4.
O SimulES-W

*Este capítulo apresenta o SimulES-W (Monsalve, 2010). Um jogo educacional de cartas que simula o processo de desenvolvimento de software. O jogo permite que um aluno assuma o papel de gerente de projeto e simule problemas que não são exercitados em aulas tradicionais. SimulES-W é um software colaborativo que implementa uma versão adaptada de SimulES (*Figueiredo et al., 2007*). Sua primeira versão em modelado e desenvolvimento é apresentado em Monsalve (2010) e usado nesta tese como uma abordagem de GBL para ensinar engenharia de software e testar os conceitos de transparência pedagógica. Diante disso, esta seção contextualiza o leitor sobre a dinâmica do jogo e sua modelagem, e também a evolução e as experiências de uso do jogo SimulES-W como ferramenta de ensino de engenharia de software.*

4.1.
Definição de Termos

Para o entendimento da modelagem usada na concepção e evolução do SimulES-W, esta seção (4.1 Definição de Termos) está fortemente baseada em Monsalve (2010) onde estes conceitos são fundamentos desse trabalho, aqui servem para contextualizar ao leitor:

Visão Geral do Framework i*

O *framework i** (*i de intencionalidade e estrela de distribuída*) (Yu, 1995) permite modelar contextos organizacionais baseado nos relacionamentos de dependência intencional entre os atores. Esse *framework* é usado para se obter um melhor entendimento dos relacionamentos organizacionais com base na interação entre atores.

Identificar estes relacionamentos entre os atores ajuda os engenheiros de requisitos na elicitação de metas nas fases iniciais. Essas metas serão identificadas respondendo a questionamentos sobre o porquê dos atores terem estas dependências organizacionais. O foco do *framework i** é possibilitar a compreensão das razões, tanto internas dos atores como aquelas que são

compartilhadas entre eles, uma vez que as mesmas são expressas explicitamente, auxiliando na escolha de alternativas durante a etapa de modelagem do software. Nesse framework, os participantes do contexto organizacional são chamados de atores. Conforme Yu (1995), atores são entidades ativas que efetuam ações para alcançar metas através do exercício de suas habilidades e conhecimentos, podendo ter dependências intencionais entre eles. Uma dependência intencional ocorre quando dois ou mais atores compartilham algum elemento de dependência definido no *framework i*.* A seguir uma descrição geral dos elementos de dependência:

(i) *Meta concreta:* é uma condição ou estados de desejos no mundo que o ator deseja alcançar. Não é especificado como a meta deve ser alcançada, o que possibilita que várias alternativas sejam consideradas.

(ii) *Tarefa:* especifica um modo especial de fazer alguma coisa. Quando uma tarefa é definida como subtarefa de outra tarefa, ela a restringe a um curso de ação em particular, especificado pela tarefa subtarefa.

(iii) *Recurso:* é uma entidade (física ou informacional) para satisfazer uma necessidade que não é considerada problemática pelo ator. Suas principais características são quem o disponibiliza e se efetivamente está disponível.

(iv) *Meta flexível:* condição ou estado no mundo que o ator deseja alcançar. É diferente de uma meta concreta, pois a condição para ser alcançada não é definida desde o início, e além disso, ela pode estar sujeita à interpretação. Conforme Cunha (2000), quando uma meta flexível é uma componente em uma decomposição de tarefa, ela serve como uma meta de qualidade para aquela tarefa, guiando (ou restringindo) a seleção entre as alternativas para a decomposição da tarefa.

Conforme Yu (1995), dois modelos dentro do *Framework i* foram propostos*: o modelo SD (sigla em inglês de *Strategic Dependency*) e o modelo SR (sigla em inglês de *Strategic Rationale*), descritos a seguir.

O Modelo SD – Strategic Dependency

O modelo SD descreve os relacionamentos de dependência estratégica entre os atores da organização, utilizando para isso uma rede de nós, para representar atores e arestas para representar as dependências entre os mesmos.

No modelo SD, um relacionamento baseado em cooperação entre dois atores representa a dependência que existe entre eles, onde o primeiro é designado "*depender*" ou "dependente" e o segundo "*dependee*" ou "de quem se depende", unidos por um elo de dependência, designado como "*dependum*". Essa relação de dependência ou *dependum* pode representar uma meta a ser atingida, uma tarefa a ser executada, um recurso a ser fornecido ou uma meta flexível, satisfeita a contento. A seguir as definições do *dependum* segundo Yu (1995):

(i) Dependência por meta: acontece quando o *depender* depende do *dependee* para que certo estado do mundo seja alcançado. Como acontece na Figura 4.1, ao *dependee* é dada a liberdade de execução da meta. Portanto, com uma dependência por meta, o *depender* ganha a habilidade de assumir que a condição ou estado do mundo será alcançado. É assim que o *depender* se torna vulnerável, pois o *dependee* pode falhar ou não realizar tal condição;

(ii) Dependência por tarefa: acontece quando o *depender* depende do *dependee* para que este último realize uma tarefa. Esse tipo de relação especifica "como" a tarefa deve ser realizada, mas não menciona "o porquê". Como na Figura 4.1, o Jogador *depender* do SimulES para que os recursos sejam disponibilizados. Isso faz com que o *depender* (Jogador) seja vulnerável, pois o *dependee* (SimulES) pode falhar ou não executar a tarefa.

(iii) Dependência por recurso: acontece quando o *depender* depende de outro, o *dependee*, para que uma entidade (física ou computacional) seja disponibilizada. Como mostrado na Figura 4.1, o *depender* (jogador) ganha a habilidade de usar a entidade (Cartão do projeto) como um recurso. Esse fato faz com que o jogador fique vulnerável, pois a entidade ou recurso pode tornar-se indisponível.

(iv) Dependência por meta flexível: acontece quando o *depender* depende do *dependee* para que certo estado do mundo seja alcançado, porém diferentemente de uma meta (rígida ou concreta), pois o critério para a condição ser alcançada não é definido a princípio, estando sujeito à interpretação. Quer dizer que, para metas flexíveis, são usados os termos "satisfeita a contento", conforme Simon (1981), ou "razoavelmente satisfeita". Esta subjetividade faz com que a satisfação de metas flexíveis seja inerente a requisitos não funcionais, por se tratar de aspectos de qualidade (Cunha, 2000); como é ilustrado na Figura 4.1, não há critérios claramente definidos para sua satisfação.

