Buzzword Bingo
Game Documentation
(Java based Game)

Meppe Patrick
Djeufack Stella
Beltran Daniel

April 15, 2011

Inhaltsverzeichnis

1 Einleitung

Nachfolgend sollen die Schritte dokumentiert werden, die maßgeblich zur Struktur und Erscheinungsbild unseres in der Aufgabenstellung beschriebenen Spiels beigetragen haben. Der Werdegang und letztendlich der Entwicklungsprozess werden hier festgehalten um das schrittweise, systematische Entstehen aufzuzeigen.

2 Aufgabenstellung

Die Aufgabenstellung verlangt ein Applet oder Applikation einer Bingo Derivation. Diese soll graphisch interaktiv umgesetzt werden. Gefordert ist die Variante Buzzword Bingo oder Bullshit Bingo. Sie ist als netzwerkfähige oder zumindest verteilte Software zu implemtieren und sollte möglichst Platform unabhängig sein.

3 Allgemeines zu Buzzword Bingo

Es werden verschiedene Wörter benutzt die Redner oder Vortragende in Präsentationen benutzen und zum eigentlichen Sinn der Rede nichts beitragen. Es können jedoch auch Wörter verwendet werden, die die Präsentierenden besonders oft benutzen. Wenn eine Person in einer Reihe, sei es nun vertikal, horizontal oder diagonal, alle Wörter gehört und markiert hat, ruft er laut ‚Bullshit'. Diejenige Person hat das Spiel gewonnen.

4 Allgemeine Überlegungen

Als erstes ist die Usability und Stabilität das Zentrum unserer Überlegungen gewesen. Wir wollten ein interaktives Programm, was anklickbar ist und farblich deutlich sichtbare Veränderungen aufzeigt sowie ein ansprechendes Erscheinungsbild hat. Es sollte möglich sein später oder direkt eine Datenbank einzubinden. Desweiteren sollte der Server unabhängig vom Client laufen können.

5 Rollenverteilung

Patrick Meppe (Product Owner)
Daniel Beltran (Architecture & Design/Scrum Master)

Stella Djeufack (Documentation)

6 Anforderungsanalyse

6.1 Muss-Anforderungen (Must-Have)

Ein grafikbasiertes Spiel der Bingo Variante Buzzword Bingo
Netzwerkfähigkeit des Spiels

