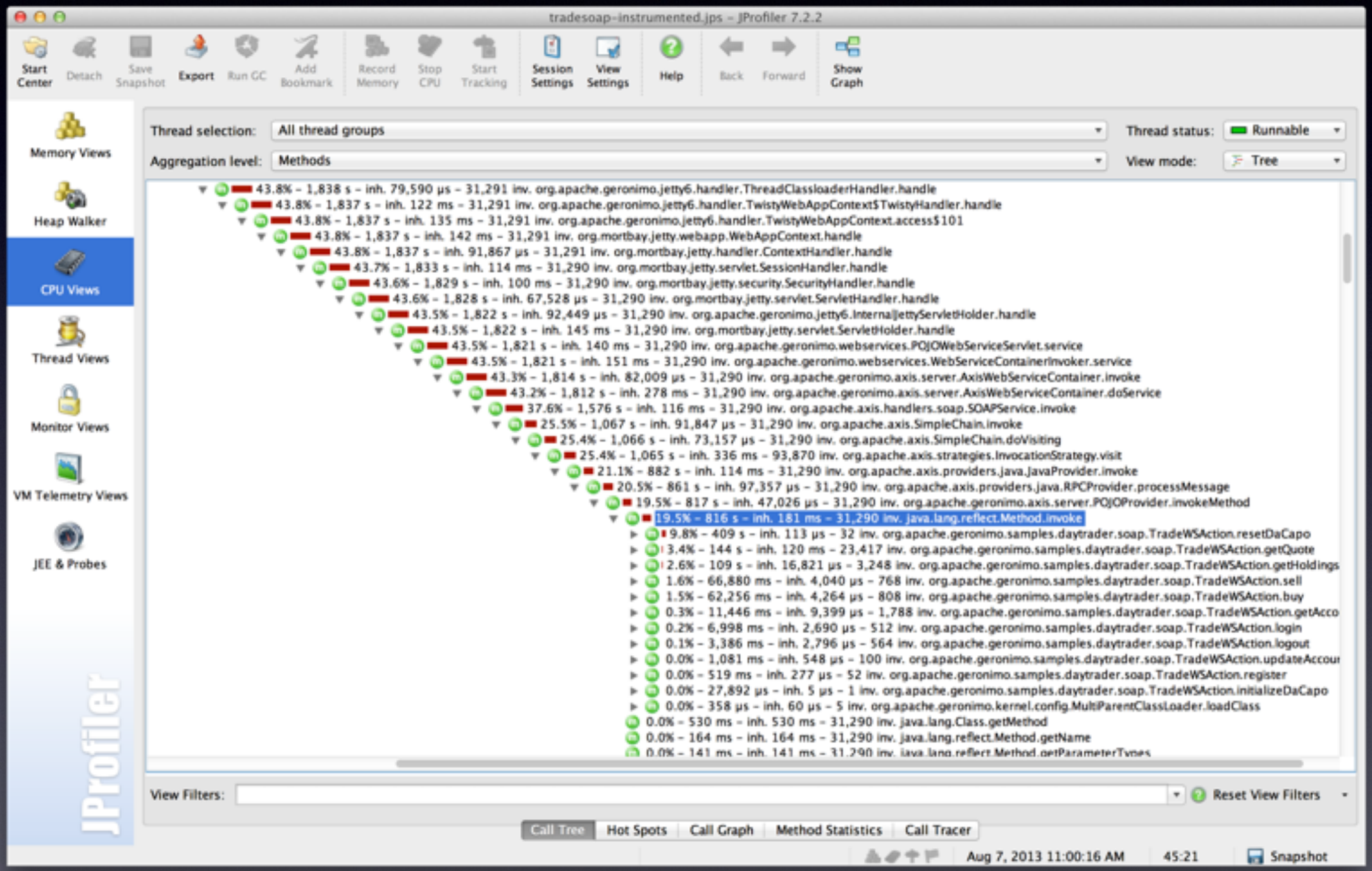
Performance Analysis for Object-Oriented Software

