Guideline Execution Engine (GEE) Specification

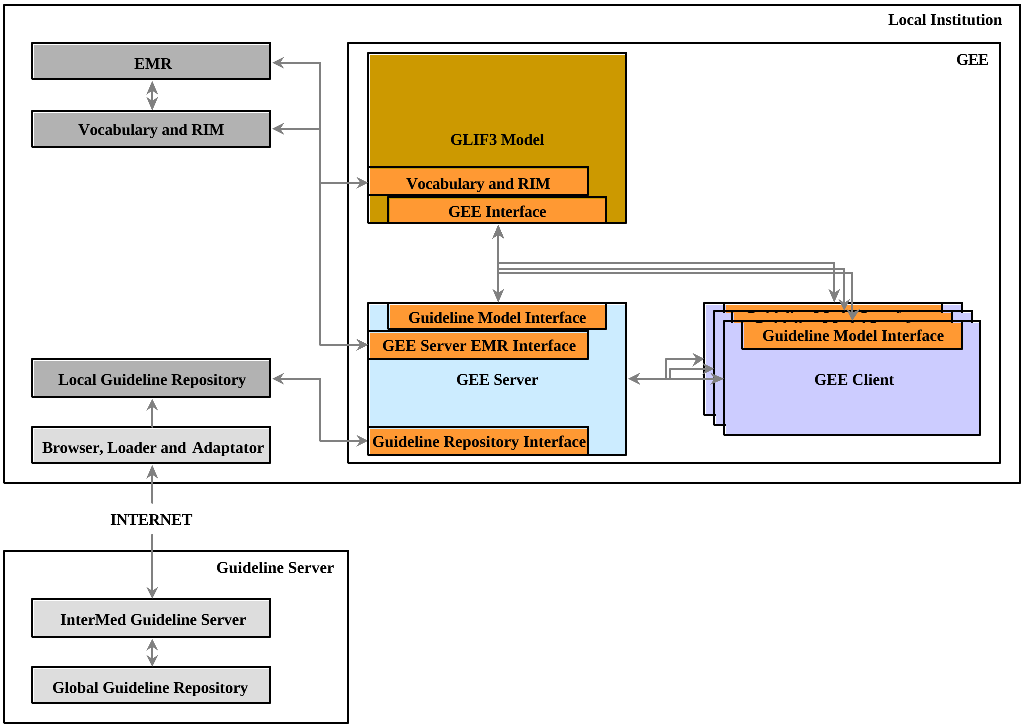
Philosophy of Design

We distinguish tasks required by a specific guideline model for proper execution from tasks required by a Guideline Execution Engine (GEE) for administrative purpose.

We provide a set of APIs for the implementation of a GEE to support execution of guidelines encoded in GLIF3. These APIs can be considered as an attachment of the GLIF3 model to regulate how the elements in GLIF3 can be used. In other words, they define the semantics of GLIF3. In object-oriented jargon, these are the methods, comparing to current elements in the GLIF3 model, which are attributes.

Institutions can implement their own GLIF3 GEEs with appropriate interfaces to these APIs. We provide one implementation of GLIF3 GEE based on these APIs.

System Architecture of GEE


Local Institution
EMR
Vocabulary and RIM
GEE
GLIF3 Model
Vocabulary and RIM
GEE Interface
Guideline Model Interface
GEE Server EMR Interface
GEE Server
Guideline Repository Interface
Guideline Model Interface
GEE Client
Local Guideline Repository
Browser, Loader and Adaptator
INTERNET
Guideline Server
InterMed Guideline Server
Global Guideline Repository

Assumptions in the design:

- Guidelines have been already loaded from the global guideline server to a local institution, adapted and stored in a local guideline repository.
- Patient data, actions, events, etc. can be mapped to a local EMR through the vocabulary and RIM used in the guideline model.

Design of our GEE takes the client-server system architecture. The system consists of one GEEServer and multiple GEEClients. The GEEServer handles all "real" execution tasks as well as administrative tasks of the whole GEE. The GEEClients only provide user interfaces and their own administrative tasks. Within each GEEClient, executions of multiple guidelines applied to multiple patients are supported.

