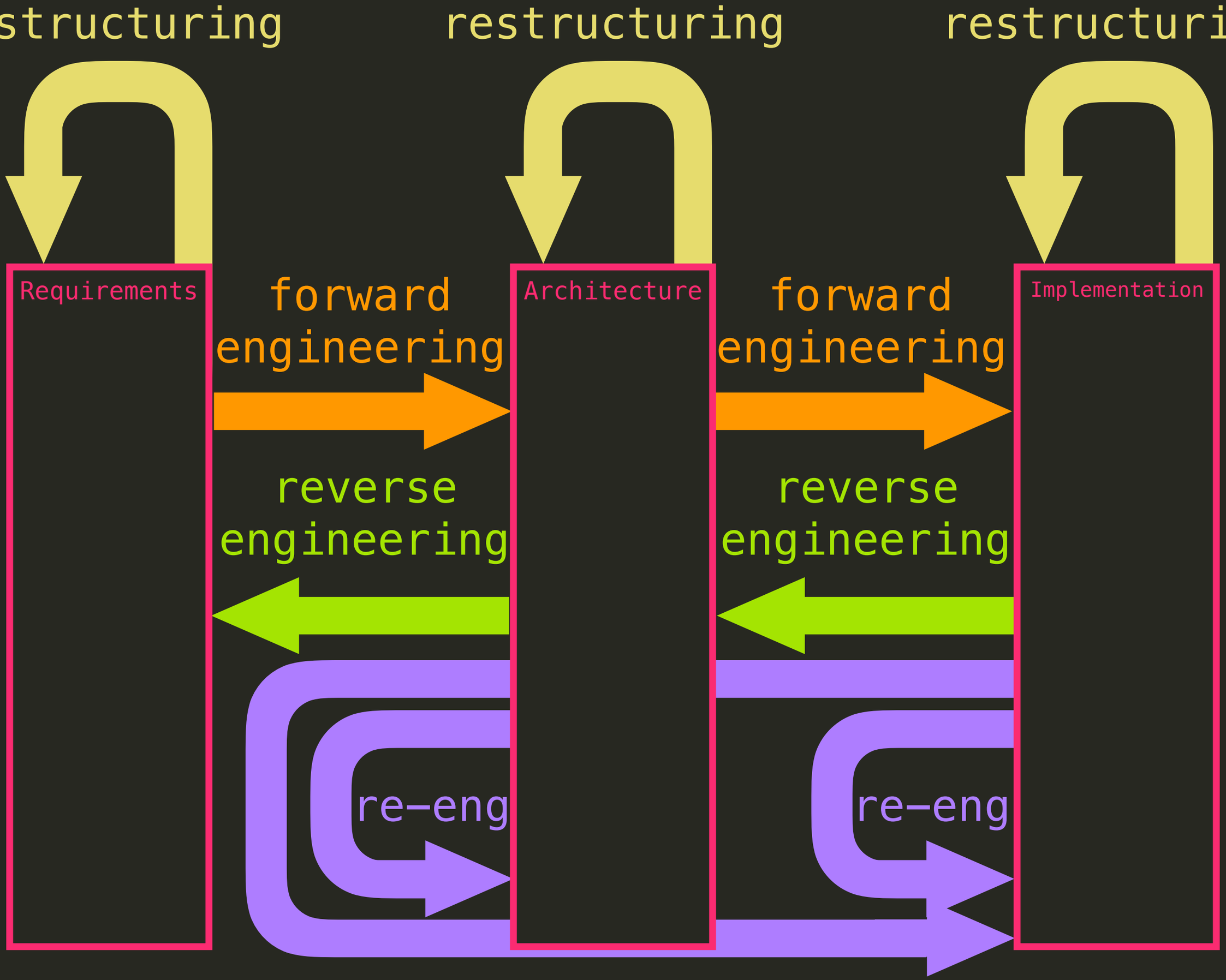
Dr. Vadim Zaytsev aka @grammarware

UvA, MSc SE, 9 November 2015

Roadmap

W44	Introduction	V.Zaytsev
W45	Metaprogramming	J.Vinju
W46	Reverse Engineering	V.Zaytsev
W47	Software Analytics	M.Bruntink
W48	Clone Management	M.Bruntink
W49	Source Code Manipulation	V.Zaytsev
W50	Legacy and Renovation	TBA
W51	Conclusion	V.Zaytsev





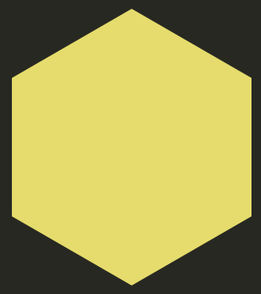
E.Chikofsky, J.H.Cross II, Reverse Engineering and Design Discovery: A Taxonomy. IEEE Software 7:1, 1990.

Objectives of reverse engineering

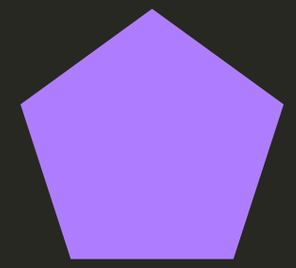
* Cope with complexity
* Generate alternate views
* Recover lost info
* Detect side effects
* Synthesise higher abstractions
* Facilitate reuse

E.Chikofsky, J.H.Cross II, Reverse Engineering and Design Discovery: A Taxonomy. IEEE Software 7:1, 1990.

