4
Estudo de Caso

Este capítulo tem por objetivo apresentar a aplicação da estratégia de validação em um estudo de caso, visando avaliar os possíveis ganhos advindos da utilização da mesma. Inicialmente é apresentada uma introdução do caso e posteriormente o detalhamento do experimento realizado.

4.1.SimulES

O SimulES (Simulador de Uso da Engenharia de Software) é um jogo educacional de cartas que simula várias situações da engenharia de software e propõe uma forma de aprendizado mais interativa dos conceitos, das técnicas e dos processos de desenvolvimento de software.

Este jogo permite que um estudante assuma o papel de gerente de projeto de software e se depare com problemas práticos que não são bem cobertos em um ambiente de sala de aula [25].

O SimulES é uma evolução de um jogo educacional chamado PnP (*Problems and Programmers*) [27], desenvolvido no Departamento de Informática na Universidade da Califórnia. Durante o estudo do PnP e dos conceitos envolvidos no jogo, foram observadas algumas questões que poderiam ser evoluídas, dando origem ao SimulES.

Como o PnP, o objetivo do SimulES é que os jogadores disputem a para terminar um projeto de software e o vencedor será quem implantar o projeto primeiro. Dentro deste contexto, os jogadores, principalmente alunos, aprendem importantes conceitos em engenharia de software [26].

Os recursos disponibilizados pelo jogo são: cartões de projeto, tabuleiros, cartas (problemas, conceitos e engenheiro de software), artefatos e um dado. A seguir, esses recursos serão explicados resumidamente.

4.1.1.Cartão de Projetos

O cartão de projetos é um recurso que contém todas as informações relevantes do projeto que serão utilizadas pelos participantes. As informações do cartão são apresentadas a seguir:

- Descrição do projeto – texto que descreve as principais características do projeto.

- Complexidade – mostra quantos pontos de tempo um engenheiro de software deve gastar para produzir um artefato de qualidade.

- Tamanho – indica a quantidade de módulos a serem integrados para que o jogo termine.

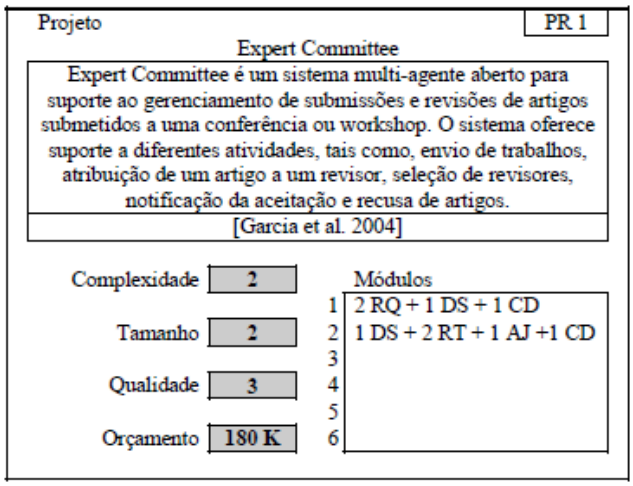


Figura 4.1 – Cartão de Projeto do SimulES [25].

- Qualidade – representa o número de módulos que devem estar livres de defeito ao término do jogo.

- Orçamento – quantidade de dinheiro que pode ser gasta durante o projeto.

4.1.2.Tabuleiros

O tabuleiro é a área onde o jogador coloca seus engenheiros de software e seus artefatos produzidos. Os artefatos podem ser do tipo: requisitos, desenho, código, rastro e ajuda. A Figura 4.2, retirada de [26] ilustra um tabuleiro individual do jogador:

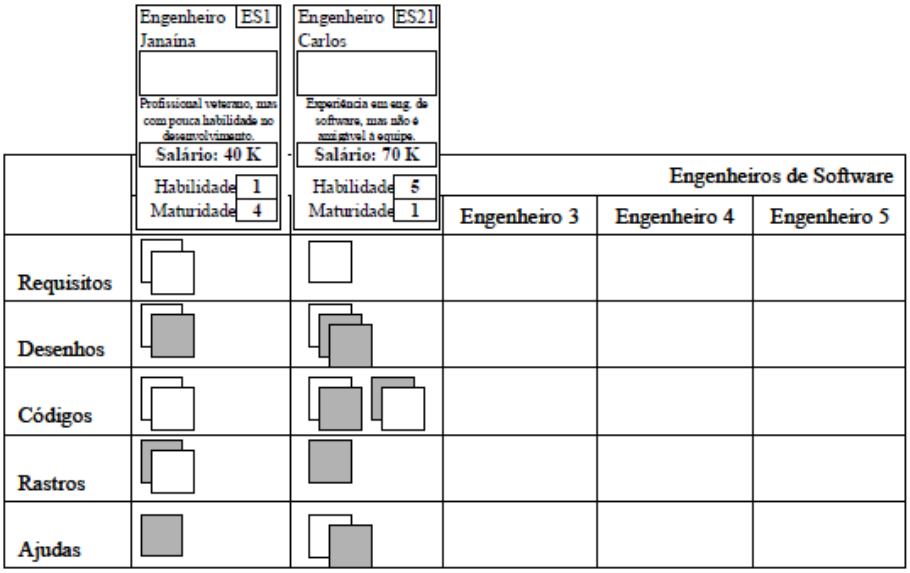


Figura 4.2 – Tabuleiro individual com engenheiros [26].

4.1.3.Cartas

As cartas do simules podem ser de três tipos: cartas de problemas, cartas de conceitos e cartas de engenheiros de software.

- Cartas de problemas – são cartas que descrevem os problemas clássicos da engenharia de software resultantes de falhas no processo de produção. Servem para criar obstáculos para os jogadores.

- Cartas de Conceitos – representas as boas práticas de engenharia de software. Podem ser utilizadas para contrapor problemas aplicados.

- Cartas de Engenheiros de Software – os engenheiros de software são responsáveis por construir, inspecionar e corrigir artefatos durante o jogo. Possuem habilidade, que são pontos de tempo e maturidade, que é a capacidade deles trabalharem em grupo.

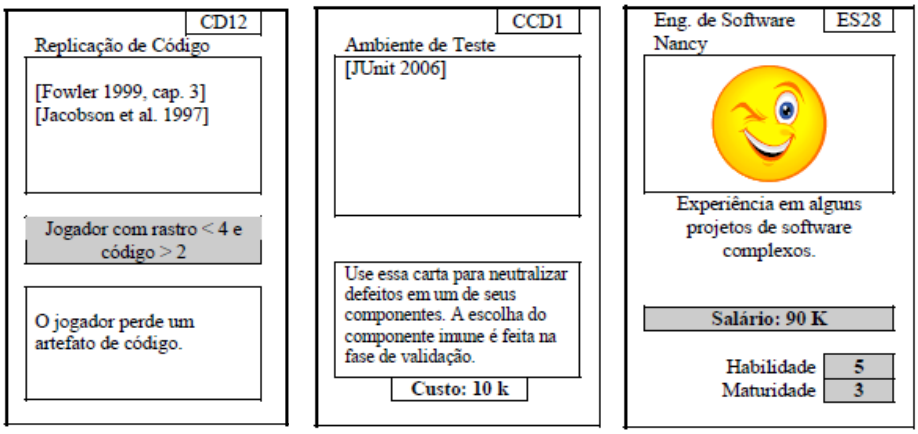


Figura 4.3 – Exemplos de cartas [25].

4.1.4.Artefatos

São os produtos que os engenheiros de software produzem durante o jogo. Podem ser de boa qualidade (branco) ou de má qualidade (cinza). Além disso, podem ou não possuir defeitos. A Figura a seguir ilustra um exemplo de artefato.