A Control-flow Centric Approach

ICSE 2014 Doctoral Symposium - David Maplesden
dmap001@aucklanduni.ac.nz

A ‘modern’ profiler



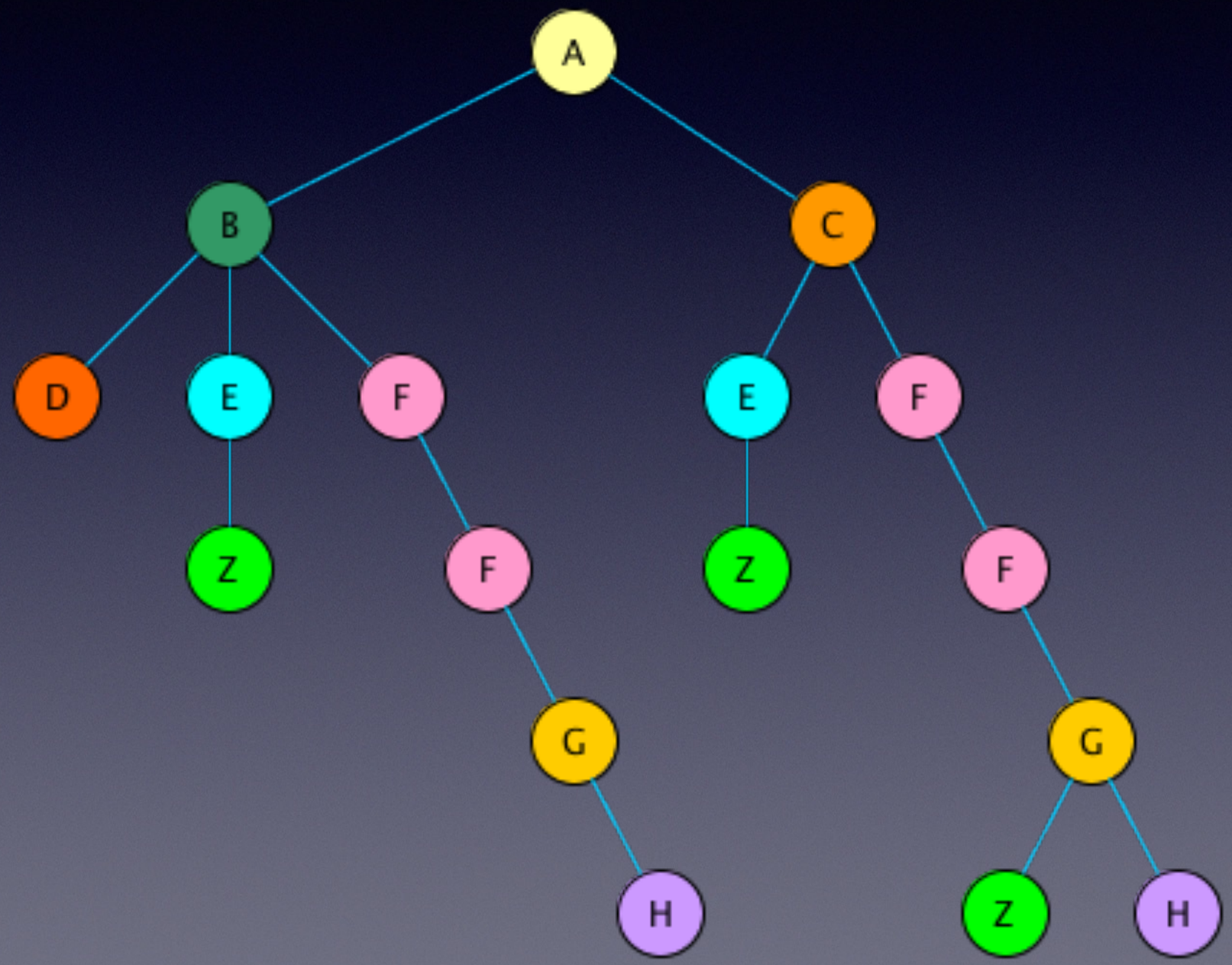
Challenges of OO software

- Many small methods
  - Inter-procedural control flow
- Heavily layered architecture
  - Engineered for maintainability and reuse
  - Reusable frameworks, more abstractions
- Complex, thinly distributed, runtime behaviour
- Challenging to provide actionable feedback

Hot methods - DaCapo tradesoap benchmark

Method	Occurrences in CCT	% Inherent Time	Number of Unique Calling Methods
java.lang.String.intern	461	15.818	15
java.net.SocketInputStream.socketRead0	27	6.576	1
java.security.AccessController.doPrivileged	390	5.719	53
org.apache.axis.encoding.SerializationContext .startElement	66	3.577	7
org.h2.jdbcx.JdbcXAConnection $PooledJdbcConnection.checkClosed	357	2.646	2
java.lang.Class.getClassLoader0	76	1.890	2