Code Reverse Engineering



Code reverse engineering

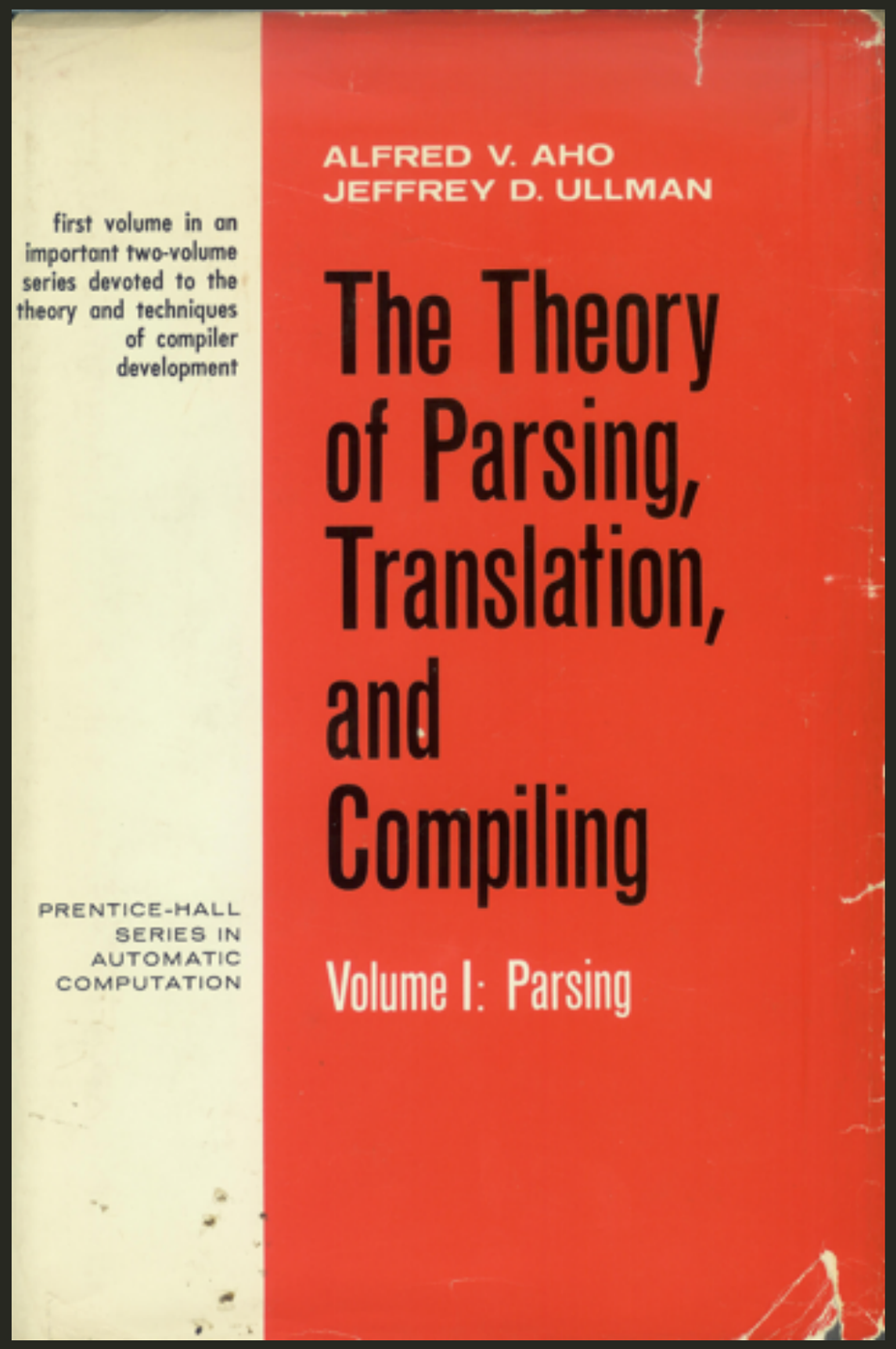


* Parsing
* Fact extraction
* Slicing
* Pattern matching
* Decomposition
* Exploration

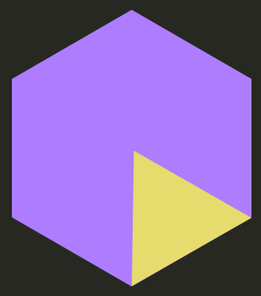
H.A.Müller, J.H.Jahnke, D.B.Smith, M.-A.Storey, S.R.Tilley, K.Wong, Reverse Engineering: A Roadmap, ICSE 2000. http://bibtex.github.io/ICSE-2000-Future-MullerJSSTW.html

Parsing

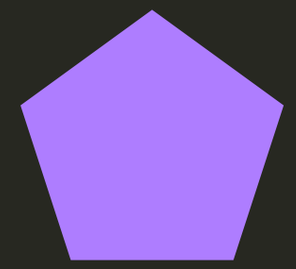
* Well-developed since…
* Recognising structure
  * text → tree
  * parse tree → AST
  * forest disambiguation
  * tokens → list
  * image → visual model



A.V. Aho & J.D. Ullman, The Theory of Parsing, Translation and Compiling, 1972.
V.Zaytsev, A.H.Bagge, Parsing in a Broad Sense, MoDELS 2014.