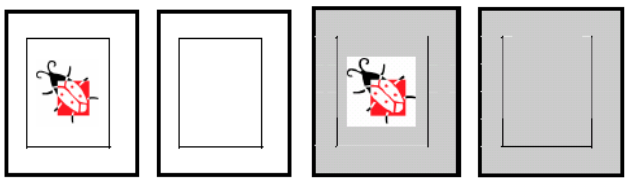


Figura 4.4 – Artefatos com e sem defeito [25].

4.2.Aplicação da Estratégia no Projeto SimulES

As seções a seguir apresentam todas as etapas para a construção dos modelos i* do SimulES, partindo do LEL, além da transformação dos cenários para entrada da ferramenta e a simulação com os interessados. O resultado da aplicação da estratégia é discutido no Capítulo 5.

4.2.1.Elicitar as Metas dos Atores

Visando facilitar o entendimento desta etapa, será fornecido o léxico dos símbolos envolvidos divididos por grupos (sujeito, objeto, verbo e estado), juntamente com suas metas elicitadas.

4.2.1.1.Preparar o LEL – Léxico Estendido da Linguagem

Nesta etapa o léxico foi construído utilizando a ferramenta C&L [18], baseado em uma série de documentos disponíveis, além das pessoas que tiveram contato com o jogo SimulES. Para construir o léxico, nós seguimos os seguintes passos, previstos em [13]:

1. Identificar os símbolos relevantes dentro de palavras ou frases peculiares e bastante usadas;
2. Classificar os símbolos (sujeito, objeto, verbo ou estado);
3. Descrever os símbolos através da noção e impacto;
4. Verificar o LEL através de inspeção;
5. Validar o LEL com os interessados pertencentes ao UdI.

4.2.1.2.Definir AGFL – Metas dos Agentes Vindas do Léxico

Passo 1: identificar atores

Os atores identificados pelo léxico foram: **jogador, jogador da vez, adversário, engenheiro de software e SimulES**. Jogador da vez e adversário são instâncias de jogador.

Passo 2: Definir as metas a partir do símbolo sujeito:

A Figura a seguir mostra a definição dos símbolos tipo sujeito do LEL:

Nome:	jogador da vez
Noção:	Papel de **jogador** que esta executando uma ação dentro de **jogo**
Classificação:	sujeito
Impacto(s):	1-joga rodada de início. 2-joga **rodada de ações**. 3-joga rodada de conceitos. 4-integra **artefatos** em um **módulo**. 5-submete produto. 6-aplica **problema**. 7-**contrata engenheiro de software**. 8-demite **engenheiro de software**.
Sinônimo(s):	

Nome:	adversario
Noção:	Papel de **jogador** que é instanciado se não é **jogador da vez**
Classificação:	sujeito
Impacto(s):	1-Recebe problemas através das **cartas de problemas** 2-Escolhe **módulo** 3-Inspeciona **módulo**
Sinônimo(s):	

Nome:	simules_
Noção:	É um **jogo** educacional que simula várias situações da Engenharia de Software.
Classificação:	sujeito
Impacto(s):	1-Fornece recursos para os **jogadores**
Sinônimo(s):	

Nome:	jogador
Noção:	Participante do **jogo simules**. Tem por objetivo vencer o **jogo**. Jogador pode asumir o papel de **jogador da vez** Jogador pode asumir o papel de **adversario**
Classificação:	sujeito
Impacto(s):	
Sinônimo(s):	jogadores.

Nome:	engenheiro de software
Noção:	**carta** do **jogo** que representa o profissional responsável por desempenhar ações. Possui **habilidade** e **maturidade**.
Classificação:	sujeito
Impacto(s):	1-Constrói **artefatos** 2-Inspeciona **artefatos** 3-Corrige **artefatos** 4-Integra **artefatos** em um modulo
Sinônimo(s):	engenheiros de software.

Figura 4.5 – LEL dos símbolos do tipo sujeito.

O *template* preenchido a seguir repete cada impacto dos símbolos e solicita a motivação envolvida na ação (resposta ao por quê).

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao porquê?	<sujeito / objeto LAL>	seja / esteja	<verbo>		<sujeito LAL>
JOGADOR DA VEZ					
-- joga rodada de início					
Porque **adversário** deseja que	jogo	seja	iniciado	por	**jogador da vez**
-- joga rodada de ações					
Porque **adversário** deseja que	rodada de ações	seja	jogada	Por	**Jogador da vez**
-- joga rodada de conceitos.					
Porque **adversário** deseja que	rodada de conceitos	seja	jogada	por	**Jogador da vez**
-- integra artefatos em um módulo.					
Porque **jogador da vez** deseja que	projeto	seja	concluído	por	**Jogador da vez**
-- submete produto.					
Porque **jogador da vez** deseja que	jogo	seja	vencido		
-- aplica problemas.					
Porque **jogador da vez** deseja que	adversário	seja	vencido		
-- contrata engenheiro de software.					
Porque **jogador da vez** deseja que	engenheiro de software	seja	contratado		
Porque **jogador da vez** deseja que	artefato	seja	construído	por	**Engenheiro de software**
Porque **jogador da vez** deseja que	artefato	seja	inspecionado	por	**Engenheiro de software**
Porque **jogador da vez** deseja que	artefato	seja	corrigido	por	**Engenheiro de software**
Porque **jogador da vez** deseja que	projeto	seja	concluído	por	**Jogador da vez**
-- demite engenheiro de software.					
Porque **jogador da vez** deseja que	engenheiro de software	seja	demitido		
Porque **jogador da vez** deseja que	engenheiro de software	seja	descartado		

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao porquê?	<sujeito / objeto LAL>	seja / esteja	<verbo>		<sujeito LAL>
ADVERSÁRIO					
-- Recebe problemas através das cartas de problemas.					
Porque **jogador da vez** deseja que	jogo	seja	vencido		
Porque **adversário** deseja que	problema	seja	tratado		
-- Escolhe Módulo.					
Porque **jogador da vez** deseja que	jogo	seja	vencido		
Porque **adversário** deseja que	módulo	seja	inspecionado		
-- Inspeciona Módulo.					
Porque **jogador da vez** deseja que	jogo	seja	vencido		

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao porquê?	<sujeito / objeto LAL>	seja / esteja	<verbo>		<sujeito LAL>
ENGENHEIRO DE SOFTWARE					
-- constrói artefatos.					
Porque **jogador da vez** deseja que	artefato	seja	construído	por	**engenheiro de software**
Porque **jogador da vez** deseja que	produto	seja	concluído	por	
Porque **engenheiro de software** deseja que	produto	seja	construído		
-- inspeciona artefatos.					
Porque **jogador da vez** deseja que	artefato	seja	inspecionado	por	**engenheiro de software**
Porque **jogador da vez** deseja que	artefato	seja	livre de defeitos		
Porque **engenheiro de software** deseja que	produto	seja	construído		
-- corrige artefatos.					
Porque **jogador da vez** deseja que	artefato	seja	corrigido	por	**engenheiro de software**
Porque **engenheiro de software** deseja que	produto	seja	construído		
-- integra artefatos em um módulo.					
Porque **jogador da vez** deseja que	módulo	seja	integrado	por	**engenheiro de software**
Porque **jogador da vez** deseja que	produto	seja	concluído	por	

TIPO: SUJEITO	<meta concreta>				ATOR
-- impacto resposta ao porquê?	<sujeito / objeto LAL>	seja / esteja	<verbo>		<sujeito LAL>
SIMULES					
-- fornece recursos para os jogadores.					
Porque **simules** deseja que	rodada de início	seja	iniciada		
Porque **simules** deseja que	rodada de ações	seja	iniciada		
Porque **simules** deseja que	rodada de conceitos	seja	iniciada		
Porque **simules** deseja que	recursos	sejam	disponibilizados		
Porque **simules** deseja que	artefatos	sejam	comprados		
Porque **simules** deseja que	cartas	sejam	compradas		

Figura 4.6 – *Template* preenchido com metas dos símbolos do tipo sujeito.

