

Software Engineering II

Übung 3

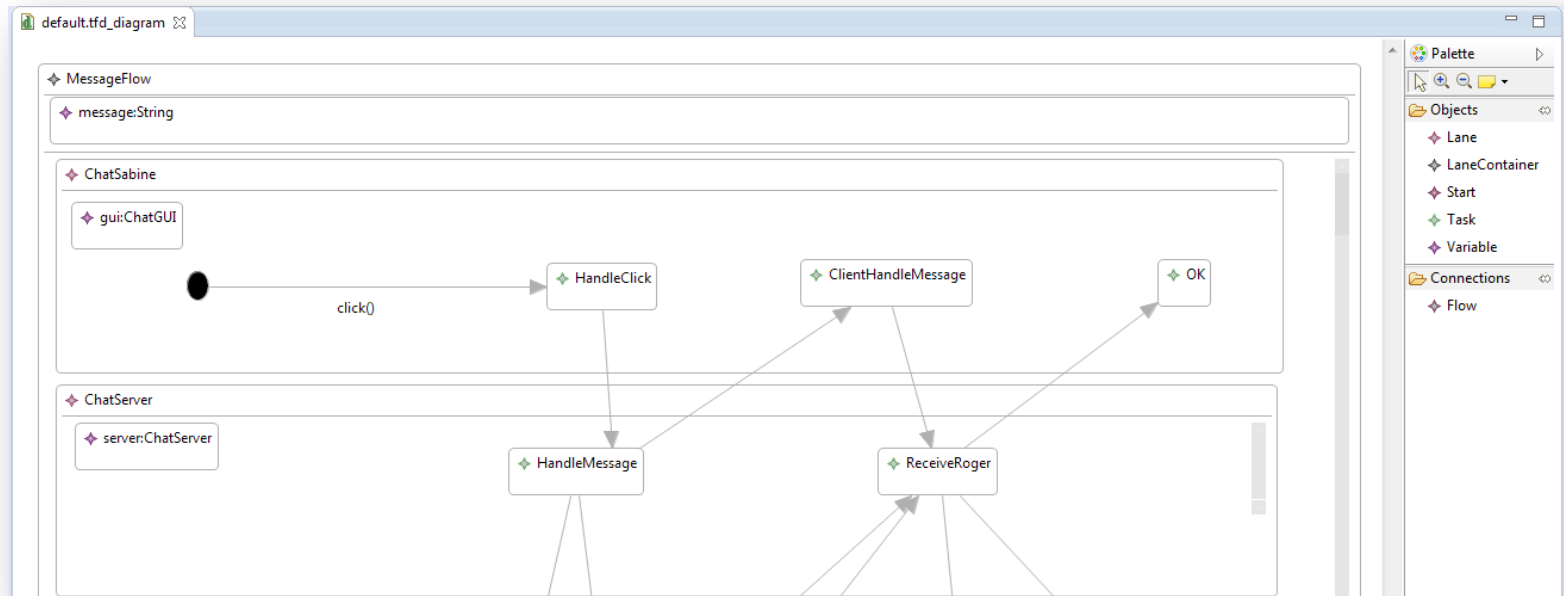
Wintersemester 12/13

Fachgebiet Software Engineering

Andreas Scharf

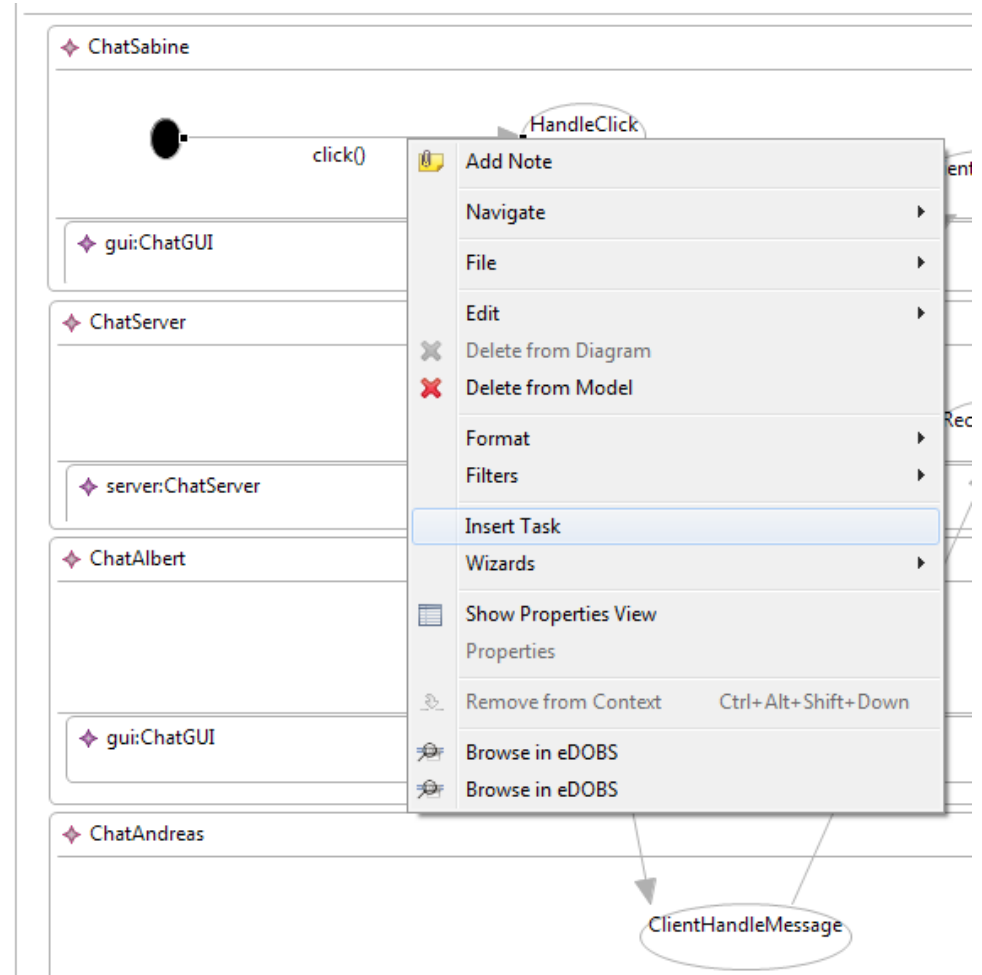
Wiederholung

- **Grafischer Editor zum Erstellen von Task Flow Diagrammen**
 - Annotationen am Fujaba Modell
 - Generierung des Quelltextes
 - Verarbeitung der Annotationen mit Eugenia
 - Automatische Generierung des grafischen Editors



Nächstes Ziel

- **Aus dem grafischen Editor heraus per Rechtsklick eine Action auf dem Modell anstoßen**
 - Umbauten am Modell vornehmen, Refactorings etc.
 - Actions werden (komplett) in Fujaba entwickelt und generiert
 - Actions können im Runtime-Eclipse auf dem grafischen Editor ausgeführt werden.



Notwendige Schritte

- **Download von JavaClass.ctr und EclipseClasses.ctr von der Blogseite**
- **Einfügen der .ctrs ins TFD Eclipse Projekt.**
- **Öffnen von JavaClasses per Doppelklick**
- **Öffnen von EclipseClasses per Doppelklick**
- **Öffnen des TFD Fujaba Projektes**

Modellierung der Actions mit Fujaba

- **Einfügen von EclipseClasses als Project Dependency ins TFD Modell**
 - File -> Project Dependencies
- **Einfügen von EObject ins Modell-Klassendiagramm**
 - Rechtsklick -> Edit Class Diagram...
- **Alle Modellobjekte erben von EObject!**
- **Anlegen eines neuen Klassendiagramms für die Actions**
- **Rechtsklick -> Edit Class Diagram...**
 - Hinzufügen von TransactionActionDelegate
- **Eigene Actions anlegen und von TransactionActionDelegate erben**
 - WICHTIG: nicht ins model package, wir wollen Java Code generieren und keinen emf Code.
- **runImpl(action:IAction) implementieren**

Generieren der Actions I

- **Generieren von Quelltext aus EclipseClasses.ctr**
 - Einfügen des generated2 Ordners in den Java BuildPath des TFD Projekts
-> das Projekt hat Fehler
