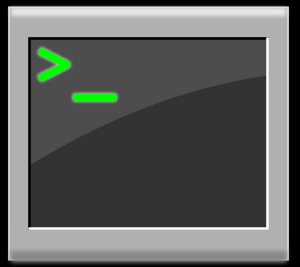
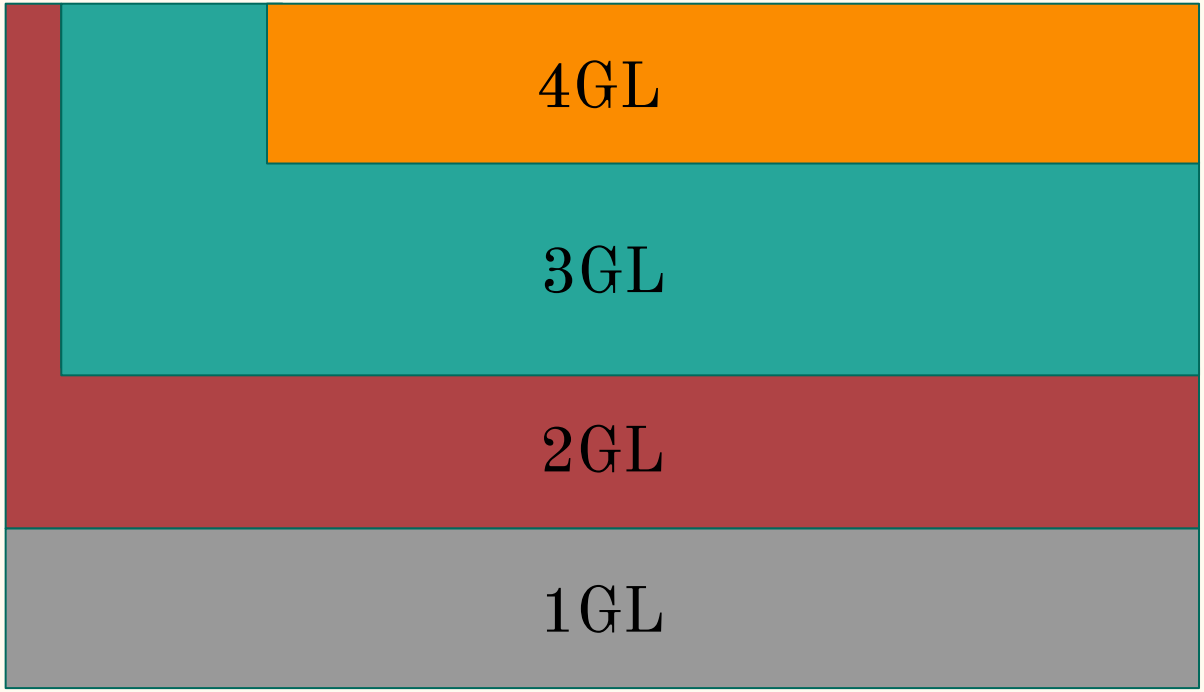
Incremental Coverage of Legacy Software Languages

