Postfuse
Eclipse Plugin zum Visualisieren von Graphen

Bachelorpraktikum - Review 3 - Qualität
02.03.2007
Gruppe: $G^{2^{2^2}}$

Gliederung

1. Werkzeuge
2. Tests
3. Stand der Dinge

Gliederung

1 Werkzeuge
2 Tests
3 Stand der Dinge

Werkzeuge

- Mantis - Bugtracking Website inkl. Integration in SVN
- SVN - inkl. Commitmails
- JUnit - Unittest für Komponenten
- “Nacktes” Eclipse zum Testen von Plugin

Werkzeuge

- Mantis - Bugtracking Website inkl. Integration in SVN
- SVN - inkl. Commitmails
- JUnit - Unittest für Komponenten
- “Nacktes” Eclipse zum Testen von Plugin

Werkzeuge

- Mantis - Bugtracking Website inkl. Integration in SVN
- SVN - inkl. Commitmails
- JUnit - Unittest für Komponenten
- “Nacktes” Eclipse zum Testen von Plugin

Werkzeuge

- Mantis - Bugtracking Website inkl. Integration in SVN
- SVN - inkl. Commitmails
- JUnit - Unittest für Komponenten
- “Nacktes” Eclipse zum Testen von Plugin

Maßnahmen

- Pair-Programming
- Code Conventions
- Plattformdiversität in der Gruppe

Maßnahmen

- Pair-Programming
- Code Conventions
- Plattformdiversität in der Gruppe

Maßnahmen

- Pair-Programming
- Code Conventions
- Plattformdiversität in der Gruppe

Gliederung

1 Werkzeuge

2 Tests

3 Stand der Dinge

Use-Cases

- werden mit Hilfe von Blackbox-Tests getestet
- definieren den normalen Ablauf
- definieren die alternativen Abläufe
- legen das jeweils erwartete Ergebnis fest
- Projekt mit Liste von wie folgt spezifizierten *.gml und Skriptdateien

Use-Cases

- werden mit Hilfe von Blackbox-Tests getestet
- definieren den normalen Ablauf
- definieren die alternativen Abläufe
- legen das jeweils erwartete Ergebnis fest
- Projekt mit Liste von wie folgt spezifizierten *.gml und Skriptdateien

Use-Cases

- werden mit Hilfe von Blackbox-Tests getestet
- definieren den normalen Ablauf
- definieren die alternativen Abläufe
- legen das jeweils erwartete Ergebnis fest
- Projekt mit Liste von wie folgt spezifizierten *.gml und Skriptdateien

Use-Cases

- werden mit Hilfe von Blackbox-Tests getestet
- definieren den normalen Ablauf
- definieren die alternativen Abläufe
- legen das jeweils erwartete Ergebnis fest
- Projekt mit Liste von wie folgt spezifizierten *.gml und Skriptdateien

Use-Cases

- werden mit Hilfe von Blackbox-Tests getestet
- definieren den normalen Ablauf
- definieren die alternativen Abläufe
- legen das jeweils erwartete Ergebnis fest
- Projekt mit Liste von wie folgt spezifizierten *.gml und Skriptdateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Laden eines Graphen (API)

Use-Cases

- Verschiedene Test *.gml Dateien vorgeben
- Korrekte Dateien
- Möglichst komplexe Dateien erstellen
- Abdecken von allen denkbaren Konstrukten
- Defekte, bewusst manipulierte Dateien ($\Rightarrow$ möglichst fehlertolerenater Parser)
- nicht *.gml Dateien

Starten eines Skriptes (GUI)

Use-Cases

- korrekte *.gml Dateien
- an Knoten und Kanten angehängte Skripte
- Skripte aus eingebetteten Code und aus Dateien
- inkorrekter Skriptcode
- testen ob richtiger Interpreter geladen wird

Starten eines Skriptes (GUI)

Use-Cases

- korrekte *.gml Dateien
- an Knoten und Kanten angehängte Skripte
- Skripte aus eingebetteten Code und aus Dateien
- inkorrekter Skriptcode
- testen ob richtiger Interpreter geladen wird

Starten eines Skriptes (GUI)

Use-Cases

- korrekte *.gml Dateien
- an Knoten und Kanten angehängte Skripte
- Skripte aus eingebetteten Code und aus Dateien
- inkorrekter Skriptcode
- testen ob richtiger Interpreter geladen wird

Starten eines Skriptes (GUI)

Use-Cases

- korrekte *.gml Dateien
- an Knoten und Kanten angehängte Skripte
- Skripte aus eingebetteten Code und aus Dateien
- inkorrekter Skriptcode
- testen ob richtiger Interpreter geladen wird

Starten eines Skriptes (GUI)

Use-Cases

- korrekte *.gml Dateien
- an Knoten und Kanten angehängte Skripte
- Skripte aus eingebetteten Code und aus Dateien
- inkorrekter Skriptcode
- testen ob richtiger Interpreter geladen wird