Além disso, o *depender* (Jogador) será vulnerável, pois o *dependee* (SimulES) pode falhar ao realizar tal condição.

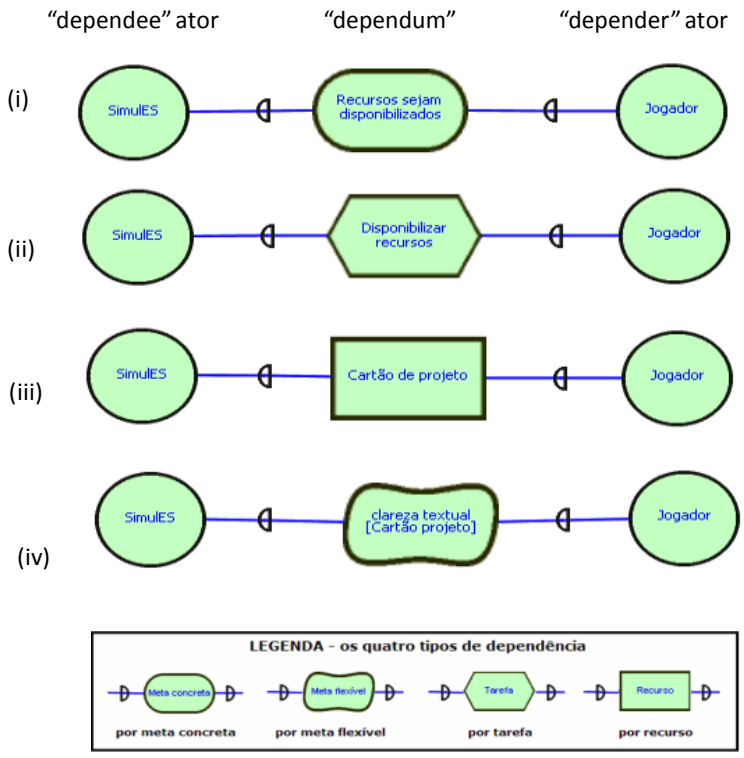


Figura 4.1 – Ilustração da dependência entre atores utilizada em modelos SD (Monsalve, 2010)

O Modelo SR – Strategic Rationale

O modelo SR tem como objetivo representar as estratégias internas de cada ator, fornecendo uma descrição intencional do contexto organizacional em termos dos elementos e das decisões e escolhas por trás dele. O raciocínio (*rationale*) dos atores é representado explicitamente no modelo SR, proporcionando um entendimento detalhado do contexto organizacional. Além disso, são elaborados de modo a expressarem o processo pelo qual: as metas são alcançadas, as tarefas são executadas, os recursos são disponibilizados e as metas flexíveis são refinadas (decompostas) e operacionalizadas. A representação destes relacionamentos intencionais, que são internos aos atores, são conhecidas como "*meios-fim*". Tal representação fornece explicitamente "o *porquê"*, o "*como"* e as alternativas.

O modelo SR é um grafo com quatro tipos de nós, idênticos aos tipos de *dependum* em um modelo SD: meta, tarefa, recurso e meta flexível. O modelo

SR possui duas classes principais de elos: elos de decomposição de tarefa e elos “meios-fim”.

(i) *Elos de decomposição de tarefa*: Uma tarefa (cuja noção está associada a “como fazer alguma coisa”) é modelada em termos de sua decomposição em suas subcomponentes, que podem ser: metas, tarefas, recursos e/ou metas flexíveis (os quatro tipos de nós). Na representação com elos de decomposição de tarefa, apresentado na Figura 4.2, a tarefa “B” é decomposta em uma (sub) meta: “submeta M”; uma (sub) tarefa: “subtarefa S”; e uma meta flexível: “meta flexível F”.

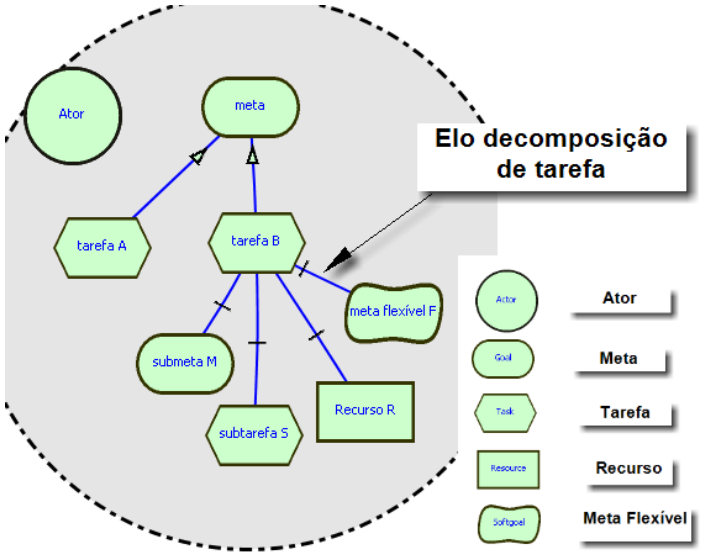


Figura 4.2 – Decomposição de tarefas (Monsalve, 2010)

(ii) Elos Meios-Fim: os elos do tipo meios-fim, segundo Cunha (2000), representam relacionamentos do tipo: uma meta a ser alcançada, uma tarefa a ser feita, um recurso a ser produzido, ou uma meta flexível a ser razoavelmente satisfeita (*satisficed*) nomeado como “fim” e os “meios” alternativos para alcançá-los. Na Figura 4.2, a “Tarefa B” representa o “meio” pois a noção de tarefa está associada a ‘uma atividade que tem que ser feita’. Além disso, a notação gráfica do *i*,* apresentada na Figura 4.3, indica-nos que uma cabeça de seta aponta dos meios para o fim. Os tipos de elos meios-fim estão descritos a seguir:

Elo Meta –Tarefa – nesta ocorrência, o “fim” é especificado como uma meta e o “meio” como uma tarefa. A tarefa especificada “como” através da decomposição em seus componentes.