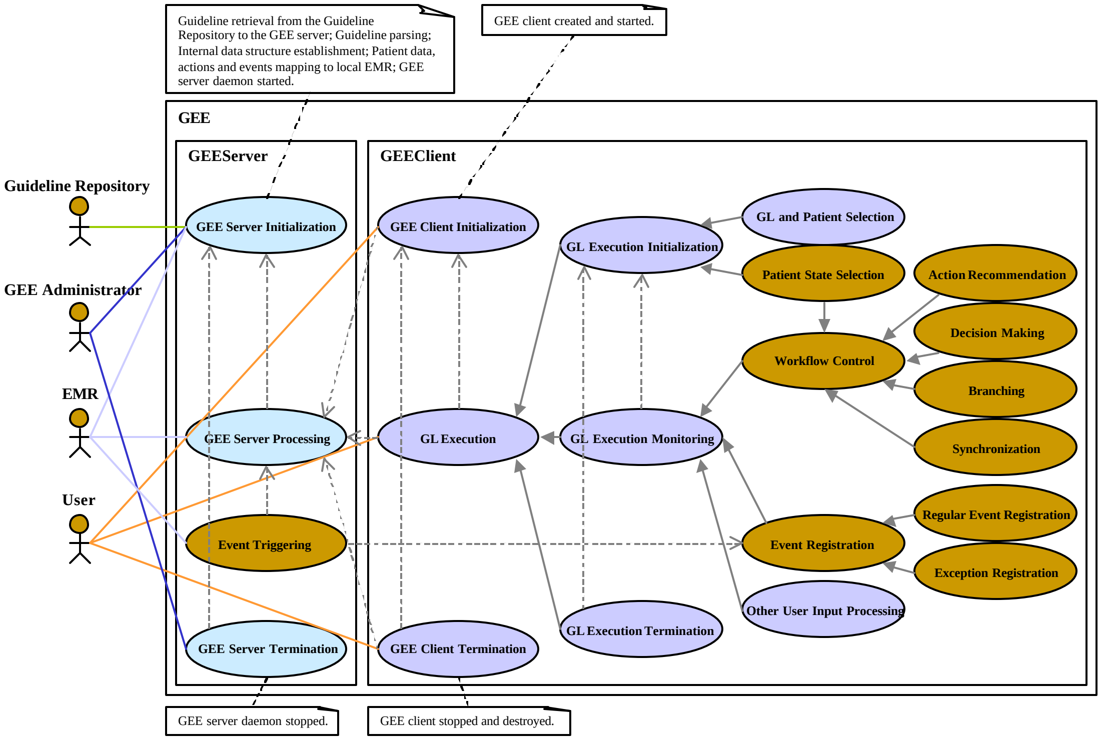
A user's interaction with a GEEClient will be sent to the GEEServer. After appropriate processing, the result will be sent back to the GEEClient as feedback to the user.

Use Cases

Use cases are shown in the following diagram.

Actors related to the GEEServer include the GEE administrator, the guideline repository and the EMR.

- GEE Server Initialization
  In this scenario, the GEE administrator starts the GEEServer. The GEEServer then loads guidelines from the guideline repository, parses them, builds the internal data structures, maps patient data, actions and events to local EMR, and starts the server daemon.
- GEE Server Processing
  In this scenario, the GEEServer handles communication from GEEClients and the EMR (for actions, events and exceptions), processes them appropriately and provides feedbacks.
- Event Triggering
  In this scenario, the GEEServer already registered events (initiated by clients) that need to be monitored. When these events happen, the EMR communicates with the GEEServer, appropriate processing are performed and results are feedbacked to the users that are using the related guidelines for specific patients.
- GEE Server Termination
  In this scenario, all GEEClients are already terminated. The GEE administrator can stop the server daemon and terminate the GEEServer.


Guideline retrieval from the Guideline Repository to the GEE server; Guideline parsing; Internal data structure establishment; Patient data, actions and events mapping to local EMR; GEE server daemon started.
GEE client created and started.
GEE
GEEServer
GEEClient
Guideline Repository
GEE Administrator
EMR
User
GEE Server Initialization
GEE Server Processing
Event Triggering
GEE Server Termination
GEE Client Initialization
GL Execution
GEE Client Termination
GL Execution Initialization
GL Execution Monitoring
GL Execution Termination
GL and Patient Selection
Patient State Selection
Action Recommendation
Decision Making
Workflow Control
Branching
Synchronization
Regular Event Registration
Event Registration
Exception Registration
Other User Input Processing
GEE server daemon stopped.
GEE client stopped and destroyed.

Actors related to the GEEClients are the users.

