



2. Our Approach to Software Quality Assessment

Rüdiger Lincke

ARiSA™ - Applied Research in System Analysis

<http://www.arisa.se>

Växjö, 19 August 2005

Goals of this talk

- ⇒ Define our approach and its relation State-of-the-Art and Standards in Software Quality Assessment
- ⇒ Discuss what's missing
 - from research perspective
 - from application perspective

Agenda

- ⇒ ARiSA – First Contact Analysis
- ⇒ ARiSA – Quality Monitor

Agenda

- ⇒ ARiSA – First Contact Analysis
- ⇒ ARiSA – Quality Monitor

ARiSA – First Contact Analysis

- ⇒ Uses a Generic Quality Model
- ⇒ Assesses internal software quality using VizzAnalyzer
- ⇒ Maps standard metrics to ISO 9126 standard criteria and factors, cf. Factor Criteria Metric
- ⇒ Advantages
 - Gives an overview and first impression on internal software quality,
 - Allows to compare different projects, software products
 - Good starting point for tailoring a specific Quality Model, cf. ARiSA Quality Monitor
 - Takes only a week (or less) to be established

Knowledge Basis

- ⇒ Compendium of Software Quality Standards and Metrics
 - Defines factors and criteria derived from ISO 9126
 - Defines well known software metrics
 - Relates metrics and factors and criteria (double index)
- ⇒ Goals: comprised community knowledge
- ⇒ Open knowledge base
 - Critical mass for being attractive (ongoing work)
 - Open for new entries
 - Edited to keep consistency
- ⇒ Available at: <http://www.arisa.se/compendium>

Compendium's Double Index

⇒ For each ISO 9126 criterion

- Reference to highly related metrics
- Reference to somewhat related metrics
- Related could be directly or inversely related
- Goal driven index

⇒ For each metric

- Reference to highly related ISO 9126 criteria
- Reference to somewhat related ISO 9126 criteria
- Technology driven index

Example mapping criteria to metrics

- ⇒ **Learnability for Reuse**: the attributes of software that bear on a software engineer/developer effort for learning how to integrate a software component in new systems
- ⇒ **Highly** Related Metrics
 - McCabe Cyclomatic Complexity
 - Number of Attributes and Methods
 - Weighted Method Count
 - Depth of Inheritance Tree ...
- ⇒ **Somewhat** Related Metrics
 - Lines of Code
 - Number of Statements ...

Example mapping metrics to criteria

- ⇒ Depth of Inheritance Tree(DIT).
- ⇒ **Highly** Related Software Quality Properties
 - Re-Usability (both directly and inversely)
 - Understandability for Reuse (inversely)
 - Learnability for Reuse (inversely)
 - Operability for Reuse – Programmability (inversely)
 - Attractiveness (directly)
 - Maintainability (inversely)
 - Changeability (inversely)
 - Testability (inversely)
 - Portability (inversely)
 - Adaptability (inversely)
 - Replaceability (inversely)

Example cont'd

...