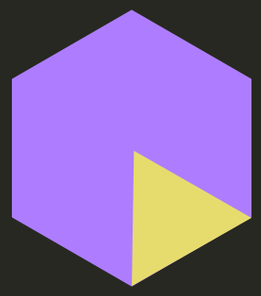


↑ Parsing

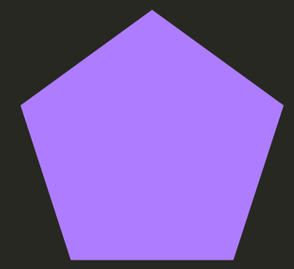


* Reduce the input back to the start symbol
* Recognise terminals
* Replace terminals by nonterminals
* Replace terminals and nonterminals by lhs

* LR(1) ::= yacc | Beaver | Eli | SableCC | Irony;
* GLR ::= bison | DMS | GDK | Tom;
* SGLR ::= ASF+SDF | Spoofax | Stratego;

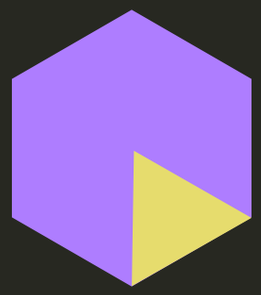


↓ Parsing

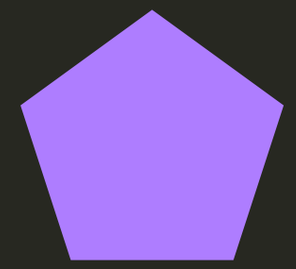


* Imitate production by rederivation
* Each nonterminal is a goal
* Replace each goal by subgoals
* Parse tree is built from top to bottom

* LL(k) ::= JavaCC;
* Earley ::= Marpa | ModelCC;
* GLL ::= Rascal | gll-combinators;
* Packrat ::= Rats! | OMeta | PetitParser;
* LL(*) ::= ANTLR | TXL;
* DCG ::= Prolog;

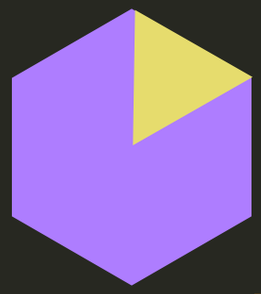


Semiparsing

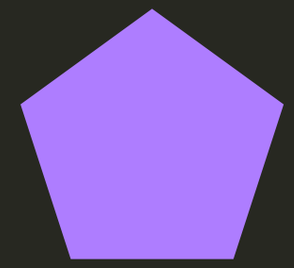


* grep
* anchor terminals
* islands & noise
* skeleton grammars
* relaxation & robustness
* multilanguage

V.Zaytsev, Formal Foundations for Semi-Parsing, CSMR-WCRE ERA, 2014
http://bibtex.github.io/CSMR-WCRE-2014-Zaytsev.html



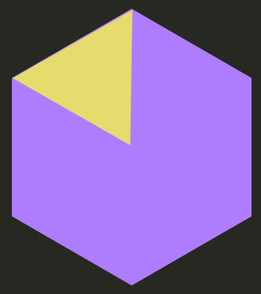
Fact extraction



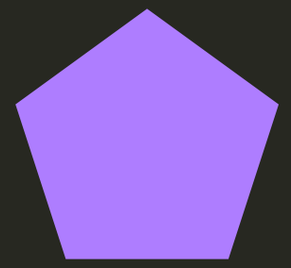
= parsing + generating a factbase
(or, sequence of graph transformations)

* e.g., metrics
* Can be language-parametric!
* Schema
  * describes form of the data
  * ASG = Abstract Semantic Graph
  * call graph
  * dependence graph
  * relations

Y.Lin, R.C.Holt, Formalizing Fact Extraction, ATEM 2003.
http://bibtex.github.io/ATEM-2003-LinH04.html

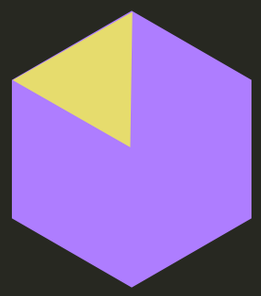


Slicing

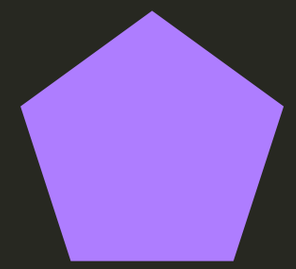


```
read(text);
read(n);
lines = 1;
chars = 1;
subtext = "";
c = getChar(text);
while (c != ‘\eof’)
      if (c == ‘\n’)
      then lines = lines + 1;
           chars = chars + 1;
      else chars = chars + 1;
           if (n != 0)
           then subtext = subtext ++ c;
                n = n – 1;
      c = getChar(text);
write(lines);
write(chars);
write(subtext);
```