- GEE Client Initialization

  In this scenario, the user starts the GEEClient. The GEEClient then communicate with the GEEServer so that a GEEClient instance will be created at the server side.
- GL Execution

  In this scenario, the user performs various interactions with the GEEClient, which are sent to the server side, processed and feedbacked.
  - GL Execution Initialization

    The user selects a guideline to be executed and a patient to be applied. In case this is a returning session, the patient state applied is also decided.
  - GL Execution Monitoring

    Interactions between the user and the GEEClient include 1) workflow control, which further consists of patient state selection, action recommendation, decision making, branching and synchronization management, 2) event registration, which include regular event registration and exception registration, and 3) other user input processing, for example, manual confirmation of action or decision making.
  - GL Execution Termination

    The user completes the execution of a guideline or a session of it applied to a specific patient. The server side will do appropriate management work.
- GEE Client Termination

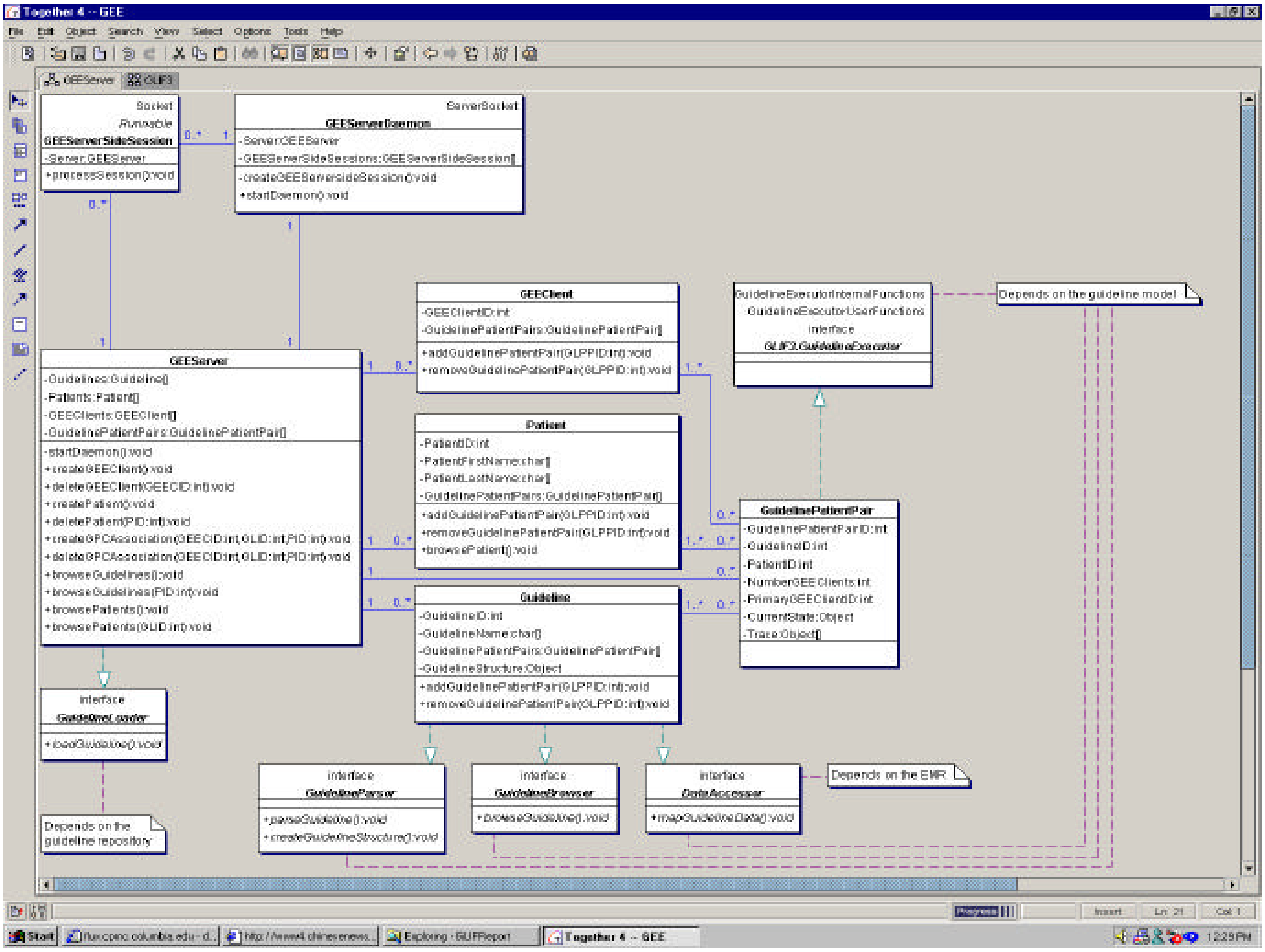
  In this scenario, the user completes all the work and stops the GEEClient. The server side will destroy the GEEClient instance.

