Software-/Programmierpraktikum WS07 Abalone

Nicolai Kamenzky, Christopher Oezbek, Olufemi Rosanwo, Jan Siwy
{kamenzky, oezbek, rosanwo, siwy}@inf.fu-berlin.de

Institut für Informatik
Freie Universität Berlin
Berlin, Germany
http://www.inf.fu-berlin.de/inst/agse/

February 2007

Version 1.1

Abstract

This document contains the requirements for the game client to be built by the participants of the software/programming laboratory class winter term 2007. The two player game to implement is called Abalone and is played on a hexagonal playing field. The client is to be written within 3 weeks by a team of 10 people. The homepage of the class can be found at projects.mi.fu-berlin.de/w/bin/view/SE/ProgrammierPraktikum2006. This document is written in English as an additional challenge for the class.

Contents

1 Introduction

1.1 Theme/purpose of the Abalone Game Client

The Abalone game client is supposed to be a simple gaming application which allows users to play the game Abalone both off- and online against other players and artificial intelligences. For playing online, the client implements the GAme Server Protocol (GASP) which is specified in a separate document and can be found on the homepage.

1.2 Requirements priorities and notation

The requirements described in this document are categorized into three different priority levels and each requirement is marked accordingly:

MUST marks an essential requirement. Unless all MUST-requirements in all MUST use cases are implemented, the system is considered inacceptable. MUST

SHOULD marks an important requirement. If some of these requirements are not implemented, the system is considered incomplete, but acceptable. SHOULD

MAY marks an optional requirement. These requirements are considered nice-to-have but need not be implemented when time is short or if their cost-benefit ratio is considered too high. MAY

Each requirement is marked by exactly one of these three terms, followed by a subscript number that is the unique reference number of that requirement (also repeated in the margin).

If a compound requirement (a use case) is marked MUST, SHOULD or MAY, it will be considered realized only if all of the MUST requirements contained in it are realized. It does not mean that all of its subrequirements must be realized.

Phrases *in italics* in the use cases below are references to further use cases, the corresponding section number is appended in parentheses as a hyperlink.

1.3 Definitions and Terms

The following terms are used in this requirements document with a specific meaning:

- A *client* is short for client application and refers to a piece of software running on the machine of a user with the possibility to connect to a server.
- A *user* is a human being using an application.

- A *server* is short for server application is a piece of software running at a remote site and handling the interaction of several clients.
- A *player* is a user that is playing a game.
- A *team* is a set of students within this class that have to implement the client.
- The *organizers* refers to the Übungsleiter and tutors for this class.
- A *match* is an instance of a game.
- A *dialog* is a window presented to the user that either asks for input or informs him/her about a problem/state/etc.

2 Functional requirements: Use Cases

The following main scenarios exist:

- The *user configures the application and possibly creates a user account* (2.1) before *connecting to the server* (2.2) and *playing online* (2.5).
- The *user plays offline against other players, an artificial intelligence or watches two AI players play against each other* (2.3) .
- The *client is run in autoplay mode* (2.4).
- The *user saves a match and loads it later on in offline mode* (2.6).
- The *user looks at a saved match in replay mode* (2.7).

2.1 Sign-Up and Configuration

MUST$_1$ M 1
The client MUST$_2$ have a dialog to configure the server used to connect to, MUST$_3$ have a dialog M 2
for signing up a new user account with the server, and MUST$_4$ have a dialog to configure the user to M 3
connect with. M 4

2.2 GASP compliant network client

The client MUST$_5$ have a full implementation of the GASP-protocol as specified in GASP.txt which M 5
can be found on the homepage.

2.3 Offline play

MUST$_6$ M 6
The user must be able to play offline against another player (so-called “Hotseat” mode because both
players switch back in front of the same computer, MUST$_7$), play against an AI (MUST$_8$) or watch M 7
two AIs play against each other (MUST$_9$). It must be possible to save the currently ongoing match M 8
(see 2.6). If two artificial intelligences are playing against each other, it MAY$_{10}$ be possible to pause M 9
the match and it MAY$_{11}$ be possible to step the match (i.e. the next AI will execute a single move and m 10
then the match is paused again). m 11

It MUST$_{12}$ be possible to decide which player has which color (with Abalone white always begins M 12 the match).