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

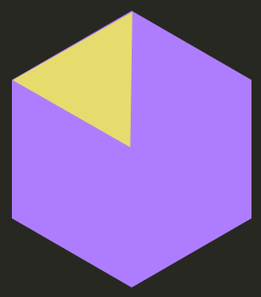


Slicing

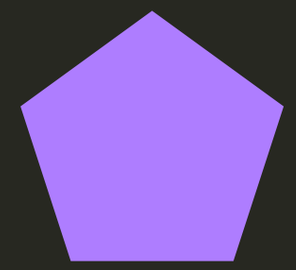


* Forward/backward slicing
* Dynamic/conditioned slicing
  * constraints on input
* Chopping
  * discover connection between I & O
* Amorphous slicing
* . . .

J. Silva, A Vocabulary of Program Slicing-Based Techniques, CSUR, 2012.

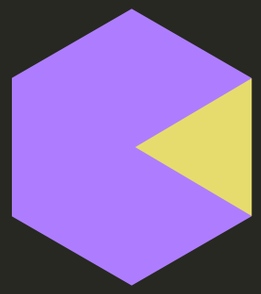


Slicing

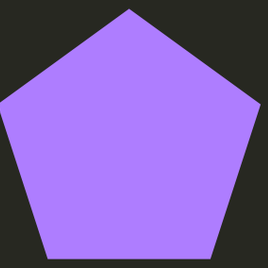


* Debugging
  * cf. Weiser CACM 1982
* Cohesion measurement
  * cf. Ott&Bieman IST 1998
* Comprehension
  * cf. De Lucia&Fasolino&Munro IWPC 1996
* Maintenance
  * e.g. reuse
* Re-engineering
  * e.g. clone detection

http://www0.cs.ucl.ac.uk/staff/mharman/sf.html

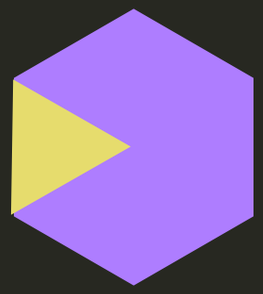


Pattern matching

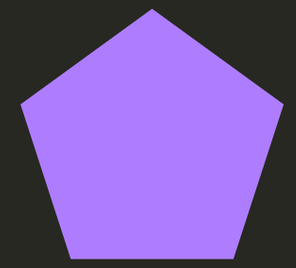


* Easy to formulate on ADTs
* In Rascal:
  * visit(){case}
  * := and !:=
  * functions
* Need traversal strategies
  * depth-first (pre-, in-, post-order)
  * breadth-first
  * topdown, bottomup, downup
  * innermost, outermost
  * . . .

E.Visser, Z.Benaissa, A.P.Tolmach, Building Program Optimizers with Rewriting Strategies, ICFP 1998.
http://bibtex.github.io/ICFP-1998-VisserBT.html



Decomposition



* Recall partitioning & equiv. classes
* Simplest form: modularisation
* Usually: some graph + SCCs
  * Given granularity
    * make a valid decomposition
    * maximising benefit
  * Applicable to packages, build targets, automata, tasks, formulae, processes, rels…

M.Vakilian, R.Sauciuc, J.D.Morgenthaler, V.Mirrokni, Automated Decomposition of Build Targets, ICSE 2015
http://bibtex.github.io/ICSE-v1-2015-VakilianSMM.html