Continued on next page

Class Diagrams

There are three packages in the system, GEEServer, GEEClient and GLIF3. Both GEEServer and GEEClient will import GLIF3.

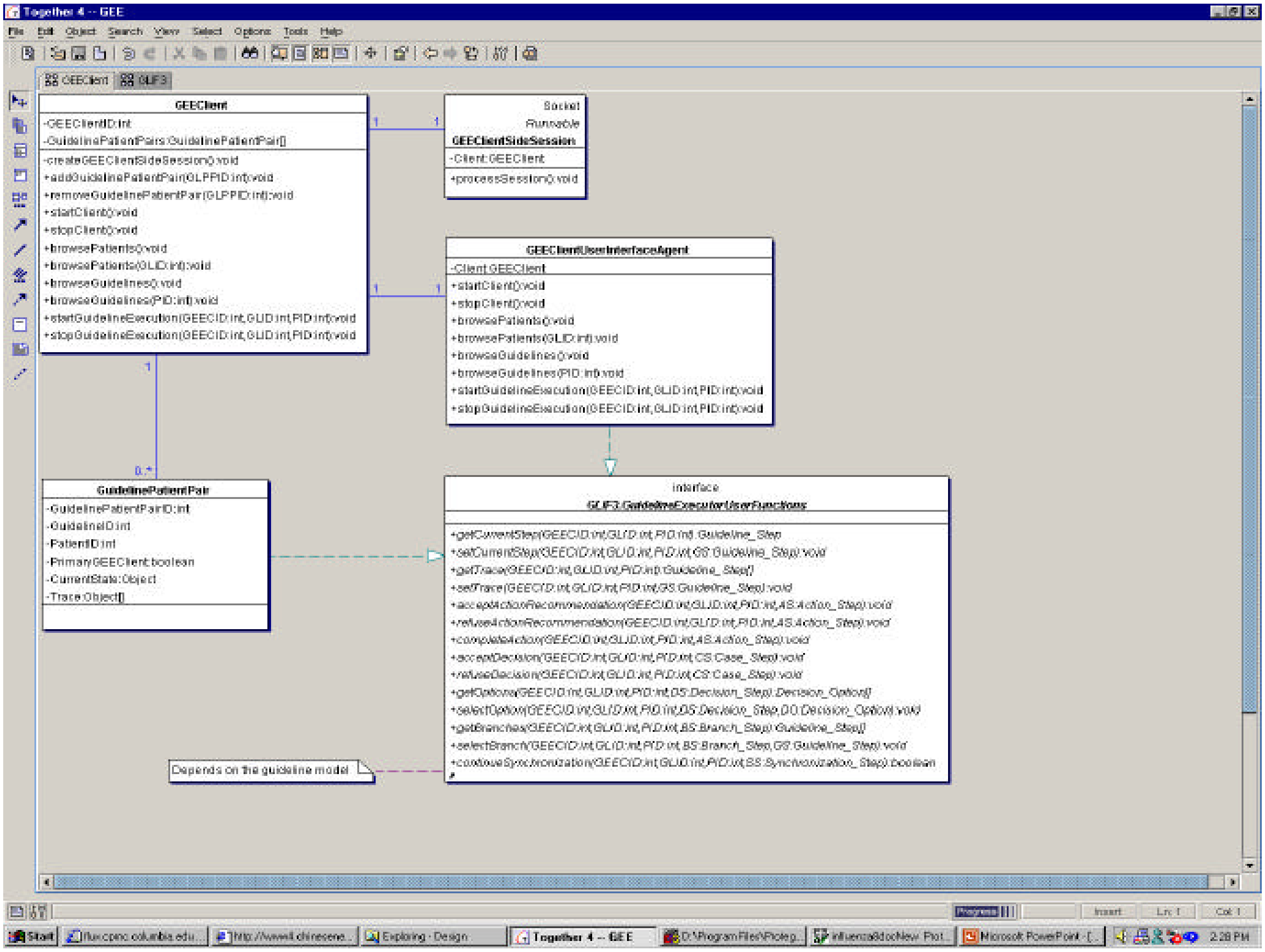
- GEEServer


Together 4 -- GEE
File Edit Object Search View Select Options Tools Help
GEEServer
GLIF3
Socket
Runnable
GEEServerSideSession
-Server:GEEServer
+processSession():void
0..*
1
ServerSocket
GEEServerDaemon
-Server:GEEServer
-GEEServerSideSessions:GEEServerSideSession[]
-createGEEServersideSession():void
+startDaemon():void
GEEServer
-Guidelines:Guideline[]
-Patients:Patient[]
-GEEClients:GEEClient[]
-GuidelinePatientPairs:GuidelinePatientPair[]
-startDaemon():void
+createGEEClient():void
+deleteGEEClient(GEECID:int):void
+createPatient():void
+deletePatient(PID:int):void
+createGPCAssociation(GEECID:int,GLID:int,PID:int):void
+deleteGPCAssociation(GEECID:int,GLID:int,PID:int):void
+browseGuidelines():void
+browseGuidelines(PID:int):void
+browsePatients():void
+browsePatients(GLID:int):void
GEEClient
-GEEClientID:int
-GuidelinePatientPairs:GuidelinePatientPair[]
+addGuidelinePatientPair(GLPPID:int):void
+removeGuidelinePatientPair(GLPPID:int):void
Patient
-PatientID:int
-PatientFirstName:char[]
-PatientLastName:char[]
-GuidelinePatientPairs:GuidelinePatientPair[]
+addGuidelinePatientPair(GLPPID:int):void
+removeGuidelinePatientPair(GLPPID:int):void
+browsePatient():void
Guideline
-GuidelineID:int
-GuidelineName:char[]
-GuidelinePatientPairs:GuidelinePatientPair[]
-GuidelineStructure:Object