6.2 Kann-Anforderungen (Nice-To-Have)

-Neustart-Funktion
-Server-Suche
-Multiplen Spielen beitreten

6.3 Nicht-Funktionale Anforderungen (Usability & Documentation)

-Multiple lauffähige Clients
-Dokumentation
-Quellcode

6.4 Aktivitätsdiagramme

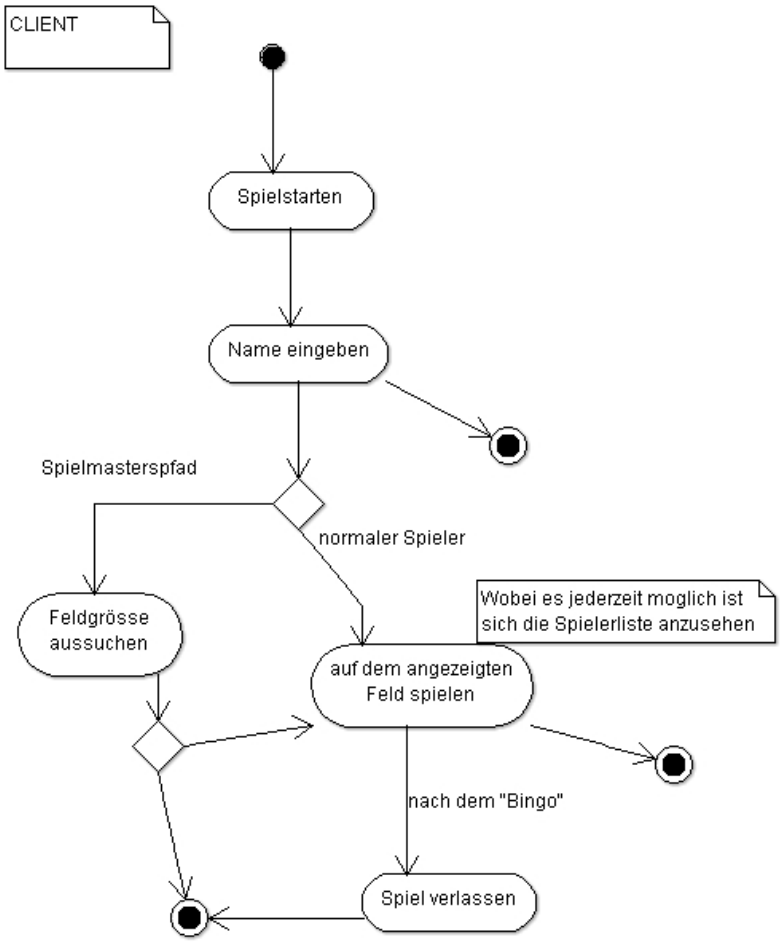


Figure 1: Aktivitätsdiagramm Client

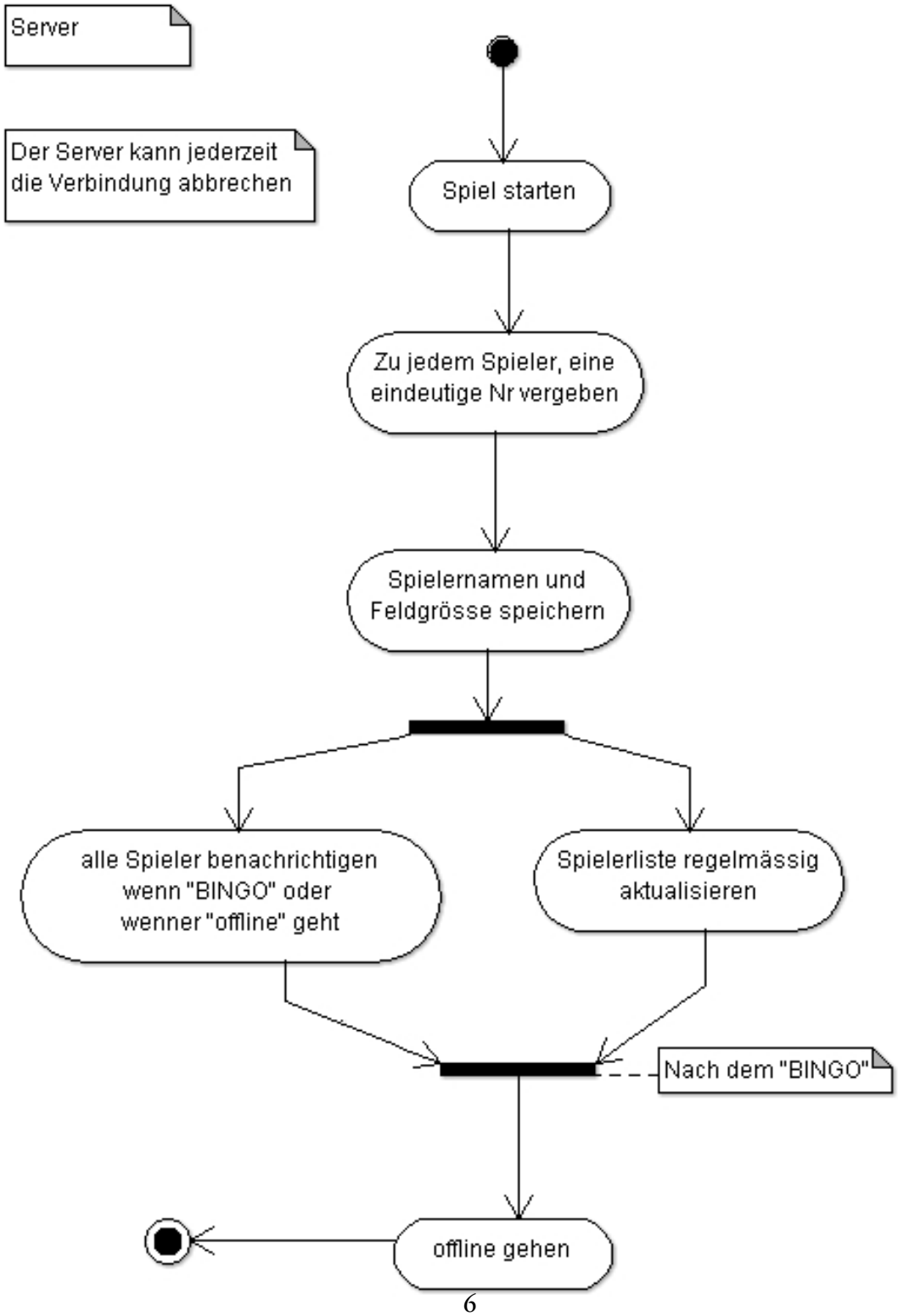


Figure 2: Aktivitätsdiagramm Server

7 Verwendete Programmiersprache

Da Java eine gängige und platform-unabhängige Sprache ist, haben wir uns für Java entschieden. Java ist ähnlich wie alle gängigen Programmiersprachen wie C, C# oder C++ von denen sie große Teilmengen an Codesprache besitzt. Java hat schon von vorneherein eine gute GUI, gute Netzwerkbibliotheken und einfache Interaktion.

8 Netzwerkfähigkeit/ -topologie

Aufgrund der schon vorhandenen Socket Bibliotheken ist die Netzwerkimplementierung kein unnötig großer Aufwand. Die Verbindung wird als Client-Server Applikation umgesetzt, wo der Client sozusagen unabhängig aber auch parallel mit dem Server auf einer Workstation laufen kann, aber auch ein Webserver einen Server bereitstellen kann und die Clients verteilt auf eine große Distanz zueinander stehen können. Der Einfachheit halber und der Erfahrungsvorteile wegen nutzen wir Eclipse und Netbeans.

