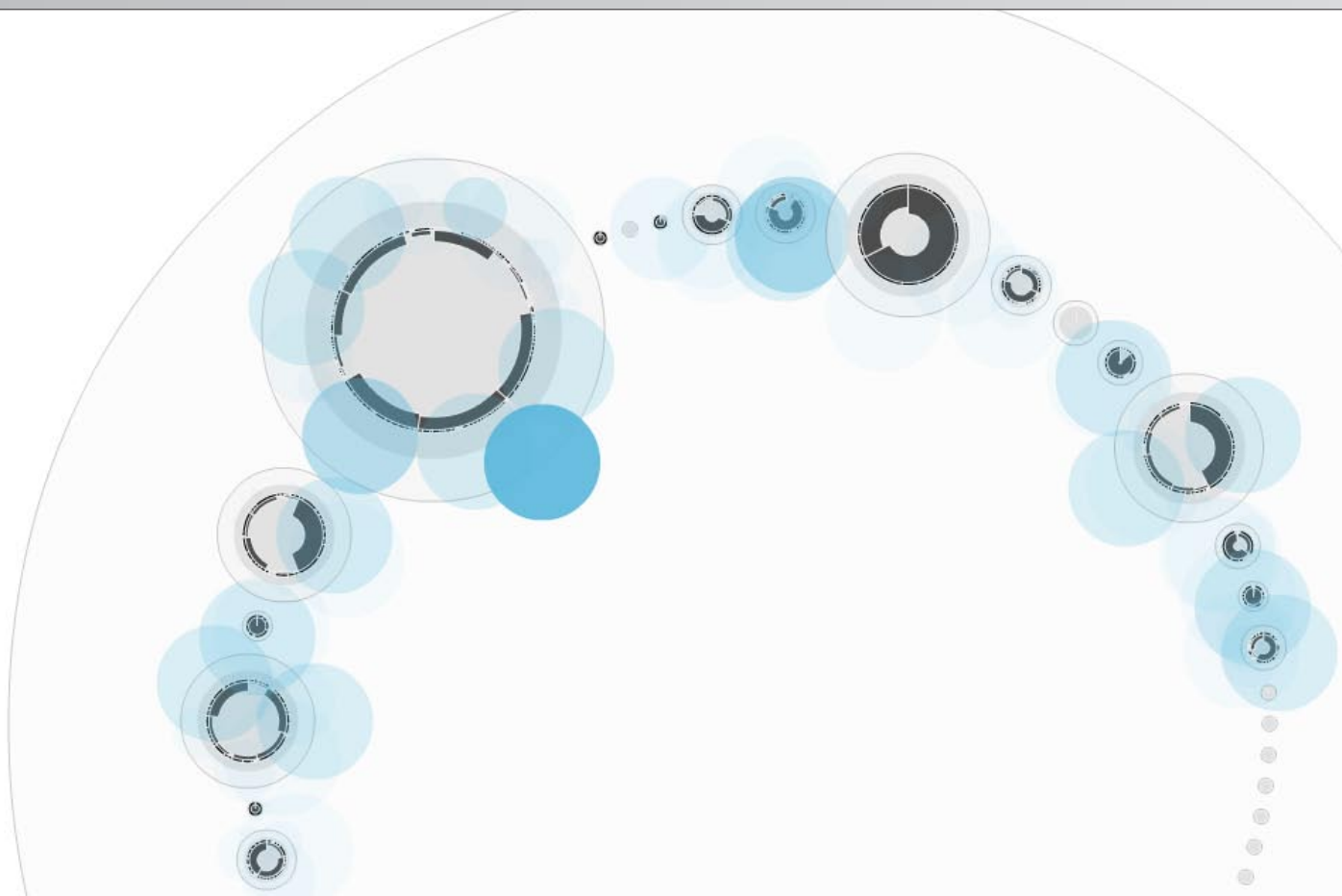
Dokumentation
ZHdK | IAD
5. Semester
HS 07/08

Modul: Applied Interaction I
Projekt: Software Visualisierung

Carlo Jörges, Mark Odermatt,
Peter Gassner, Christian Siegrist

Prof. Jürgen Späth, ZHdK IAD
Prof. Dr. Harald Gall, Uni Zürich IFI



Inhalt

Projektbeschrieb

Kurzbeschreibung:

Evaluation und Realisation von Software Visualisierungen.

Lernziele/Lerninhalte:

In Zusammenarbeit mit Informatikstudenten der Universität Zürich werden Visualisierungsansätze aus dem Kurs «Dynamic Data» evaluiert und mit Realdaten realisiert.

Die Erarbeitung von Lösungen in der Gruppe steht im Vordergrund.

Zielsetzungen der Gruppe:

Aufbauend auf dem Kurs «Dynamic Data» wird das Konzept von Christian Siegrist weiterentwickelt. Die Zielsetzung besteht darin, die entworfene Grundlage - das Softwarepaketzeichen - in ein Gesamtsystem zu übertragen und dieses mit Realdaten zu überprüfen.

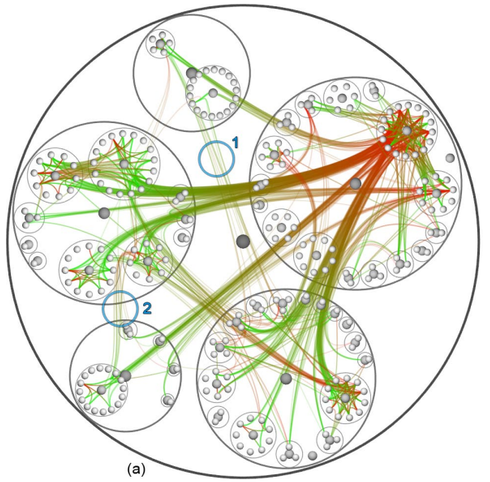
Beim Konzeptentwurf stellen wir die Abstraktion und Reduktion einer grossen Datenmenge durch Farbe und Form, die Filterung durch einfache Interaktionen und die Hervorhebung von relevanten Metriken durch Kontrastrierung in den Vordergrund.

Recherche

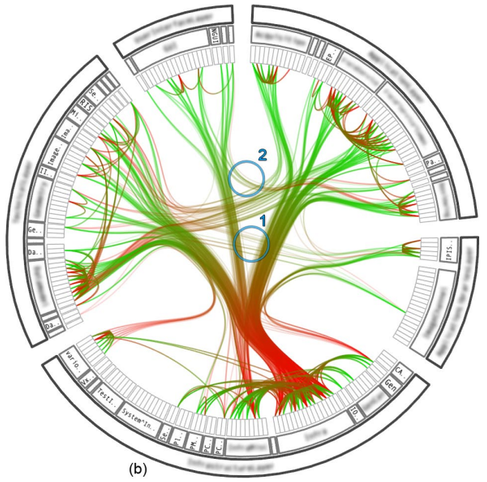
Hierarchical Edge Bundles:

«A software system and its associated call graph (caller = green, callee = red). (a) and (b) show the system with bundling strength b = 0.85 using a balloon layout (node labels disabled) and a radial layout, respectively. Bundling reduces visual clutter, making it easier to perceive the actual connections [...]. Bundled visualizations also show relations between sparsely connected systems more clearly (encircled regions); these are almost completely obscured in the non-bundled versions. The encircled regions highlight identical parts of the system for (a), (b).»

Holten, Danny: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, Seite 6.



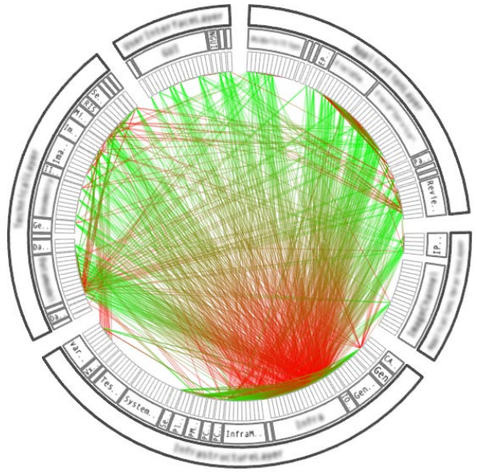
(a)



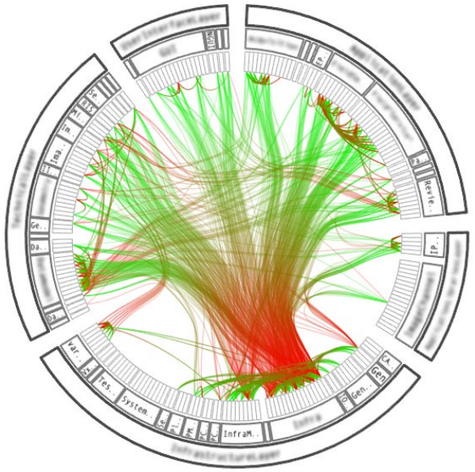
(b)

«Using the bundling strength b to provide a trade-off between low-level and high-level views of the adjacency relations. The value of beta increases from left-to-right; low values mainly provide low-level, node-to-node connectivity information, whereas high values provide high-level information as well by implicit visualization of adjacency edges between parent nodes that are the result of explicit adjacency edges between their respective child nodes.»

Holten, Danny: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, Seite 7.



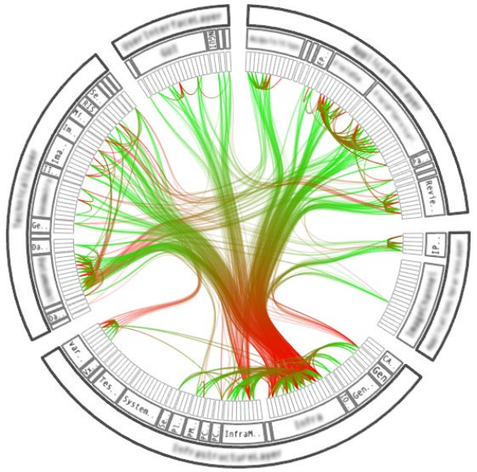
$\beta = 0$



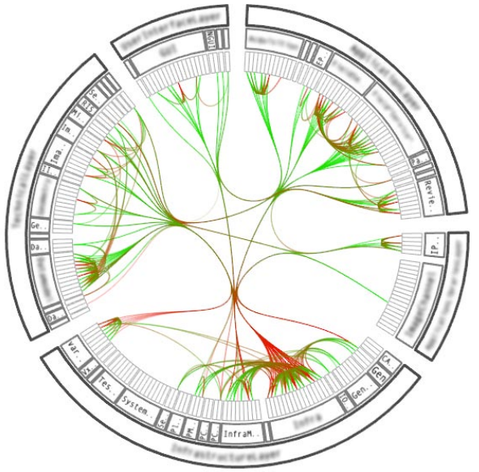
$\beta = 0.5$

«An important aspect of visualizing the curves is the order in which they are drawn. Since short curves only occupy a small amount of screen space, they tend to become obscured by long curves. This problem can be resolved by drawing short curves on top of long curves. In addition, we use alpha blending to further mphasize short curves by drawing long curves at a lower opacity than short curves.»

Holten, Danny: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, Seite 4.