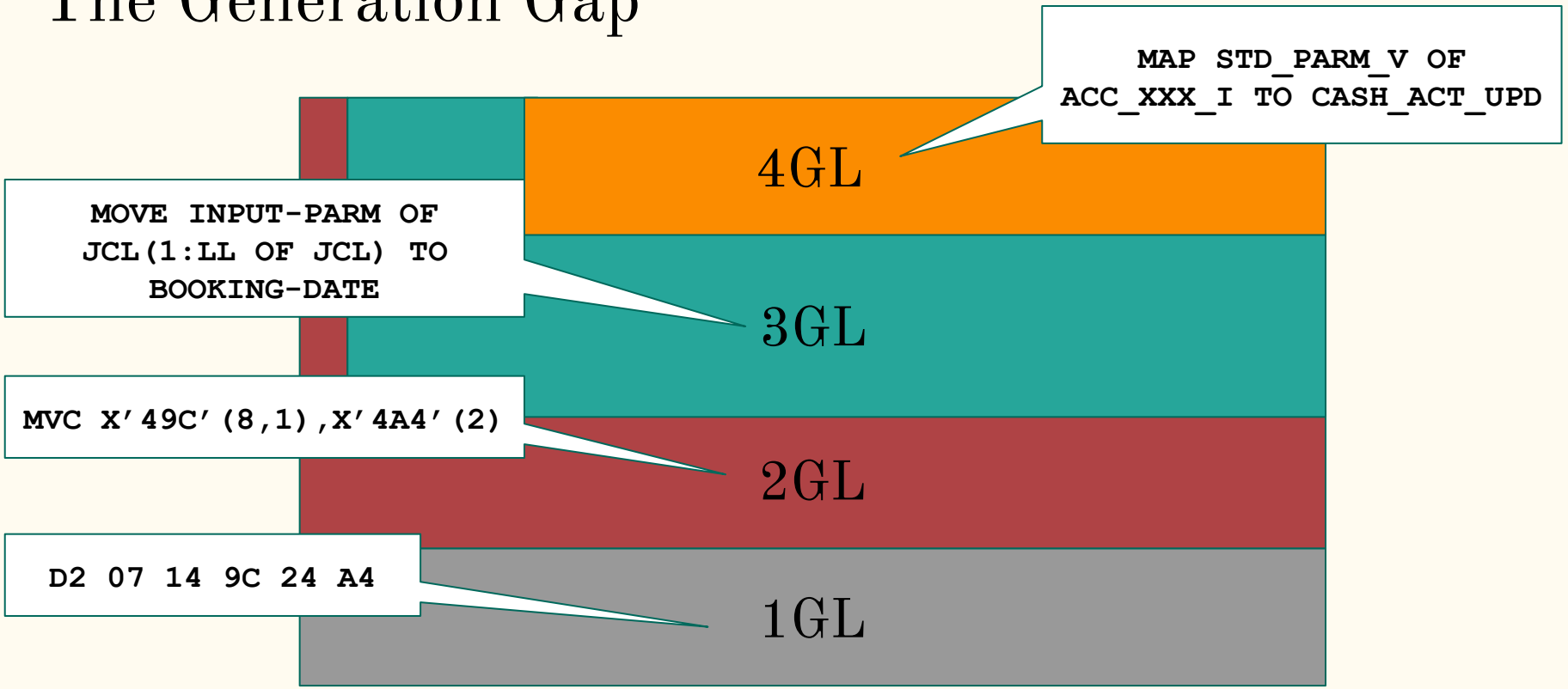
V. Zaytsev @ PX/17.2 @ SPLASH 2017



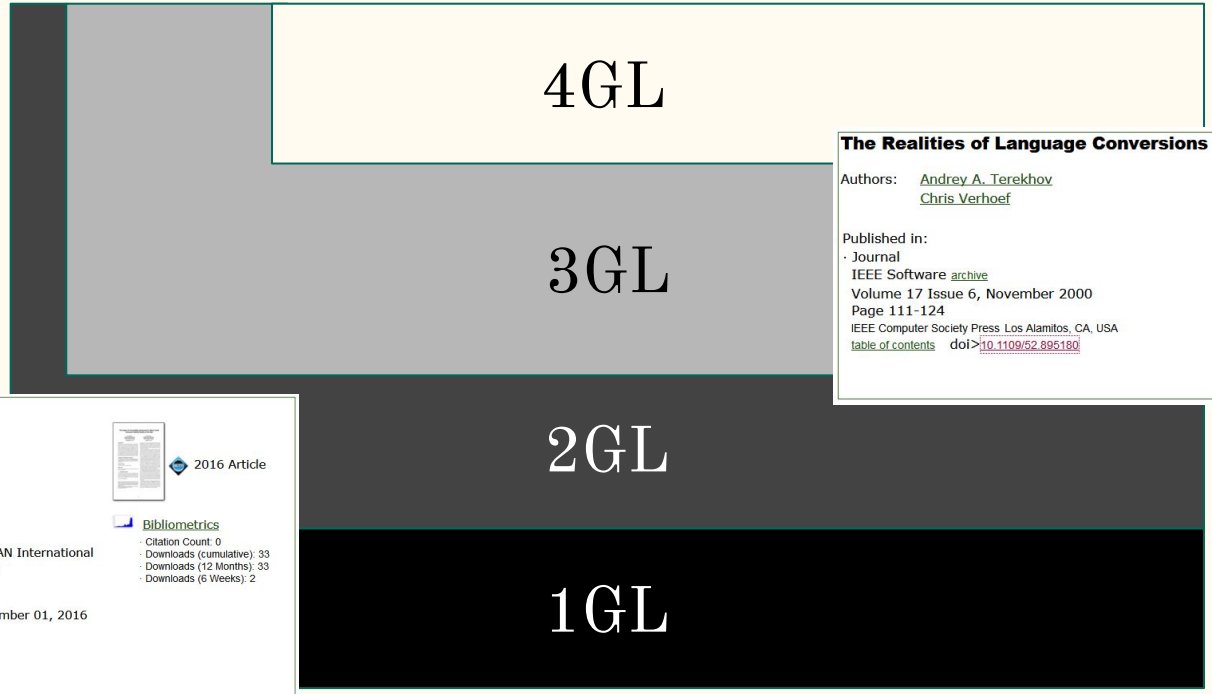
The Generation Gap



The Generation Gap



Language Migration



The Realities of Language Conversions

Authors: Andrey A. Terekhov
Chris Verhoef

Published in:
· Journal
IEEE Software archive
Volume 17 Issue 6, November 2000
Page 111-124
IEEE Computer Society Press Los Alamitos, CA, USA
table of contents doi>10.1109/52.895180