Elo Recurso-Tarefa – nesta ocorrência, o "fim" é especificado como recurso e o "meio" como uma tarefa.

Elo Meta Flexível-Tarefa – nesta ocorrência, o "fim" é especificado como uma meta flexível, e o "meio" é especificado como uma tarefa.

Elo Meta Flexível-Meta Flexível – nos elos deste tipo, tanto o "fim" quanto o "meio" são metas flexíveis, permitindo o desenvolvimento de uma hierarquia meio-fim de metas flexíveis. Isso permite que as metas flexíveis sejam cada vez mais refinadas, até chegar a metas flexíveis mais fáceis de operacionalizar (Cunha, 2000).

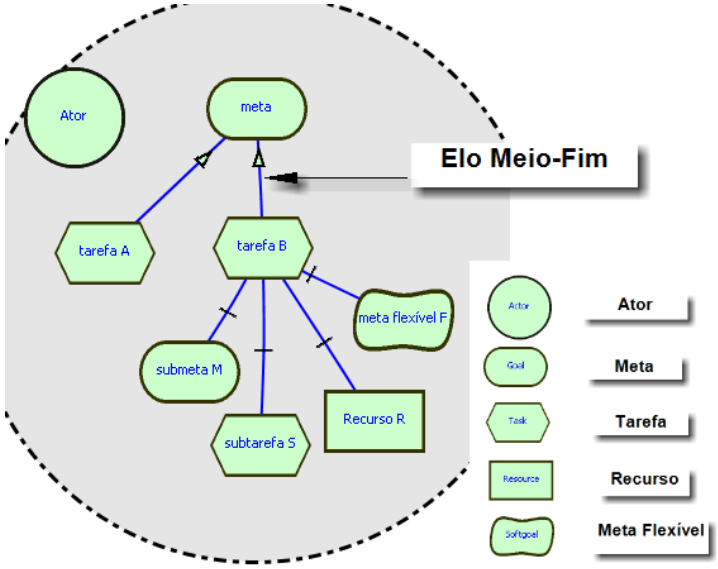


Figura 4.3 – Elos meios-fim (Monsalve, 2010)

Na Figura 4.3, representa-se o elo do tipo Meta-Tarefa, onde a meta "fim" tem duas tarefas "meio" "A" e "B".

O Modelo SA – Strategic Actor

O modelo SA é uma extensão do framework *i** e tem como objetivo modelar atores em *i*.* Ele auxilia no entendimento dos atores e seus relacionamentos, do ponto de vista estrutural (Oliveira, 2008).

O modelo SA utiliza os conceitos de agente, posição e papel, definidos em Yu (1995), para refinamento de atores e seus relacionamentos. A seguir serão descritos alguns conceitos, conforme Yu (1995):

Ator - um ator é uma entidade ativa que executa atividades para atingir suas metas conforme o exercício de seu conhecimento (know-how) e habilidades (Yu, 1995). O termo ator é usado para fazer referência a qualquer unidade à qual se possa atribuir dependências intencionais (Yu, 1995), (Cunha, 2000).

Posição - uma posição é formada por um conjunto de papéis desempenhados por um agente (Oliveira, 2008). Ele localiza-se em um nível intermediário de abstração entre um papel e um agente.

Papel - um papel é uma caracterização abstrata do comportamento de um ator social em algum contexto especializado (Yu, 1995), onde suas características podem eventualmente ser transferidas para outros atores sociais.

Agente - um agente é um ator com manifestações físicas concretas, tal qual um ser humano (Yu, 1995). O termo agente pode fazer referência tanto para humanos quanto para agentes artificiais (hardware/software). Suas características geralmente são intransferíveis para outros indivíduos, como habilidades, limitações e experiências (Yu, 1995) .

Agente Real - conforme Oliveira (2008), um novo tipo de ator foi proposto para o relacionamento de instanciação entre agentes: o agente real. Assim, é mais específico que o agente, um agente genérico, e pode ser identificado, como por exemplo, uma pessoa específica ou um software específico.

Um resumo dos relacionamentos descritos por Oliveira e Cysneiros (2007), os quais fazem uso das definições e exemplos de Yu (1995) são descritos a seguir:

É um (*ISA*): relação de especialização de ator para ator, papel para papel, posição para posição e/ou agente para agente.

Instância (*INS*): relacionamento de agente real para agente. Ou seja, um agente pode ter diferentes agentes reais como instância.

Ocupa (*OCCUPIES*): relacionamento de agente para posição. Um agente pode ocupar mais de uma posição e uma posição pode ser ocupada por mais de um agente (Cunha, 2000).

Cobre (*COVERS*): relacionamento de posição para papel. Uma posição cobre mais de um papel e um papel pode ser coberto por mais de uma posição.

Desempenha (*PLAYS*) – relacionamento de agente para papel. Um agente pode desempenhar mais de um papel e um papel pode ser desempenhado por mais de um agente.

Parte de (*is–Part–of*): relacionamento de agente para agente, posição para posição e de papel para papel. Agentes, posições e papéis podem ser ou podem ter mais de uma subparte. Conforme Cunha (2000), há uma restrição para este relacionamento entre posições. Uma posição é parte de outra se todos os papéis desta posição também são cobertos pela outra posição.

A Figura 4.4 mostra um exemplo de um modelo de atores estratégicos e seus relacionamentos.