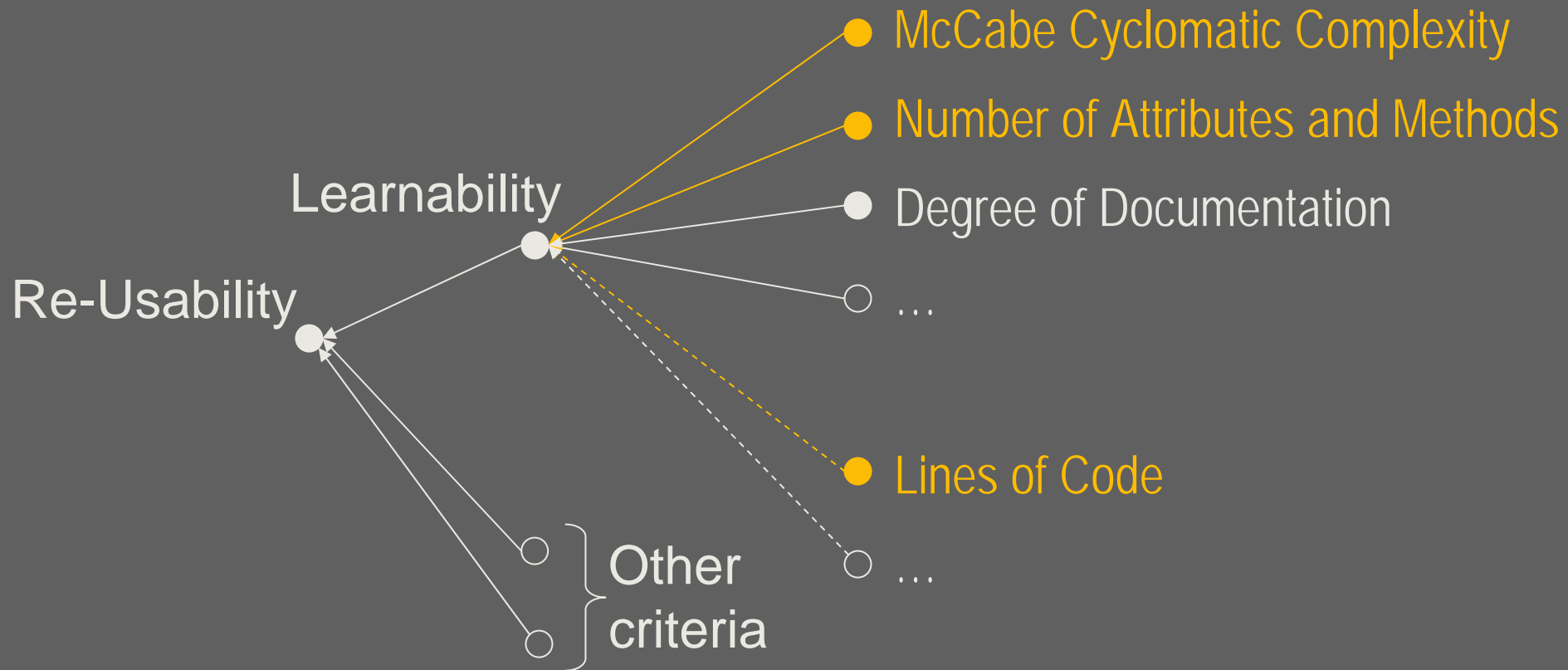
➡ Somewhat Related Software Quality Properties

- Efficiency (inversely)
 - Time Behavior (inversely)
 - Resource Utilization (inversely)
- Maintainability
 - Stability (inversely)

Generic Quality Model

- ⇒ Compendium induces a Generic Quality Model
- 1. Normalized (0..1) metrics for each metric (sub-)class
 - Complexity – size, structure, interfaces
 - Architecture – cohesion, coupling on class/package level
 - Design – documentation, conventions
- 2. Attach edges of FCM structure with weights:
 - 1 (-1) for highly directly (**inversely**) related criteria
 - $\frac{1}{2}$ (- $\frac{1}{2}$) for somewhat directly (**inversely**) related criteria
- 3. Compute the weighted sum for factors