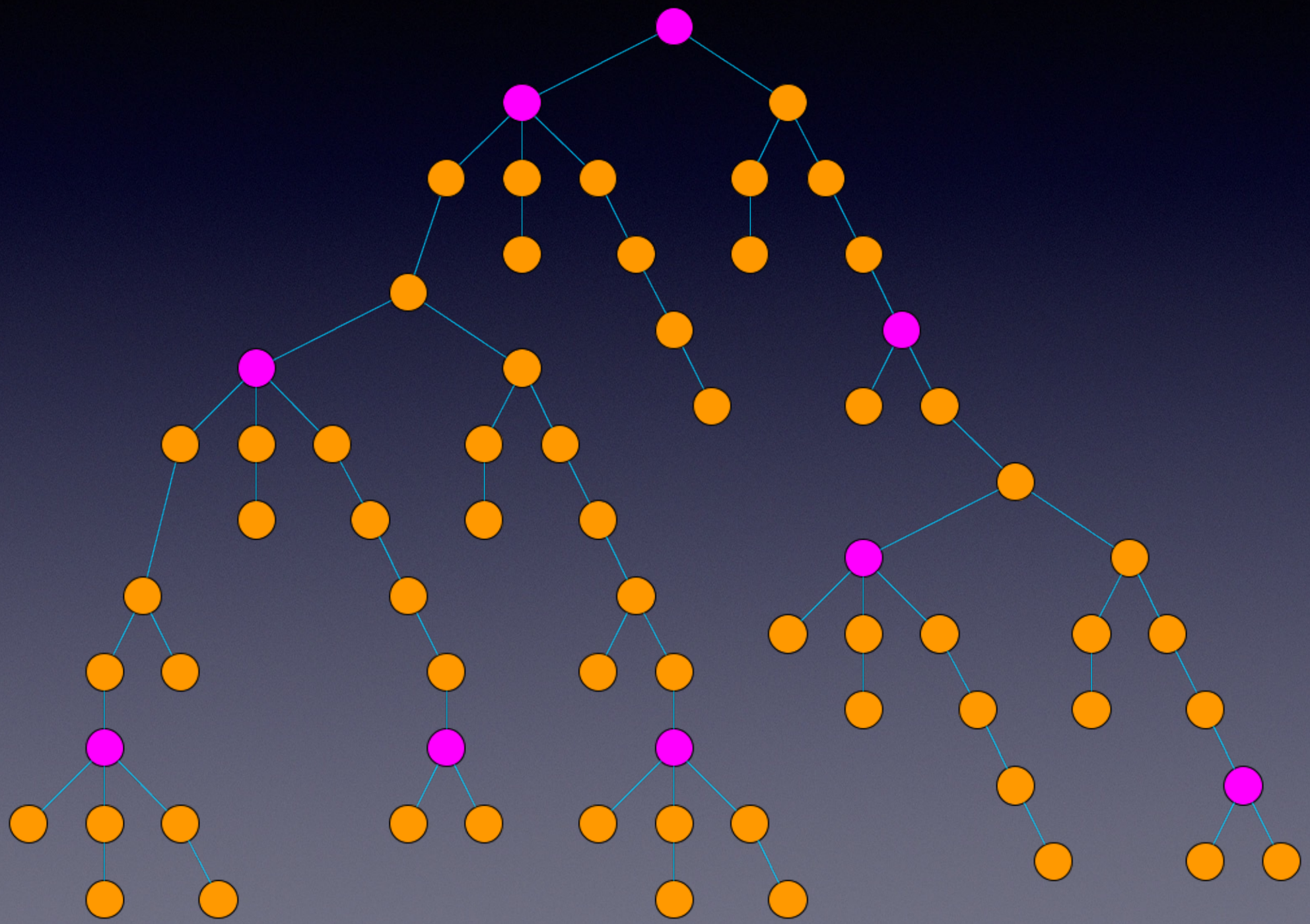
Calling Context Trees



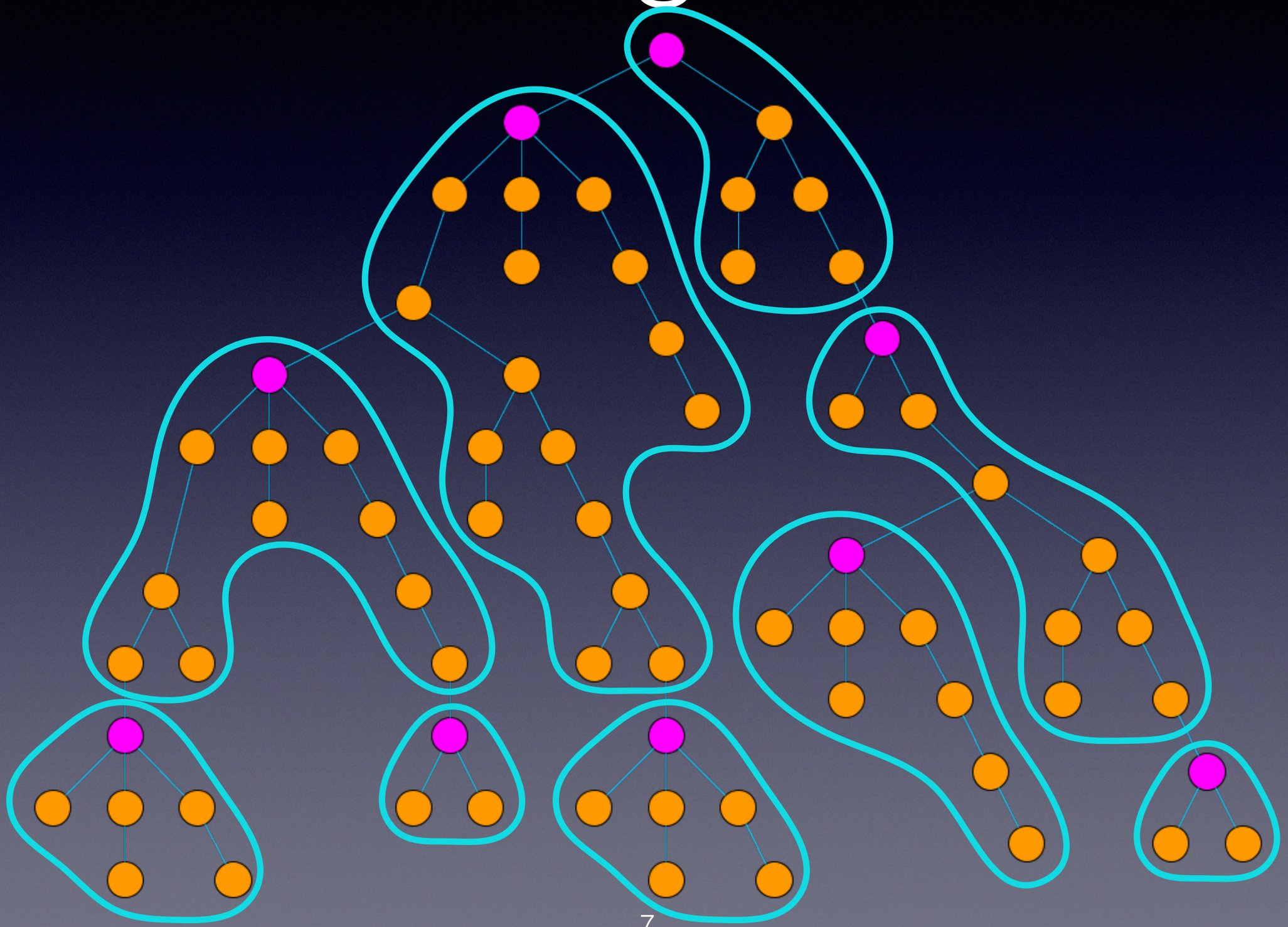
Key Idea

- CCTs aren’t just random data
- There are patterns within the calling context tree
  - induced by the design of the software
  - compact
  - repeated in multiple locations
  - expensive when aggregated