The following requirements also must be implemented for 2.5 (playing onlilne): The users MUST$_{13}$ M 13
M 14 be able to see the current number of stones captured by both players, MUST$_{14}$ be able to see which
M 15 player's turn it is, MUST$_{15}$ be able to see the current board and after the match is over, MUST$_{16}$ be
M 16 able to see who won the match.

2.4 Artificial Intelligence Autoplay

M 17 MUST$_{17}$
M 18 It MUST$_{18}$ be possible to start the client in artificial intelligence autoplay mode using the following command line string:

```
<app> autoplay <server:port> <user> <pass> [<# of matches>]
```

M 19 The client MUST$_{19}$ then connect to the server using the provided credentials, and MUST$_{20}$ play by us-
M 20 ing the AUTOPLAY command of the GASP protocol (and following the defined sequence afterwards).
M 21 After finishing a match, the player MUST$_{21}$ directly start another match (again using AUTOPLAY) until the given number of matches is reached.

M 22 After the client has played the matches, it MUST$_{22}$ output each match played in the format given in
M 23 the Abalone specification (Abalone.txt) to standard output. The client then MUST$_{23}$ output aggregate statistics about the matches played containing number of matches won, lost, that ended in error and the total number of matches played.

M 24 To play in a tournament the autoplay command-line mode also MUST$_{24}$ accept that the number of
M 25 matches parameter is ommitted. In this case the client MUST$_{25}$ play indefinitely (using the AUTOPLAY command) until it is sent the BYE command by the server after a match.

S 26 The client SHOULD$_{26}$ make use of the 'resume match'-feature of the GASP-protocol for instance when the connection to the server was lost or the client crashes (this is highly recommended since the client must not assume a reliable connection / server).

S 27 The client SHOULD$_{27}$ open a graphical user interface that SHOULD$_{28}$ show the currently ongo-
S 28 ing match and SHOULD$_{29}$ display statistics about the results of the previous matches. The user
S 29 SHOULD$_{30}$ be able to cancel the autoplay in progress. The client MAY$_{31}$ provide a way to start the
S 30 autoplay mode from the GUI in addition to the command-line.
m 31

2.5 Playing online

M 32 MUST$_{32}$

M 33 If the user wants to play on the server, the client MUST$_{33}$ provide a way to connect and SHOULD$_{34}$
S 34 have an indicator about the status of the connection. The client then MUST$_{35}$ display to the user a
M 35 list of all users currently available to play against on the server (refreshed every 15 to 60 seconds). It
M 36 MUST$_{36}$ be possible to ask these users to play a match, and it MUST$_{37}$ be possible to accept or reject
M 37 such a request by another user. Playing online against another user follows the same Abalone rules as
S 38 on a single machine. The client SHOULD$_{38}$ provide the possibility to reconnect to a interrupted match
M 39 (for instance because of a lost internet connection) and MUST$_{39}$ provide the possibility to answer the

M 40 request to resume a interrupted match. The user MUST$_{40}$ the possibility to resign from a match and MUST$_{41}$ the possibility to disconnect from the server. M 41

2.6 Saving and Loading a Match

MUST$_{42}$ M 42

It MUST$_{43}$ be possible for the user to save a match that is currently on-going at any time (i.e. during the match or after the match is done) to disk using the match-format described in the Abalone.txt file. It MUST$_{44}$ be possible to load a stored match that has not been finished, even if created with a different client but in conformance to the format definition, and resume the match in off-line mode against an artificial enemy or using the hotseat mode. Games that are stored after they have finished are only interesting for the replay feature described in 2.7. M 43 M 44

2.7 Replay

MAY$_{45}$ m 45

It would be a nice feature if the client supports loading any match in *replay mode*. It MUST$_{46}$ be possible to step through the match move by move, rewind to the start (MUST$_{47}$) and go to the end (MUST$_{48}$). It MAY$_{49}$ be an interesting feature for AI developers to take a stored match and go to a specific move in the match and then let the artificial intelligence make a next move (to see if it comes up with a better response now). M 46 M 47 M 48 m 49

3 Non-functional requirements

3.1 User interface & Usability

M 50 The user interface MUST$_{50}$ conform to the above-mentioned requirements (as far as they are realized at all) in a sensible way with respect to the arrangement and markup of the user controls, views, labels, prompts and explanations that guide the user. Within those limits each group is free to organize and design the interface as seen appropriate.

