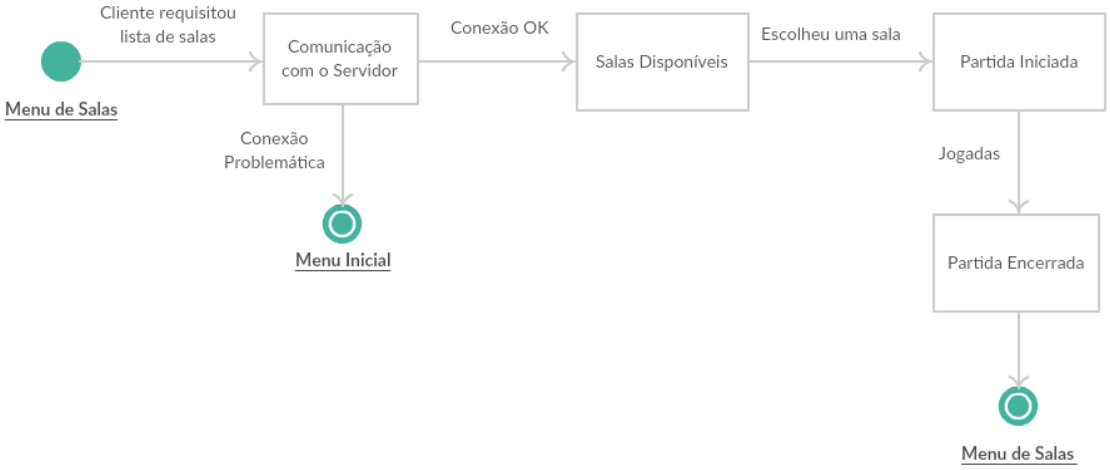
Engenharia de Software – Profª Ana C. V. de Melo
Projeto Kindred – Fase 2

Bruno Guilherme Ricci Lucas (4460596)
Leonardo Pereira Macedo (8536065)
Vinícius Bitencourt Matos (8536221)

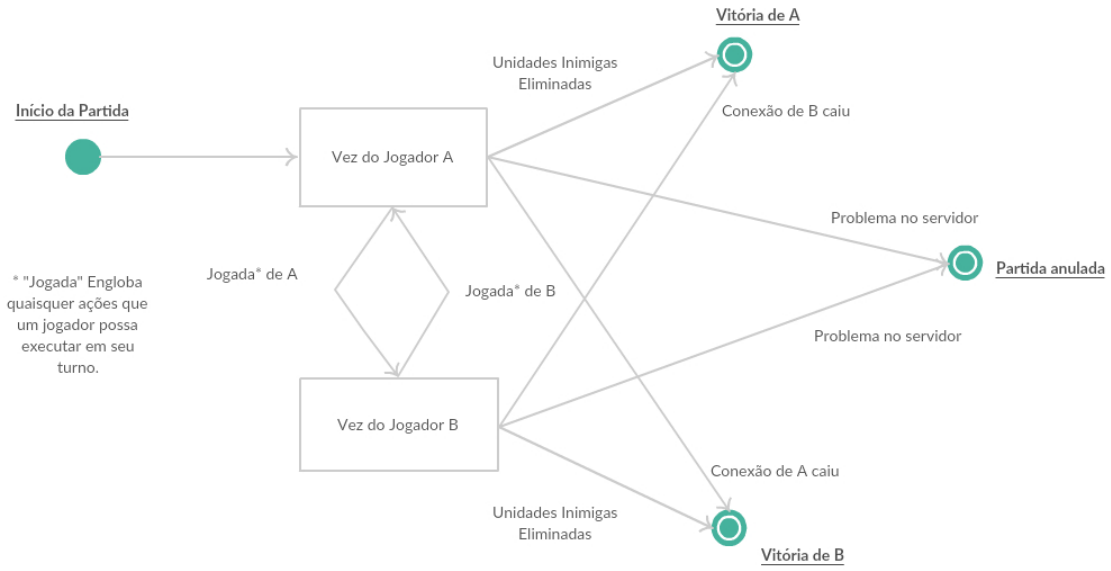
17 de novembro de 2015

1 Statecharts

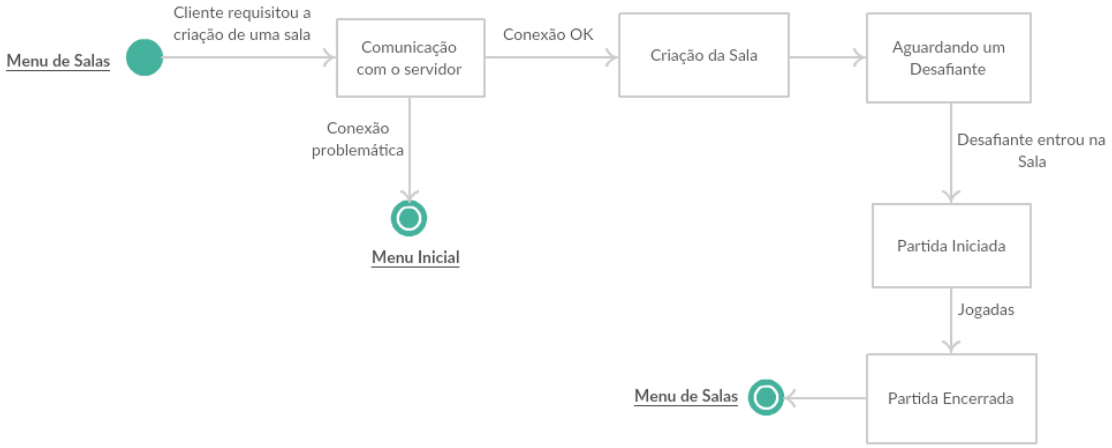
1.1 Busca por sala



1.2 Partida



1.3 Criação de sala



2 Projeto arquitetônico

2.1 Estrutura: sistemas e subsistemas

2.2 Projeto lógico de controle

3 Projeto da interface do sistema

3.1 Interface interna (entre objetos)

3.2 Interface humano-máquina (telas do sistema – diálogos)

4 Projeto de classes do sistema

5 Implementação – código-fonte e executável

A implementação foi feita em Java 7 e encontra-se .....
Para compilar, ...