A Figura a seguir exibe o LEL dos símbolos tipo objeto, a fim de manter o entendimento do processo.

Nome:	artefato
Noção:	**cartas** que simbolizam os produtos que são produzidos pelos engenheiros de sofware e que podem ou não ter defeitos. Artefatos com defeito possuem um inseto. Podem ser de boa qualidade(**artefato branco**) ou de má qualidade(**artefato cinza**)
Classificação:	objeto
Impacto(s):	1-**engenheiro de software** constrói artefato 2-**engenheiro de software** inspeciona artefato 3-**engenheiro de software** corrige artefato
Sinônimo(s):	artefatos.

Nome:	carta
Noção:	Retângulo em papel que representa um **problema**, um conceito, um **artefato** ou um **engenheiro de software**.
Classificação:	objeto
Impacto(s):	1-**jogador da vez** compra cartas 2-**jogador da vez** descarta cartas 3-**jogador da vez** coloca carta no tabuleiro
Sinônimo(s):	cartas.

Nome:	carta de conceito
Noção:	**cartas** que representam boas práticas de Engenharia de Software. Possuem referências para pesquisas do conceito citado. Podem neutralizar **cartas de problemas**.
Classificação:	objeto
Impacto(s):	1-**jogador** compra **carta** de conceito. 2-**jogador** descarta **carta** de conceito. 3-**jogador** aplica conceito através da **carta** de conceito. 4-**jogador** coloca **carta** de conceito no tabuleiro.
Sinônimo(s):	cartas de conceito.

Nome:	carta de problema
Noção:	**cartas** que descrevem problemas clássicos de Engenharia de Software resultantes de faltas no processo de produção. Algumas **cartas** de problema usam a **maturidade** como condição de aplicação do problema. São utilizadas para criar obstáculos ao progresso dos adversários. Podem ser neutralizadas por **cartas de conceito**.
Classificação:	objeto
Impacto(s):	1-**jogador** compra **carta** de problema. 2-**jogador** descarta **carta** de problema. 3-**jogador** submete problema através de **carta** de problema. 4-**jogador** coloca **carta** de problema no tabuleiro.
Sinônimo(s):	cartas de problemas, cartas de problema, problema.

Nome:	cartão de projeto
Noção:	Retângulo em papel que informa detalhes sobre o projeto a ser desenvolvido no **jogo**. Um cartão de projeto contém uma descrição em linguagem natural com as principais características do projeto, a **complexidade do projeto**, o **tamanho do projeto**, a **qualidade do projeto** e o **orçamento do projeto**. Um cartão de projeto deve ser selecionado por um **jogador** para ser desenvolvido ao longo do **jogo**. O cartão de projeto selecionado deve ser colocado sobre a **mesa** de forma que todos os **jogadores** tenham acesso as suas informações. A escolha do cartão de projeto é um pré-requisito para dar início ao **jogo**.
Classificação:	objeto
Impacto(s):	1-**jogador** escolhe cartão de projeto que será considerado ao longo do **jogo**. 2-**adversario** concorda com a escolha do projeto. 3-**jogador** obedece as especificações do cartão de projeto.
Sinônimo(s):	cartão, cartões de projeto, cartões, projeto.

Nome:	complexidade do projeto
Noção:	Indica quantos **pontos de tempo** um **engenheiro de software** precisa gastar para completar um **artefato** de boa qualidade. **artefatos** de má qualidade custam metade deste valor. Possui valor 2 ou 4.
Classificação:	objeto
Impacto(s):	1-**jogador** produz **artefatos** de acordo com as informações de complexidade do **cartão de projeto**.
Sinônimo(s):	complexidade.

Nome:	dado
Noção:	Cubo utilizado para sortear aleatoriamente números entre 1 e 6.
Classificação:	objeto
Impacto(s):	1-**jogador** joga o dado. 2-**jogador** obtêm **resultado do lançamento do dado**.
Sinônimo(s):	dados.

Nome:	habilidade
Noção:	Número que indica o quanto o **engenheiro de software** possui para, a cada rodada, desempenhar ações no **jogo**. Número que limita as ações a serem desempenhadas pelo **engenheiro de software**. Número inteiro no intervalo [1,5].
Classificação:	objeto
Impacto(s):	1-**engenheiro de software** usa habilidade para **construir artefato**. 2-**engenheiro de software** usa habilidade para **corrigir artefato**. 3-**engenheiro de software** usa habilidade para **inspecionar artefato**.
Sinônimo(s):	pontos de tempo, ponto de tempo.

Nome:	informações do projeto
Noção:	Especificações de um **projeto** apresentadas no **cartão** do **projeto** e compostas de descrição, **complexidade**, **tamanho**, qualidade e **orçamento**. As informações do **projeto** devem ser colocadas sobre a **mesa**. As informações do **projeto** ficam sobre a **mesa** ao longo do **jogo**. As informações do **projeto** devem ser seguidas pelos **jogadores**. Todos os **jogadores** podem ter acesso as essas informações a qualquer momento do **jogo**.
Classificação:	objeto
Impacto(s):	**jogador da vez** joga de acordo com as informações do **projeto**. **jogador da vez** usa estratégias de acordo com as informações do **projeto**. **jogador da vez** precisa das informações do **projeto** para ganhar o **jogo**.
Sinônimo(s):	informações dos projetos.

Nome:	jogo
Noção:	Ferramenta lúdico-pedagógica desenvolvida para facilitar o ensino da Engenharia de Software. Possui de 1 a 8 **jogadores**. Possui **cartas**. Possui tabuleiros. Possui **cartões de projeto**. Possui início. Possui rodadas.
Classificação:	objeto
Impacto(s):	1-**jogador** joga o jogo.
Sinônimo(s):	partida.

Nome:	maturidade
Noção:	Número inteiro entre [1,5]. Quanto maior o número, maior a facilidade do engenheiro em trabalhar com os outros. Indica um engenheiro com personalidade que facilita o trabalho em conjunto.
Classificação:	objeto
Impacto(s):	**cartas** de **problema** podem usar a maturidade como condição para aplicação do **problema**.
Sinônimo(s):	

Nome:	mesa
Noção:	Define o estado do **jogo**. Lugar onde são colocados os tabuleiros dos **jogadores**.
Classificação:	objeto
Impacto(s):	**jogadores** usam mesa.
Sinônimo(s):	mesa de jogo.

Nome:	módulo
Noção:	Parte que compõe um **projeto**. Pode ser formado por **artefatos** de requisitos(RQ), desenho (DS), código (CD), rastro (RT) ou ajuda (AJ).
Classificação:	objeto
Impacto(s):	**jogador da vez** integra **artefatos** em modulos.
Sinônimo(s):	módulo do projeto, módulos, módulos do projeto.

Nome:	monte
Noção:	Conjunto de **cartas** ou **cartões** disponíveis para seleção, compra ou descarte por parte dos **jogadores**. **jogadores** podem comprar **cartas** dos montes de **cartas**. **jogadores** podem **descartar cartas** nos montes de **cartas**. **jogadores** podem selecionar um **cartão** do monte de **cartões**. Montes devem ficar sobre a **mesa**. Montes devem ser acessíveis por todos os **jogadores**.
Classificação:	objeto
Impacto(s):	**jogadores** colocam os montes sobre a **mesa de jogo**. **jogador** compra **carta** do monte de **cartas**. **jogador** descarta **carta** no monte de **cartas**. **jogador** seleciona um **projeto** do monte de **cartões** de projetos.
Sinônimo(s):	monte de cartas de engenheiros de software, montes, monte de cartas, monte de artefatos, monte de cartas engenheiros.