Example: Learnability



Evaluation

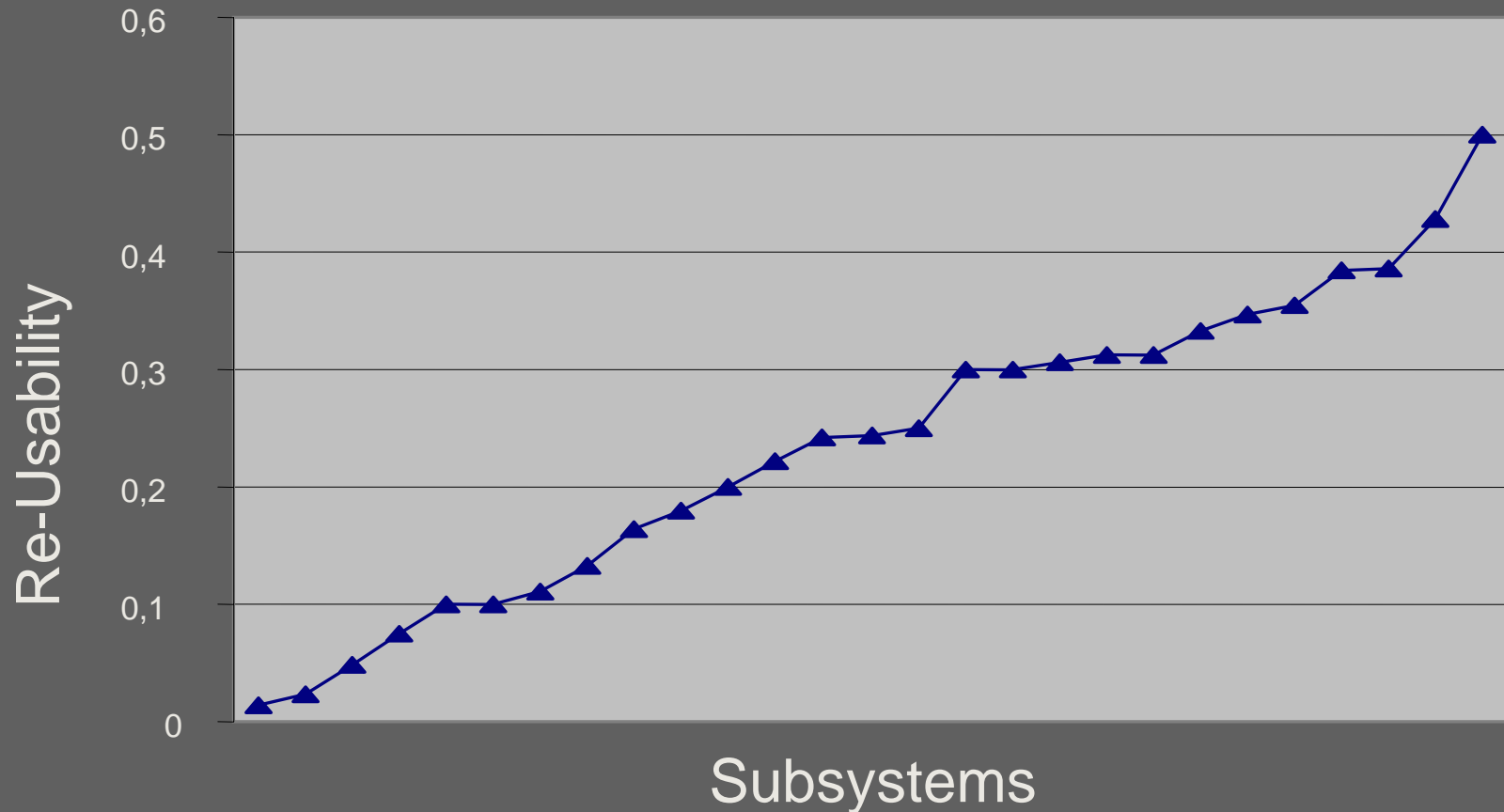
⇒ Absolute

- Compare weighted sum of factors to a normative range
- Questionable

⇒ Self-related

- Compute a distribution of weighted sum of factors for all software entities (classes, files, packages, components) in the system
- Look for outliers (considered harmful)

Example: A database server



Open Issues / Questions

- ⇒ Achieve a critical mass in the compendium.
- ⇒ Which metrics should be selected?
- ⇒ Are the weights reasonable?
- ⇒ Is there a constant range for “healthy” and “challenged” projects
 - generally?
 - depending on project size/company/programming paradigm?

Agenda

- ⇒ ARiSA – First Contact Analysis
- ⇒ ARiSA – Quality Monitor

ARiSA – Quality Monitor

- ⇒ Tailored Quality Model
- ⇒ Designed in co-operation with a customer
- ⇒ Assesses internal software quality according to specific goals of the customer
 - Maps standard and even tailored metrics to answer customer's specific questions, cf. Goal Question Metric
 - Takes more time to install
- ⇒ Advantages
 - More detailed assessment of quality
 - Specific to customer's requirements

Tailored Quality Model

- ⇒ Start with the Generic Quality Model
- ⇒ Select relevant part including standard metrics
- ⇒ **Add** new information extractors for extracting information from (design documents, documentations, CVS, Bug-Database etc.)
- ⇒ **Add** new metrics based on this information
- ⇒ Map software entities, their relations and metrics (software model) to a multi-dimensional picture (software view) for easy evaluation

Models of Software Systems

⇒ Relational System

- Entities like classes, methods, statements, (call-) expressions
- Syntactic and semantic relations like contain, define-use, inherits, calls
- Metrics attached to both
- Changes over time in the development process (special relation)

⇒ Models captured by graph structures, in our design

- nodes modeling entities
- edges between two nodes modeling binary relations (n -ary relations are modeled by introducing special nodes),
- arbitrary properties of the whole graph and its nodes and edges modeling metric values.