Exploration

Software visualisation

Algorithm visualisation

Static algorithm visualisation

Algorithm animation

Program visualisation

Data animation

Static code visualisation

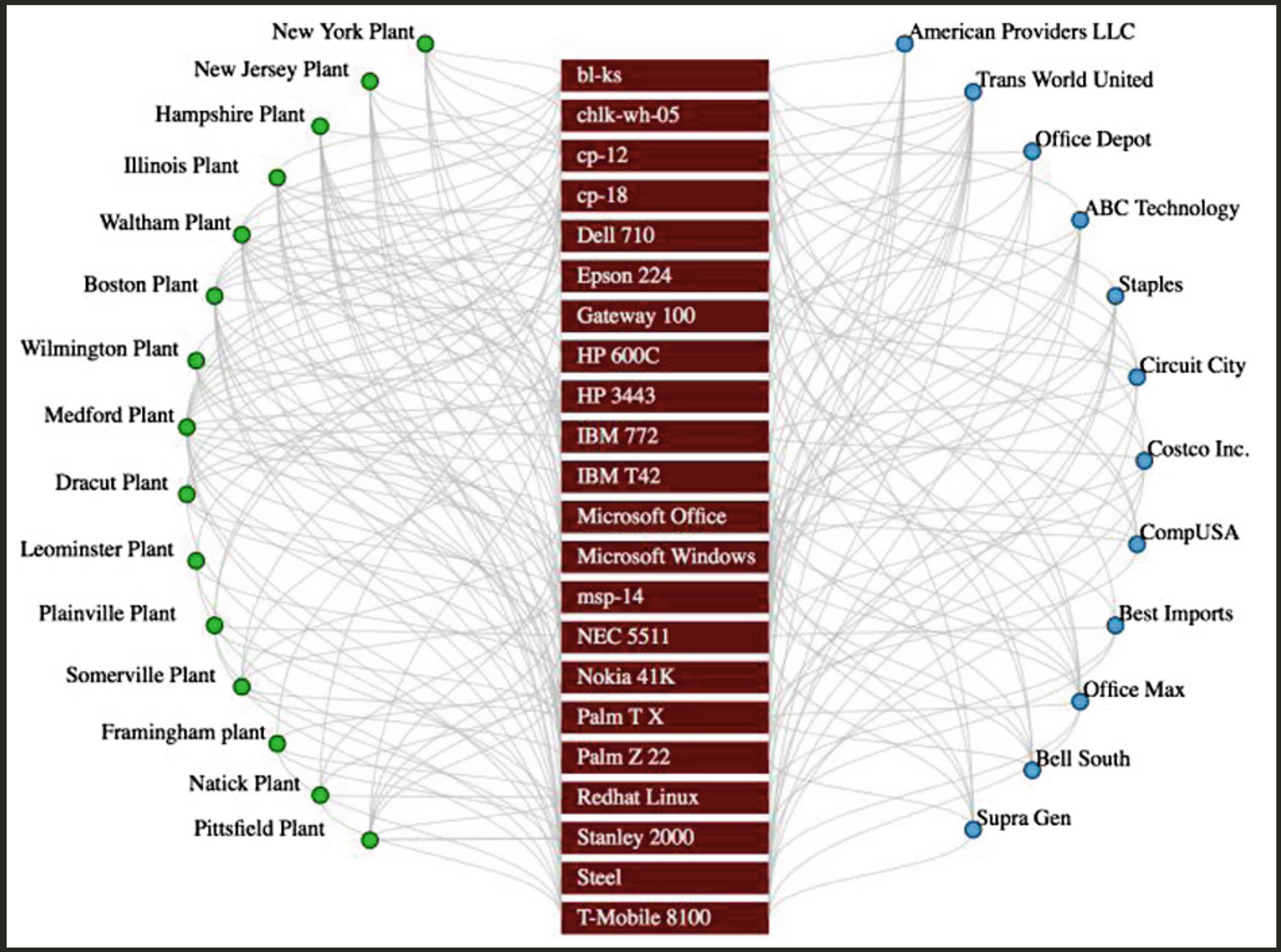
Static data visualisation

Visual programming

Code animation

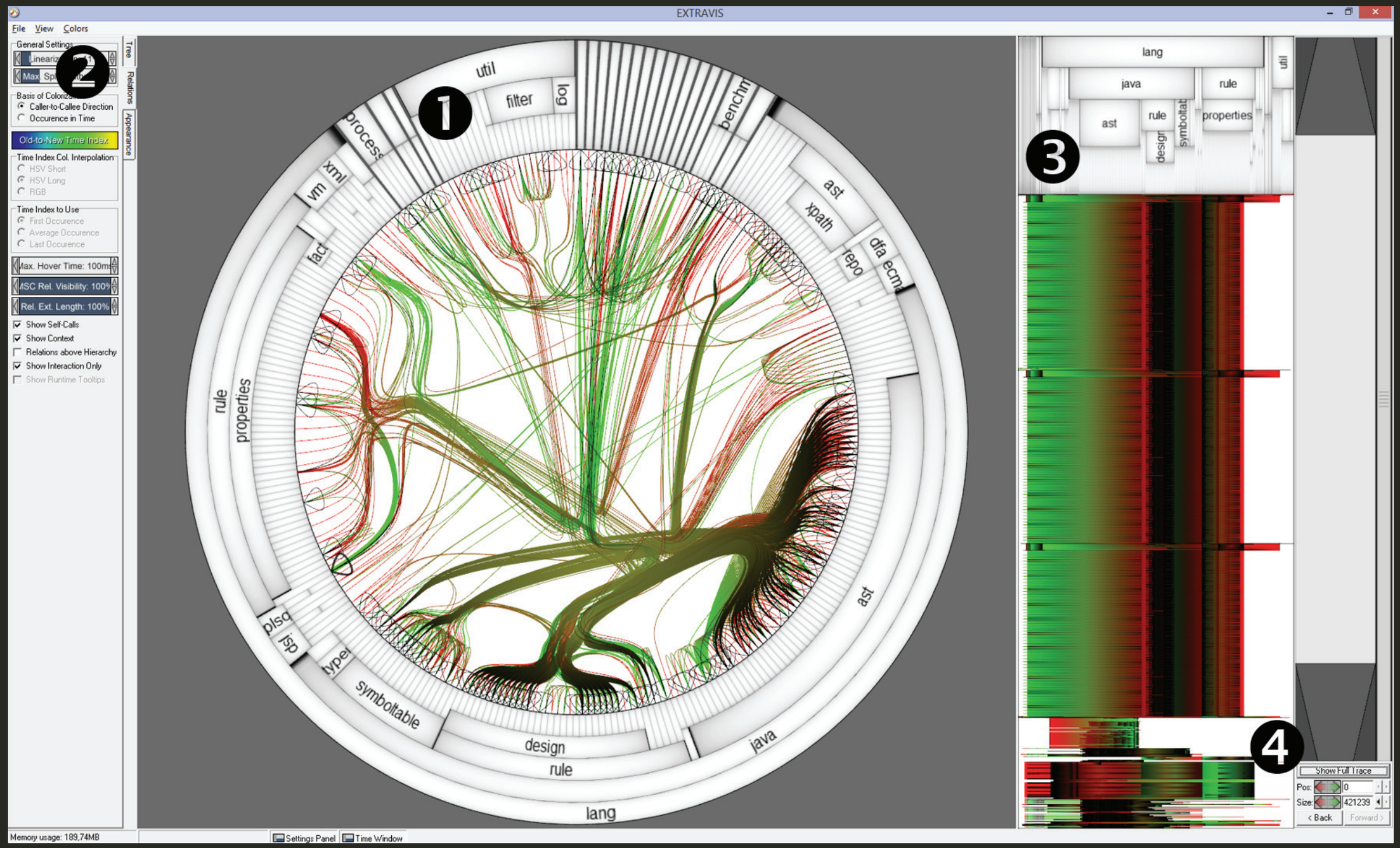
B.A.Price, R.M.Baecker, I.S.Small, A Principled Taxonomy of Software Visualization, JVLC 1993