+addGuidelinePatientPair(GLPPID:int):void
+removeGuidelinePatientPair(GLPPID:int):void
GuidelineExecutorInternalFunctions
GuidelineExecutorUserFunctions
interface
GLIF3.GuidelineExecuter
Depends on the guideline model
GuidelinePatientPair
-GuidelinePatientPairID:int
-GuidelineID:int
-PatientID:int
-NumberGEEClients:int
-PrimaryGEEClientID:int
-CurrentState:Object
-Trace:Object[]
1..*
0..*
interface
GuidelineLoader
+loadGuideline():void
Depends on the guideline repository
interface
GuidelineParsor
+parseGuideline():void
+createGuidelineStructure():void
interface
GuidelineBrowser
+browseGuideline():void
interface
DataAccessor
+mapGuidelineData():void
Depends on the EMR
Insert
Ln 21
Col 1
Start
Exploring - GLIFReport
Together 4 -- GEE
12:29 PM

Major classes in the GEEServer package include GEEServer, GEEClient, Patient, Guideline and GuidelinePatientPair. The latest one corresponds to a specific guideline applied to a specific patient. Other classes include GEEServerDaemo n, which monitors the incoming connections, and GEEServerSideSession, whose instance is a specific communication session with GEEClient or EMR (for actions, events and exceptions).

Guideline model specific methods are wrapped in the GLIF3.GuidelineExecutor interface, which is implemented by the GuidelinePatientPair. Other interfaces whose implementations depend on the guideline model include GuidelineParsor, GuidelineBrowser and DataAccessor. DataAccessor also depends on the EMR. GuidelineLoader interface depends on the guideline repository.