$\beta = 0.75$

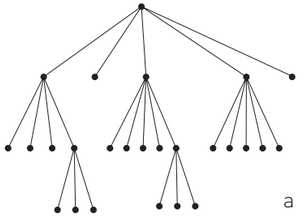


$\beta = 1$

Systementwicklung

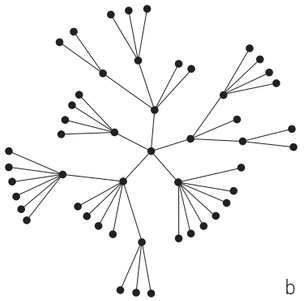
Baumdiagramm-Techniken:

Um die einzelnen Bestandteile einer Software zu visualisieren, muss deren hierarchische Struktur massgebend beachtet werden. Die einzelnen Methoden können zu Klassen zusammengefasst, die Klassen wiederum in verschiedene Unterpakete organisiert werden, welche auf ihrer obersten Stufe addiert das Gesamtsystem bilden.



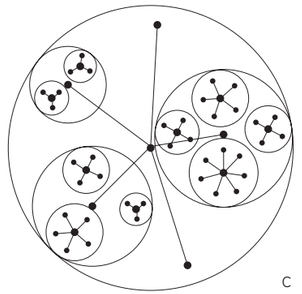
a

Um eine Software Visualisierung effizient zu gestalten, sollten die Aufrufe von Methode zu Methode und die wichtigsten Metriken jedes Elementes einer jeden Hierarchestufe dargestellt werden können. Es scheint zudem hilfreich, die Software ganzheitlich und der Orientierung wegen statisch abzubilden.



b

Aufgrund dieser Erkenntnisse und der erwartet grossen Anzahl von Bottom-Level Elementen einer komplexeren Software, scheint die Variante [c] am geeignetsten für die Visualisierung zu sein, im Gegensatz zu Variante [a], welche bei vielen Bottom-Level Elementen zu breit und unübersichtlich ausfallen würde. Variante [b] ist auf Grund der schlechten Erfassung der Hierarchiestufen problematisch.



c

Ausgangslage:

Im Modul «Dynamic Data» im 3. Semester IAD wurde das Grundkonzept des Repräsenanten für ein einzelnes Software-Paket geschaffen. Es besteht aus einem Ring von einzelnen Methoden und Funktionen. Der Segmentwinkel macht eine Aussage über die Anzahl Codelinien, welche für dieses Bottom-Level Element verwendet wurde.

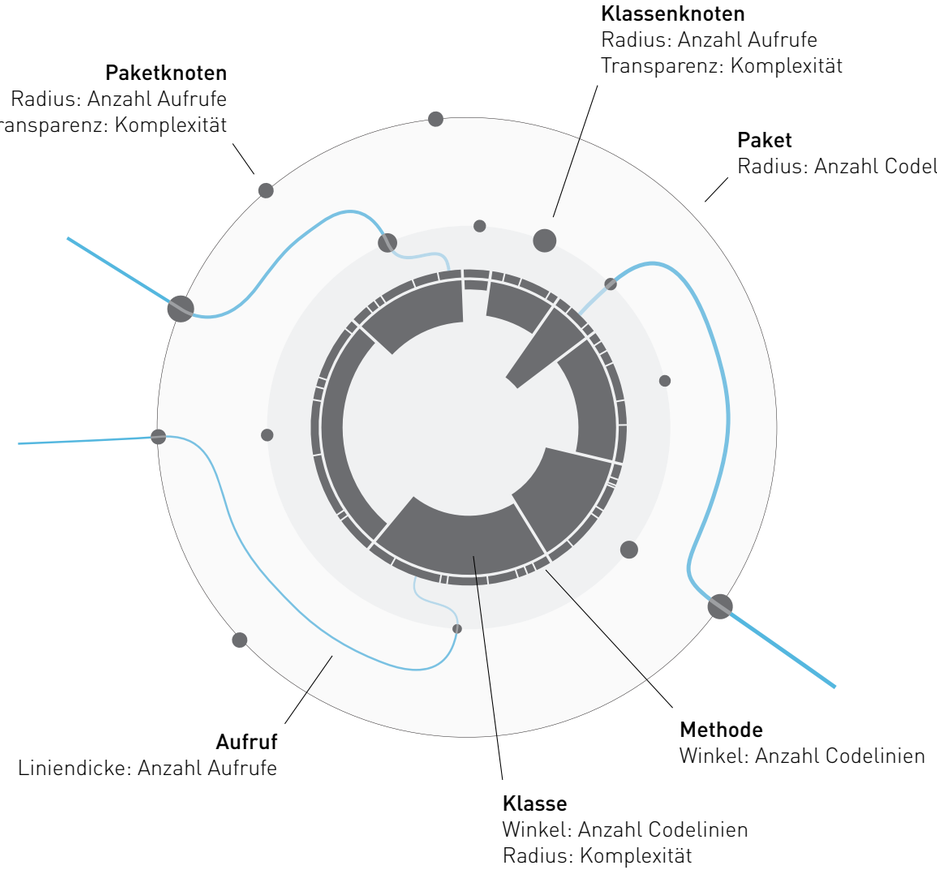
Die inneren, grauen und im Radius ungleichen Segmente fassen die kleinsten Elemente in Klassen zusammen. Der Radius wird durch die Klassenmetriken, wie die zyklomatische Komplexität, bestimmt.

Jeder Klasse wird auf einem äusseren Sphärenradius ein Klassenknoten zugeschrieben. Jede Kommunikation zwischen den Methoden muss durch die entsprechenden Knoten fliessen, um zum Empfänger zu gelangen.

Auf dem äussersten Sphärenradius liegen die Paketknoten. Falls der Kommunikationsstrang das Innere des Paketes verlassen muss um eine Methode eines anderen Paketes aufzurufen, wird der zusändige Paketknoten richtungsweisend.

Dem Radius eines Knotens kann die Anzahl der Kommunikationsstränge und seiner Transparenz die addierte, empfängerbedingte, zyklomatische Komplexität abgelesen werden. Mit diesem Konzept soll dem Rezipienten ermöglicht werden, eine Aussage über das Fehlerrisiko zu machen und so Besonderheiten in der Softwarearchitektur zu lokalisieren.

Die Herausforderung besteht nun darin, dieses einzelne Zeichen in ein Gesamtsystem zu bringen und das Verhältnis von Information und Abstraktion zu finden, welches das visuelle Erfassen einer Software am einfachsten und intuitivsten ermöglicht, ohne Verlust an Aussagekraft.



Clustering:

Die Intensität an Kommunikation zwischen den einzelnen Paketen organisiert die Zugehörigkeit zu den Subsystemen, dessen Anzahl der Rezipient vorhergehend definieren kann. So liegen diejenigen Pakete, welche die meissten Kommunikationsstränge zueinander haben, im selben Cluster.

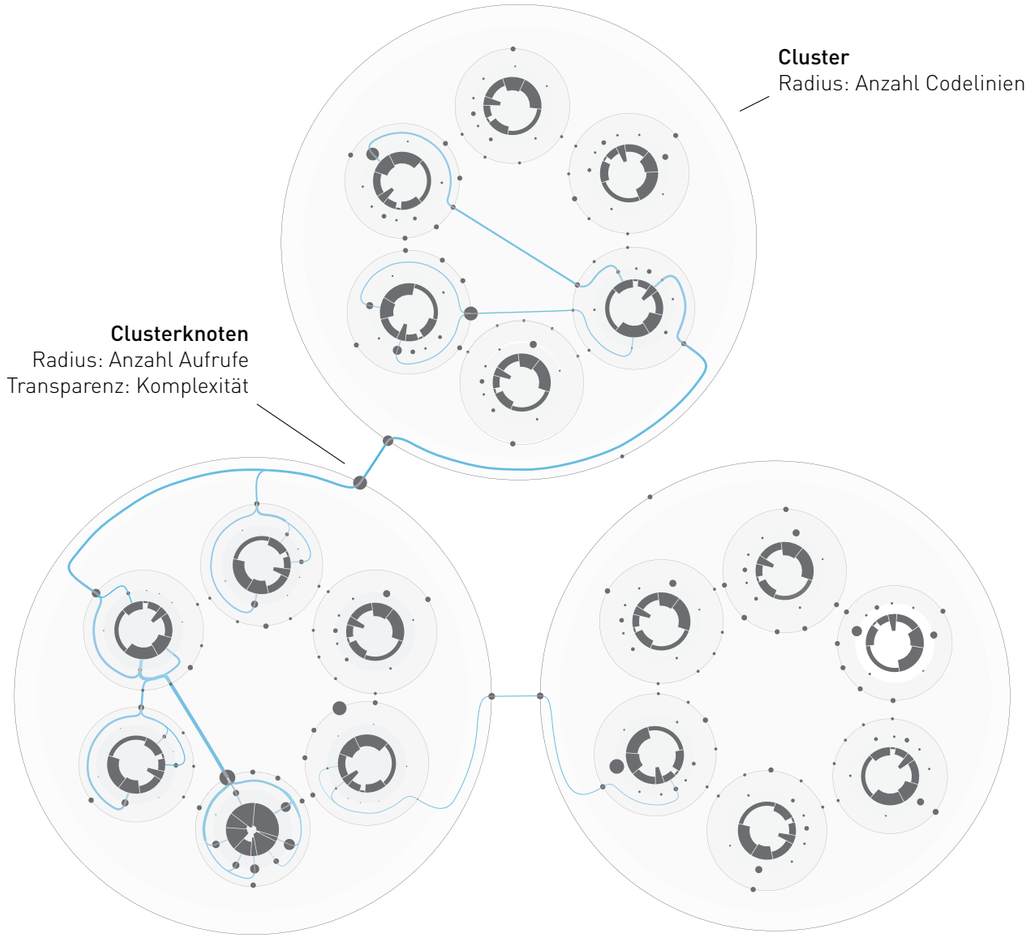
Jedem Paket und Cluster werden soviele Knoten zugewiesen, wie andere Elemente sich auf der selben Hierarchiestufe befinden. Diese ordnen sich auf dem äussersten Ring an und nehmen den Winkel zum benachbarten Element ein. Zudem wird dem Paket ein zusätzlicher Knoten, der für die Kommunikation zu den anderen Cluster zuständig ist, angefügt.

Die Bahnen für die Kommunikationsstränge verlaufen radial um die Pakete und Cluster. Um die Bahnen zu verlassen, muss der Strang durch den entsprechenden Knoten. Die Liniendicke repräsentiert die Anzahl der Aufrufe von Methode zu Methode.

Um die Informationsmenge zu reduzieren, sind die Methodensegmente bei der Systemansicht unsichtbar.

Durch den Grauwert sind auch bei vielen Paketen, diejenigen mit hoher Komplexität sehr gut erkennbar.

Der Grössenunterschied der Knoten ist noch zu gering, um gewichtige Informationen zu erhalten. Überprüft werden muss auch die hohe Zahl an Knoten, welche möglicherweise die Lesbarkeit erschweren.



Reduktion:

Die Zahl der Knoten wurde auf Paket- und Clusterebene auf einen einzigen reduziert. Dieser ist für jegliche Kommunikation nach Aussen zuständig.