Visualisation



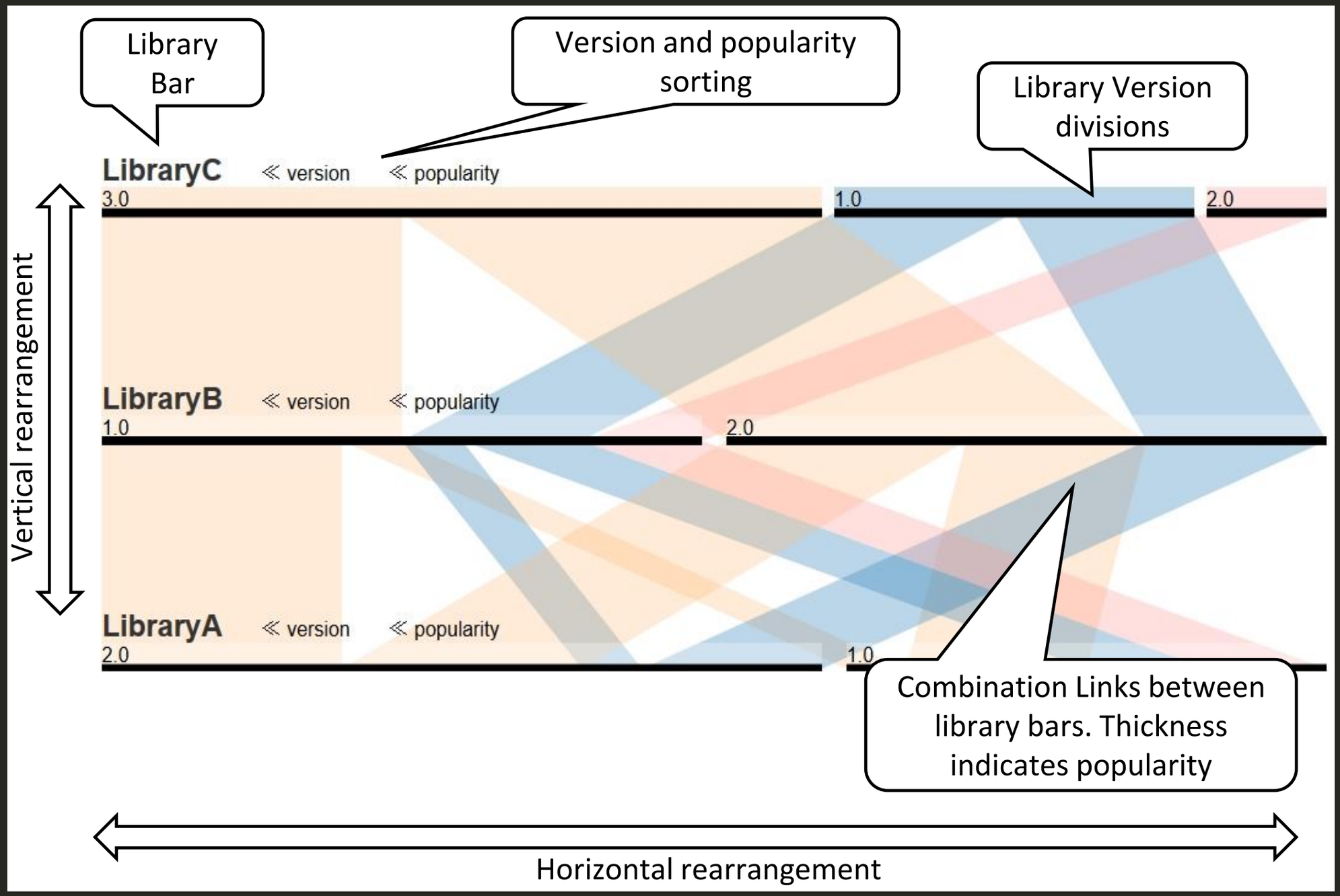
T.Babaian, W.T.Lucas, M.Li, Modernizing Exploration and Navigation in Enterprise Systems with Interactive Visualizations, HCI 2015. http://bibtex.github.io/HIMI-IKD-2015-BabaianLL.html