- **Manifest.MF öffnen und unter Dependencies die Projektabhängigkeiten anpassen.**
 - org.eclipse.core.expressions, org.eclipse.gmf.runtime.notation, org.eclipse.gef, org.eclipse.ui, org.eclipse.emf.transaction, org.eclipse.gmf.runtime.diagram.ui
- **Hinzufügen von Fujaba Bibliotheken zum Plugin-In Classpath**
 - de.uni_kassel.features
 - de.uni_paderborn.runtimetools

Generieren der Actions II

- **generated Ordner löschen**
- **.edit, .editor und .diagram Projekte löschen. (wegen Änderung am Modell → EObject)**
- **Quelltext aus TFD Projekt neu generieren**
- **.edit, .editor und .diagram neu generieren**

Actions zum Laufen bringen

- **Manifest.MF öffnen**
- **Unter Extensions hinzufügen:**
 - org.eclipse.popupMenus
 - org.eclipse.core.runtime.adapters
- **In org.eclipse.ui.popupMenus mit Rechtsklick -> New -> Object contribution**
 - Jeweils für alle Klassen auf denen Actions ausgeführt werden sollen
 - Adaptable auf true setzen!
- **in org.eclipse.core.runtime.adapters**
 - adaptableType: org.eclipse.gef.EditPart
 - class: de.uni_kassel.eclipse.adapter.EditPartAdapterFactory

Actions zum Laufen bringen II

- **Adapter zur factory einstellen/hinzufügen für jeden Typ, auf dem Actions aufgerufen werden können sollen:**
 - type: die Klasse für die der Adapter gelten soll
- **Zu jeder objectContribution:**
 - Rechtsklick -> New -> action
 - class: implementierte Action Klasse
 - label: Text der im Menu auftauchen soll.

Zu implementierende Refactorings

- **Convert Task to Start**
 - Auf einem Task aufrufbar
 - Alle ausgehenden Flows erhalten
- **Convert Start to Task**
 - Auf einem Start Knoten aufrufbar
 - Alle ausgehenden Flows erhalten
- **Insert Task**
 - Auf einem Flow aufrufbar
 - Neuen Task erstellen. Container ist die Lane des Source Knotens!
 - 2 neue Flows erstellen
 - Flow 1: Source -> Neuer Task
 - Flow 2: Neuer Task -> Target