2000 Article
• orig-researc

Bibliometrics
· Citation Count: 13
· Downloads (cumulative): 0
· Downloads (12 Months): 0
· Downloads (6 Weeks): 0

Raincode assembler compiler

Full Text: PDF

Authors: Volodymyr Blagodarov Raincode, Belgium
Yves Jaradin Raincode, Belgium
Vadim Zaytsev Raincode, Belgium

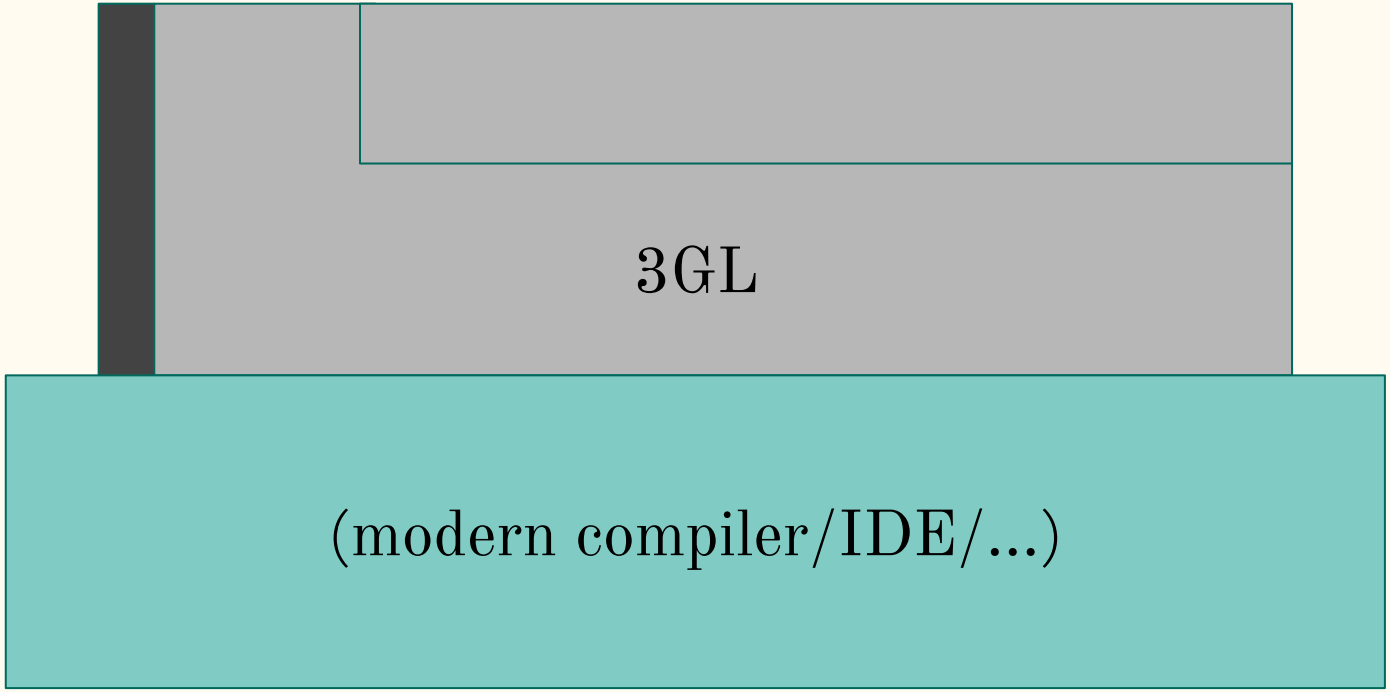
Published in:
· Proceeding
SLE 2016 Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering
Pages 221-225

Amsterdam, Netherlands — October 31 - November 01, 2016
ACM New York, NY, USA ©2016
table of contents ISBN: 978-1-4503-4447-0
doi>10.1145/2997364.2997387