Trace vis



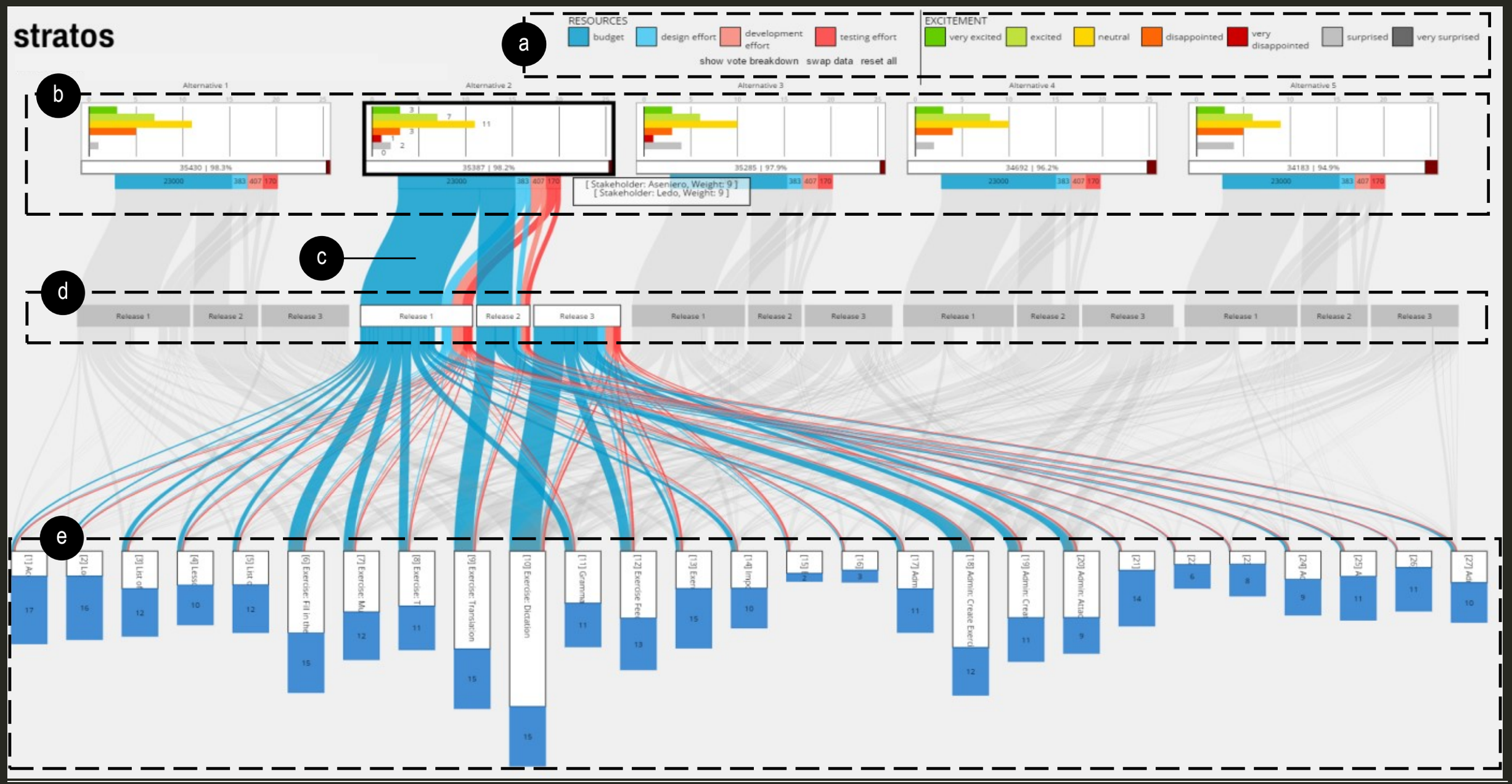
F.Fittkau, S.Finke, W.Hasselbring, J.Waller, Comparing trace visualizations for program comprehension through controlled experiments, ICPC 2015. http://bibtex.github.io/ICPC-2015-FittkauFHW.html

Versioning vis



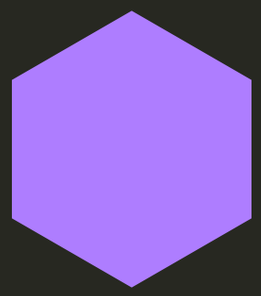
Y.Yano, R.G.Kula, T.Ishio, K.Inoue, VerXCombo: an interactive data visualization of popular library version combinations, ICPC 2015. http://bibtex.github.io/ICPC-2015-YanoKII.html

Release vis



B.A.Aseniero, T.Wun, D.Ledo, G.Ruhe, A.Tang, S.Carpendale, STRATOS: Using Visualization to Support Decisions in Strategic Software Release Planning, CHI 2015. http://bibtex.github.io/CHI-2015-AsenieroWLRTC.html