Nome:	orçamento do projeto
Noção:	Quantidade de dinheiro disponível para gastar com o **projeto**. Funciona como restrição para **contratação de engenheiro de software** bem como para uso de **cartas de conceito**.
Classificação:	objeto
Impacto(s):	**jogador** usa orçamento do **projeto**
Sinônimo(s):	orçamento.

Nome:	qualidade do projeto
Noção:	Representa o quão livre de defeitos deve estar o produto final. O valor varia de 1 a 5 indicando o número mínimo de **módulos** sem defeitos necessários para vencer o **jogo**.
Classificação:	objeto
Impacto(s):	**jogador** usa qualidade do **projeto**. **jogador** deve possuir um número mínimo de **módulos** sem defeito.
Sinônimo(s):	

Nome:	resultado do lançamento do dado
Noção:	Número no intervalo [1,6] resultante do lançamento do **dado**.
Classificação:	objeto
Impacto(s):	resultado do lançamento do **dado** determina quantas **cartas** serão compradas. resultado do lançamento do **dado** determina o tipo de **carta** que será comprada.
Sinônimo(s):	

Nome:	sentido de jogo
Noção:	Forma pela qual os **jogadores** se sucedem ao longo do **jogo**. O sentido de **jogo** no **simules** é horário. O sentido de **jogo** não pode ser desrespeitado.
Classificação:	objeto
Impacto(s):	**jogadores** obedecem sentido de **jogo**.
Sinônimo(s):	

Nome:	tabuleiro central
Noção:	Área central da **mesa** definida por um papel impresso onde as **cartas** (problemas, conceitos, **artefatos**, **engenheiros de software** e **cartão de projeto**)são disponibilizadas no **jogo**. Um tabuleiro é colocado na **mesa de jogo** para cada **jogador**.
Classificação:	objeto
Impacto(s):	Tabuleiro central orienta as rodadas do **jogo**.
Sinônimo(s):	

Nome:	tabuleiro individual
Noção:	Área na **mesa** definida por um papel impresso onde as **cartas** (problemas, conceitos, **artefatos** e **engenheiros de software**)de um determinado **jogador** são colocadas em **jogo**. **jogador** coloca suas **cartas** no tabuleiro. Tabuleiro pertence a um **jogador**. Um tabuleiro é colocado na **mesa de jogo** para cada **jogador**.
Classificação:	objeto
Impacto(s):	**jogador** coloca **artefatos** no tabuleiro. **jogador** coloca **cartas** no tabuleiro. **jogador** retira **artefatos** do tabuleiro. **jogador** retira **cartas** do tabuleiro. **jogador** desvira **artefatos** do tabuleiro.
Sinônimo(s):	

Nome:	tamanho do projeto
Noção:	Indica quantos **módulos** integrados devem ser completados para empacotar o produto e vencer o **jogo**. Um **projeto** pode ter tamanho máximo 6.
Classificação:	objeto
Impacto(s):	**jogador** usa informações de tamanho do **projeto**.
Sinônimo(s):	tamanho.

Figura 4.7 – LEL dos símbolos do tipo objeto.

O *template* a seguir exibe as metas dos símbolos do tipo objeto. Pode se perceber a diferença comparado ao *template* usado nos símbolos do tipo objeto, pois segundo [2], como os símbolos do tipo objeto sofrem ações, sua intencionalidade necessita ter um formato diferente do anterior.

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
ARTEFATO					
-- Engenheiro de software constrói artefato.					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	empacotado	por	**Jogador da vez**
-- Engenheiro de software inspeciona artefato.					
Para que	artefato	seja	livre de defeitos		
Para que	produto	seja	construído	por	**Engenheiro de Software**
-- Engenheiro de software corrige artefato.					
Para que	artefato	seja	corrigido	por	**Engenheiro de software**
Para que	produto	seja	construído	por	**Engenheiro de Software**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CARTA					
-- jogador da vez compra cartas.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**
Para que	problema	seja	aplicado	por	**Jogador da vez**
Para que	engenheiro de software	seja	contratado	por	**Jogador da vez**
-- jogador da vez descarta cartas.					
Para que	conceito	seja	Descartado	por	**Jogador da vez**
Para que	problema	seja	Descartado	por	**Jogador da vez**
Para que	engenheiro de software	seja	demitido	por	**Jogador da vez**
-- jogador da vez coloca carta no tabuleiro.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**
Para que	problema	seja	aplicado	por	**Jogador da vez**
Para que	engenheiro de software	seja	contratado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CARTA DE CONCEITO					
-- Jogador da vez compra carta de conceito.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**
-- Jogador da vez descarta carta de conceito.					
Para que	conceito	seja	descartado	por	**Jogador da vez**
-- Jogador da vez aplica conceito através de carta de conceito.					
Para que	problema	seja	tratado	por	**Jogador da vez**
Para que	jogo	seja	vencido	por	**Jogador da vez**
-- Jogador vez coloca carta de conceito no tabuleiro.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CARTA DE PROBLEMA					
-- Jogador da vez compra carta de problema.					
Para que	problema	seja	aplicado	por	**Jogador da vez**
-- Jogador da vez descarta carta de problema.					
Para que	problema	seja	descartado	por	**Jogador da vez**
-- Jogador submete problema através de carta de problema.					
Para que	problema	seja	aplicado	por	**Jogador da vez**
-- jogador coloca carta de problema no tabuleiro.					
Para que	problema	seja	tratado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
CARTÃO DE PROJETO					
-- jogador escolhe cartão de projeto que será considerado ao longo do jogo.					
Para que	jogo	seja	iniciado	por	**Jogador da vez**
-- adversário concorda com escolha do projeto.					
Para que	jogo	seja	iniciado	por	**Jogador da vez**
-- jogador obedece as especificações do cartão de projeto.					
Para que	produto	seja	empacotado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
COMPLEXIDADE DO PROJETO					
-- jogador produz artefatos de acordo com as informações de complexidade do cartão de projeto					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	concluído	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
DADO					
-- jogador lança o dado.					
Para que	resultado do lançamento	seja	obtido	por	**simules**
Para que	jogo	seja	iniciado	por	**Jogador da vez**
Para que	cartas	sejam	compradas	por	**Jogador da vez**
-- jogador obtém resultado do lançamento do dado.					
Para que	cartas	sejam	compradas	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
HABILIDADE					
-- Engenheiro de Software usa habilidade para construir artefato.					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	empacotado	por	**Jogador da vez**
-- Engenheiro de Software usa habilidade para inspecionar artefato.					
Para que	artefato	seja	livre de defeitos		
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	artefatos	sejam	inspecionados	por	**Engenheiro de software**
-- Engenheiro de Software usa habilidade para corrigir artefato.					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	artefatos	sejam	corrigidos	por	**Engenheiro de software**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
INFORMAÇÕES DO PROJETO					
-- Jogador da vez joga de acordo com informações do projeto.					
Para que	produto	seja	empacotado	por	**Jogador da vez**
-- Jogador da vez usa estratégias de acordo com as informações do projeto.					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	empacotado	por	**Jogador da vez**
Para que	jogo	seja	vencido	por	**Jogador da vez**
-- jogador vez precisa das informações do projeto para ganhar o jogo.					
Para que	jogo	seja	vencido	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
JOGO					
-- Jogador joga o jogo.					
Para que	jogo	seja	vencido	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
MATURIDADE					
-- cartas de problema podem usar maturidade como condição para aplicação do problema.					
Para que	problema	seja	aplicado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
MESA					
-- jogadores usam mesa.					
Para que	recurso	seja	disponibilizado	por	**simules**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
MÓDULO					
-- Jogador da vez integra artefatos em módulos.					
Para que	produto	seja	empacotado	por	**Jogador da vez**
Para que	modulo	seja	escolhido	por	**adversário**
Para que	modulo	seja	inspecionado	por	**adversário**
Para que	jogo	seja	vencido	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
MONTE					
-- Jogador coloca os montes sobre a mesa de jogo.					
Para que	recurso	seja	disponibilizado	por	**simules**
Para que	jogo	seja	iniciado	por	**Jogador da vez**
-- Jogador compra carta do monte de cartas.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**
Para que	problema	seja	aplicado	por	**Jogador da vez**
Para que	engenheiro de software	seja	contratado	por	**Jogador da vez**
-- Jogador descarta carta no monte de cartas.					
Para que	conceito	seja	Descartado	por	**Jogador da vez**
Para que	problema	seja	Descartado	por	**Jogador da vez**
Para que	engenheiro de software	seja	demitido	por	**Jogador da vez**
-- Jogador seleciona um projeto do monte de cartão de projetos.					
Para que	jogo	seja	iniciado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
ORÇAMENTO DO PROJETO					
-- Jogador usa orçamento do projeto					
Para que	Engenheiro de software	seja	contratado	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
QUALIDADE DO PROJETO					
-- Jogador usa qualidade do projeto					
Para que	artefatos	sejam	integrados	por	**Jogador da vez**
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	empacotado	por	**Jogador da vez**
Para que	jogo	seja	vencido	por	**Jogador da vez**
-- jogador deve possuir um número mínimo de módulos sem defeito					
Para que	jogo	seja	vencido	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
RESULTADO DO LANÇAMENTO DO DADO					
-- resultado do lançamento do dado determina quantas cartas serão compradas.					
Para que	cartas	sejam	compradas	por	**Jogador da vez**
-- resultado do lançamento do dado determina o tipo de carta que será comprada.					
Para que	cartas	sejam	compradas	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
SENTIDO DE JOGO					
-- Jogadores obedecem sentido de jogo.					
Para que	rodada de ações	seja	jogada	por	**Jogador da vez**
Para que	rodada de conceitos	seja	jogada	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
TABULEIRO CENTRAL					
-- Tabuleiro central orienta as rodadas do jogo.					
Para que	rodada de ações	seja	jogada	por	**Jogador da vez**
Para que	rodada de conceitos	seja	jogada	por	**Jogador da vez**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
TABULEIRO INDIVIDUAL					
-- Jogador coloca artefatos no tabuleiro individual.					
Para que	jogo	seja	disponibilizado	Por	**adversário**
Para que	artefatos	sejam	construídos	por	**Jogador da vez**
-- Jogador coloca cartas no tabuleiro individual.					
Para que	conceito	seja	aplicado	por	**Jogador da vez**
Para que	problema	seja	aplicado	por	**Jogador da vez**
-- Jogador retira artefatos do tabuleiro individual.					
Para que	artefatos	sejam	destruídos	por	**Engenheiro de software**
-- Jogador retira cartas do tabuleiro individual.					
Para que	cartas	sejam	descartadas	por	**Jogador da vez**
-- Jogador desvira artefatos do tabuleiro.					
Para que	artefatos	sejam	inspecionados	por	**Engenheiro de software**