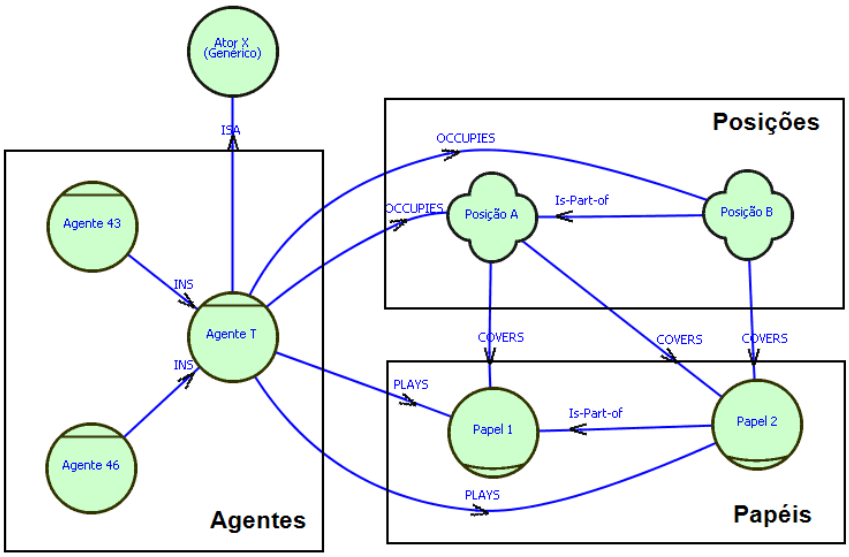


Figura 4.4 – Ilustração de um modelo SA (Oliveira, 2008)

Situação de Dependência Estratégica – SDsituations

As SDsituations foram propostas por (Oliveira e Cysneiros, 2007) como uma representação estruturada de situações de dependência estratégica entre atores dentro de um contexto organizacional. Conforme Oliveira (2008), uma SDsituation reúne um bloco de situações que compartilham um objetivo em comum.

Segundo Oliveira (2008), elementos de dependência como meta, tarefa, recurso ou meta flexível, os quais envolvem atores, não estão isolados. Portanto, é uma situação bem definida de colaboração, conhecida como “situação de dependência estratégica”. Além disso, uma SDsituation pode ser identificada separadamente das outras SDsituations, formando uma cadeia de interdependências entre elas.

Conforme Oliveira (et al., 2008), existem três tipos de interdependências entre SDsituations: física, lógica e temporal. Entretanto, mais de um tipo de interdependência pode ocorrer simultaneamente. Conforme Cunha (2000) os três tipos de interdependências serão descritos a seguir:

Física – “ocorre quando um recurso é preparado por uma SDsituation e é solicitado por outra SDsituation” (Cunha, 2000).

Lógica – “ocorre quando uma ou mais SDsituations necessitam da conclusão de outra SDsituation para sua iniciação, ou quando uma ou mais SDsituations dependem da conclusão de outra SDsituation para sua própria conclusão” (Cunha, 2000).

Temporal – “ocorre quando uma ou mais SDsituations necessitam esperar algum tempo após o início de outra SDsituation ou quando uma ou mais SDsituations necessitam esperar algum tempo após a conclusão de outra SDsituation” (Cunha, 2000).

4.2. SimulES-W: Um Jogo para o Ensino de Engenharia de Software

Semelhante à seção anterior, esta seção está fortemente baseada nos conceitos apresentados em Monsalve (2010) e serve para contextualizar ao leitor.

SimulES-W é uma abordagem de aprendizagem baseada em jogos, como jogo representa um projeto de software e tem como objetivo criar um produto de software. Foi modelado usando uma modelagem intencional, pois esta considera a interação entre os atores (Yu, 1995), adotando-se o método ERi*c (Oliveira, 2008) para facilitar a criação dos modelos iniciais. A descrição dessa modelagem pode ser encontrada em Monsalve (2010).

SimulES-W foi desenvolvido em Monsalve (2010), mapeando as informações dos modelos intencionais até o código, para isso foi escolhida uma arquitetura do tipo MVC (Netbeans 6.5, 2008) e como ferramenta de desenvolvimento a linguagem Java (Monsalve, Werneck e Leite, 2010b). Além

disso, foram utilizados os seguintes softwares de apoio: MySQL, Visual Web JavaServer Faces, Hibernate e jMaki Ajax. A seguir apresentaremos detalhes de sua origem e evolução.

As primeiras versões do jogo, antes do SimulES-W foram compiladas em um jogo de tabuleiro, conhecido como SimulES (Figueiredo, 2007). Ele está direcionado para o ensino de engenharia de software e foi concebido a partir da evolução do jogo "*Problems and Programmers*" (PnP) (Baker, 2003). A evolução do SimulES para um software que roda na Web, foi chamado de SimulES-W. Esta evolução foi realizada com base em um estudo de literatura sobre jogos educacionais na engenharia de software e nas experiências de uso do jogo (Serrano et al., 2007; Monsalve, 2010; Monsalve, Werneck e Leite, 2010a; Monsalve, Werneck e Leite, 2011), entre outras.

As diferentes versões do jogo conservam a filosofia geral, que é construir um produto de software baseado em um projeto comum para todos os jogadores. Para isso, o aluno assume o papel de gerente de projeto e deve lidar com o orçamento, contratação e demissão de engenheiros de software, de engenheiro de software ao decidir se inspeciona ou não, se coloca a carta como requisitos e construção de diversos artefatos necessários ao término do projeto, Além de criar uma estratégia e realizar “jogadas” que desestabilizem o jogo dos adversários, visando melhorar o seu jogo frente aos adversários. No entanto, os adversários podem responder com conceitos de engenharia de software e bloquear o ataque de um jogador. A partida tem um vencedor quando um jogador constrói todos os artefatos necessários para completar o projeto, satisfazendo os requisitos de qualidade estipulados, ou seja, é preciso completar os módulos do projeto definidos no cartão de projeto, selecionado no início do jogo, na qualidade exigida. Após serem construídos os artefatos, estes devem ser empacotados na jogada chamada de *rodada de ações* e depois submetidos como produtos finais.