Die Bahnen um die Cluster werden weggelassen, womit jegliche Kommunikation durch das Zentrum des Clusters geleitet wird.

Von der Zusammenfassung der Knoten wird eine Verbesserung der Lesbarkeit der Informationen erwartet, wobei jedoch so auf die Aussage des Fehlerrisikos zwischen den einzelnen Elementen verzichtet und als Pauschal angesehen werden muss.

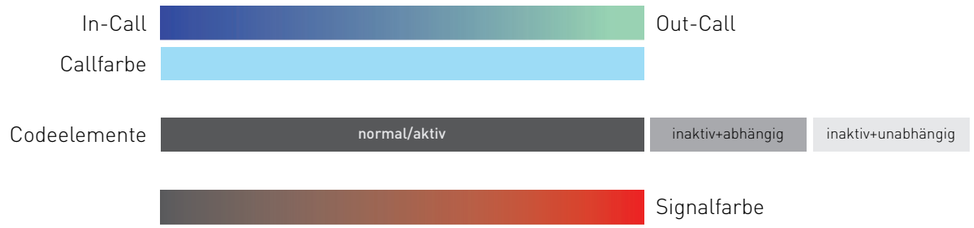
Farbmethodik

Kontraste:

Die Codeelemente, also die Methoden, Klassen und Pakete werden in Grauwerten dargestellt.

Als Kontrast zu den Codeelementen werden die Calls in Farbe dargestellt. Das kalte Hellblau zeigt lediglich die Bahn des Aufrufes, der Farbverlauf differenziert den Sender und den Empfänger.

Als Kontrast zu den Calls und Code-elementen, werden die verschiedenen Metriken je nach ihrer Gewichtung durch Beimischung der warmen Signalfarbe zur Grundfarbe hervorgehoben.



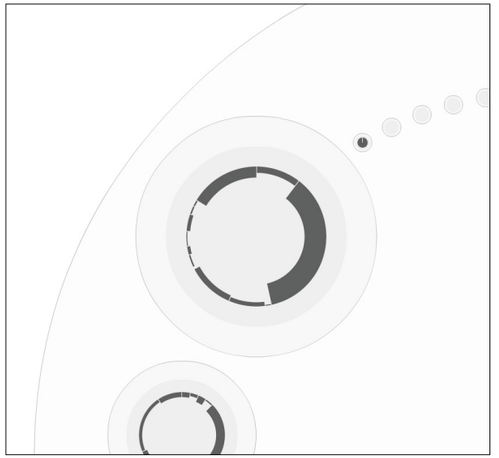
User Interface

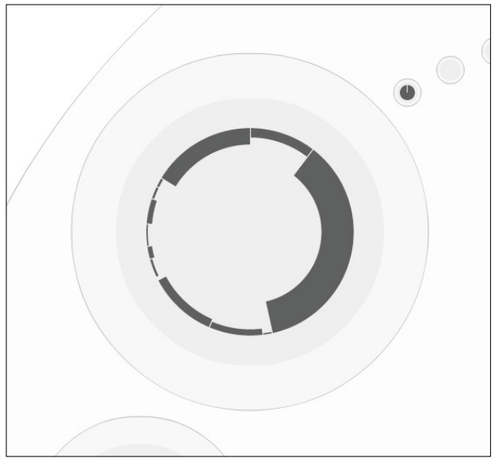
Softwarearchitektur:

Die Visualisierung der Softwarearchitektur durch die einzelnen Hierarchiestufen stellt das Zentrum dar. Sie dient zur Orientierung im Gesamtsystem und als erste Instanz, um die Architektur auf ihre Qualität zu prüfen.

Zoomstufen:

Die Visualisierung der Software kann durch verschiedenen Zoomstufen genauer untersucht werden. So wird ermöglicht, dass auch kleinere Elemente überprüf- und auswählbar sind.





Filter:

Der Rezipient kann mittels Dropdown Menu auf verschiedene Filterarten zugreiffen.

Das Filtern selbst wird durch einen Schieberegler ermöglicht. In seiner horizontalen Achse kann der Auswahlbereich verschoben, durch eine Mausbewegung in vertikaler Richtung skaliert werden.

Sobald ein Filter aktiv ist, wird durch einen zweiten, noch passiven Balken einen weiteren Filter angeboten. Die Filter können jederzeit entfernt oder ausgeblendet werden.

Farbindex:

Die Farbindizes können wie die Filter per Dropdown Menu ausgewählt werden.

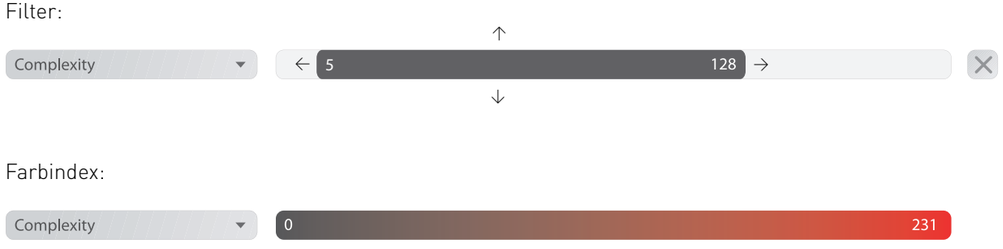
Der Balken kann hier nicht verändert werden und dient nur als Indikator für die Einfärbung der Softwareelemente.

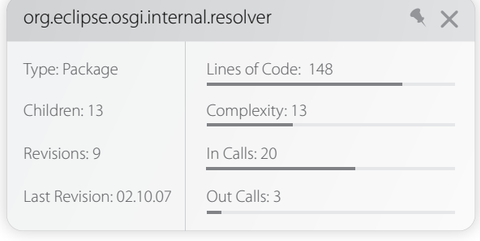
Die Farbindizes können nur mit Filter oder dem In-/Outcall Farbindex kombiniert werden, welcher ausserhalb dieses Menüs angeboten wird.

Detailinformationen:

Durch die Auswahl eines Softwareelementes werden seine Detailinformationen in einem Widgetartigen Werkzeug angezeigt.

Es können mehrere Info-Widgets auf dem Bildschirm belassen werden und so als Vergleichsinstrument einzelner Elemente und als Verknüpfung dienen.





Interaktion

Hervorheben:

Metriken können durch einen Farbindex, respektive durch Beimischung der Signalfarbe Rot ihrer Gewichtung nach hervorgehoben werden. Folgende Farbindizes können angewendet werden:

- Komplexitätsfarbindex
- Aktualitätsfarbindex
- Fehlerrisikofarbindex

Die Farbindizes können nur einzeln verwendet werden, jedoch parallel zu den Filtermöglichkeiten.

Ein weiterer Farbindex differenziert die Sender- und Empfängerseite eines Kommunikationsstranges durch einen Farbverlauf:

- In-Outcall-Farbindex

Dieser Index kann als einziger mit jedem Filter und Index kombiniert werden.

Filtern:

Die erste Filtermöglichkeit ergiebt sich, sobald man ein Paket, eine Klasse oder ein Knoten durch Anklicken auswählt. Dabei werden alle inaktive, unabhängige Elemente ausgeblendet und alle inaktive, aber abhängige Elemente leicht in der Transparent reduziert.

Weitere Filtermöglichkeiten bieten sich durch Schieberegler, welche alle parallel angewendet werden können:

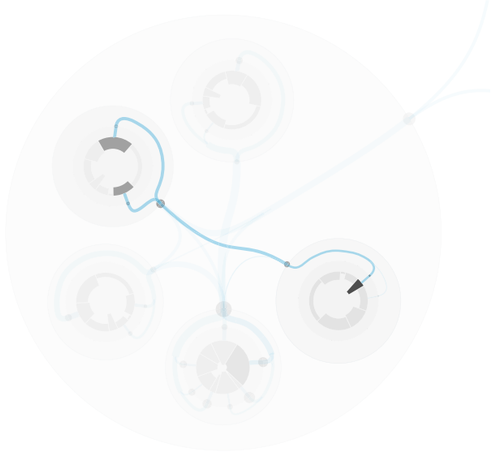
- Komplexitätsfilter
- Aktualitätsfilter
- Fehlerrisikofilter
- Callfilter

Paketauswahl:

Die einzelnen Pakete können ausgewählt werden, wobei ausser den abhängigen Elementen alles ausgeblendet wird. So hat der Rezipient die Möglichkeit, sich ein Paket und seine aufgerufenen oder aufrufenden Klassen und alle tangierten Knoten zu untersuchen.

Klassenauswahl:

Wenn sich der Rezipient auf nur eine Klasse und seine abhängigen Elementen konzentrieren möchte, kann dies auch durch Anklicken erreicht werden.



Knotenauswahl:

Wenn ein einzelner Knoten ausgewählt wird, zeigen sich nur diejenigen Calls, welche durch ihn fliessen und diejenigen Klassen, welche die Calls bedingen. So kann als Beispiel ein Paket auf seine Kommunikation nach Aussen untersucht werden.

Komplexitätsfarbindex:

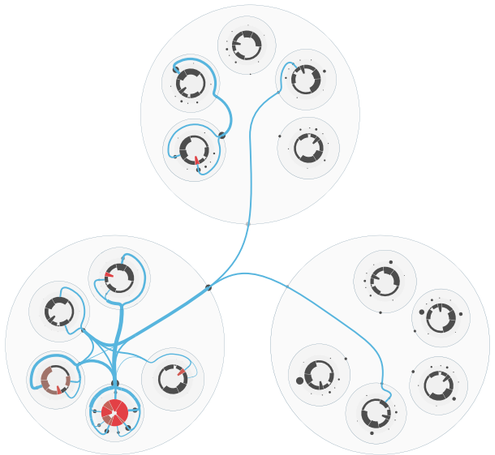
Durch das Anwenden des Komplexitätsfarbindex werden den Klassen relativ zu der Klasse mit grösster zyklomatischer Komplexität ein Rotfarbwert zugewiesen und so die Extremwerte hervorgehoben, was die Lesbarkeit unterstützt. Als kleinster Wert wird die Komplexität 0 verwendet.

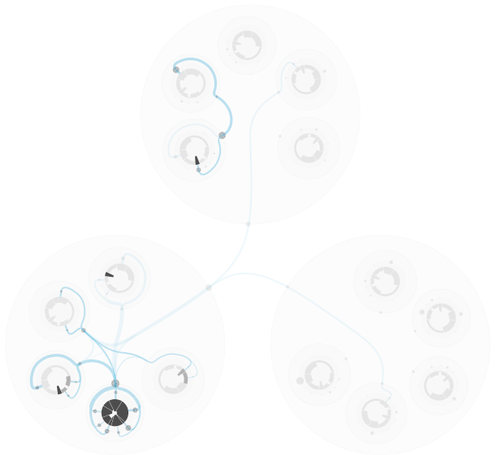
Komplexitätsfilter:

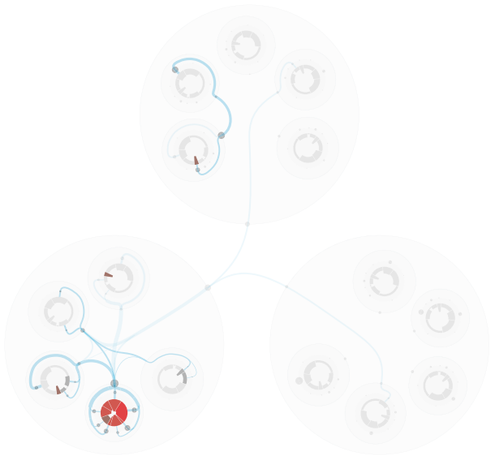
Durch das Anwenden des Komplexitätsfilters werden nur die Klassen angezeigt, die in einem gewissen, vom Rezipienten definierten Komplexitätswertebereich liegen. Die abhängigen Kommunikationsstränge, deren aufgerufenen oder aufrufenden Klassen und alle tangierten Knoten werden in einer leicht reduzierten Transparenz dargestellt.