- GEEClient


Together 4 -- GEE
File
Edit
Object
Search
View
Select
Options
Tools
Help
GEEClient
GLIF3
GEEClient
-GEEClientID:int
-GuidelinePatientPairs:GuidelinePatientPair[]
-createGEEClientSideSession():void
+addGuidelinePatientPair(GLPPID:int):void
+removeGuidelinePatientPair(GLPPID:int):void
+startClient():void
+stopClient():void
+browsePatients():void
+browsePatients(GLID:int):void
+browseGuidelines():void
+browseGuidelines(PID:int):void
+startGuidelineExecution(GEECID:int,GLID:int,PID:int):void
+stopGuidelineExecution(GEECID:int,GLID:int,PID:int):void
1
1
Socket
Runnable
GEEClientSideSession
-Client:GEEClient
+processSession():void
1
1
GEEClientUserInterfaceAgent
-Client:GEEClient
+startClient():void
+stopClient():void
+browsePatients():void
+browsePatients(GLID:int):void
+browseGuidelines():void
+browseGuidelines(PID:int):void
+startGuidelineExecution(GEECID:int,GLID:int,PID:int):void
+stopGuidelineExecution(GEECID:int,GLID:int,PID:int):void
1
0..*
GuidelinePatientPair
-GuidelinePatientPairID:int
-GuidelineID:int
-PatientID:int
-PrimaryGEEClient:boolean
-CurrentState:Object
-Trace:Object[]
interface
GLIF3.GuidelineExecutorUserFunctions
+getCurrentStep(GEECID:int,GLID:int,PID:int):Guideline_Step
+setCurrentStep(GEECID:int,GLID:int,PID:int,GS:Guideline_Step):void
+getTrace(GEECID:int,GLID:int,PID:int):Guideline_Step[]
+setTrace(GEECID:int,GLID:int,PID:int,GS:Guideline_Step):void
+acceptActionRecommendation(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+refuseActionRecommendation(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+completeAction(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+acceptDecision(GEECID:int,GLID:int,PID:int,CS:Case_Step):void
+refuseDecision(GEECID:int,GLID:int,PID:int,CS:Case_Step):void
+getOptions(GEECID:int,GLID:int,PID:int,DS:Decision_Step):Decision_Option[]
+selectOption(GEECID:int,GLID:int,PID:int,DS:Decision_Step,DO:Decision_Option):void
+getBranches(GEECID:int,GLID:int,PID:int,BS:Branch_Step):Guideline_Step[]
+selectBranch(GEECID:int,GLID:int,PID:int,BS:Branch_Step,GS:Guideline_Step):void
+continueSynchronization(GEECID:int,GLID:int,PID:int,SS:Synchronization_Step):boolean
Depends on the guideline model
Insert
Ln: 1
Col: 1
Start
Together 4 -- GEE
Microsoft PowerPoint
2:28 PM

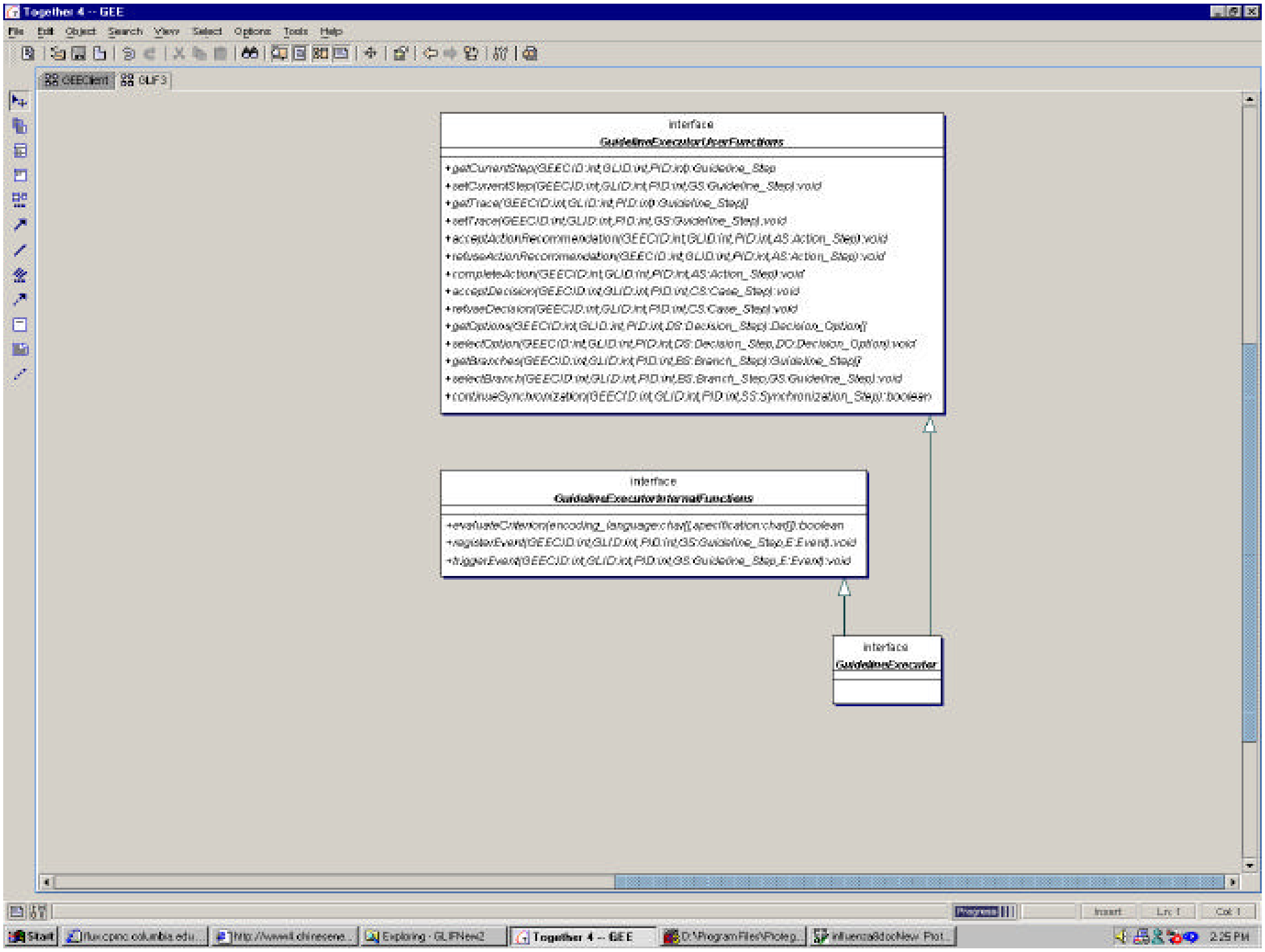
Major classes in the GEEClient package include GEEClient, GuidelinePatientPair and GEEClientUserInterfaceAgent. Another class, GEEClientSideSession, is used for communication with the GEEServer.