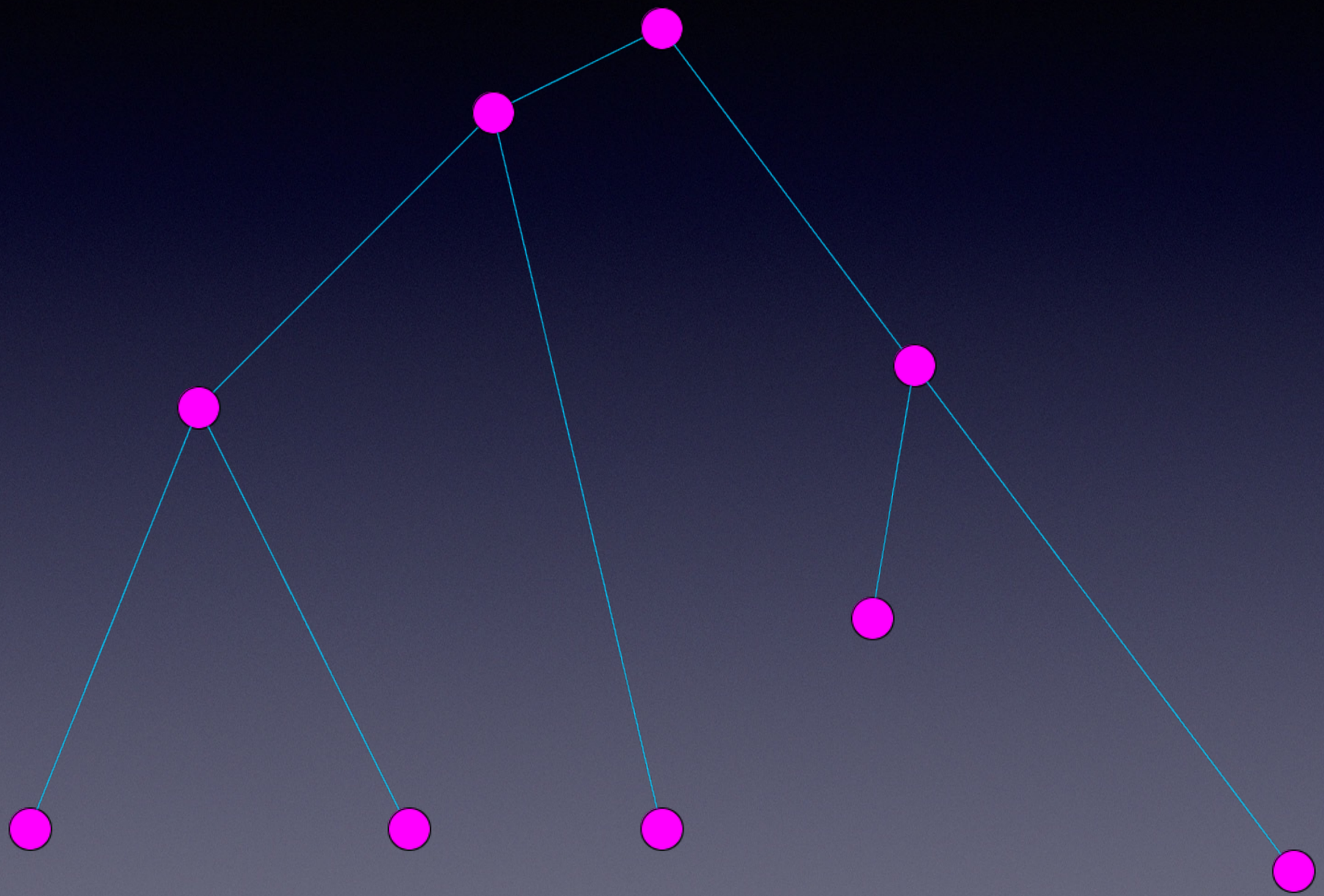
Subsuming Methods



Subsuming Methods

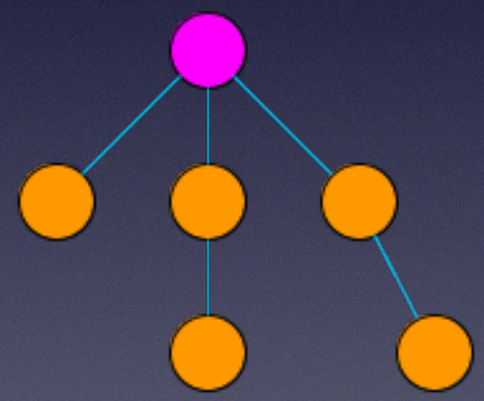


Consolidated Tree



Subsuming Methods

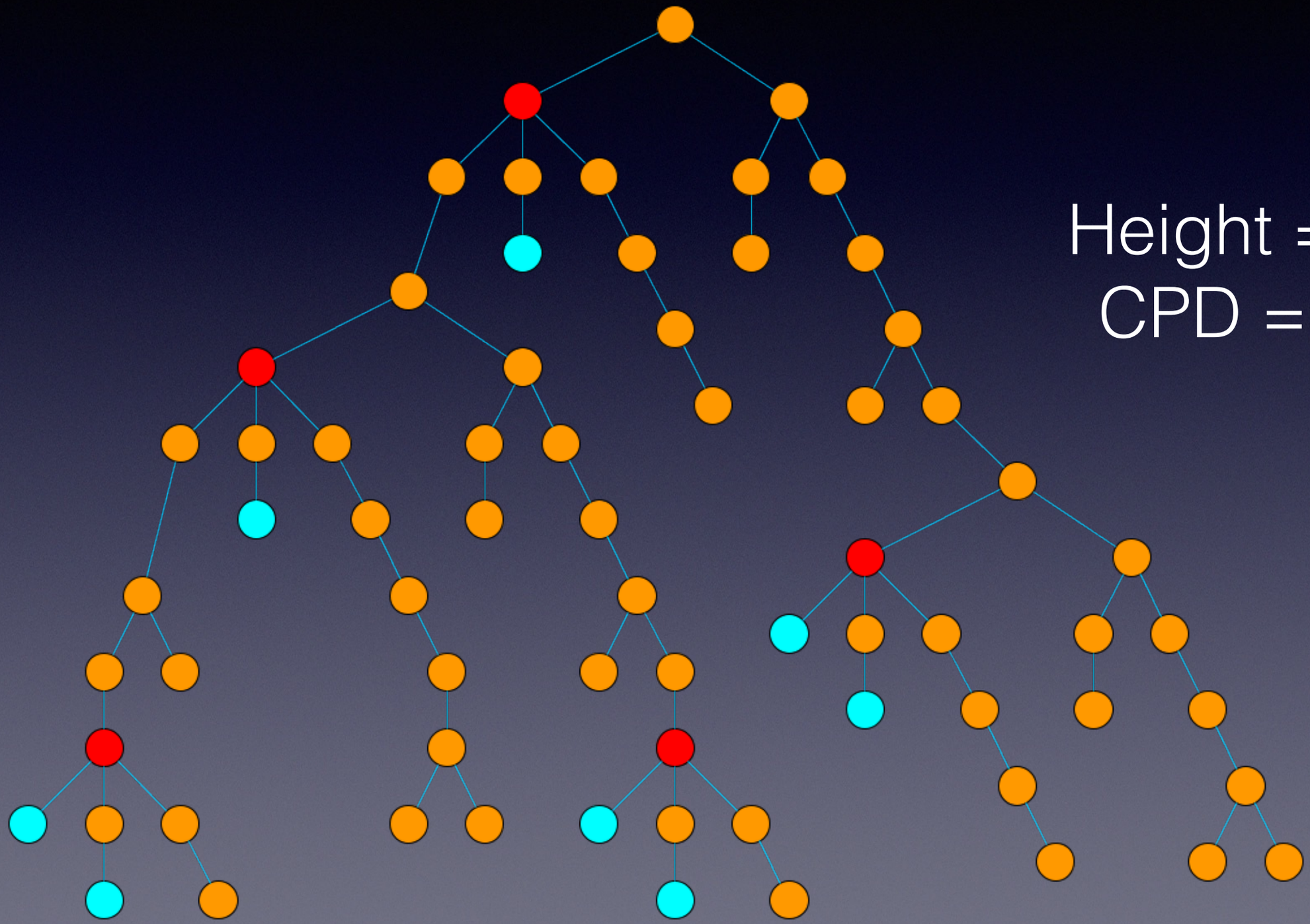
- Subsuming methods form a course grained partition of the CCT
- Each subsuming method represents a pattern consisting of itself and the subsumed methods it calls
- How do we choose our subsuming methods?



Subsuming Attributes

- ‘Elementary’ methods - induce a limited range of behaviour
  - We have used the height of the method as a measure of the range of behaviour it induces
    - Trivial case (height = 0) - method never calls any other method - a leaf method
- ‘Subordinate’ methods - called in a predictable manner
  - Every call to the method can be attributed to a (nearby) common parent which is responsible for their invocation
  - The trivial case is when a method is only ever called from a single call site

Height and Common Parent Distance



Height = 0
CPD = 2

Results - DaCapo tradesoap benchmark

Constraints: height > 4 and CPD > 4

	Full CCT	Subsuming CCT
Nodes	64093	18349
Height	145	60
Unique Methods	8162	587
90% Method count	659	73

Top Subsuming Methods - DaCapo tradesoap benchmark

Method	Occurrences in CCT	% Inherent Time	% Subsumed Time
com.sun.org.apache.xerces.internal.parsers.XMLParser.parse	36	0.000	11.534
org.apache.axis.encoding.SerializationContext.startElement	66	3.577	8.114
java.io.BufferedInputStream.fill	21	0.009	6.594
java.security.AccessController.doPrivileged	390	5.719	6.134
org.apache.axis.message.SAX2EventRecorder.replay	70	0.341	4.380
org.apache.axis.encoding.DeserializationContext.startElement	53	0.296	4.080