Kombination:

Der Komplexitätsfarbindex kann mit jedem beliebigen Filter kombiniert werden. Der absoluten Komplexitätswert wird nun durch diejenige, noch eingeblendete Klasse mit der grössten Komplexität, ermittelt und alle anderen Rotwerte relativ angepasst.

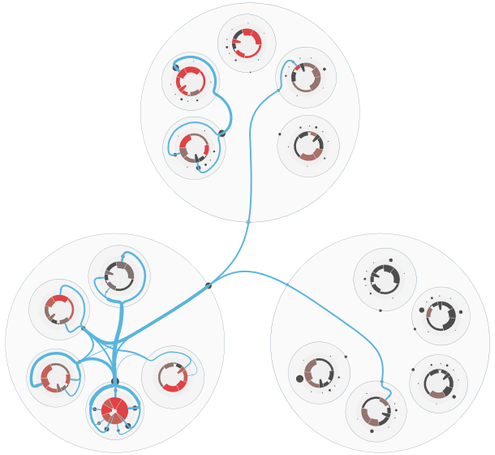






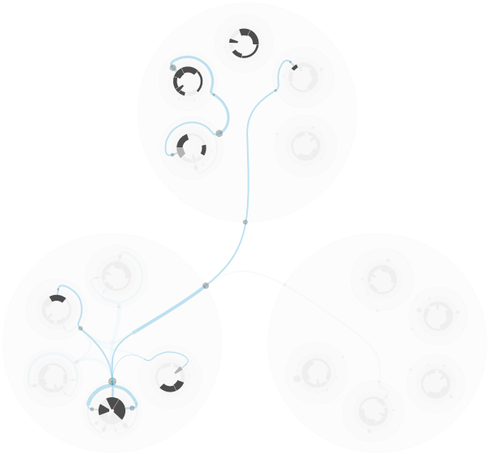
Aktualitätsfarbindex:

Durch das Anwenden des Aktualitätsfilters werden den Klassen relativ zu der Klasse mit aktuellstem Bearbeitungsdatum ein Rotfarbwert zugewiesen. Als kleinster Wert wird das älteste Bearbeitungsdatum verwendet.



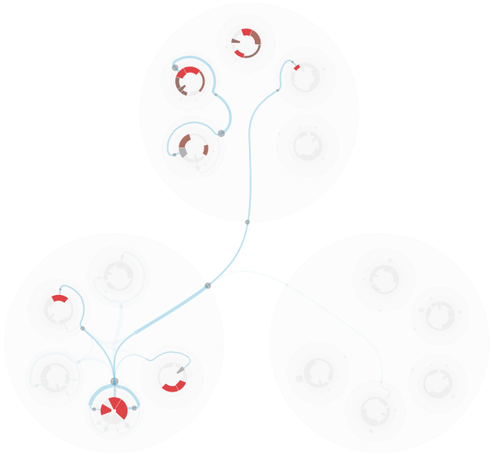
Aktualitätsfilter:

Durch das Anwenden des Aktualitätsfilters werden nur diejenigen Klassen angezeigt, die während eines bestimmten Zeitbereiches zum letzten Mal bearbeitet wurden. So können als Beispiel die aktuellsten oder die ältesten Klassen untersucht werden. Die abhängigen Kommunikationsstränge, deren aufgerufenen oder aufrufenden Klassen und alle tangierten Knoten werden in einer leicht reduzierten Transparenz dargestellt.



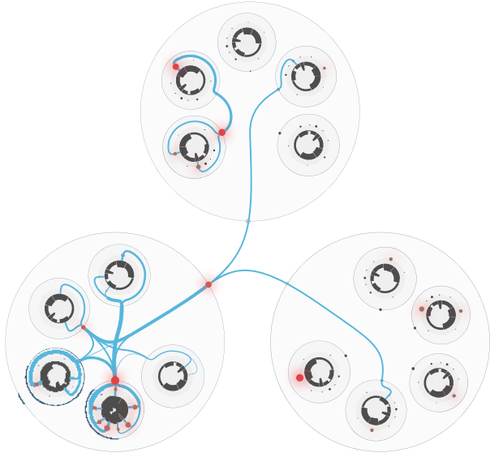
Kombination:

Der Aktualitätsfarbindex kann mit jedem beliebigen Filter kombiniert werden. Der absolute Wert wird nun das Datum der noch eingeblendeten, aktuellsten Klasse und alle anderen Rotwerte werden relativ angepasst.



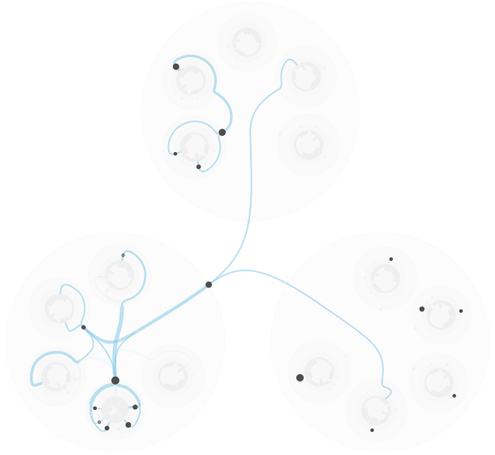
Fehlerrisikofarbindex:

Durch das Anwenden des Fehlerrisikofarbindex werden die Knoten relativ zu dem Knoten mit dem höchsten Fehlerrisiko ein Rotfarbwert zugewiesen. Das Fehlerrisiko wird durch die Anzahl der Aufrufe und deren addierten, empfänger bedingten zyklomatischen Komplexität berechnet. Als kleinster Wert wird das Fehlerrisiko 0 verwendet.



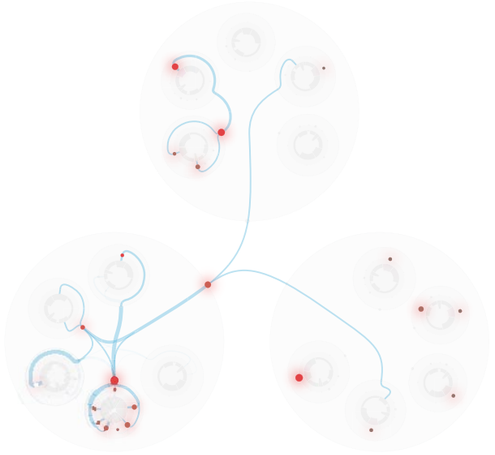
Fehlerrisikofilter:

Durch das Anwenden des Fehlerrisikofilters werden nur diejenigen Knoten angezeigt, die innerhalb eines, vom Rezipienten festgelegten Fehlerrisikobereichs liegen. Die Kommunikationssränge, die durch diese Knoten fliessen, deren aufgerufenen oder aufrufenden Klassen und alle tangierten Knoten werden in einer leicht reduzierten Transparenz dargestellt.



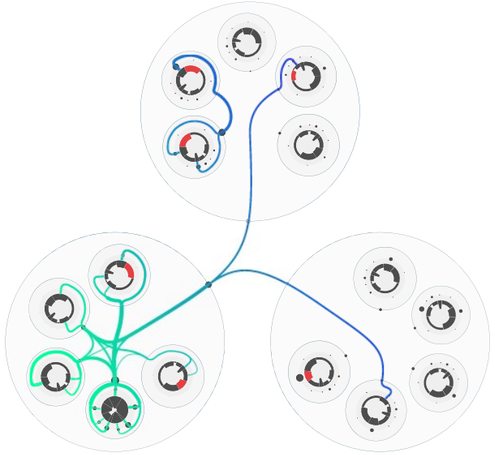
Kombination:

Der Fehlerrisikofarbindex kann mit jedem beliebigen Filter kombiniert werden. Das absoluten Fehlerrisiko wird nun durch denjenigen, noch eingeblendeten Knoten mit dem grössten Risiko, ermittelt und alle anderen Rotwerte relativ angepasst.



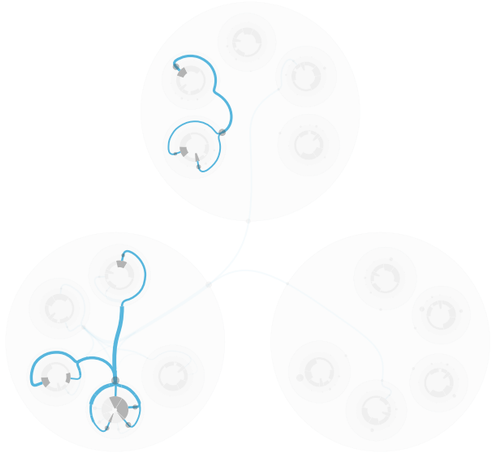
In-/Outcall-Farbindex:

Durch das Anwenden des In-/Outcall-Farbindex wird die Sender-Empfänger Bezieghung der Aufrufe sichtbar gemacht. Die hellere und leuchterende Seite des Farbverlaufs repräsentiert die Aktion, also den Outcall, die dunklere Seite den Incall, welcher dann als Reaktion einen Wert zurückgibt.



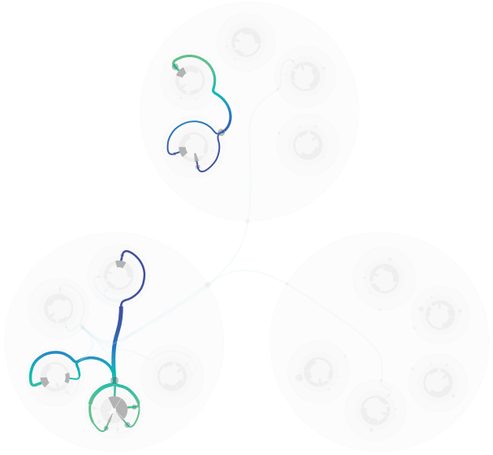
Callfilter:

Durch das Anwenden des Callfilters werden nur diejenigen Kommunikationsstränge angezeigt, welche innerhalb eines, vom Rezipienten definierten Aufrufeanzahlbereiches liegen.



Kombination:

Der In-/Outcall-Farbindex kann mit jedem beliebigen Filter und Farbinex kombiniert werden.