TIPO: OBJETO	<meta>				ATOR
-- impacto resposta ao por quê?	<sujeito / objeto LAL>	seja/ esteja	<verbo>		<sujeito LAL>
TAMANHO DO PROJETO					
-- Jogador usa informações de tamanho do projeto.					
Para que	produto	seja	construído	por	**Engenheiro de Software**
Para que	produto	seja	empacotado	por	**Jogador da vez**

Figura 4.8 – *Template* com metas concretas dos símbolos do tipo objeto.

	<meta flexível>		
-- impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
-- Engenheiro de software inspeciona artefato	ação flexível		
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
-- Engenheiro de software corrige artefato	ação flexível		
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software
-- jogador usa qualidade do projeto	ação flexível		
Porque	qualidade	[projeto]	produto seja concluído jogador da vez

Figura 4.9 – *Template* com metas flexíveis dos símbolos do tipo objeto.

O melhor momento para definir metas flexíveis é após a definição de metas concretas. A partir de agora, os *templates* utilizados serão os de metas flexíveis. A Figura abaixo mostra o LEL dos símbolos tipo verbo.

Nome:	aplicar conceito
Noção:	Procedimento ou execução de ação do **jogador da vez** para contrapor outra ação que perjudica seu **jogo**, ou para melhorar seu **jogo**.
Classificação:	verbo
Impacto(s):	1- **jogador da vez** neutraliza **problema**
Sinônimo(s):	

Nome:	aplicar problema
Noção:	**jogador da vez** pode escolher até 3 adversários e submeter a cada um deles uma **carta de problema**.
Classificação:	verbo
Impacto(s):	Adversário recebe **problema**. **jogador da vez** trata **problema**.
Sinônimo(s):	aplicar carta de problema.

Nome:	comprar carta
Noção:	Pegar **carta** de um **monte** (**engenheiro de software**, problemas e conceitos) e mantê-la nas mãos. As **cartas** são compradas pelo **jogador da vez**. As **cartas** são compradas de acordo com o **resultado do lançamento do dado**.
Classificação:	verbo
Impacto(s):	**jogador da vez** adquire **carta** de conceitos. **jogador da vez** adquire **carta** de problemas. **jogador da vez** adquire **engenheiro de software**. **jogador da vez** adquire **artefato**.
Sinônimo(s):	

Nome:	construir artefato
Noção:	**jogador da vez** através de seu(s) engenheiro(s) de software coloca em seu tabuleiro as **cartas** de **artefatos** (brancas ou cinzas), respeitando os pontos de **habilidade** do engenheiro e **complexidade do projeto**. o jogado da vez pode escolher construir **artefatos** de requisitos, desenho, código, ajuda e rastro. **artefato** é construído por **engenheiro de software**. **artefatos** construídos podem ser de qualidade (branco) ou sem qualidade (cinza).
Classificação:	verbo
Impacto(s):	**artefato** é colocado no **tabuleiro individual** do **jogador da vez** de acordo com seu tipo e **engenheiro de software** que o produziu.
Sinônimo(s):	construir artefatos.

Nome:	contratar engenheiro de software
Noção:	**jogador da vez** seleciona o **engenheiro de software** em suas mãos que que deseja contratar, colocando-o em seu **tabuleiro individual**. No início do **jogo** é obrigatório cada **jogador** contratar um **engenheiro de software**. **engenheiro de software contratado** só poderá produzir **artefatos** no próximo turno.
Classificação:	verbo
Impacto(s):	**engenheiro de software** pode **construir artefato**. **engenheiro de software** pode **inspecionar artefato**. **engenheiro de software** pode **corrigir artefato**. **jogador da vez** pode **demitir engenheiro de software**.
Sinônimo(s):	contrata engenheiro de software, contratação de engenheiro de software.

Nome:	corrigir artefato
Noção:	O **artefato** a ser corrigido é substituído por um novo **artefato** da mesma cor não inspecionado.
Classificação:	verbo
Impacto(s):	**jogador da vez** possui um novo **artefato** em seu tabuleiro.
Sinônimo(s):	corrigir artefatos.