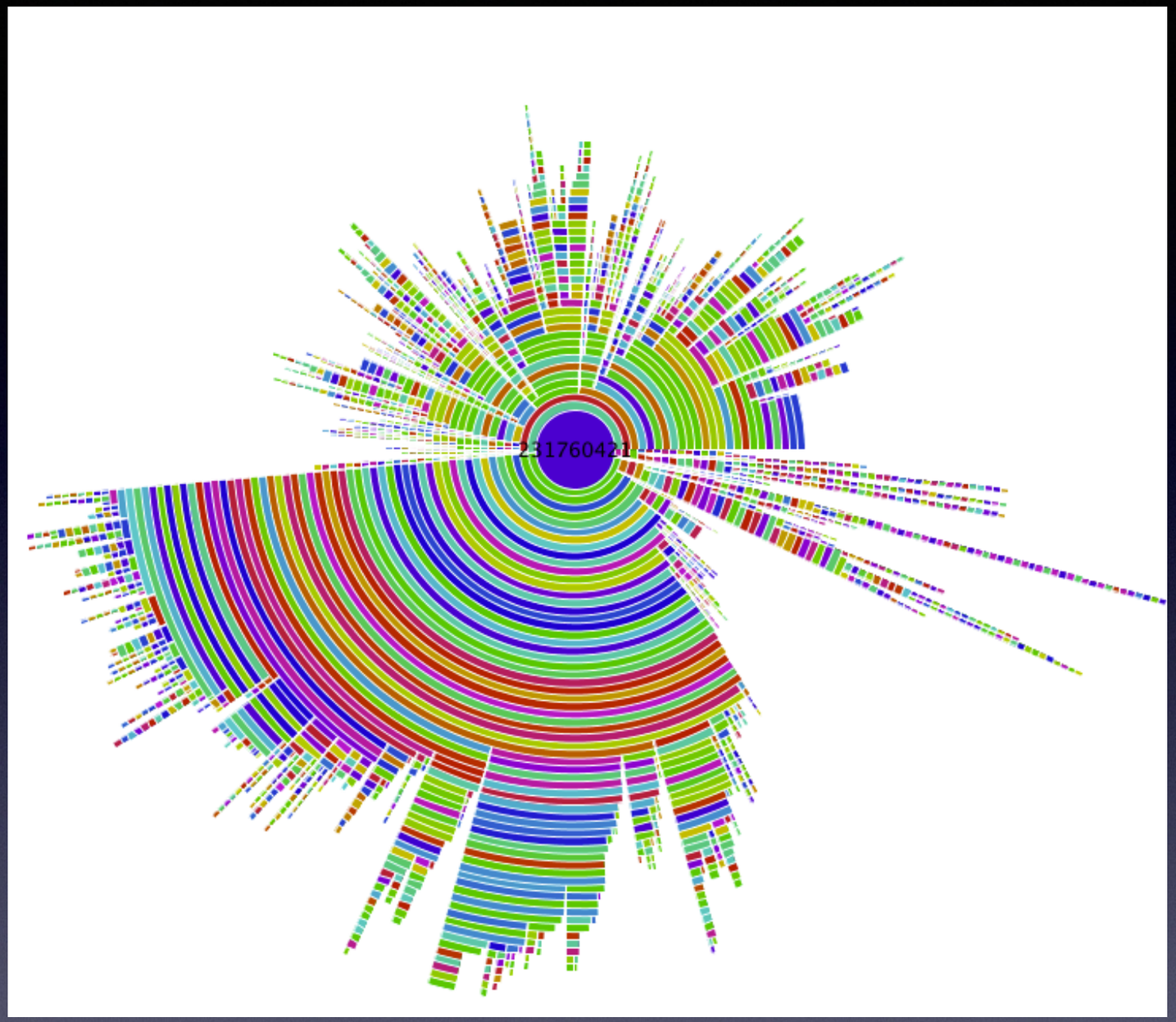
Evaluation Approach

- Evaluate the tool prototype
- Empirical evaluation of concrete metrics
  - Number of patterns found
    - Proportion of overall costs for which patterns are responsible
    - Use standard open benchmarks - DaCapo, SPECjvm
    - Industrial case study
- Controlled user study and questionnaire

Research Plan

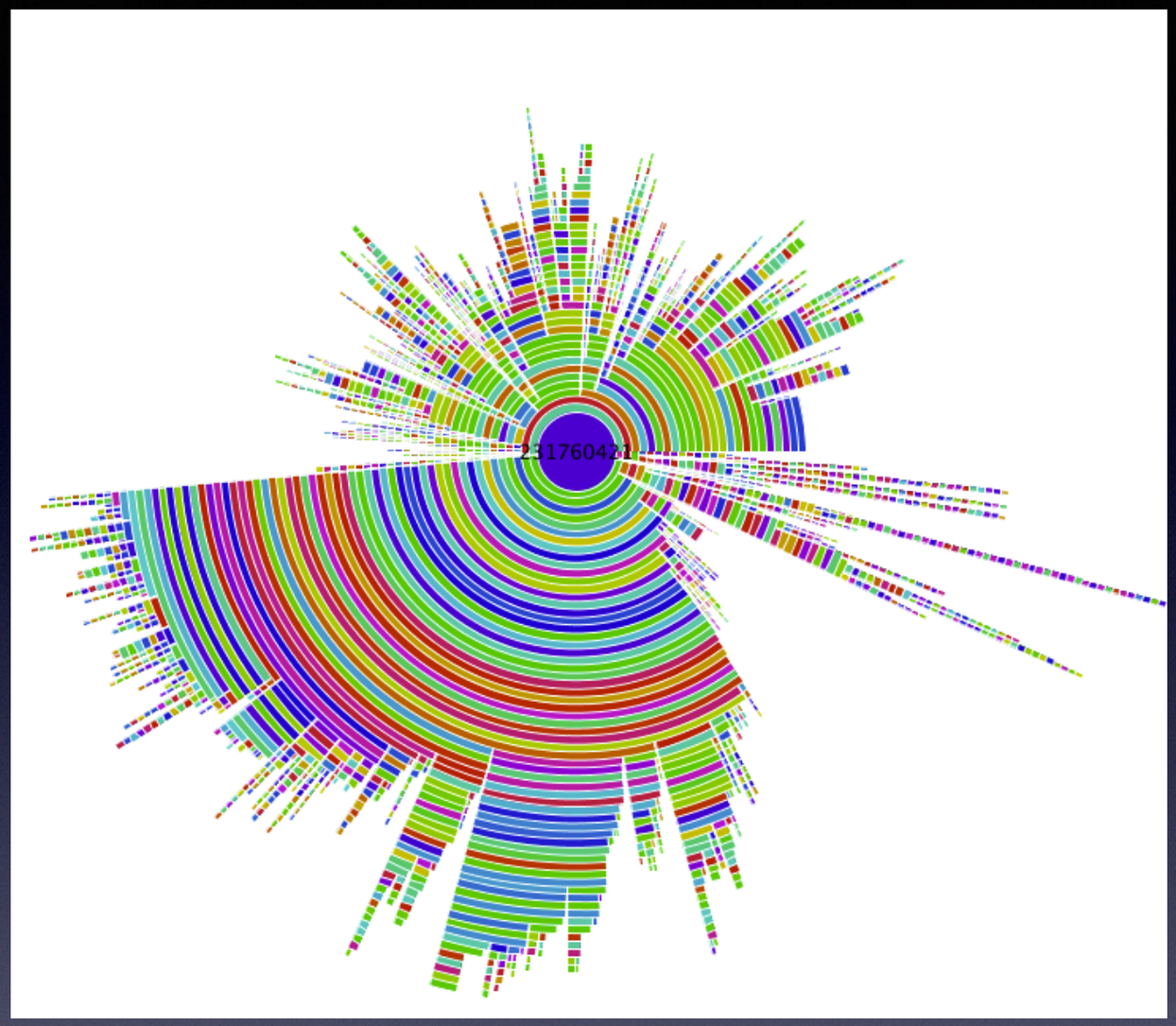
- Next Steps
  - Thorough evaluation of subsuming methods
- Later this year
  - Other subsuming characteristics - static attributes
  - Other pattern matching or aggregation approaches
  - Characteristics of identified patterns
- Next Year
  - User study evaluation