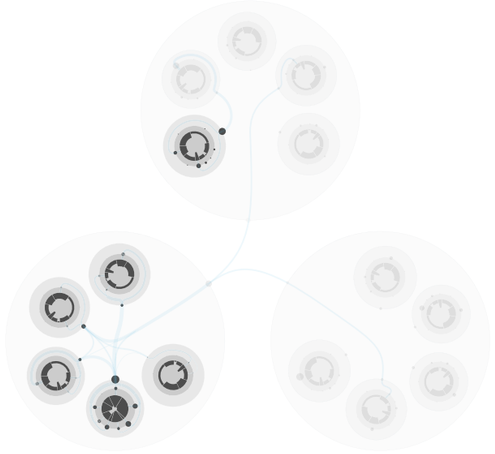


Weitere Interaktionsmöglichkeiten:

Die infolge vorgestellten Interaktionen sind auf rein konzeptioneller Ebene entwickelt und würden noch weiteren Überlegungen bedürfen, um sie in der Anwendung miteinzubauen.

Clusterzugehörigkeit:

Die von den Softwareentwicklern benutzte Projekt- und Ordnerstruktur ergibt die Softwarearchitektur. Diese kann auf ihre Qualität überprüft werden, indem alle Softwarepakete derselben, von den Entwicklern definierten Clusterzugehörigkeit angezeigt werden. Für die Clusterbildung unserer Software Visualisierung verwenden wir die Kommunikationsintensität zwischen den einzelnen Paketen als massgebenden Parameter. Wenn die beiden Strukturen nicht übereinstimmen und ein einzelnes Paket oder mehrere ausserhalb des Clusters liegen, kann möglicherweise über eine Restrukturierung der Architektur besser entschieden werden.



Vererbungsstruktur:

Eine weitere Idee ist die Visualisierung der Vererbungsstruktur. Doch kann diese nicht wie dargestellt auf Paketebene funktionieren, da nur Klassen ihre Eigenschaften vererben können.

Altersstruktur:

- Zusammenhang zwischen Komplexität und Alter von Paketen.
- Zusammenhang zwischen Konnektivität und Alter von Paketen.

Fragestellungen:
- Wie alt ist das System?
- Gibt es Pakete, die lange nicht mehr bearbeitet wurden, im Gesamtsystem aber wichtig sind?
- Scheuen die Entwickler die Komplexität von alten Paketen?
- Werden alte Pakete isoliert oder nur noch zum Teil genutzt?

Komplexität:

- Höhe des Kreissegments zeigt die Komplexität der Klassen eines Paketes.
- Komplexität als Indikator für Fehleranfälligkeit.

Fragestellungen:
- Welche Module sind komplex, welche nicht?
- Muss bestimmten Paketen mehr Aufmerksamkeit geschenkt werden, da sie komplex sind und mit vielen anderen Paketen in Verbindung stehen?
- Wie fehleranfällig sind bestimmte Bereiche eines Systems?
- Sind bestimmte Klassen komplex, aber schlecht strukturiert; das heisst, die Klasse hat nur wenige unterscheidbare Abschnitte?

Calls zwischen Paketen, Klassen, Methoden:

- Verbindungen visualisieren Aufrufe von einzelnen Elementen untereinander.
- Dicke der Verbindung symbolisiert Häufigkeit der Aufrufe.

Fragestellungen:
- Welche Pakete stehen im Zentrum der Kommunikation?
- Kommunizieren bestimmte Pakete nur untereinander?
- Gibt es Klassen, die als Kommunikator zur Aussenwelt fungieren (Kapselung)?

Clusterung/Paketzugehörigkeit:

- Die Pakete werden nach der Häufigkeit der Aufrufe geclustert.
- Die Anzahl der Cluster wird entweder vom Benutzer eingestellt, oder durch das Programm optimiert.
- Im Interface werden die Packages des Sourcecodes angezeigt. Wenn ein Package ausgewählt wird, werden die entsprechenden Pakete gehighlightet.

Fragestellungen:
- Was ist die optimale Paket-Anzahl für das System?
- Ist das System gut gekapselt?
- Kommunizieren bestimmte Pakete viel mit Paketen eines anderen Clusters, daher schlechte Paketaufteilung?

Allgemeine Funktionen des Interface:

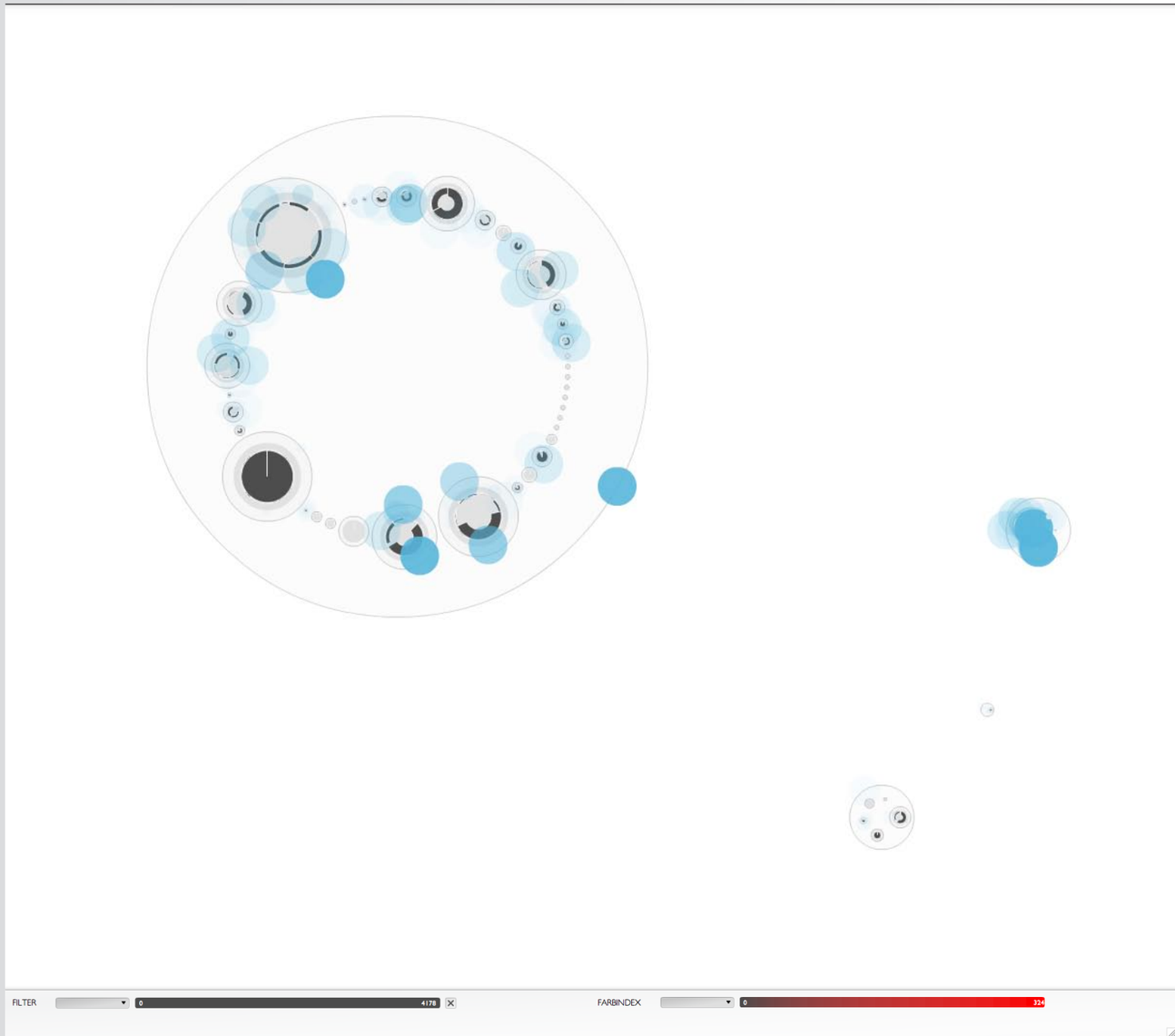
Semantiches Zooming:
Detaillierung der Information steigt mit der Vergrößerung der Visualisierung. Verbindungen und Codesegmente werden bei großem Zoom angezeigt.

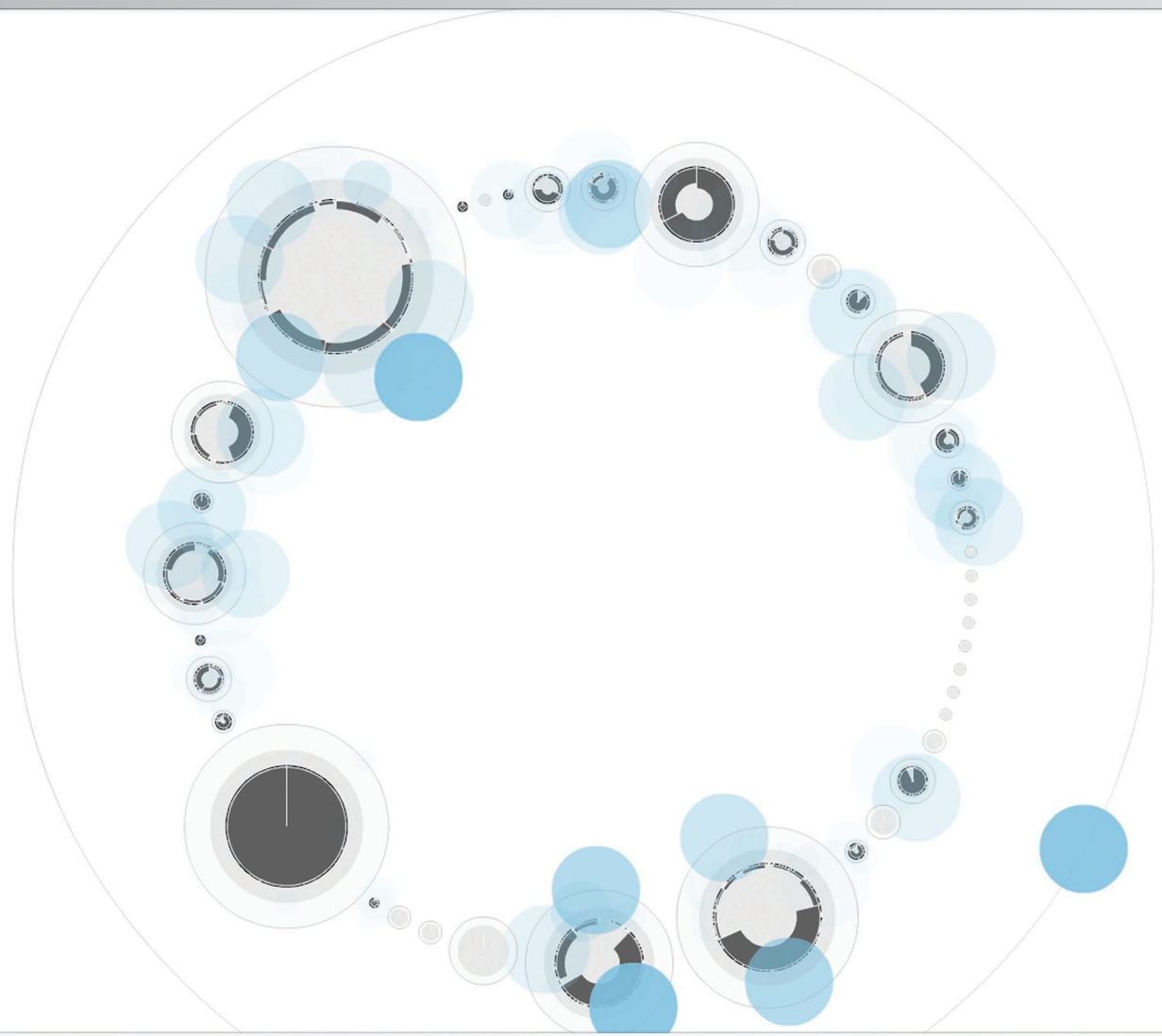
Highlighting:
Pakete, Klassen und Verbindungen sind auswählbar und zeigen dann ihren Zusammenhang durch IN/OUT Calls im System.