Para realizar as jogadas, o jogo dispõe de diferentes *rodadas*. A primeira é a *jogada de inicio*. Nela é escolhido aleatoriamente o projeto que deve ser tratado durante todo o jogo, como também o primeiro *engenheiro de software* para cada um dos jogadores. A segunda é a *jogada de ações*, que começa com o lançamento do dado. De acordo com o número tirado, retira-se de 1 a 3 Cartas de *conceito* e *problema*. Se o valor do lançamento do dado for maior que 4, pode-se pegar cartas de *engenheiro de software* e a quantidade será determinada pelo resultado da diferença do valor do lançamento do dado – 3.. A terceira é a jogada de conceitos. Nela o jogador da vez deve escolher a

subatividade de construir, inspecionar ou empacotar os artefatos dispostos no tabuleiro individual. Na construção, pode-se retirar novas *cartas de artefato* (brancas ou cinzas) dependendo da complexidade do projeto e da habilidade de sua equipe de *engenheiros de software*. A habilidade determina quantos "pontos de tempo" ele tem e, portanto, quantas ações ele pode desempenhar por vez. Analogamente, o jogador pode inspecionar, dependendo também, da habilidade da sua equipe. Continuando, na *jogada de conceitos,* o jogador faz também tratamento de problemas e onde os jogadores podem atacar e serem atacados por seus adversários usando as *cartas conceitos* e *cartas problemas* com conceitos típicos da engenharia de software. Por último, *submissão do produto*, sendo que o primeiro jogador a chegar nesta instância sem defeitos no seu produto de software e que não tenha penalidade alguma a ser resolvida, ganha o jogo.

Para ilustrar, a Figura 4.5 apresenta a tela principal do SimulES-W onde são dispostas as mensagens trocadas entre os jogadores e as informações sobre as jogadas e estado do jogo. Nessa tela, temos também, as informações do projeto escolhido para a jogada e os jogadores on-line, e finalmente a lista das jogadas aceitadas ou não, entre os jogadores.

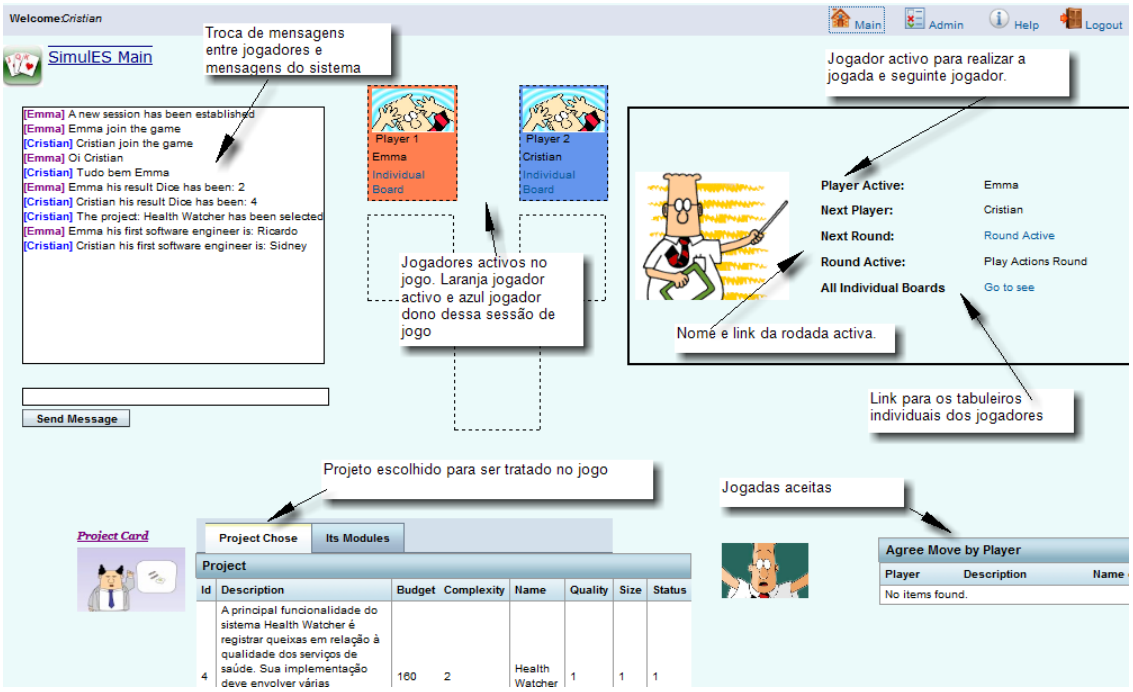
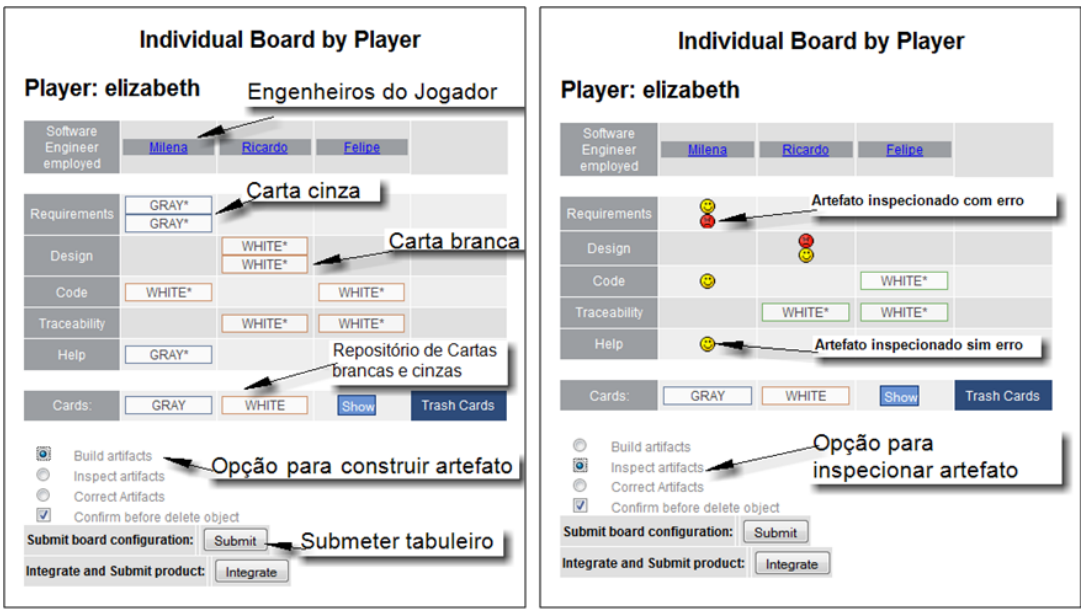