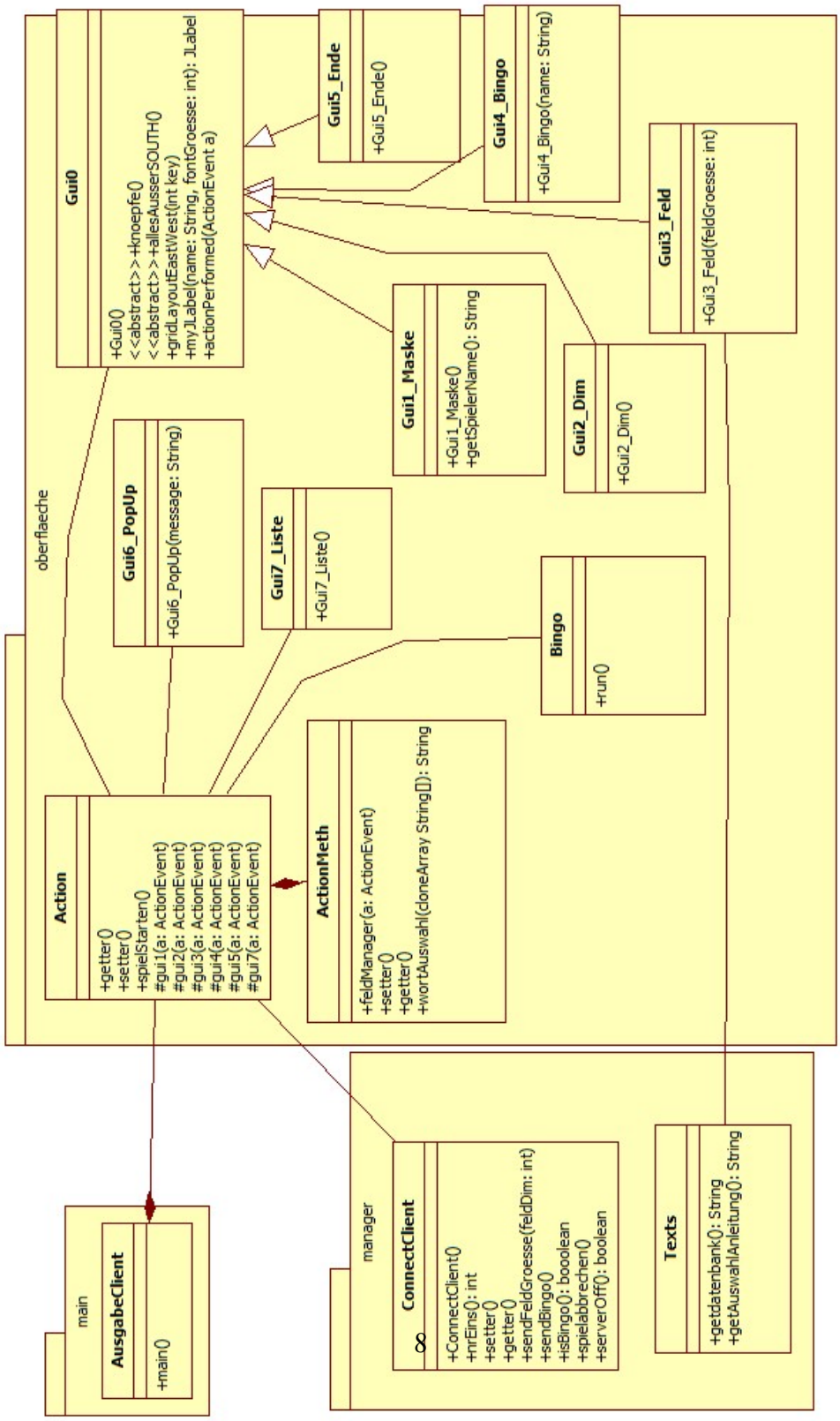
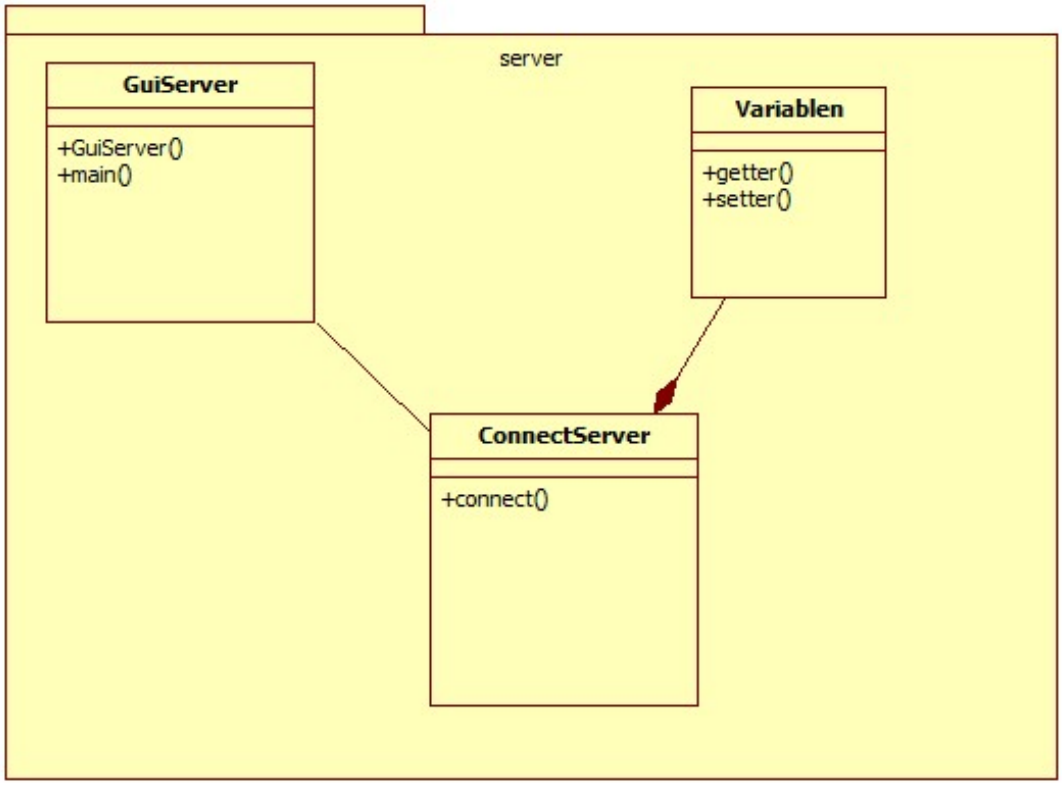


Figure 3: Klassendiagramm Client

9 Struktur der Lösung



9.1 Server-Seitige Klassen

GuiServer.java
Die Klasse mit der main-Funktion und der graphischen Oberfläche
/Klasse die den Server instanziert

ConnectServer.java
Server implementierung mit automatischer Abwicklung

Variablen.java
Alle enthaltenen Paramet die über das
Netzwerk geschickt werden und für die Clients relevant sind.