Nome:	demitir engenheiro de software
Noção:	**jogador da vez** retira **carta** de engenheiro de seu **tabuleiro individual** e a retorna para a última posição do **monte de cartas** de **engenheiro de software**.
Classificação:	verbo
Impacto(s):	**jogador da vez** pode usar **orçamento** novamente para a contratação de um novo engenheiro
Sinônimo(s):	

Nome:	descartar carta
Noção:	**jogador da vez** retorna a(s) **carta**(s) de suas mãos a posição final dos respectivos **montes** situados no **tabuleiro central**.
Classificação:	verbo
Impacto(s):	**cartas** retornam para os **montes**.
Sinônimo(s):	descartar cartas.

Nome:	empacotar produto
Noção:	produzir todos os **módulos** necessários de acordo com o **cartão de projeto**.
Classificação:	verbo
Impacto(s):	Adversários escolhem **módulos** de acordo com a **qualidade do projeto**. Adversários inspecionam **módulos** selecionados. Se **módulos** verificados não apresentarem defeitos, **jogador** vence o **jogo**.
Sinônimo(s):	

Nome:	jogar dado
Noção:	arremeçar o **dado** no **tabuleiro central** para iniciar rodada.
Classificação:	verbo
Impacto(s):	**jogador da vez** obtem **resultado do lançamento do dado**. **jogador da vez** compra **cartas**.
Sinônimo(s):	lançar dado.

Nome:	receber problema
Noção:	receber **carta de problema** submetida pelo **jogador da vez** durante a rodada de conceitos
Classificação:	verbo
Impacto(s):	**jogador da vez** trata **problema**.
Sinônimo(s):	receber carta de problema.

Nome:	inspecionar artefato
Noção:	Desvirar os **artefatos** selecionados para verificar a existência de defeito. o custo de inspecionar o **artefato** é metade do custo de construir um **artefato**. **artefato** é inspecionado por **engenheiro de software**.
Classificação:	verbo
Impacto(s):	**jogador da vez** conhece o **artefato inspecionado**. **jogador da vez** pode **corrigir artefato**, caso ele seja defeituoso.
Sinônimo(s):	

Nome:	jogar rodada de ações
Noção:	Nesta rodada o **jogador** pode adquirir **cartas** de conceitos, **cartas de problemas**, e/ou **cartas** de **engenheiros de software**, de acordo com o **resultado do lançamento do dado**. O **jogador** também poderá **construir artefatos**, inspecionar **artefatos** e/ou **corrigir artefatos**.
Classificação:	verbo
Impacto(s):	SimulES inicia rodada de ações **jogador da vez** joga rodada de ações **jogador da vez** obtém **resultado do lançamento do dado**
Sinônimo(s):	rodada de ações.

Nome:	jogar rodada de conceitos
Noção:	Nesta rodada o **jogador** descarta **cartas** excedentes, contrata e/ou demite **engenheiros de software**, submete problemas e trata problemas.
Classificação:	verbo
Impacto(s):	SimulES inicia rodada de conceitos Se **jogador da vez** puder **empacotar produto**, **jogador da vez** submete produto.
Sinônimo(s):	

Nome:	receber problema
Noção:	receber **carta de problema** submetida pelo **jogador da vez** durante a rodada de conceitos
Classificação:	verbo
Impacto(s):	**adversario** trata **problema**.
Sinônimo(s):	receber carta de problema.

Nome:	submeter produto
Noção:	mostrar o conteudo de alguns ou todos os modulos construidos, de acordo com a **qualidade do projeto**.
Classificação:	verbo
Impacto(s):	se **módulos** inspecionados não apresentarem defeitos, **jogador** vence o **jogo**.
Sinônimo(s):	

Nome:	tratar problema
Noção:	Procedimento ou execução de ação do **jogador da vez** para contrarestar outra ação que perjudica seu **jogo**
Classificação:	verbo
Impacto(s):	**adversario** pode contrapor **problema** aplicado. **adversario** atende a demanda do **problema**.
Sinônimo(s):	

Figura 4.10 – LEL dos símbolos do tipo verbo.

A seguir, serão exibidas as metas (concretas e flexíveis) elicitadas dos símbolos do tipo verbo.

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
APLICAR CONCEITO			
-- jogador da vez neutraliza problema	ação concreta		Conceito seja aplicado Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
APLICAR PROBLEMA			
-- adversário recebe problema	ação concreta		Problema seja aplicado Jogador da vez
-- adversário trata problema	ação concreta		Problema seja tratado adversário

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
COMPRAR CARTA			
-- jogador da vez adquire carta de conceitos	ação concreta		Rodada de ações seja jogada Jogador da vez
-- jogador da vez adquire carta de problemas	ação concreta		Rodada de ações seja jogada Jogador da vez
-- jogador da vez adquire engenheiro de software	ação concreta		Rodada de ações seja jogada Jogador da vez
-- jogador da vez adquire artefato	ação concreta		Rodada de ações seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
CONSTRUIR ARTEFATO			
-- artefato é colocado no tabuleiro individual do jogador da vez de acordo com seu tipo e engenheiro de software que o produziu	ação concreta		Rodada de ações seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
CONTRATAR ENGENHEIRO DE SOFTWARE			
-- Engenheiro de software pode construir artefato.	ação concreta		Produto seja construído Engenheiro de Software
-- Engenheiro de software pode inspecionar artefato.	ação concreta		Rodada de ações seja jogada Jogador da vez
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
-- Engenheiro de software pode corrigir artefato.	ação concreta		Rodada de ações seja jogada Jogador da vez
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software
-- jogador da vez pode demitir engenheiro de software.	ação concreta		Engenheiro de software seja demitido Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
CORRIGIR ARTEFATO			
-- jogador da vez possui um novo artefato em seu tabuleiro.	ação concreta		Artefato seja corrigido Engenheiro de Software
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
DEMITIR ENGENHEIRO DE SOFTWARE			
-- Jogador da vez pode usar orçamento novamente para a contratação de um novo engenheiro	ação concreta		Engenheiro de software seja demitido Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
DESCARTAR CARTA			
-- cartas retornam para os montes	ação concreta		

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
EMPACOTAR PRODUTO			
-- adversários escolhem módulos de acordo com a qualidade do projeto.	ação concreta		Produto seja empacotado Jogador da vez
-- adversários inspecionam módulos selecionados	ação concreta		produto seja concluído
-- se módulos verificados não apresentarem defeitos, jogador vence o jogo	ação concreta		jogo seja vencido

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ESCOLHER PROJETO			
-- Jogador da vez usa informação do cartão de projeto.	ação concreta		Projeto seja escolhido Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
INSPECIONAR ARTEFATO			
-- jogador da vez conhece o artefato inspecionado			
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
-- jogador da vez pode corrigir artefato caso seja defeituoso			
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
JOGAR DADO			
-- jogador da vez obtém resultado do lançamento do dado	ação concreta		Jogo seja iniciado
-- jogador da vez compra cartas	ação concreta		Rodada de ações seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
JOGAR RODADA DE AÇÕES			
-- simules inicia rodada de ações	ação concreta		Rodada de ações seja iniciada por SimulES
-- jogador da vez joga rodada de ações	ação concreta		Projeto seja concluído Jogador da vez
-- jogador da vez obtém resultado do lançamento do dado	ação concreta		Rodada de conceitos seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
JOGAR RODADA DE CONCEITOS			
-- simules inicia rodada de conceitos	ação concreta		Rodada de conceitos seja iniciada por SimulES
-- Se jogador da vez puder empacotar produto, jogador da vez submete produto.	ação concreta		produto seja concluído Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
RECEBER PROBLEMA			
-- Adversário trata problema	ação concreta		Problema seja tratado Adversário

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
SUBMETER PRODUTO			
-- se módulos inspecionados não apresentarem defeitos, jogador vence o jogo	ação concreta		Jogo seja vencido Jogador da vez
Porque	qualidade	[artefato]	Rodada de ações seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
TRATAR PROBLEMA			
-- Adversário pode contrapor problema aplicado	ação concreta		Problema seja tratado Adversário
-- Adversário atende demanda do problema	ação concreta		Problema seja tratado Adversário

Figura 4.11 – *Template* com metas dos símbolos do tipo verbo.