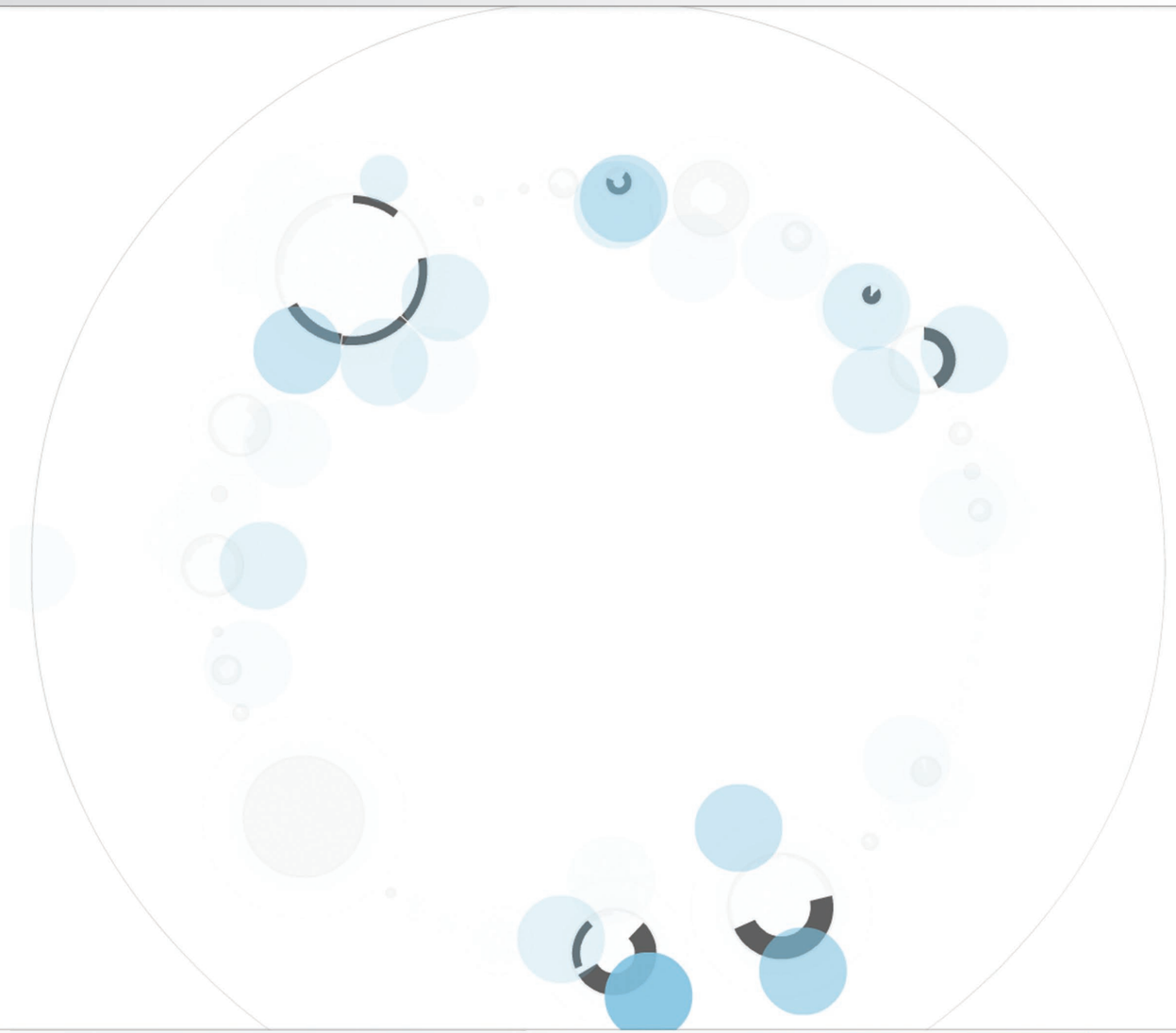
Suche:
Suche nach bestimmten Pakete oder Klassen durch Texteingabe.

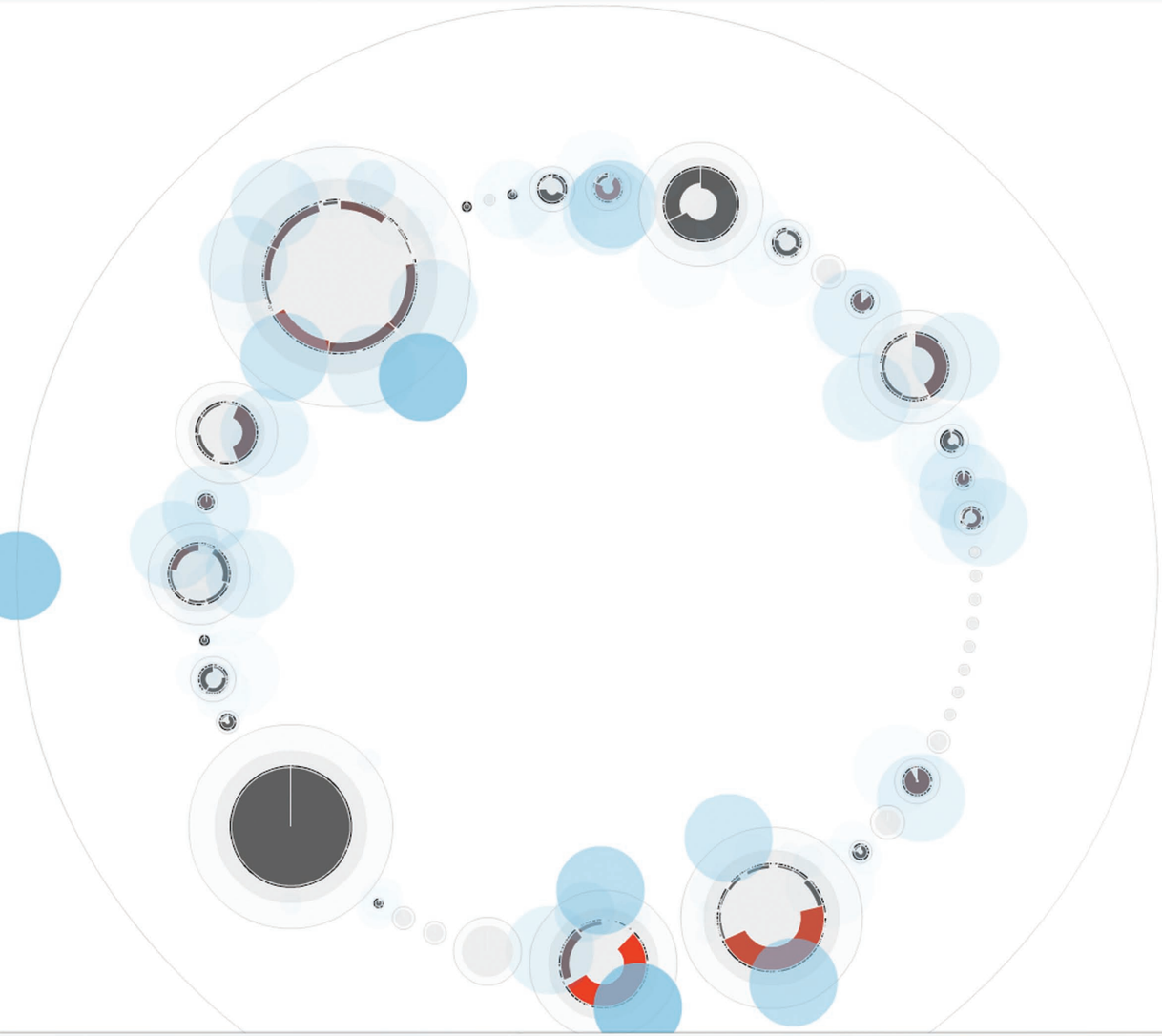
Code-Anzeige:
bei entsprechend großem Zoom kann der Code eines Segments angezeigt werden.