S 51 The user interface SHOULD$_{51}$ provide sufficient explanation of all uncommon concepts to guide a user who does not have prior knowledge about these topics, for instance as how to play the game,
M 52 save and load matches, etc. Make use of external links where needed. You MUST$_{52}$ not include copyrighted material without permission of the copyright holders. All copyrighted material that you
M 53 do use (for instance material under a software license that is compatible with yours) MUST$_{53}$ be mentioned in the about dialog.

S 54 The user interface SHOULD$_{54}$ match general usability guidelines and be simple to use. The user
m 55 interface certainly MAY$_{55}$ be fancy, creative or special, just make sure that you cover the MUSTs first.

S 56 Error handling of the application SHOULD$_{56}$ be nonintrusive to the user, i.e. that only those message should be displayed to the user that are important and cannot be handled by the application itself.

3.2 Platform compatibility

M 57 The client MUST$_{57}$ work fully under Linux and Windows. The client MUST$_{58}$ provide webstart
M 58 functionality if supported by the programming language and platform. It MUST$_{59}$ be possible to
M 59 install and run the client on a machine on which the user does not have admin rights (installation may be local for the user in this case).

3.3 Scalability & Session timeout

M 60 The GUI client MUST$_{60}$ run at acceptable speed (comparable to other GUI applications) on an average PC with more than 1 GHz and 512 MB of RAM.

M 61 The artificial intelligence MUST$_{61}$ support a mode where it returns a move within 10 seconds (for the
M 62 tournaments). In (non-tournament) autoplay mode on the server, the artificial intelligence MUST$_{62}$
S 63 return a move within 30 seconds. For offline play, the artificial intelligence SHOULD$_{63}$ return a move within 10 to 30 seconds.

M 64 If the connection to the server is lost, the client MUST$_{64}$ reconnect within 60 seconds to prevent losing

M 65 the match. If the server is restarted, clients MUST$_{65}$ try to reconnect every 60 seconds. If a client does not reconnect within 5 minutes after server-startup the match is lost in autoplay mode. The number of resumes and time-violations is limited to 10 per match. A time-violation occurs if the client requires more than 3 seconds longer than the given deadline or a multiple of the deadline (on a 10 second time limit, a violation occurs at 13 seconds, 23 seconds, 33 seconds ...).

3.4 Persistence

The following information must be stored persistently:

- All connection information like username, password and server setting (MUST$_{66}$). M 66
- Information necessary to resume an interrupted match (MUST$_{67}$). M 67

3.5 Programming style & License

Both all identifiers and all comments in the source code and helper files MUST$_{68}$ be either completely in English OR German. The group should make a decision about this at the beginning of the project. M 68

Prior to commit to the version repository the source-code MUST$_{69}$ be formatted using the Eclipse standard format options (in a Java editor do right-click, Source, Format). M 69

The project must contain user documentation (how to run, install, uninstall, use the application, MUST$_{70}$) and developer documentation (how to check-out, build, develop and create a new release, M 70
MUST$_{71}$) in a top-level folder called `docs`. M 71

Each non-trivial public program element (such as a method) MAY$_{72}$ be documented (purpose, us- m 72
age) and every source code file SHOULD$_{73}$ be documented at least globally (purpose, usage, role in S 73
design).

The project MUST$_{74}$ be licensed under an OSI certified Open Source license. It SHOULD$_{75}$ be M 74
licensed under the GPLv2. All source code project files MUST$_{76}$ bear the license header of the chosen S 75
license (for the GPL you can find this information at the bottom of the license `http://www.gnu.org/licenses/gpl.txt`) and the associated copyright. It is recommended to make this choice at the beginning of the project and modify the templates for source code files in the IDE accordingly. M 76

If you include third party material you MUST$_{77}$ ensure that you do not violate copyright restrictions M 77
and you MUST$_{78}$ mark all such third party material by using the name of the original authors and a M 78
sentence that declares that this was not your work.