Figura 4.5 – Tela principal de SimulES-W (Monsalve, Werneck e Leite, 2013a)

Na Figura 4.6, temos o Tabuleiro Individual em duas situações: a) *construção de artefato,* onde o jogador tem os *engenheiros de software* contratados para a construção dos diferentes artefatos, cada um deles com um

link que apresenta as informações dos *engenheiros de software*. Na parte a) dessa figura, as cartas de artefato brancas e as cartas de artefato cinza aparecem sem terem sido desviradas ainda. Na parte inferior dessa tela aparecem, também, as operações possíveis de serem realizadas dentro do tabuleiro. A Figura 4.6 - parte b), apresenta uma situação de *inspeção de artefatos*, pois o jogador já construiu o artefato (parte a) e ele pode inspecioná-lo. Nessa segunda parte da Figura 4.6, é ilustrado o resultado de uma inspeção quando o artefato tem defeito e quando não tem. SimulES-W define, aleatoriamente, a qualidade da carta de artefato, branca ou cinza, sendo que as cartas brancas têm menos possibilidades de apresentarem defeito do que as cartas cinzas.

Uma das principais características do SimulES-W é a possibilidade de edição das cartas de problemas e das cartas de conceitos. Isso permite que o material a ser enfatizado pelo curso seja customizado pelo professor.



a) Construção de artefato b) Inspeção de artefato

Figura 4.6 – Tabuleiro Individual nas situações de Construção de Artefato e Inspeção de Artefato (Monsalve, 2010) *corrigir: “Artefato inspecionado SEM erro”

4.3. Experiências de Uso do SimulES-W

A evolução do SimulES para um software chamado SimulES-W foi realizada com base em um estudo de literatura sobre jogos educacionais em engenharia de software.

A primeira versão do SimulES-W surge no trabalho (Monsalve, 2010). A observação dos estudantes, jogando o SimulES de tabuleiro, ajudou no entendimento da dinâmica do jogo, na identificação da preferência dos estudantes pelo uso de jogos nas aulas de engenharia de software e na elicitação dos primeiros requisitos para o jogo web (Monsalve, Werneck e Leite, 2010a).

O jogo contava com um modelo situacional como fonte de informação (Serrano et al., 2007) e partiu para um modelo intencional com o *Framework i** (Napolitano, 2009). Isso também foi parte das fontes de informação para a versão web que seria criada. Esses modelos também foram evoluídos para representar o jogo web e melhor descrever a interação dos jogadores (Monsalve, 2010). Os progressos do SimulES-W que fazem parte desta tese são detalhadas a seguir:

A evolução do SimulES-W continuou com a aplicação do Método da Inspeção Semiótica (MIS) (de Souza et al., 2010). O MIS permite avaliar o artefato computacional desde a perspectiva do designer, ou seja, o engenheiro como um ator ativo através de sua metacomunicação com o usuário. Esta avaliação é feita através de uma verificação realizada pelo próprio engenheiro ou por um inspetor, com o objetivo de avaliar a interface, sem ter que envolver o usuário no processo. Para isso, um *framework* de perguntas preestabelecidas procura justificar as escolhas de signos e símbolos na interface. Deste modo, a inter-relação dos atributos da usabilidade podem ser avaliados com o método MIS e os critérios de usabilidade definidos em Leite e Cappelli (2010) para criar um software mais transparente. Como consequência, o MIS permitiu, no caso do SimulES-W, analisar como as escolhas e o entendimento do engenheiro influenciariam os usuários nos processos de significação, ou seja, foram observados e analisados os signos e como eles podem interferir no comportamento dos usuários frente ao software. Em sequência a esse trabalho, também foram realizados testes de usabilidade e de interface do usuário.

Continuando, conforme o reportado em Monsalve, Werneck e Leite (2011), as experiências do uso do SimulES-W permitiram verificar elementos de melhoria como: a) os alunos deviam receber mais formação sobre SimulES-W antes do jogo, b) o conteúdo das cartas precisava ser revisto, porque alguns eram de difícil compreensão pelos alunos; c) algumas regras do jogo precisavam ser melhores explicadas; d) conceitos / problemas deviam ser ajustados conforme o grupo de estudantes. O experimento conduzido foi importante para confirmar a aceitação do jogo pelos alunos. Eles relataram que SimulES-W era

divertido e gerava uma competição positiva, além de motivante. Em Monsalve, Werneck e Leite (2011), um dos jogadores escreveu: "... o jogo promove a capacidade de desenvolver artefatos com um critério consistente para requisitos de projeto, monitoramento e testes. Além disso, incentiva a interação entre os jogadores e trabalho em equipe, criando uma competição saudável ".

Outra experiência do uso, do recém criado SimulES-W, foi reportada em Meirelles (et al., 2011). Esse foi realizado com um grupo de cinco alunos de graduação em Ciência da Computação da UERJ. Todos apresentavam conhecimentos básicos e tinham interesse em aprender novos conceitos de engenharia de software. Esse estudo foi realizado no dia 10 de dezembro de 2010 e teve os seguintes objetivos: (i) analisar a interação dos alunos e do software no ambiente real, passando conceitos de engenharia de software e (ii) identificar pontos fortes e fracos do SimulES-W, através da observação dos alunos e dos relatos de suas experiências com o jogo. Como resultado, oportunidades de melhoria do SimulES-W foram identificadas: (i) apresentar um menu disponível em todas as telas do jogo, (ii) evitar barras de rolagens, (iii) as mensagens entre jogadores devem aparecer em tempo real, (iv) habilitar as abas somente no momento certo, (v) tornar a interface gráfica toda na língua nativa do local, (vi) explicar melhor as cartas de conceitos e (vii) permitir que o manual de ajuda seja consultado durante o jogo, e com isso, novas melhorias foram incorporadas ao jogo.