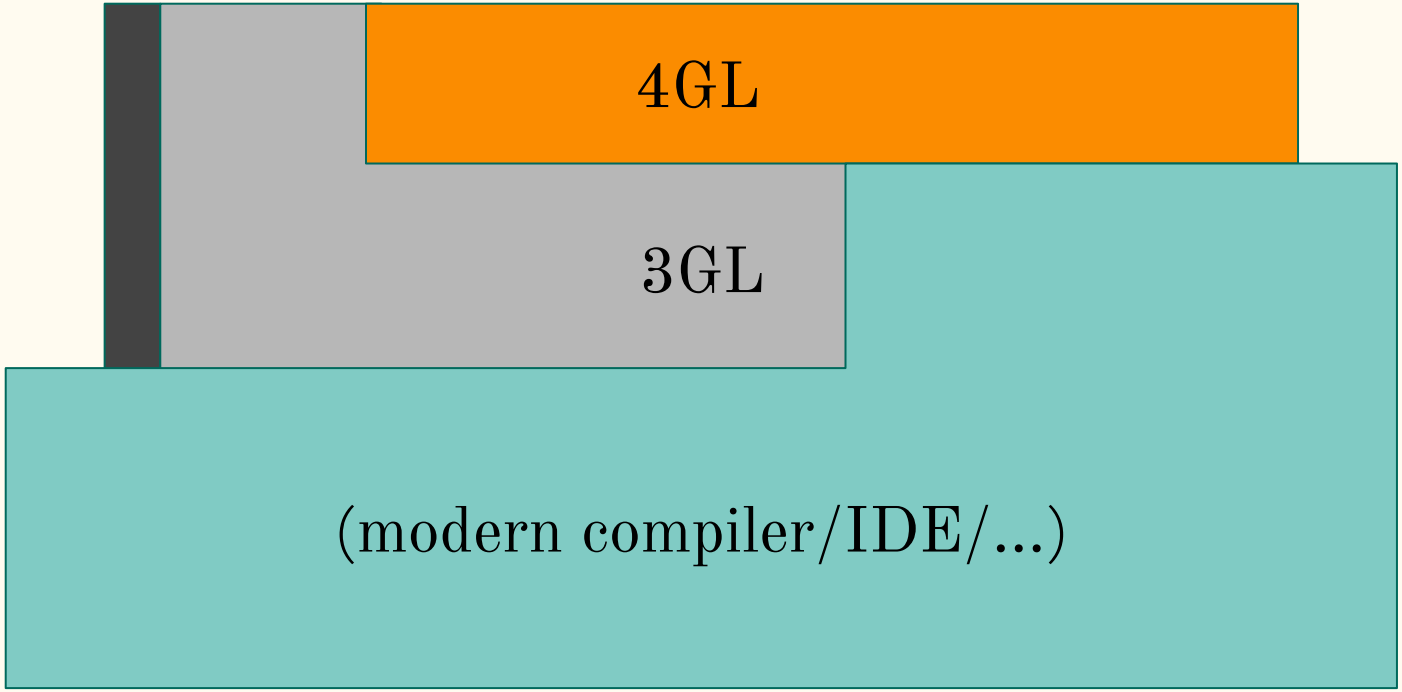
2016 Article

Bibliometrics
· Citation Count: 0
· Downloads (cumulative): 33
· Downloads (12 Months): 33
· Downloads (6 Weeks): 2

Language Migration with Generated Code



Language Migration with 4GL Code



Keep in Mind

- No language design, 100% implementation
- Documentation is not (a) given
- Domain experts = language experts/devs
- Many iterations with domain experts
- Months and years of effort, even with advanced tech
- Don’t try this at home!

Challenge: Regression Parsing

- regression parsing in general works well
- also in industrial settings
- great for the nightly build
- sometimes suitable only for weekly builds
- takes too long for continuous processes
- incrementality is ad hoc and limited

Challenge: Test Suite Inference

- first days of the compiler: nothing parses
- first months of the compiler: nothing runs
- customers grow impatient
- need to measure progress
- extensive test suites take tremendous time to create
- need coverage analysis, iterative refinement, etc

Challenge: Grammar Impact Analysis

- grammars are great
- finite specs of complex infinite artefacts
- if one nonterminal changes, what is the impact?
- no readily useful techniques, but no foreseeable showstoppers
- knowing the change impact enables many incremental techniques

Challenge: Grammar/Samples Dependencies

- for some languages, grammar inference is feasible and useful
- cf. "*Parser Generation by Example for Legacy Pattern Languages*" @ GPCE
- very few studies on incremental grammar inference
- needed both ways: codebase are updated, grammars too
- many opportunities to research and make great tools

Challenge: Neighbour Analysis

- the dark data of compiler construction: near misses
- cannot parse: “totally against expectations” vs “missing comma”
- useful for error tolerance and recovery
- done manually when exploring a new 4GL
- practical parsers often distinguish between success and commit
- differential testing + fuzzing?