Por fim, o mesmo será feito com os símbolos do tipo estado:

Nome:	artefato construído
Noção:	1-É um **artefato** de software. 2-Pode ser de requisitos, desenho, código, rastro ou ajuda. 3-Pode ser branco ou cinza. 4-Pode ter defeito ou não.
Classificação:	estado
Impacto(s):	1- **engenheiro de software** pode **inspecionar artefato**. 2- **engenheiro de software** pode **corrigir artefato**.
Sinônimo(s):	

Nome:	artefato corrigido
Noção:	1-É um **artefato**. 2-Pode ter defeito ou não. 3-permanece com a mesma cor do **artefato inspecionado** que apresentou defeito.
Classificação:	estado
Impacto(s):	1-**engenheiro de software** pode inspecionar o **artefato** corrigido.
Sinônimo(s):	

Nome:	artefato defeituoso
Noção:	1-É um **artefato** 2-Pode ser branco ou cinza 3-Possui um inseto simbolizando um defeito.
Classificação:	estado
Impacto(s):	1-**engenheiro de software** pode **corrigir artefato**.
Sinônimo(s):	

Nome:	artefato inspecionado
Noção:	1-É um **artefato** 2-Pode possuir defeito ou não.
Classificação:	estado
Impacto(s):	1-avalia a qualidade do **artefato**.
Sinônimo(s):	

Nome:	artefato livre de defeito
Noção:	1-É um **artefato** 2-Pode ser branco ou cinza 3-Possui um OK representando um **artefato** sem defeitos.
Classificação:	estado
Impacto(s):	1-**jogador da vez** pode construir mais **artefatos**.
Sinônimo(s):	artefato livre de defeitos.

Nome:	engenheiro de software contratado
Noção:	1-É um **engenheiro de software** 2-engenheiro contratado é colocado no **tabuleiro individual** do **jogador**
Classificação:	estado
Impacto(s):	1-**engenheiro de software** pode **construir artefato**. 2-**engenheiro de software** pode **inspecionar artefato**. 3-**engenheiro de software** pode **corrigir artefato**.
Sinônimo(s):	

Nome:	engenheiro de software demitido
Noção:	1-É um **engenheiro de software** 2-Engenheiro demitido retorna para o **monte** de engenheiros.
Classificação:	estado
Impacto(s):	1-**jogador da vez** pode contratar outro **engenheiro de software**.
Sinônimo(s):	

Nome:	problema persistente
Noção:	1-O **problema** acompanha o **jogador** por todo o **jogo**, a não ser se for contraposto por uma **carta de conceito**. 2-Deve ser colocado ao lado do **tabuleiro individual** do **jogador**.
Classificação:	estado
Impacto(s):	1-**adversario** trata **problema**. 2-Pode ser contraposto por **carta de conceito**.
Sinônimo(s):	

Nome:	problema temporário
Noção:	1-Atrapalha o **jogo** do **jogador da vez** 2-O adversário sofre a penalidade no momento de sua aplicação. 3-Após ser aplicado, o **problema** retorna para o **monte** de problemas e conceitos.
Classificação:	estado
Impacto(s):	1-Adversário trata **problema**
Sinônimo(s):	

Figura 4.12 – LEL dos símbolos do tipo estado.

Em sequência, as metas elicitadas:

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ARTEFATO CONSTRUÍDO			
-- Engenheiro de software pode inspecionar artefato.	ação concreta		Rodada de ações seja jogada Jogador da vez
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
-- Engenheiro de software pode corrigir artefato.	ação concreta		Rodada de ações seja jogada Jogador da vez
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ARTEFATO CORRIGIDO			
-- Engenheiro de software pode inspecionar o artefato corrigido			Rodada de ações seja jogada Jogador da vez
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ARTEFATO DEFEITUOSO			
-- Engenheiro de software pode corrigir artefato	ação concreta		artefato seja corrigido Engenheiro de Software

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ARTEFATO INSPECIONADO			
-- avalia a qualidade do artefato.	ação concreta		produto seja construído Engenheiro de software
Porque	qualidade	[artefato]	produto seja concluído Jogador da vez

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ARTEFATO LIVRE DE DEFEITO			
-- Engenheiro de software pode construir mais artefatos.			
Porque	qualidade	[artefato]	produto seja concluído jogador da vez

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ENGENHEIRO DE SOFTWARE CONTRATADO			
--Engenheiro de software pode construir artefato	ação concreta		rodada de ações seja jogada Jogador da vez
--Engenheiro de software pode inspecionar artefato	ação concreta		rodada de ações seja jogada Jogador da vez
--Engenheiro de software pode corrigir artefato	ação concreta		rodada de ações seja jogada Jogador da vez

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
ENGENHEIRO DE SOFTWARE DEMITIDO			
-- jogador da vez pode contratar outro engenheiro de software	ação concreta		Engenheiro de software seja contratado Jogador da vez

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
PROBLEMA PERSISTENTE			
-- Adversário trata problema	ação concreta		Problema seja tratado Adversário
-- pode ser contraposto por carta de conceito	ação concreta		Problema seja tratado Adversário

TIPO: ESTADO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
PROBLEMA TEMPORÁRIO			
-- Adversário trata problema	ação concreta		Problema seja tratado Adversário

Figura 4.13 – *Template* com metas dos símbolos do tipo estado.

Passo 3: refinar as metas

Neste passo são apresentadas as metas refinadas agrupadas por ator, ordenadas cronologicamente e com as metas repetidas excluídas.

DEPENDER					DEPENDEE
JOGADOR DA VEZ					
	jogo	seja	iniciado		
	projeto	seja	escolhido		
	jogada de ações	seja	iniciada	Por	**SimulES**
	rodada de ações	seja	jogada		
	dado	seja	jogado		
	artefato	seja	construído	por	**engenheiro de software**
	artefato	seja	inspecionado	por	**engenheiro de software**
	artefato	seja	corrigido	por	**engenheiro de software**
	rodada de conceitos	Seja	iniciada	Por	**SimulES**
	rodada de conceitos	seja	jogada		
	cartas	sejam	descartadas		
	engenheiro de software	seja	contratado		
	engenheiro de software	seja	demitido		
	engenheiro de software	seja	descartado		
	jogo	seja	disponibilizado		
	problema	seja	aplicado		
	engenheiro de software	Seja	escolhido		
	conceito	seja	aplicado		
	habilidade do engenheiro	Seja	analisada		
	artefato	seja	destruído	por	**engenheiro de software**
	produto	seja	submetido		
	projeto	seja	concluído		
	modulo	seja	escolhido	Por	**adversário**
	modulo	seja	inspecionado	Por	**adversário**
qualidade [artefato]	artefato	seja	livre de defeitos		
qualidade [projeto]	projeto	seja	concluído	por	**Engenheiro de software**
	jogo	seja	vencido		

DEPENDER					DEPENDEE
ENGENHEIRO DE SOFTWARE					
	artefato	seja	escolhido		
	artefato	seja	construído		
	artefato	seja	inspecionado		
	artefato	seja	corrigido		
	artefato	seja	destruído		
	produto	seja	construído		
	modulo	seja	integrado		

DEPENDER					DEPENDEE
SIMULES					
	rodada de início	seja	iniciada		
	rodada de ações	seja	iniciada		
	rodada de conceitos	seja	iniciada		
	recursos	sejam	disponibilizados		
	cartas	sejam	compradas	por	**jogador da vez**
	artefatos	sejam	comprados	por	**jogador da vez**

DEPENDER					DEPENDEE
ADVERSÁRIO					
	projeto	seja	escolhido	por	**Jogador da vez**
	Projeto	seja	aceito	por	**Adversário**
	Jogo	seja	disponibilizado		
	problema	Seja	tratado		
	modulo	Seja	escolhido		
	modulo	Seja	inspecionado		

Figura 4.14 – Metas agrupadas por ator e cronologicamente organizadas.