Both GuidelinePatientPair and GEEClientUserInterfaceAgent need to implement GLIF3.GuidelineExecutorUserFunction. The implementation by GuidelinePatientPair focuses on the communication with GEEServer, while the implementation by GEEClientUserInterfaceAgent focuses on the presentation at the client user interface.

- GLIF3

There are three interfaces in the GLIF3 package, GuidelineExecutorUserFunction, GuidelineExecutorInternalFunction and GuidelineExecutor. The latest one extends the previous two. GuidelineExecutorUserFunction specifies the functions users directly used when execute GLIF3 guidelines, while GuidelineExecutorInternalFunction specifies the functions that need to be implemented by the GEE to support the user functions.

The whole GLIF3 model should also be presented as classes in the GLIF3 package. The methods in the above interfaces will specify which classes and attributes will be used in the implementation. This defines the semantic of GLIF3. This work is in progress.



```
interface
GuidelineExecutorUserFunctions

+getCurrentStep(GEECID:int,GLID:int,PID:int):Guideline_Step
+setCurrentStep(GEECID:int,GLID:int,PID:int,GS:Guideline_Step):void
+getTrace(GEECID:int,GLID:int,PID:int):Guideline_Step[]
+setTrace(GEECID:int,GLID:int,PID:int,GS:Guideline_Step):void
+acceptActionRecommendation(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+refuseActionRecommendation(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+completeAction(GEECID:int,GLID:int,PID:int,AS:Action_Step):void
+acceptDecision(GEECID:int,GLID:int,PID:int,CS:Case_Step):void
+refuseDecision(GEECID:int,GLID:int,PID:int,CS:Case_Step):void
+getOptions(GEECID:int,GLID:int,PID:int,DS:Decision_Step):Decision_Option[]
+selectOption(GEECID:int,GLID:int,PID:int,DS:Decision_Step,DO:Decision_Option):void
+getBranches(GEECID:int,GLID:int,PID:int,BS:Branch_Step):Guideline_Step[]
+selectBranch(GEECID:int,GLID:int,PID:int,BS:Branch_Step,GS:Guideline_Step):void
+continueSynchronization(GEECID:int,GLID:int,PID:int,SS:Synchronization_Step):boolean
```

```
interface
GuidelineExecutorInternalFunctions

+evaluateCriterion(encoding_language:char[],specification:char[]):boolean
+registerEvent(GEECID:int,GLID:int,PID:int,GS:Guideline_Step,E:Event):void
+triggerEvent(GEECID:int,GLID:int,PID:int,GS:Guideline_Step,E:Event):void
```

```
interface
GuidelineExecutor
```