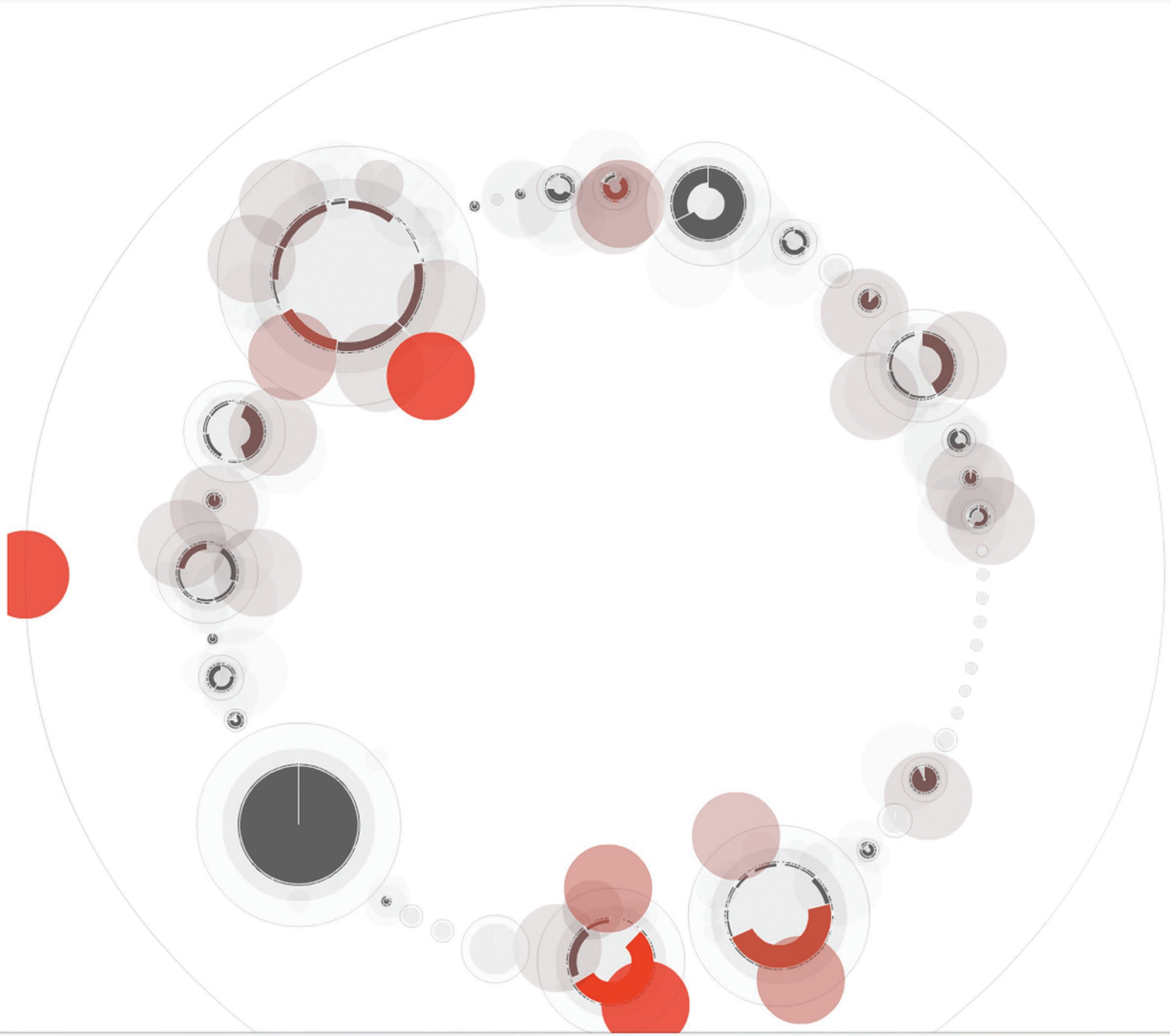
Prototyp

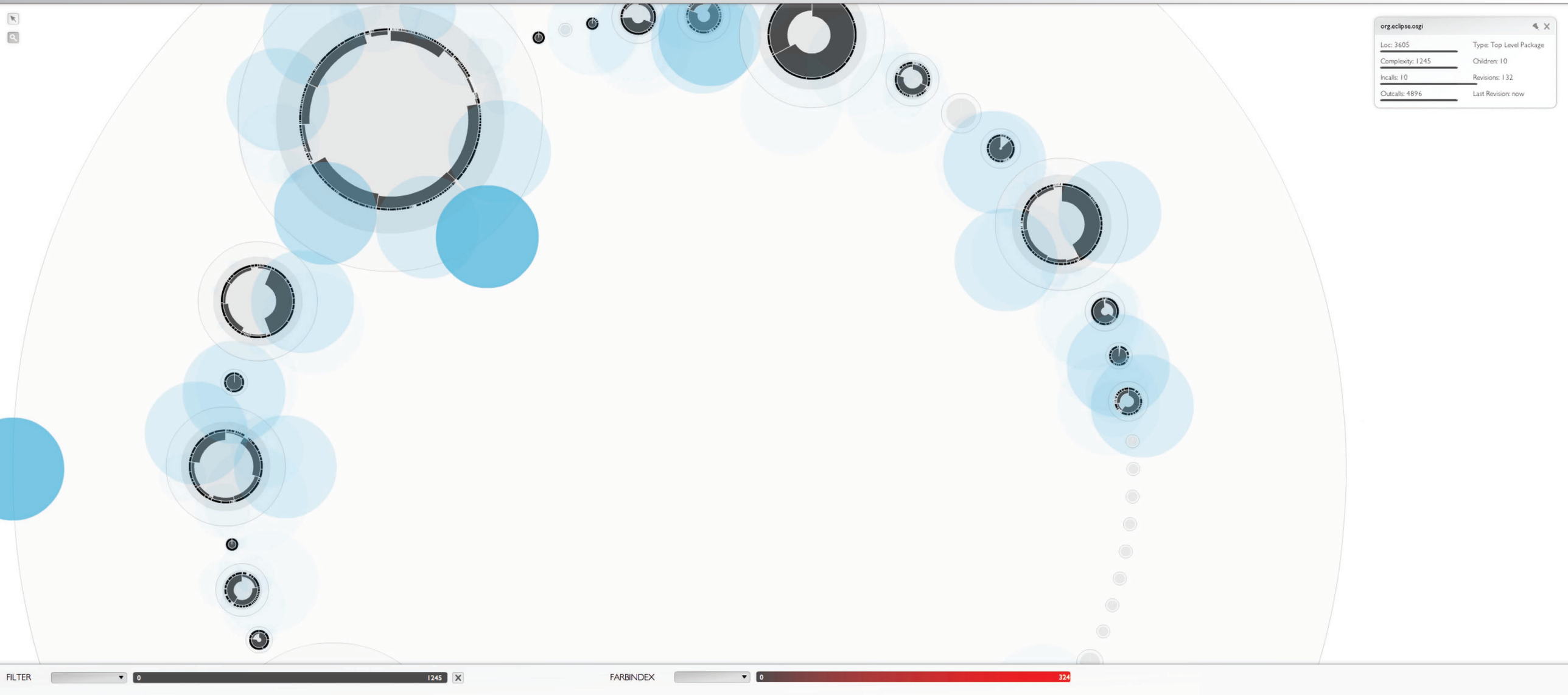


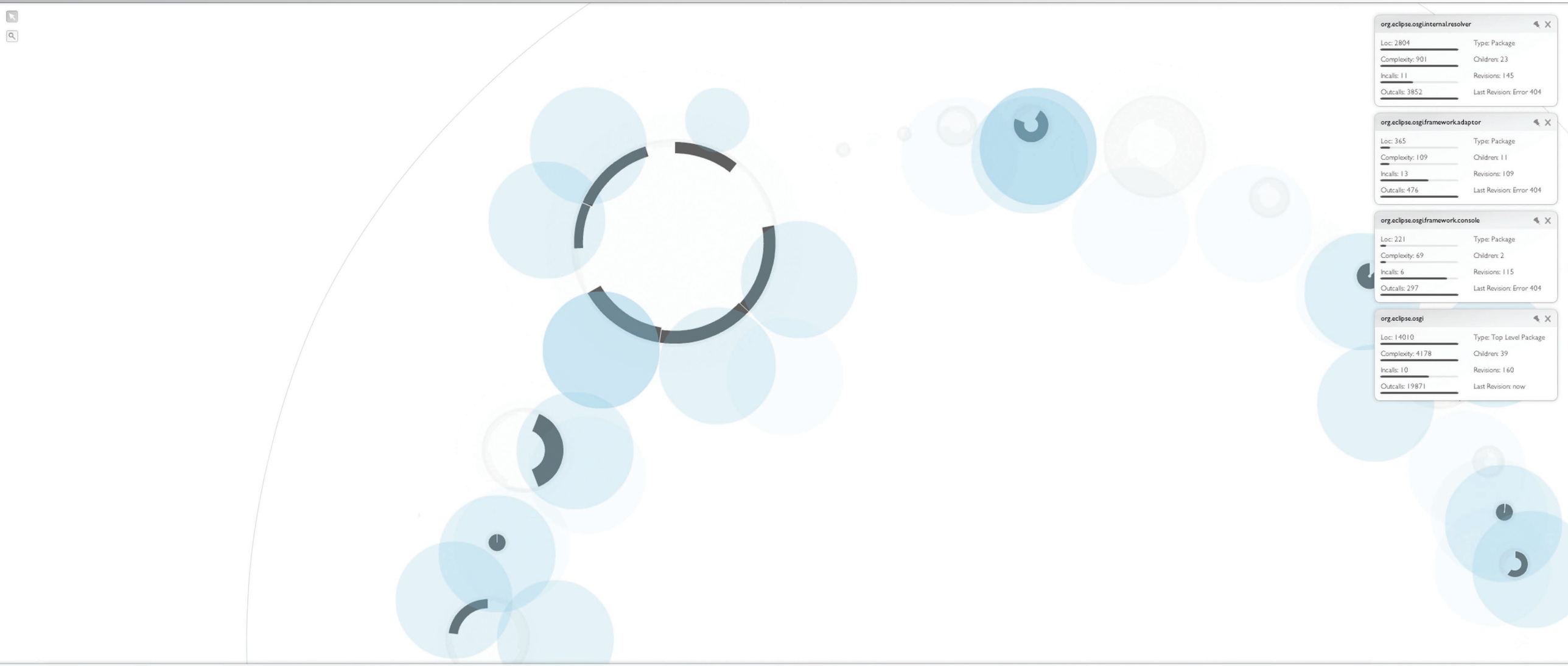

FILTER
0
4178
FARBINDEX
0
324

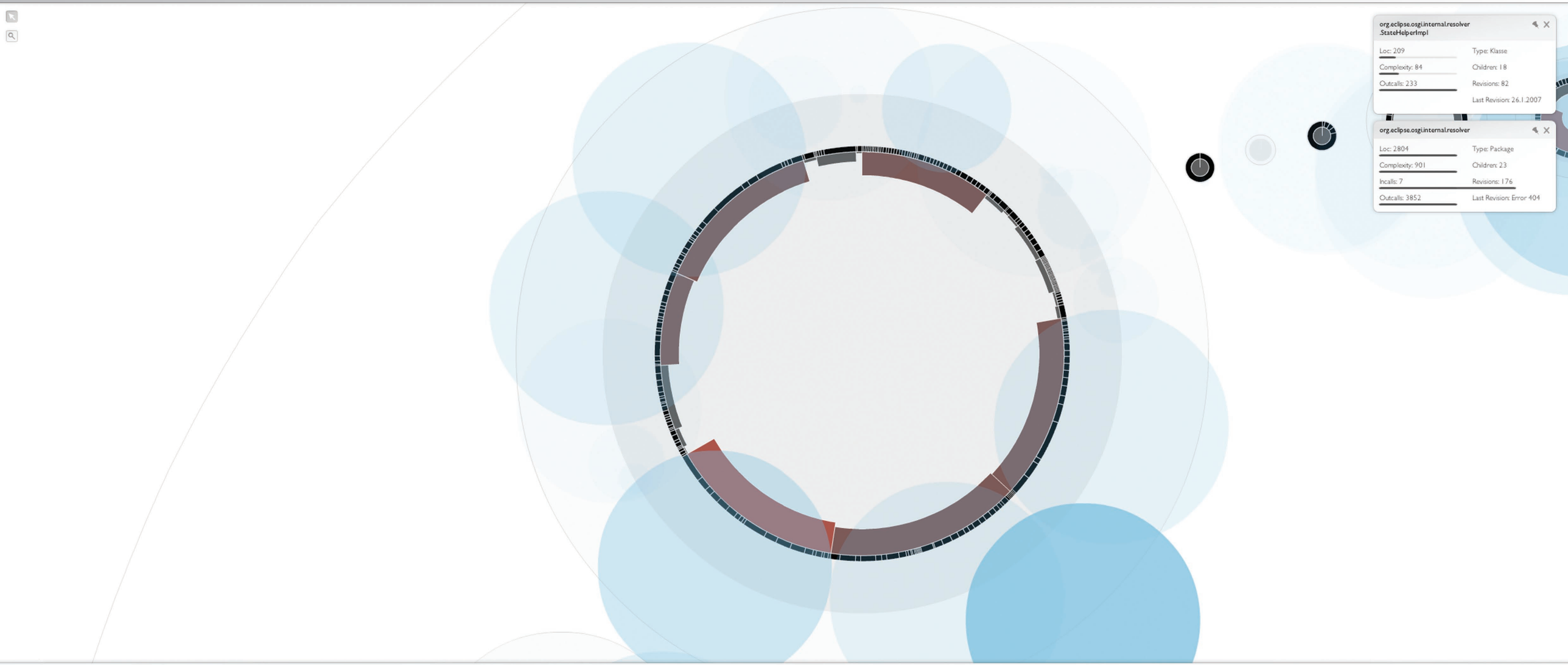

FILTER
100518
FARBINDEX
0
324

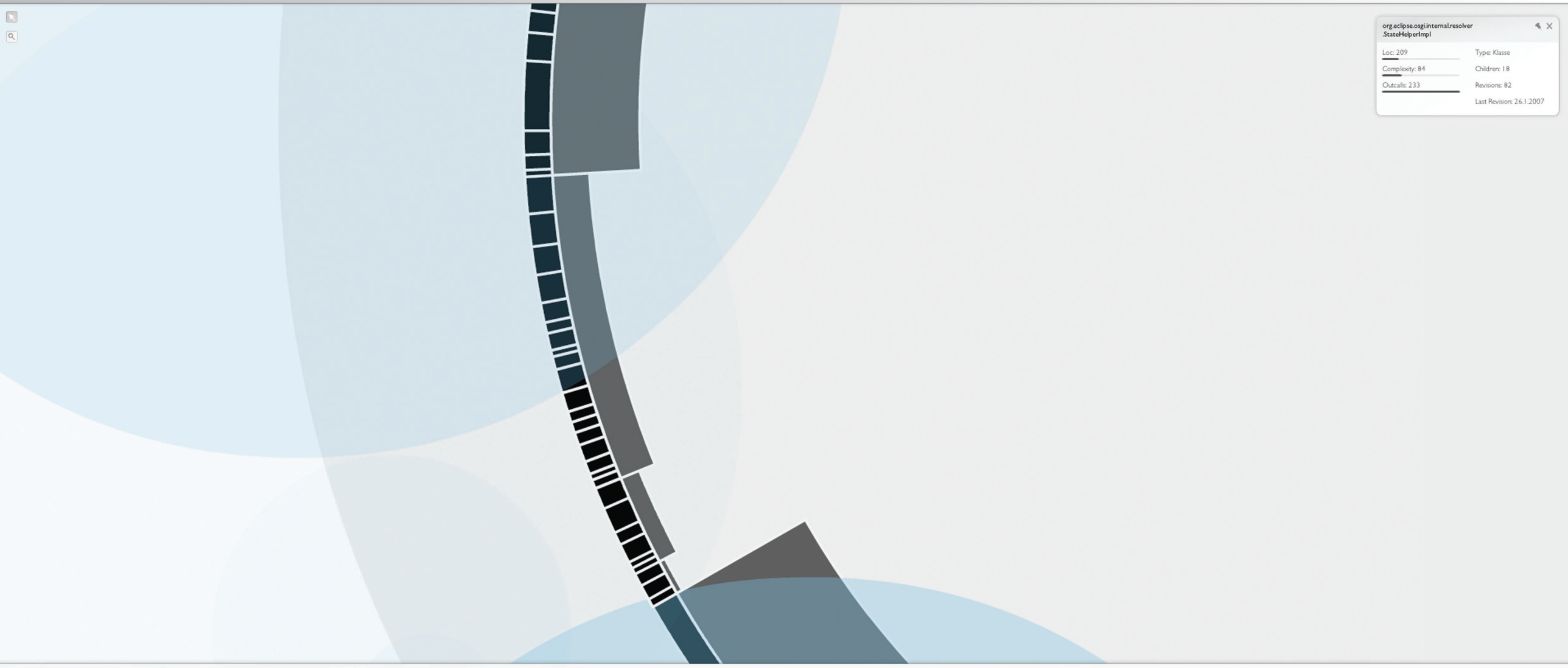

FILTER
100518
FARBINDEX
Complexity
0
324

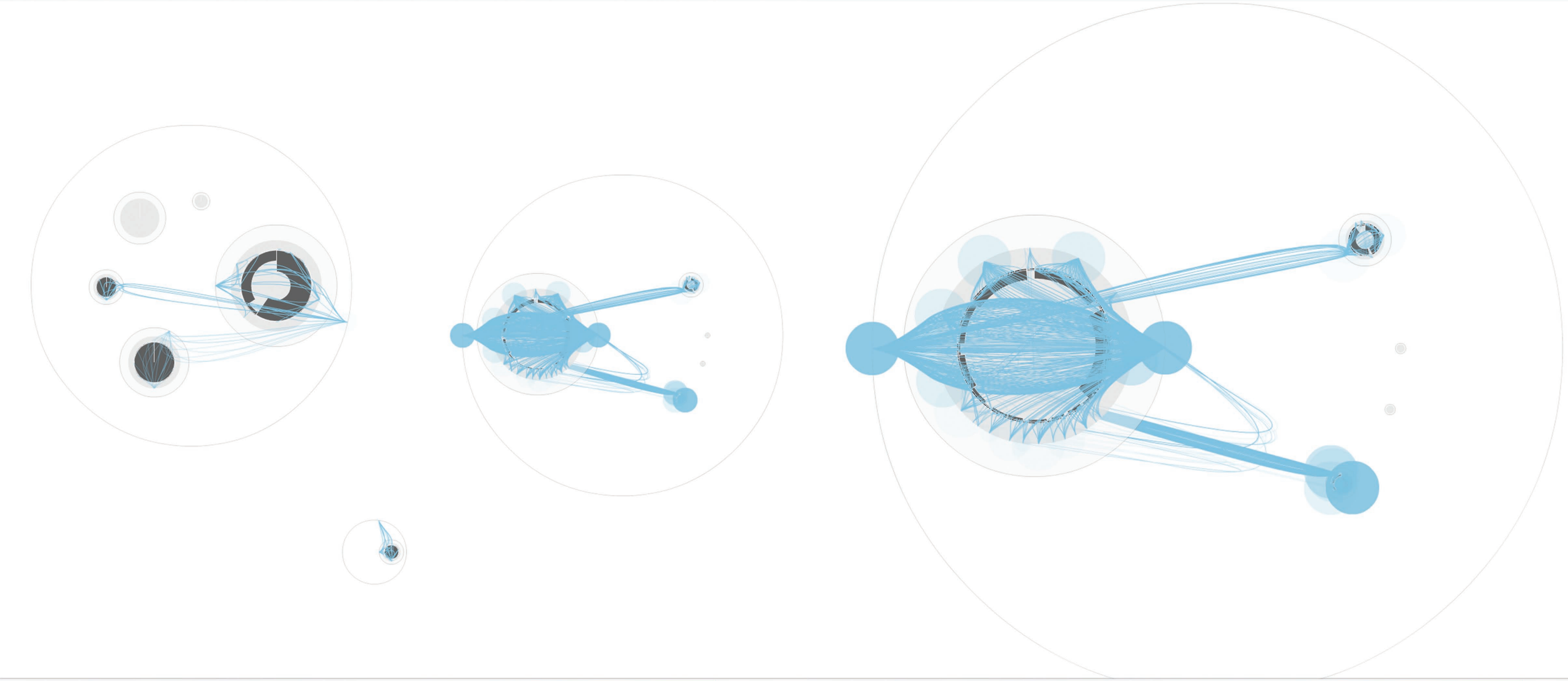

FILTER
0
4153
FARBINDEX
Bugrisk
0
324


org.eclipse.osgi
Loc: 3605
Type: Top Level Package
Complexity: 1245
Children: 10
Incalls: 10
Revisions: 132
Outcalls: 4896
Last Revision: now
FILTER
0
1245
FARBINDEX
0
324


org.eclipse.osgi.internal.resolver
Loc: 2804
Type: Package
Complexity: 901
Children: 23
Incalls: 11
Revisions: 145
Outcalls: 3852
Last Revision: Error 404
org.eclipse.osgi.framework.adaptor
Loc: 365
Type: Package
Complexity: 109
Children: 11
Incalls: 13
Revisions: 109
Outcalls: 476
Last Revision: Error 404
org.eclipse.osgi.framework.console
Loc: 221
Type: Package
Complexity: 69
Children: 2
Incalls: 6
Revisions: 115
Outcalls: 297
Last Revision: Error 404
org.eclipse.osgi
Loc: 14010
Type: Top Level Package
Complexity: 4178
Children: 39
Incalls: 10
Revisions: 160
Outcalls: 19871
Last Revision: now
FILTER
84 518
FARBINDEX
0
324


org.eclipse.osgi.internal.resolver
.StateHelperImpl
Loc: 209
Type: Klasse
Complexity: 84
Children: 18
Outcalls: 233
Revisions: 82
Last Revision: 26.1.2007
org.eclipse.osgi.internal.resolver
Loc: 2804
Type: Package
Complexity: 901
Children: 23
Incalls: 7
Revisions: 176
Outcalls: 3852
Last Revision: Error 404
FILTER
0
1245
FARBINDEX
Complexity
0
324


org.eclipse.osgi.internal.resolver
.StateHelperImpl
Loc: 209
Complexity: 84
Outcalls: 233
Type: Klasse
Children: 18
Revisions: 82
Last Revision: 26.1.2007
FILTER
0
1245
FARBINDEX
0
324


FILTER
0
4178