6 Manual de uso do sistema (para o usuário)

Encontra-se no documento *UserManual.pdf*.

7 Relatório de modificações

7.1 Introdução

Abaixo encontram-se observações sobre mudanças feitas em relação à fase anterior, bem como outras informações sobre a implementação do projeto.

7.2 Sobre a parte gráfica

A principal mudança a ser observada foi a ausência da prometida parte gráfica do jogo. Acreditávamos, na Fase 1, que seria possível implementá-la usando a biblioteca Pygame, da linguagem Python, a qual um dos integrantes do grupo possuía já uma boa experiência. Ligaríamos esta biblioteca ao projeto através da linguagem Jython. Infelizmente, devido à complexidade envolvida na mistura de duas linguagens diferentes, percebemos que tal tarefa seria complexa e demandaria muito tempo.

Foi considerada a possibilidade de montar uma interface gráfica por meio de alguma ferramenta em Java. Esta iniciativa pode ser vista no próprio código, através da criação de uma classe abstrata *AbstractView.java* para generalizar a interface com o usuário (CLI ou GUI). Havíamos criado uma classe *GUI.java*, mas devido à falta de tempo não foi possível implementá-la, apesar de que já existe compatibilidade atual no projeto para isso ser feito.

Com isso, acabamos nos concentrando em fazer primeiro uma CLI através do Terminal dos sistemas Unix. Fez-se uso de cores de fundo para diferentes *terrenos*, uso de cores de fonte para diferenciar as *unidades* dos jogadores e o caractere para diferentes *unidades*. Ficamos um pouco limitados às possibilidades oferecidas pelo Terminal, mas a adaptação ficou boa. A jogabilidade acabou ficando limitada a digitar comandos e esperar uma mensagem de resposta. Não muito confortável, mas era a opção disponível para a CLI feita.

7.3 Adaptações e simplificações

Outro problema foi decorrido de um dos integrantes do grupo ter se afastado da disciplina por problemas de saúde, reduzindo a eficiência e aumentando o tempo necessário para realizar as tarefas, pois ficamos com apenas 3 integrantes no grupo.

O próprio jogo acabou ficando bem complexo, com aproximadamente 4000 linhas de código ao total. Infelizmente, por falta de tempo, não foi possível realizar muitos testes no código. Pretendemos, se possível e tivermos tempo, realizar algumas correções para a Fase 3 (fase de testes).

Por consenso entre o grupo, decidimos fazer o código em Inglês. Por isso, pode haver uma pequena estranheza entre os nomes em Português na Fase 1 e os nomes em Inglês da Fase 2, mas na grande maioria dos casos só houve tradução dos termos. Neste relatório, o uso de termos em Português refere-se à fase anterior, enquanto o uso de Inglês refere-se à fase atual.

Simplificações, que serão explicadas um pouco abaixo, foram feitas para não tomar tempo demais no código fonte.

Em relação às classes a serem implementadas no jogo, as mudanças principais ocorridas foram:

1. Junção de *Unidade*, *Classe* e *Arma* numa classe só, **Unit**, para simplificação. Como o jogo não ficaria muito sofisticado, o desenvolvimento de uma *Classe* e *Arma* para a *unidade* só levaria a complicações desnecessárias; foi decidido que *Classe* iria se referir ao tipo a qual a *unidade* pertence (dando exemplos do próprio jogo: *Archer* e *Knight*). A ideia geral do jogo, no entanto, permaneceu inalterada: atributos para *unidades*, o mapa e a ideia de um jogo por turnos permaneceram.

2. Especificação melhor da parte Online do jogo. Por causa disso, do lado do cliente, as antigas classes Online e Jogador foram juntadas numa só, **Client**. A junção fez sentido, uma vez que estipulamos que o jogo seria somente online desde a Fase 0.

3. Definição melhor do servidor e do que ele deveria guardar dos clientes. Decidimos que o servidor iria trocar mensagens com o cliente quando este estivesse fora de uma partida. Durante um jogo, o servidor apenas passaria o comando feito pelo usuário para seu adversário, sem executar nada. Ou seja, o servidor não guarda estado das partidas, servindo apenas como um intermediador de informações que não realiza nenhuma verificação sobre comandos ligados ao jogo.

4. Simplificação do servidor, pois este śó possui um comando para interagir diretamente com quem está controlando-o (“CLOSE”, que fecha o servidor com segurança). Por falta de tempo, o servidor não produz arquivos de log informando tudo que ocorre, apesar de ele imprimir na saída padrão quando um cliente se conecta ou se disconecta.

Sobre a máquina de estados, melhor retratada no *Diagrama de Atividades* da Fase anterior, a principal mudança foi:

1. Simplificação no início de uma partida: primeiramente, quando um usuário se conecta a uma sala, não há pedido de confirmação entre ambos os lados; os dois jogadores já iniciam a partida. Em segundo lugar, os jogadores não posicionam suas *unidades* no início da partida; estes estão pré-configurados no mapa, ou seja, a quantidade, a *Classe* e a posição de cada *unidade* estão fixos e independem dos jogadores.