Views on Software Systems

- ⇒ Visualize software model, i.e. graph structure
- ⇒ Visual language which has finite many dimensions
 - x, y, z positions of nodes and edges,
 - height, width, depth of nodes and edges,
 - color, shape and texture of nodes and edges,
 - label of nodes and edges, and
 - One version property indicating their changes over time

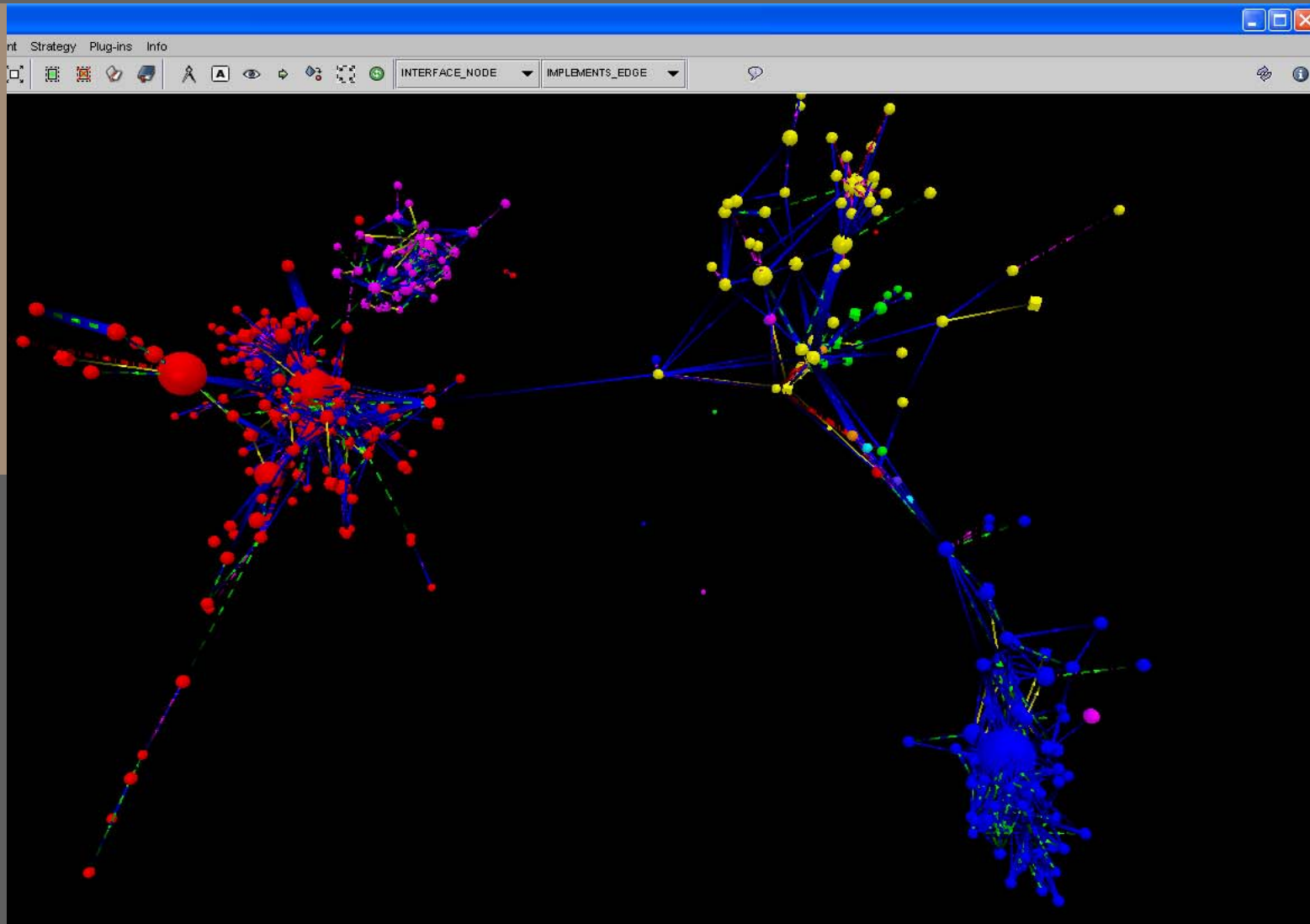
Components vs. Packages

Model

Class nodes,
Call edges
Complexity metric
Package property

View

Complexity – size
Package – color



Maintainability

Model

Classes

LOC, WMC, LOD

Package property

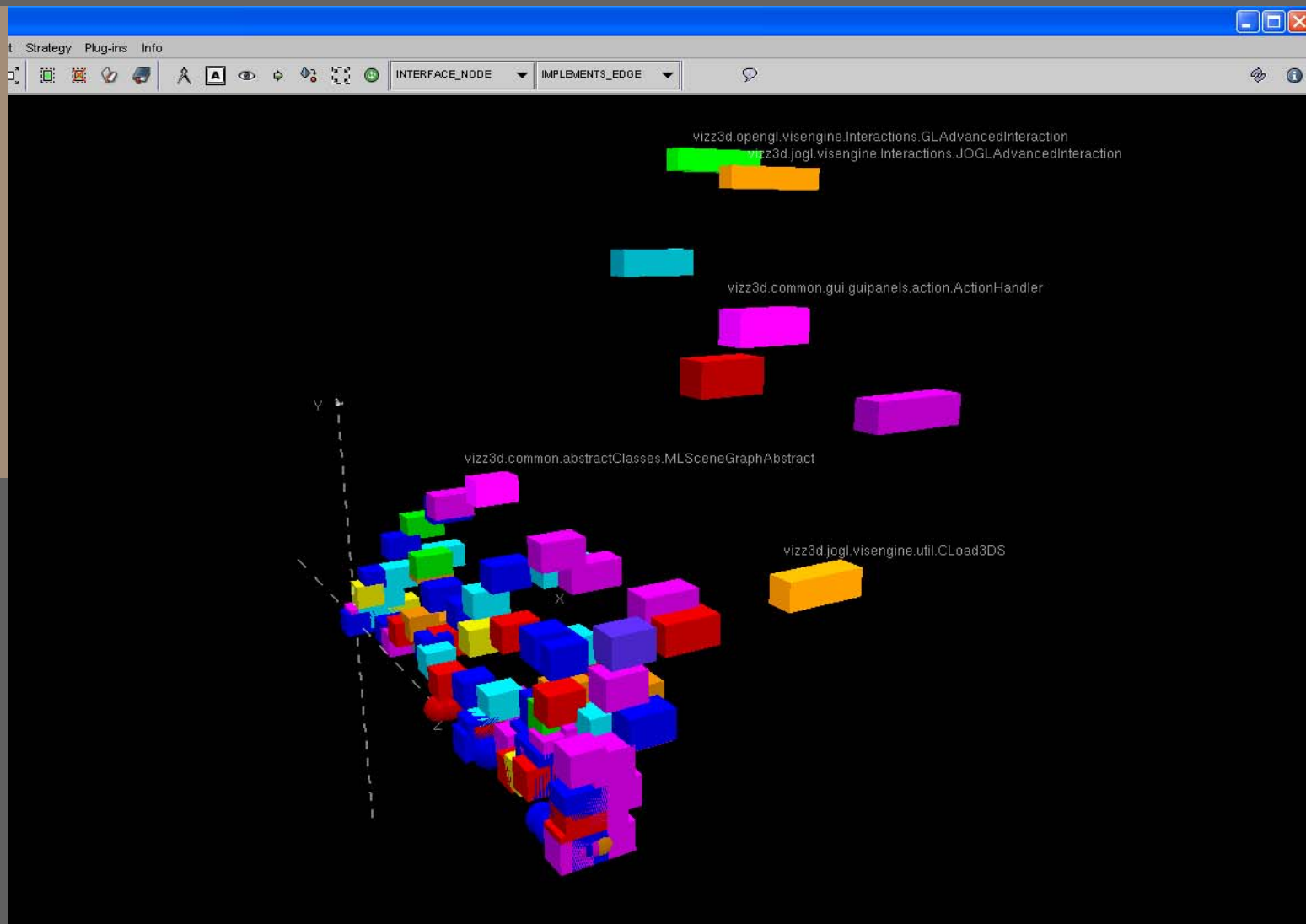
View

Package – color

LOC – x

WMC – y

LOD – z



Maintainability (alternative view)