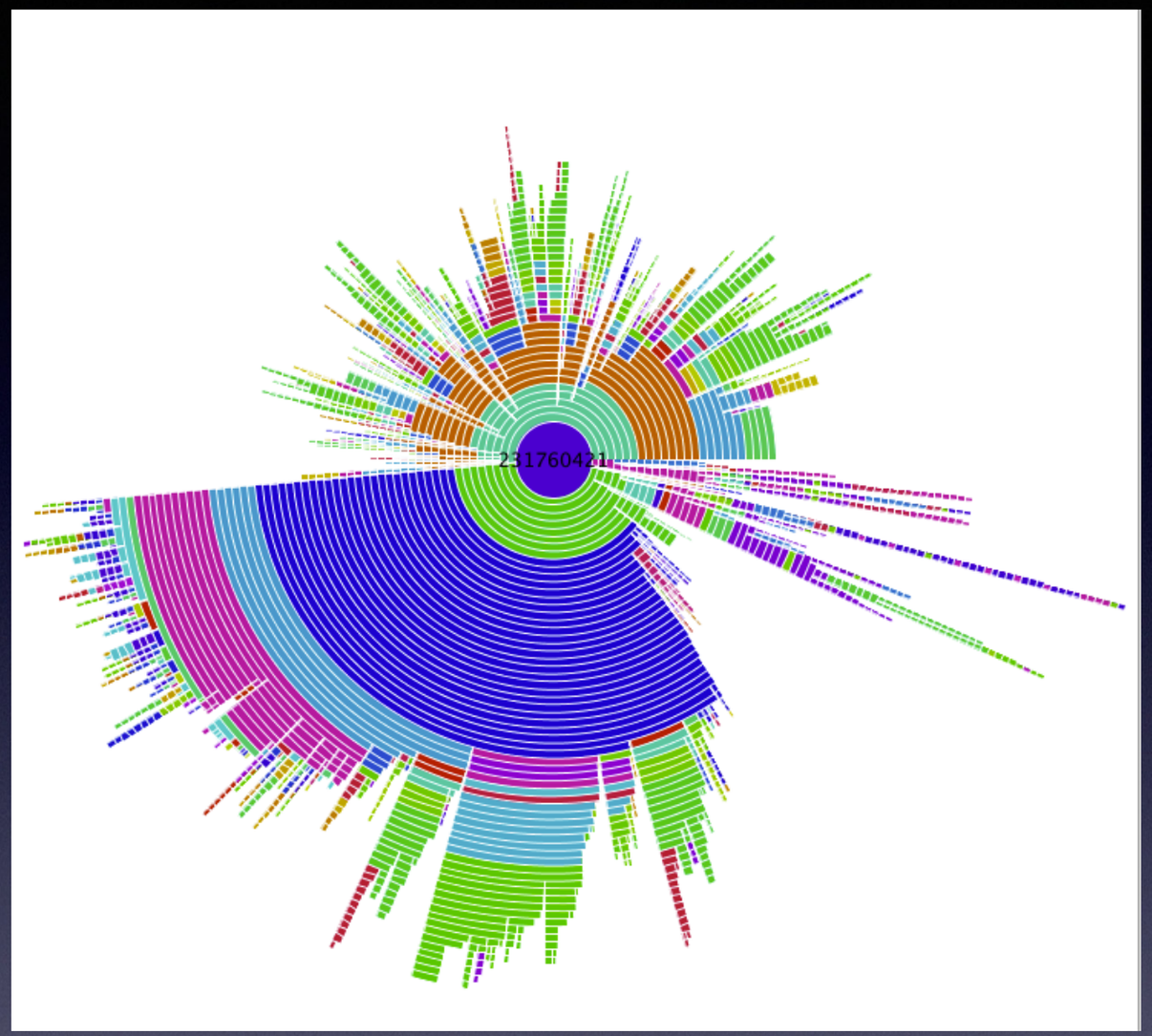
Questions?