Komponententest

- Sichtprüfung der Graphen $\Rightarrow$ Menge von Beispielen
- JUnit-Test von Scriptkomponenten
- Jeweils Vor- und Nachbedingungen spezifizieren

Komponententest

- Sichtprüfung der Graphen $\Rightarrow$ Menge von Beispielen
- JUnit-Test von Scriptkomponenten
- Jeweils Vor- und Nachbedingungen spezifizieren

Komponententest

- Sichtprüfung der Graphen $\Rightarrow$ Menge von Beispielen
- JUnit-Test von Scriptkomponenten
- Jeweils Vor- und Nachbedingungen spezifizieren

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Graph
Komponententest

Da nur Interface, testen mit *GRootGraph*:

- Node addNode()
- Node addNode(NodeDesign nd)
- abstract Subgraph addSubgraph()
- Edge addEdge(BasicNode n1, BasicNode n2)
- Edge addEdge(BasicNode n1, BasicNode n2, EdgeDesign ed)
- void setDefaultEdgeDesign(EdgeDesign ed)
- void setDefaultNodeDesign(NodeDesign nd)
- EdgeDesign getDefaultEdgeDesign()
- NodeDesign getDefaultNodeDesign()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Script
Komponententest

Da nur Interface, testen mit *BeanShellScript*, erzeugen mit zugehöriger Factory:

- void run()
- String getLabel()
- String getCode()
- boolean getCodeEmbedded()
- void setCodeEmbedded(boolean flag)
- IFile getFile()

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRenderer.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRenderer.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRender.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRender.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRender.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRender.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Methodentest

- White Box
- JUnit und Sichtprüfung
- Zweigüberdeckung $\Rightarrow$ ausführen jeder Kante im Kontrollflussgraphen
- Auswahl der Testcases anhand von Codeanalyse
- zu Testende Mehthoden:
  - GNodeRender.render() $\Rightarrow$ Design von Kanten und Knoten
  - ExtGraphMLWriter.writeGraph(Graph graph, OutputStream os) $\Rightarrow$ Menge von Bsp.

Benutzbarkeitstest

- Benutzbarkeit der GUI
- Benutzbarkeit der API
- Tester aus unserer Gruppe $\Rightarrow$ Demo Programme
- Externe Tester (Programmierer & GUI-Anwender)
- Fragenkatalog zusammenstellen

Benutzbarkeitstest

- Benutzbarkeit der GUI
- Benutzbarkeit der API
- Tester aus unserer Gruppe $\Rightarrow$ Demo Programme
- Externe Tester (Programmierer & GUI-Anwender)
- Fragenkatalog zusammenstellen

Benutzbarkeitstest

- Benutzbarkeit der GUI
- Benutzbarkeit der API
- Tester aus unserer Gruppe $\Rightarrow$ Demo Programme
- Externe Tester (Programmierer & GUI-Anwender)
- Fragenkatalog zusammenstellen

Benutzbarkeitstest

- Benutzbarkeit der GUI
- Benutzbarkeit der API
- Tester aus unserer Gruppe $\Rightarrow$ Demo Programme
- Externe Tester (Programmierer & GUI-Anwender)
- Fragenkatalog zusammenstellen

Benutzbarkeitstest

- Benutzbarkeit der GUI
- Benutzbarkeit der API
- Tester aus unserer Gruppe $\Rightarrow$ Demo Programme
- Externe Tester (Programmierer & GUI-Anwender)
- Fragenkatalog zusammenstellen

Gliederung

1 Werkzeuge

2 Tests

3 Stand der Dinge

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Was geht bereits
Stand der Dinge

- Export: SVG, PNG
- Laden / Speichern GraphML (noch nicht endgültiges Dateiformat)
- Anzeigen von Graphen ohne Subgraphen incl. Zoom und Pan
- Verschiedene Node Designs
- Auf- und Zuklappen von Knoten
- Skripte ausführen
- Eclipse: Update Site Installation
- Eclipse: MultiView mit *.gml Dateiassoziation
- Eclipse: Nutzen des Eclipse File Konzeptes
- ...

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Arbeitszeiten
Stand der Dinge

- 1000 h Zielmarke
- 500 h Bisher
  - 15% Toolsuche und Installation
  - 10% Organisationstreffen
  - 30% Dokumentation
  - 35% Implementierung
  - 10% Qualitätssicherung

Weitere Planungen & Probleme
Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
    - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Weitere Planungen & Probleme

Stand der Dinge

- Renderer für Subgraphen
- Neuer Layouter (Mehrfachkanten, nichtüberlappende Knoten, Schleifen, Subgraphen)
- HTML in Nodes
- Eclipse:
  - MacOS-X
  - UI-Thread Problem
- Dokumentation
- Tests

Fragen?

- ...