A efetividade e o benefício do jogo para a aprendizagem também foi testada. Em Monsalve, Werneck, e Leite (2013c), foi apresentado um relatório do jogo na perspectiva colaborativa e, também, uma avaliação do ponto de vista pedagógico. Neste trabalho, foi concluído que o jogo é adequado como ferramenta de apoio ao ensino de engenharia de software e que fornece um impacto positivo no desempenho acadêmico do alunos. Esta experiência também confirmou o que os outros autores confirmaram sobre como os jogos estabelecem um ambiente motivador e agradável para os alunos. Finalmente, este trabalho mostrou que o uso do SimulES-W não comprometeu a habilidade dos alunos na aquisição do conhecimento e pode ser uma forma de aprender desenvolvimento de software. Também foi concluído que no ensino algumas ferramentas podem confundir, ao invés de ajudá-los a melhorar o seu conhecimento, neste experimento, identificamos um resultado positivo, que SimulES-W melhorou o conhecimento geral da engenharia de software nos estudantes que participaram. No entanto, acredita-se que mais experiências devem ser desenvolvidas para obter uma conclusão mais firme.

Do mesmo modo, em Monsalve, Werneck e Leite (2013b) é apresentada a estratégia de retroalimentação como mecanismo de melhoria do processo de ensino na engenharia de software. Com isto em mente, neste trabalho foi apresentada uma nova evolução do jogo com base em uma experiência que usou o jogo em tabuleiro (SimulES) e o digital (SimulES-W) para achar novas evidências de melhoria e funcionalidades. Contudo, também foi possível concluir que o uso de jogos possibilita a retroalimentação feita pelos estudantes e ela serve de mecanismo de melhoria no processo de ensino, julgando assim o desempenho da atividade e dos conteúdos usados nesse processo. Consequentemente, avaliar a qualidade do ensino é difícil; no entanto, jogos podem ser usados pelos estudantes para expressar a eficácia e propor melhorias naquelas atividades que eles participam.

Outra experiência com foco pedagógico é a relatada em Monsalve, Pereira e Werneck (2013). O experimento foi realizado no curso de engenharia de software do Departamento de Ciência da Computação na *Universidade do Estado do Rio de Janeiro* (UERJ). O tópico a ser ensinado foi gestão de riscos em projetos de software e este experimento foi projetado para analisar como SimulES-W poderia influenciar os alunos a aprender sobre esses conceitos. Neste trabalho com SimulES-W, demonstrou-se a utilidade de jogos no ensino, como eles estimulam os alunos a explorar ideias, forçando-os a criar teorias e pensar em testes. Além disso, eles permitem que os alunos assumam papéis sociais e tarefas colaborativas. Essa última característica pôde ser sentida não somente pelos alunos, mas também pelo grupo todo de participantes no jogo, tais como professores, designers e desenvolvedores.

Por último, o trabalho apresentado em Monsalve, Werneck e Leite (2014) teve como objetivo apresentar a instanciação do conceito de transparência na pedagogia, conceito que será ampliado no próximo capítulo. No entanto, alguns aspectos dessa experiência serão descritos a seguir complementando os relatos de experiências onde SimulES-W foi usado. Portanto, a experiência foi focada na avaliação de diferentes aspectos, tais como motivação, engajamento, aquisição de conhecimento, expectativas dos estudantes. Para a elaboração do experimento, foram feitas melhoras no software, principalmente na interface e na navegação.

Como mecanismo de apoio na experiência relatada em Monsalve, Werneck e Leite (2014), os modelos do jogo foram utilizados, porém, foram revisados e em alguns a reengenharia foi aplicada. Como resultado, a Figura 4.7 apresenta o diagrama SDsituations do SimulES-W do jogo atual. No tempo T1,

são apresentadas as informações gerais do jogos e a Rodada de Início. Essas informações gerais estão presentes durante todo o jogo. Continuando, no tempo T2 é apresentada a Rodada de Ações e nele o jogador pode escolher entre: Construir, Inspecionar, Corrigir, Integrar Artefatos ou Submeter produto. E finalmente, há um tempo T3, onde é apresentada a Rodada de Conceitos. Nela o jogador faz Gestão do seus engenheiros, Submete problemas para os demais jogadores, descarta cartas e trata problemas, se os tiver. Acreditamos que a apresentação deste diagrama fornece clareza sobre a dinâmica do jogo.