Data Reverse Engineering



Data reverse engineering

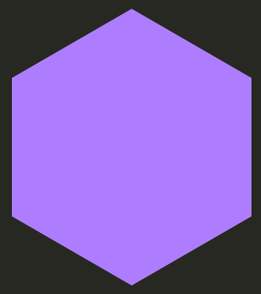
* Database design recovery
* Pattern recognition
* Information retrieval
* Clustering
* Mining unstructured data

H.A.Müller, J.H.Jahnke, D.B.Smith, M.-A.Storey, S.R.Tilley, K.Wong, Reverse Engineering: A Roadmap, ICSE 2000. http://bibtex.github.io/ICSE-2000-Future-MullerJSSTW.html

Database design recovery

* Forward database engineering
  * Conceptual design
  * Logical design
    * Simplification
    * Optimisation
    * Translation
  * Physical design
  * View design

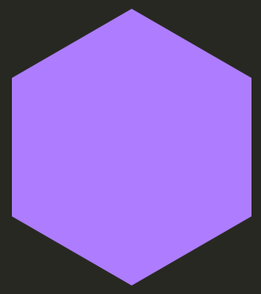
J.-L.Hainaut, J.Henrard, J.-M.Hick, D.Roland, V.Englebert, Database Design Recovery, CAiSE, 1996.
http://bibtex.github.io/CAiSE-1996-HainautHHRE.html



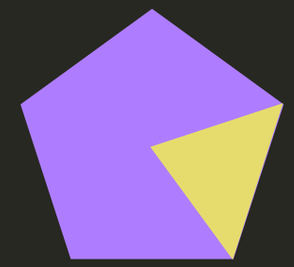
Database design recovery

* Data structure extraction
  * Program analysis
  * Data analysis
    * Schema integration
* Data structure conceptualisation
  * Untranslation
  * Deoptimisation
  * Conceptual normalisation

J.-L.Hainaut, J.Henrard, D.Roland, V.Englebert, J.-M.Hick, Structure Elicitation in Database Reverse Engineering, WCRE 1996
http://bibtex.github.io/WCRE-1996-HainautHREH.html



Pattern recognition



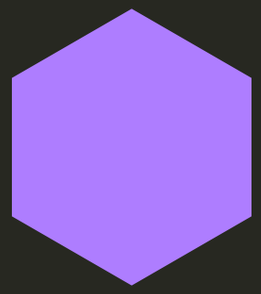
* Pattern = feature vector
* Quantitative features
  * continuous / discrete / interval
* Qualitative features
  * nominal / ordinal
* Find most descriptive/discriminatory

K.C.Gowda, E.Diday, Symbolic clustering using a new dissimilarity measure. IEEE TSMC 22, 1992.

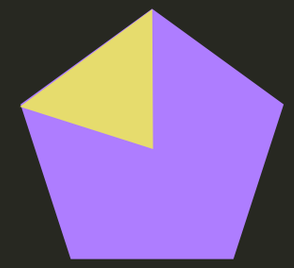
Information retrieval

- Knowledge discovery
- Data mining
- Usually statistical methods
  - = require training
- WEKA = Waikato Environment for Knowledge Analysis
  - Java, 1992–2015
  - http://www.cs.waikato.ac.nz/ml/weka/
  - good with Groovy, Scala, Jython…

M.Hall, E.Frank, G.Holmes, B.Pfahringer, P.Reutemann, I.H.Witten, The WEKA data mining software: an update, SIGKDD Explorations Newsletter 11:1, 2009.

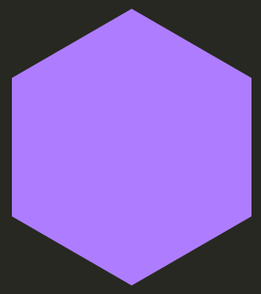


Clustering



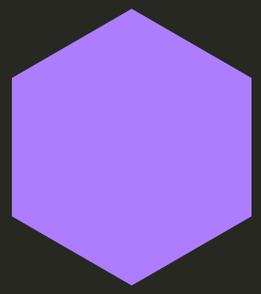
* Pattern recognition & representation
  * similarity/proximity measure
    * Minkowski / edit / statistical
  * Clustering techniques
    * hierarchical / partitional
    * agglomerative / divisive
    * hard / fuzzy
    * incremental / non-incremental
  * Dendrograms

A.K.Jain, M.N.Murty, P.J.Flynn, Data clustering: a review, CSUR 31:3, 1999.



MUD

* Mixture
  * natural language text
  * technical artefacts
* Unstructured data
  * dev communication
  * issue reports
  * documentation
  * meeting notes



MUD

* Can fish for
  * code fragments
  * class names
  * stack traces
  * patches
  * jargon
* State of the art
  * heuristic-based idiosyncratic tools

N.Bettenburg, B.Adams, A.E.Hassan, M.Smidt, A Lightweight Approach to Uncover Technical Artifacts in Unstructured Data, ICPC 2011. http://bibtex.github.io/ICPC-2011-BettenburgAHS.html

Conclusion

* Besides forward engineering
  * there is reverse engineering
* Software comprehension
  * Code reverse engineering
    * parsing, slicing, matching, visualising
  * Data reverse engineering
    * design recovery, PR, IR, clustering, MUD
* Mature yet active field