DaCapo tradesoap	
Nodes	64093
Height	145
Unique Methods	8162

DaCapo tradesoap - CCT

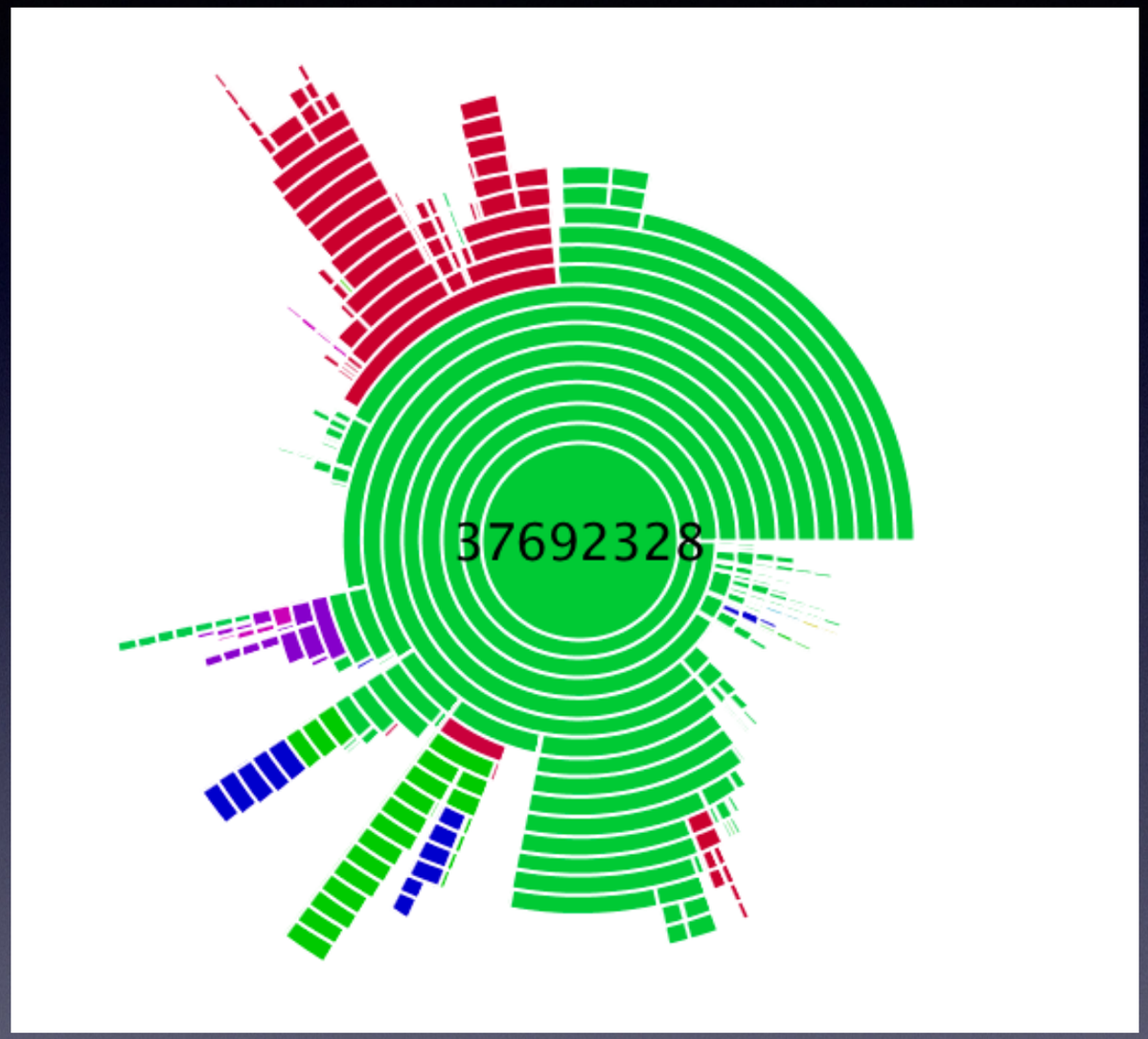




CCRC - Subsuming

com.sun.org.apache.xerces.internal.parsers.XMLParser.parse

Occurrences	36
Height	36
Subsuming Height	15
Exclusive Time	0.000%
Inclusive Time	16.263%
Subsumed Time	11.534%



Research Questions

- Can we identify repeated patterns of method calls within the dynamic behaviour of an object-oriented application that represent performance critical sections of code?
- Are these identified repeated patterns useful in guiding the performance optimisation process? Do they aid in the comprehension of dynamic behaviour/performance?
- Can we automatically generate actionable feedback on how to improve an application's performance based on the identified repeated patterns?

Related Work

- Very broad domain (100 venues in our SLR)
- Relevant work from HPC, Compiler, Programming Language domains
- Majority of approaches provide simple metrics
  - Lack of actionable feedback
- Very few approaches leverage static analysis
- Runtime bloat research focussed on data-flow

Runtime Bloat Research

- Tackle problem of excessive activity to achieve seeming simple functionality
- Data-flow centric approaches
- Efficiency of data structures
- Object pooling opportunities
- Copy profiling
- Reference propagation profiling

Other research directions

- Does aggregation improve accuracy?
- Is the common parent idea useful for other applications?
- Dominating methods

Schedule

January - 2014	Complete initial tool prototype development and evaluation
February	
March	Write first paper covering ideas and evaluation in first tool prototype
April	Prototype development - second iteration
May	
June	*ICSE 2014 PhD Symposium*
July	Second paper covering second tool prototype
August	Prototype development - third iteration
September	Focus on industrial case study evaluation
October	
November	Write up paper on updated tool and industrial case study
December	Prototype development - fourth iteration
January - 2015	
February	
March	Design and conduct user study evaluation of tool
April	
May	
June	
July	Write paper covering updated tool and user study evaluation
August	Thesis writing
September	
October	
November	
December	
January - 2016	

Expected Contributions

- Techniques for identifying repeated patterns of dynamic behaviour within the calling context tree
- A tool that implements these techniques
- An evaluation of these techniques
- The development of actionable feedback for improving the performance critical sections of code represented by the identified repeated patterns

Subsuming Attribute - Height

- Methods that induce a limited range of behaviour
  - Difficult to optimise
- We have used the height of the method as a measure of the range of behaviour it induces
- method height = max height of any sub-tree with an instance of the method as its root
- Trivial case (height = 0) - method never calls any other method - a leaf method

Subsuming Attribute - Common Parent Distance

- Methods that are called in a constrained manner
- Every call to the method can be attributed to a (nearby) common parent which is responsible for their invocation
- We have used the common parent distance (CPD) as a measure of this characteristic
- The trivial case (CPD = 1) is when a method is only ever called from a single call site