Vizz3D - OpenGL (JOGL)

File Edit Navigation Layout Binding Environment Strategy Plug-ins Info

INTERFACE_NODE FIELD_ACCESS_EDGE

Label Type

Label	Type
vizz3d.common.gui.g...	CLASS_NODE
vizz3d.jogl.visengine...	CLASS_NODE
visgraph.VisualProper...	CLASS_NODE
visgraph.j3d.interface...	INTERFACE_NODE
VAutils.bindings.gene...	CLASS_NODE
vizz3d.common.gui...	CLASS_NODE
VAutils.bindings.gene...	CLASS_NODE
visgraph.Version	CLASS_NODE
VAutils.bindings.gene...	CLASS_NODE
vizz3d.common.gui.g...	CLASS_NODE
vizz3d.common.gui.g...	CLASS_NODE
VAutils.bindings.gene...	CLASS_NODE
VAutils.bindings.gene...	CLASS_NODE
vizz3d.common.gui.g...	CLASS_NODE
vizz3d.common.interf...	INTERFACE_NODE
vizz3d.java3d.visengi...	CLASS_NODE
vizz3d.opengl.visengi...	CLASS_NODE
VAutils.bindings.j3d.f...	CLASS_NODE
vizz3d.util.GLDebug	CLASS_NODE
vizz3d.opengl.visengi...	CLASS_NODE
visgraph.gl.utilities.G...	CLASS_NODE
visgraph.visualgraph.i...	CLASS_NODE
visgraph.gl.visTypes...	CLASS_NODE
vizz3d.java3d.visengi...	CLASS_NODE
vizz3d.jogl.visengine...	INTERFACE_NODE
vizz3d.common.abstr...	CLASS_NODE

vizz3d.opengl.visengine.Interactions.GLAdvancedInteraction
vizz3d.jogl.visengine.Interactions.JOGLAdvancedInteraction
vizz3d.common.gui.guipanel.action.ActionHandler
vizz3d.java3d.visengine.Interactions.J3dAdvancedInteraction
vizz3d.jogl.visengine.util.CLoad3DS

Layout

Method Size Nesting Level Algorithm, uses LOC(Lines of Code).
2004-10-18, Mathias Peltonen, Thomas P...
anas

Package_Sorting
Sorting Metric: LOC (relative)
Y Metric: WMC (relative)
Z-Coord Multipli... 1
Items on X: 20
Spacing X: 3
Spacing Z: 3
Animation: 15

Graph Info

Vizz3D

AST (Graph)	Value
Binding 0	Label-Label
Binding 1	Type-Node Shape
Binding 2	Type-Node Texture
Binding 3	LOC (relative)-X
Binding 4	WMC (relative)-Y
Binding 5	LOD Undocumented (Class/Interface) (relative)-Z
Binding 6	LOC (relative)-Width
Binding 7	Multiplicity-Label
Binding 8	Type-Edge Shape

Open Issues / Questions

- ⇒ Does it answer the initial questions?
- ⇒ What does it take to tailor a quality model in terms of our and customer's effort?
- ⇒ What are relevant views?
 - Pictures vs. statistics?
 - If pictures, how many dimensions? Which one should be the domination (color, size, position)?