FARBINDEX
0
324
FILTER
0
4178
FARBINDEX
0
324

Fazit

Konzeptüberprüfung:

Die Datenbankanbindung gestaltet sich äusserst komplex und erfordert detaillierte Angaben über die einzelnen Methoden und Klassen. Diese unabdingbaren Realdaten wurden erst verzögert und unvollständig zur Verfügung gestellt, was eine effiziente Arbeit und eine ergiebige Überprüfung des Konzeptes - welche auch das eigentliche Ziel des Projektes war - sehr erschwert.

Was man anhand der bereits visualisierten Softwarefragmenten beurteilen kann, sind wir überzeugt ein funktionierendes Grundkonzept für die Darstellung einer grossen, hierarchisch strukturierten und vernetzten Datenmenge entworfen zu haben. Die Filtermöglichkeiten, die farbliche Kontrastierung und das Zoomkonzept machen die Informationsmenge überschau- und lesbar.

Es entseht ein planähnliche Abbildung einer Softwarearchitektur, deren Bausteine bezüglich ihrer Position statisch, wohl aber ihrer Erscheinung dynamisch sind. Dieses interaktive Werkzeug kann sich als hilfreich erweisen, um den Entstehungsprozess und die Weiterentwicklung einer Software zu begleiten und sie auf ihre Qualität zu überprüfen. Einzelne Abbildungen können seriell gesehen die Entwicklung aufzeigen und dokumentieren.

Eine wichtige Eigenschaft der Visualisierung ist die Wiederspiegelung komplexer, lokaler Gefüge. So erlaubt sie Parallelen zwischen den erschwert lesbaren Bereichen der Abbildung und der wirklichen Softwarearchitektur zu machen und lässt eine Benennung ihrer Beschaffenheit zu.

Die Farbmethodik und der Algorithmus für die Lymphknotengrösse, die Filtermöglchkeiten und die Darstellung der Calls konnten nur skizzenhaft umgesetzt werden. In einer nächsten Projektphase wären sie Gegenstand weiterer und spannender Überarbeitungen.