9.2 Client-seitige Klassen:

AusgabeClient.java:
Die Klasse mit der main-Funktion
/Klasse die den Client instanziert

ConnectClient.java:
Die Klasse mit dem Client und den Schnittstellen zur GUI

Texts.java:
Die Klasse für Text und Wörte, als Platzhalter und Schnittstellenklasse
für eine Erweiterung mit einer Datenbank

Paket Oberfläche:
Enthaltene Klassen sind primär für die GUI implementiert.
Eine weitere Klasse überprüft den ‚Bingo Status'

GUI1-GUI7
Menüführung durch das Spiel

Logo.png
Spielfrontlogo.

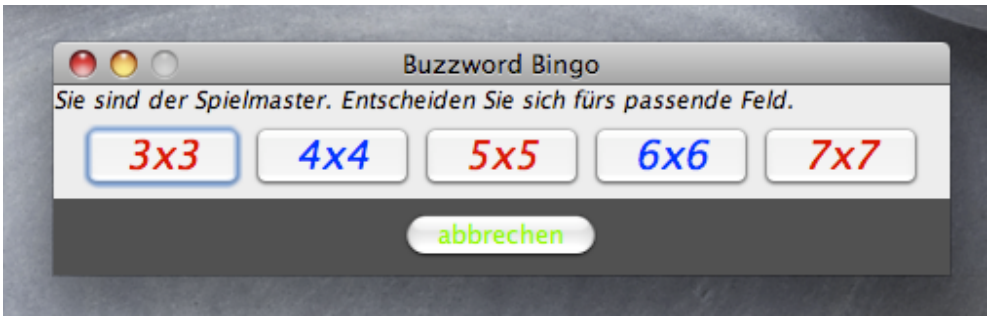
10 Produkteinsatz

Die Software ist als Client-Server Anwendung realisiert. Genutzt wurde ein Datagramm Server. Die Spiellogik erfolgt Clientseitig und wird nur in Intervallen mit dem Server synchronisiert. Spielinteraktion und Gameplay erfolgt auf der Workstation des Clients. Gewinnt ein Spieler leitet der Client das an den Server weiter, und der Server an alle noch verbundenen Clients. Danach wird das Spiel beendet. Der Server weißt die Bibliotheken zu und verwaltet diese. Clientseitig werden diese dann randomisiert wieder auf den Bildschirm ausgegeben. Alle Spieler sind mit Namen in einer Liste gespeichert die dynamisch groß werden kann.

11 Benutzeroberfläche

11.1 Graphische Oberfläche

11.2 Auswahl der Feldgröße



11.3 Graphische Oberfläche

11.4 Gameplay

11.5 Spielende

12 Schwierigkeiten und Problemestellungen

Die jeweiligen Schwierigkeiten ergeben sich im Detail. Die anfänglichen Schwierigkeiten sind die Nice-To-Haves gewesen. -Anklickbare Felder mit Markierung und Demarkierung -Datenbankimplementierung ja oder nein (Zweckmäßigkeit und Nutzen) -Apache Tom-cat ja oder nein -Neustart-Button, in der ersten Version vorhanden, im Update gelöscht -Chat für die Mitspieler (Ja/Nein)

13 Fehlerdokumentation und Glossar

13.1 Fehlerdokumentation

00ff Fehler beim Beenden des Spieles ohne einen Spieler angelegt zu haben. Nullpointer Exception weil kein Spieler aus dem Array entfernt werden kann.

00fd Fehler weil der Bingo Thread nicht beendet wird. Läuft selbst nach dem Beenden aller Clients und des Spiels weiter und verursacht dann eine Nullpointer Exception.

00ee Fehler beim Erzeugen von Namen die doppelt sind. Angehängtes Zeichen wird nur für die ersten genommen aber nicht für schon doppelte.

00ss Beim Erneuten Starten ist der Server nicht resettet, das heisst der Server muss jedesmal neu gestartet werden.

.
.
.
.
.

13.2 Fazit

Allgemein gesehen eine lustige durchaus Unterhaltsame Anwendung, die mit einer professionelleren Server-Client Anbindung und einer überarbeiteten GUI sowie erweiterten Test bestimmt kommerzieller Natur sein könnte. Eine klare Aufgabenverteilung und gute Dokumentation sowie eine gute und schnelle Kommunikation sind Pflicht für solch ein Projekt. Viel Spaß beim Gameplay!