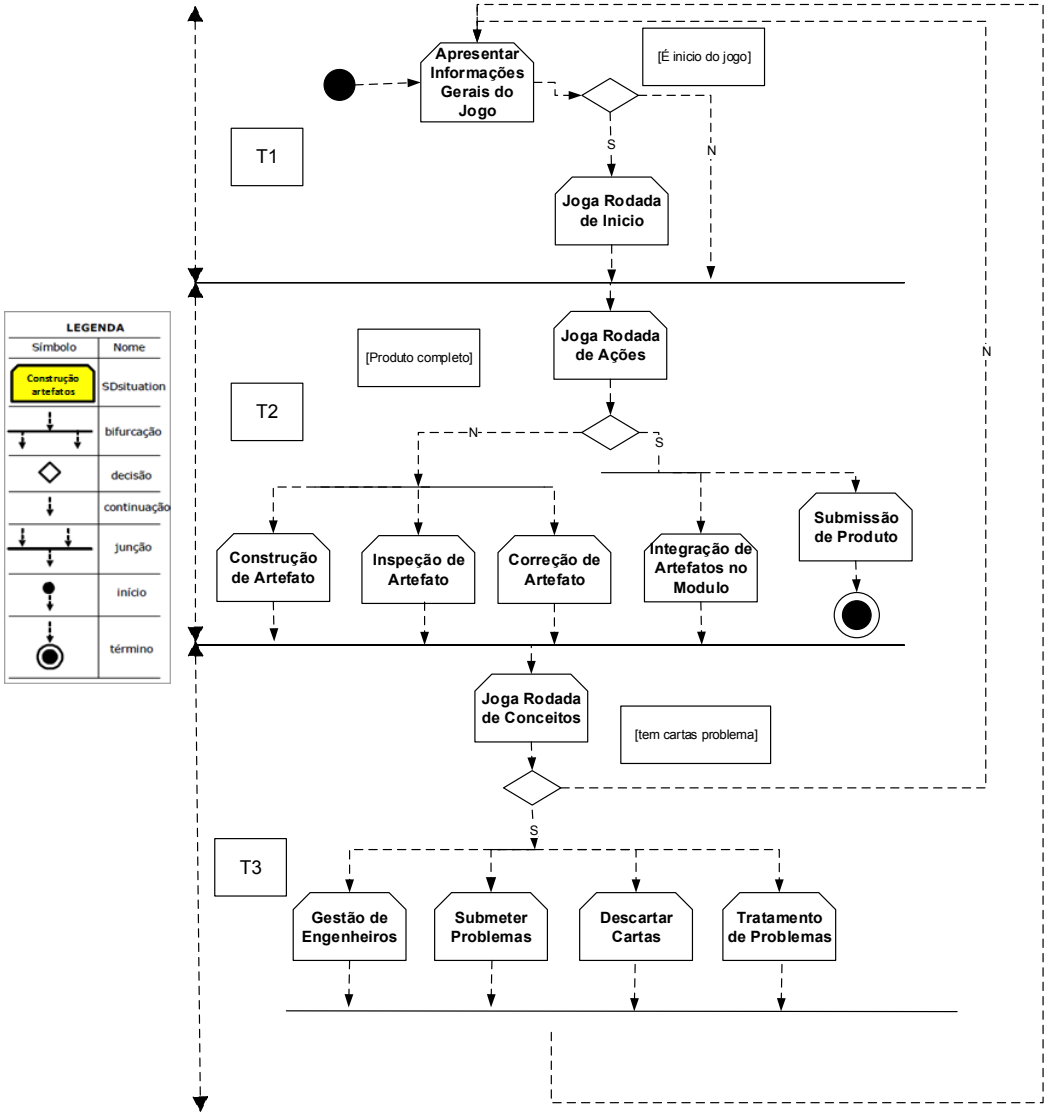


Figura 4.7 – SD Situations SimulES-W adaptado de Monsalve (2010)

Similarmente, a Figura 4.8 também apresenta a melhoria do diagrama de atores. Foi mudado o estereótipo de agente para ator. O jogador pode ser competidor, administrador ou jogador da vez. Finalmente, no Apêndice A, são

apresentados os modelos SD e SR do jogo, que representam cada uma das SDsituations da Figura 4.7. É importante lembrar que o *framework i** é, habitualmente, usado para modelar contextos organizacionais baseados nos relacionamentos entre os atores. Portanto, modelar em *i** tem permitido obter uma melhor compreensão dos relacionamentos dos jogadores e demais atores do jogo. A primeira versão desta modelagem aparece no trabalho de Napolitano (2009) e foi utilizada para representar o jogo do tabuleiro SimulES. Depois, uma nova versão foi apresentada em Monsalve (2010) e nela é representada a primeira versão do jogo web (SimulES-W). Finalmente, para este trabalho, uma nova versão é apresentada (Apêndice A) como parte da evolução do jogo, que também serve para apresentar a dinâmica do jogo aos estudantes e como mecanismo de explanação aos alunos que usaram SimulES-W. Esta última iniciativa será ampliada em capítulos posteriores.

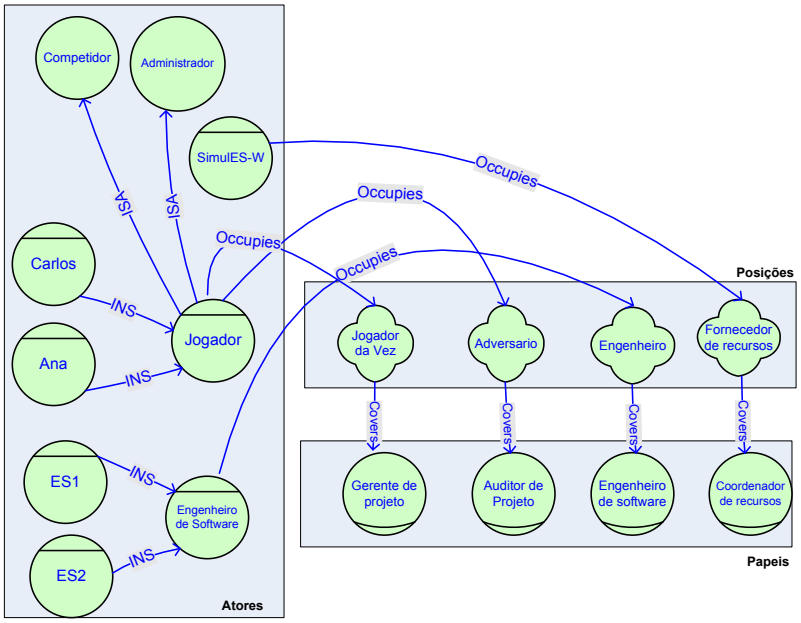


Figura 4.8 – Diagrama de Atores adaptado de Monsalve (2010)

4.4. Considerações Finais

A engenharia de software possui conceitos complexos de serem transmitidos se somente aulas teóricas são fornecidas para os estudantes. Para contornar este problema, esses conceitos teóricos podem ser apresentados através de experiências práticas, como é o uso de jogos para ensino. E neste contexto, SimulES-W é uma abordagem de aprendizagem baseada em jogos atrativa para ser usada em ambientes de ensino dessa disciplina.

A tendência de que os jogos sejam considerados como um ambiente de aprendizagem em determinadas áreas do conhecimento já é uma realidade. SimulES-W é uma resposta para os novos modelos de ensino da engenharia de software. Como apresentado, utilizamos SimulES-W como ferramenta de aprendizagem, auxiliando na simulação de um ambiente real. Identificamos como o aluno é estimulado a participar e resultados obtidos relacionados com o melhoramento do desempenho dos alunos são bastante positivos, além disso, as experiências estimularam geração de experiências individuais, de grupo e sociais, que ajudaram na formação dos alunos como profissionais.

É fato, conforme as experiências relatadas, o jogo SimulES-W tem por objetivo fornecer uma plataforma promissora para alcançar resultados eficazes de disseminação de conhecimento em engenharia de software.