Each member of the group SHOULD$_{79}$ commit code to the repository using his or her own splineforge handle, pair programming being the only reason that this is a SHOULD and not a MUST. Groups in which only one or two developers write all the code will not be tolerated. S 79

All programming artifacts MUST$_{80}$ be written in the Java programming language using a JDK version 1.5 or below (unless it is specifically permitted by the organizers to do otherwise). M 80

3.6 Website

M 81 Each group MUST$_{81}$ have a representative website with at least the following content:

- S 82 M 83 Possibility to download source code (SHOULD$_{82}$) and binaries (MUST$_{83}$).
- M 84 M 85 m 86 M 87 Link to bug-tracker (MUST$_{84}$), repository (MUST$_{85}$), mailing-list (MAY$_{86}$), user documentation (MUST$_{87}$), etc.
- M 88 List of all developers, their contact-details and responsibilities (MUST$_{88}$).
- M 89 Screenshots of the application (MUST$_{89}$).
- M 90 A link to the website of the class (MUST$_{90}$).

4 Rules for development

4.1 What is allowed

During the three weeks of the class you may:

- Use any tool, library, framework, and other software you find helpful to write the software described above unless it is Abalone specific, which is not allowed.
- Discuss with all other participants about the requirements, solutions and implementation possibilities.
- Use any development process you deem useful as long as you keep to the daily stand-up meeting.
- Ask the organizers any question you like regarding the requirements, priorities, implementation possibilities, technical and managerial problems (see 4.3). We are there to help you.

4.2 What is not allowed

During the contest you must not:

- Disturb other teams in their work.
- Attack or sabotage the development of other teams, the webserver or the game server. If you suspect vulnerabilities you are encouraged to publically disclose them on the mailing-list.
- Include work products (code, documentation, etc.) from other teams in your solution without explicit permission unless the code is publically posted in either the forum or the mailing-list, you mark the sections in your solution appropriately ("The following code was taken from the email by ... on Monday, the "... "End Included Code") AND the total length of third party source code (excluding source-code of precompiled binaries from outside the class) does not exceed 5% of the total number of formatted source code lines. If in doubt, check with the organizers.

Teams which are in violation of one of these points risk losing their certificate for the class.

4.3 Requests for Clarificiation and Updates

If problems arise with this specification or any of the sub-documents you should contact the organizers directly or even better using the mailing-list so that everybody benefits from uncertainties found.

We will happily answer questions regarding clarification of the meaning of requirements (but beware: if our advice and this document should ever be in conflict, it is this document that is relevant for the grading, not the advice). The organizer team will discuss with you questions regarding requirements priorities or regarding which of two concrete solution ideas would be better anytime, but prefer if this discussion takes place during the technical sessions, the forum or mailing-list.

We will issue clarifications to the mailing-list and post new document versions on the homepage if necessary, so make sure that you are subscribed to the mailing-list.

4.4 Final Delivery

You are allowed to finish your development at any time you think appropriate, but no later than tuesday 13.03.2007 midnight 16:00.

Send an email to {oezbek, siwy, rosanwo, kamenzky}@inf.fu-berlin.de containing the following and complying with the following requests:

1. The version number of the SVN repository that we should consider to be the final one.
2. A url to the website where users can download your application, report bugs, look at screenshots etc.
3. A copy of the developer documentation mentioned in 3.5.
4. S 91 All members of your team SHOULD$_{91}$ participate in the Lehrevaluation at Institute für Informatik. Tokens will be emailed to you within the last week of the laboratory class and should be used by 18.03.2007. If you want to give personal or direct feedback, please feel free to stop by in room 008 or email oezbek@inf.fu-berlin.de. Thanks!
5. YOU ARE DONE. Go and sleep or party or bang your head against a wall — whatever you feel most like doing. We hope you had fun!

5 Clarification and Updates

- Version 1.0 - 19.02.2007 - 12:00 - Initial Release
- Version 1.1 - 26.02.2007 - 14:47 - Documentation Clarification
  - All identifiers and comments in the source code and helper files must be either in English OR German is relaxed to the following interpretation: You can either have (1) both the comments and identifiers in English or (2) both in German or (3) the identifiers in English and the comments in German or (4) the comments in English and the identifiers in German.