Trend Measurements

- ⇒ Self reference gives an overview of critical subsystems, but not “Continuous quality improvement”
- ⇒ Requires trend measurement:
 - Tailored Quality Model over time
 - Relate software entities of different versions (classes, packages, files, components)
 - Compute metric values for each version or/and sequences thereof
 - Redesign the mapping to picture
- ⇒ Full version of ARiSA – Quality Monitor

Trend of Stability

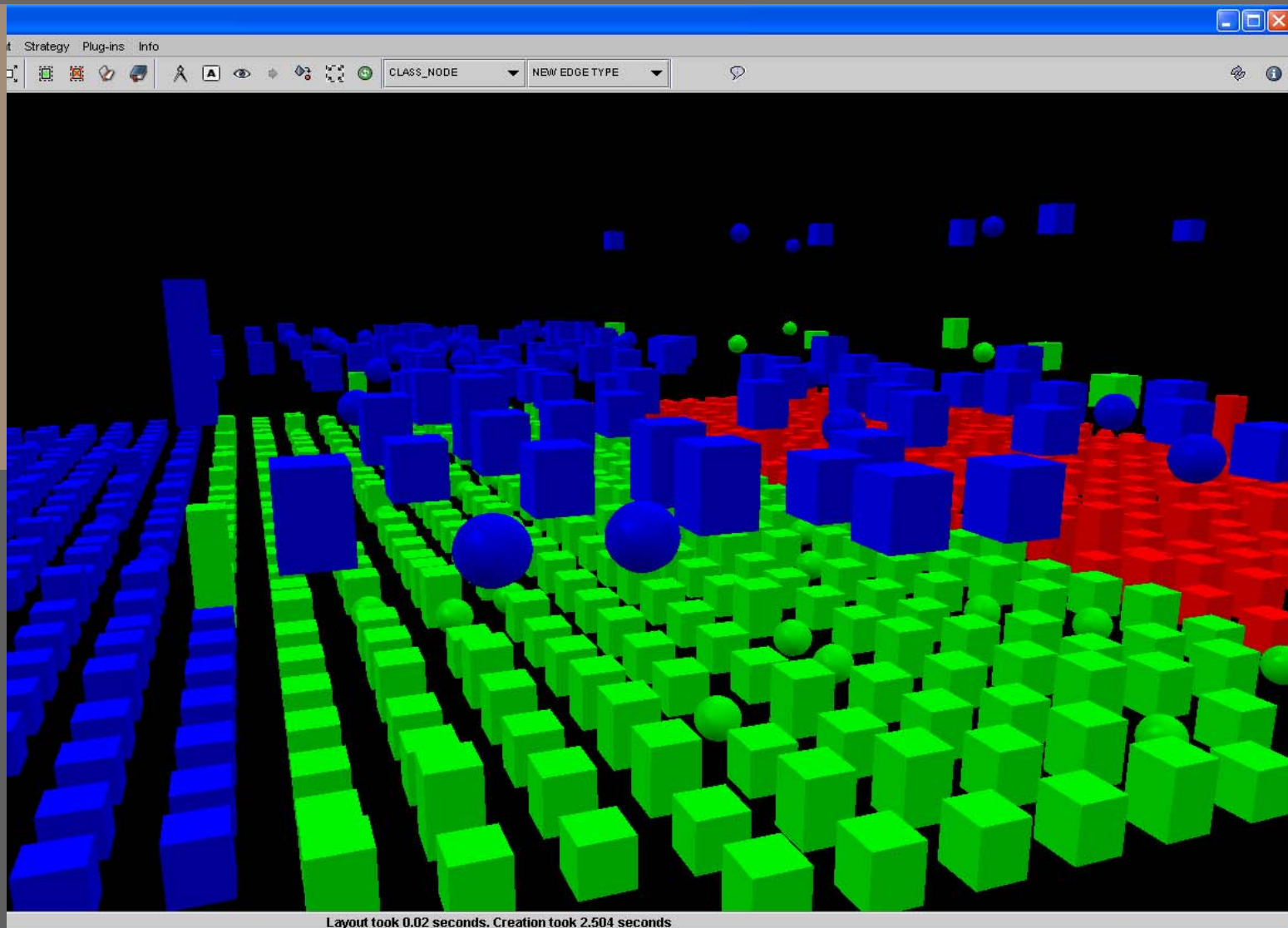
Model

Class and
Interface nodes,
Complexity metric

View

Complexity – size

Version – color



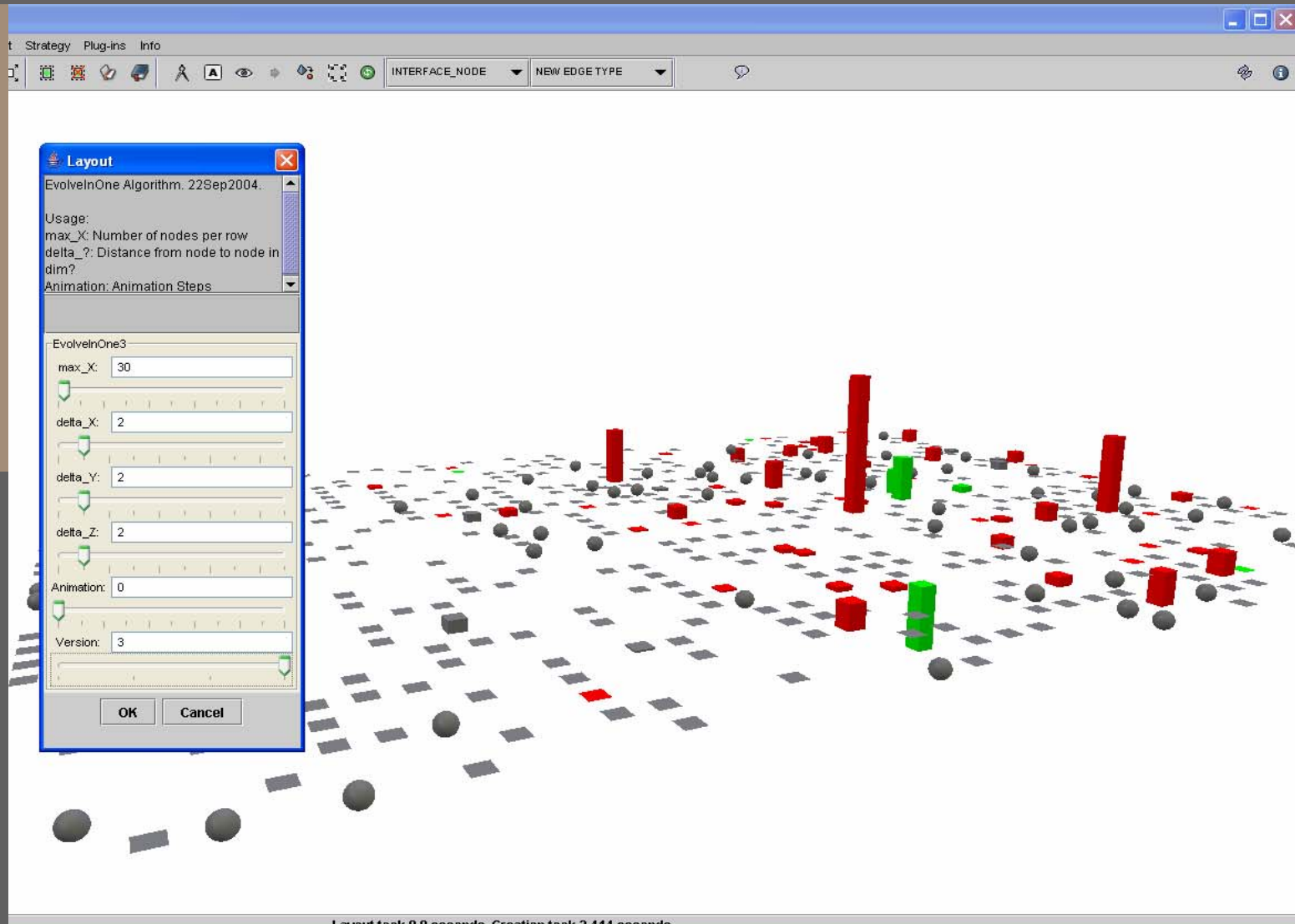
Trend of Maintainability

Model

Class and
Interface nodes,
Delta of
LOC, WMC

View

LOC – height & y
WMC – color



Open Issues / Questions

- ⇒ Do we need new/more complex metrics for trends?
Metrics on history of metrics?
- ⇒ Conclusions from trends to future, are they valid?
- ⇒ What are relevant views?

Conclusion

⇒ ARiSA – First Contact Analysis

- Quantitative reasonable statistical control over internal product quality
- Internal quality of software entities related to absolute norm values or self related

⇒ ARiSA – Quality Monitor

- Quantitative basis for continuous internal product quality improvement
- Internal quality of software entities related to previous versions