4.2.2.Identificar as Situações de Dependência Estratégica

Passo 1: Distinguir *SDsituations*

Foram identificadas nove situações de dependência estratégica no projeto SimulES, com seus respectivos participantes:

1. **Joga Rodada de Início** [SimulES, jogador da vez, engenheiro de software, adversário]
2. **Joga Rodada de Ações** [SimulES, jogador da vez, engenheiro de software]

3. **Construção de Artefatos** [SimulES, jogador da vez, engenheiro de software]
4. **Inspeção de Artefatos** [jogador da vez, engenheiro de software]
5. **Correção de artefatos** [SimulES, jogador da vez, engenheiro de software]
6. **Integração de Artefatos em Módulo** [SimulES, jogador da vez, engenheiro de software]
7. **Joga Rodada de Conceitos** [SimulES, jogador da vez, engenheiro de software, adversário]
8. **Tratamento de Problemas** [jogador da vez, adversário]
9. **Submissão de Produtos** [SimulES, jogador da vez, adversário, engenheiro de software]

DEPENDER	SITU					DEPENDEE
JOGADOR DA VEZ						
	1	jogo	seja	iniciado		
	1	projeto	seja	escolhido		
	2	jogada de ações	seja	iniciada	Por	**SimulES**
	2	rodada de ações	seja	jogada		
	7	rodada de conceitos	Seja	iniciada	Por	**SimulES**
	7	rodada de conceitos	seja	jogada		
	2,3,4,5	projeto	seja	concluído		
	7	jogo	seja	disponibilizado		
	1,2	dado	seja	jogado		
	3	artefato	seja	construído	por	**engenheiro de software**
	4	artefato	seja	inspecionado	por	**engenheiro de software**
	5,9	artefato	seja	corrigido	por	**engenheiro de software**
	7,8	cartas	sejam	descartadas		
	7,8	engenheiro de software	seja	demitido		
	7	engenheiro de software	seja	descartado		
	1,7	engenheiro de software	seja	contratado		
qualidade [artefato]	2,4,5,7	artefato	seja	livre de defeitos		
qualidade [projeto]	2,4,5,7	projeto	seja	concluído	por	**Engenheiro de software**
	9	jogo	seja	vencido		
	3	habilidade do engenheiro	Seja	analisada		
	4,5	engenheiro de software	Seja	escolhido		
	7	artefato	seja	destruído	por	**engenheiro de software**
	7	problema	seja	aplicado		
	7,8	conceito	seja	aplicado		
	9	produto	seja	submetido		
	9	modulo	seja	escolhido	Por	**adversário**
	9	modulo	seja	inspecionado	Por	**adversário**

DEPENDER	SITU					DEPENDEE
ENGENHEIRO DE SOFTWARE						
	3	artefato	seja	construído		
	4	artefato	seja	inspecionado		
	5,9	artefato	seja	corrigido		
	7	artefato	seja	destruído		
	2,6	modulo	seja	integrado		
	2,3,4,5,6	produto	seja	construído		
	5,6	artefato	seja	escolhido		

DEPENDER	SITU					DEPENDEE
SIMULES						
	1	rodada de início	seja	iniciada	por	**jogador da vez**
	2	rodada de ações	seja	iniciada	por	**jogador da vez**
	7	rodada de conceitos	seja	iniciada	por	**jogador da vez**
	1,2,3,5,6, 7,9	recursos	sejam	disponibilizados		
	2,3,5	artefatos	sejam	comprados	por	**jogador da vez**
	2	cartas	sejam	compradas	por	**jogador da vez**

DEPENDER	SITU					DEPENDEE
ADVERSÁRIO						
	1	projeto	seja	escolhido	por	**Jogador da vez**
	1	Projeto	seja	Aceito		
	7	Jogo	seja	disponibilizado		
	8	problema	Seja	tratado		
	9	modulo	Seja	escolhido		
	9	modulo	Seja	inspecionado		

Figura 4.15 – Metas organizadas em *SDsituations.*

Passo 2: Reconhecer Interdependências entre *SDsituations*

A *SDsituation Joga Rodada de Início* é a situação inicial. Em seguida, acontece *Joga Rodada Ações* onde se pode optar por *Construir Artefato*, *Inspecionar Artefato*, *Corrigir Artefato* ou *Integrar Artefato em Módulo*. Em

sequência, ocorre *Joga Rodada de Conceitos*, onde pode ocorrer *Tratamento de Problema*. Por fim, tem–se *Submissão de Produtos*.

Passo 3: Construir Diagrama de *SDsituation*

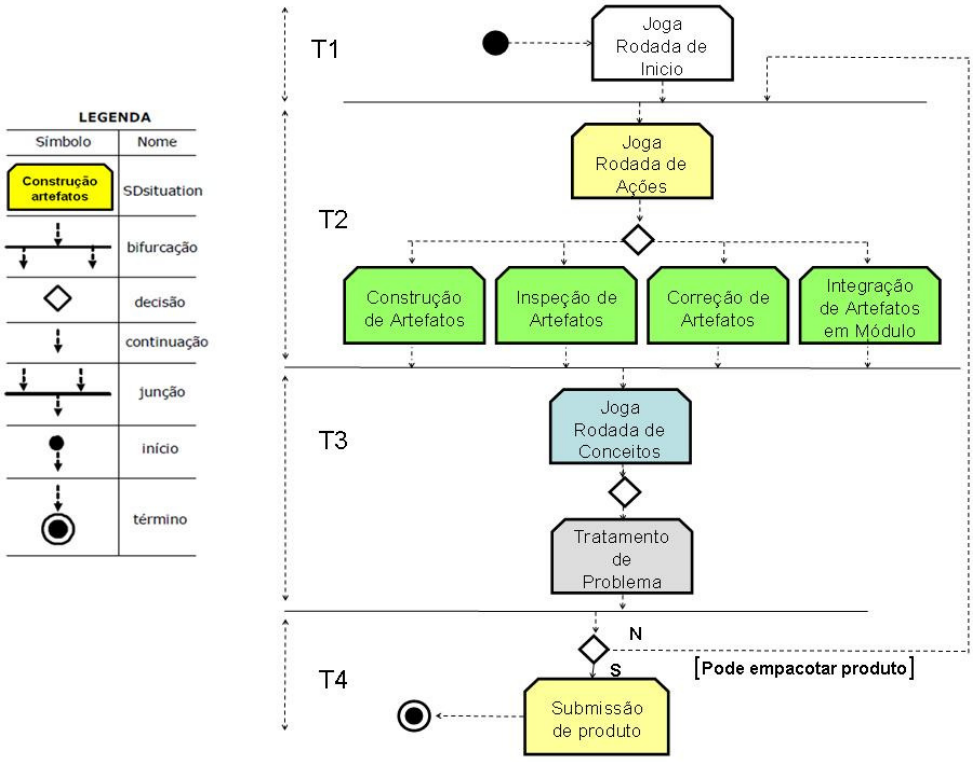


Figura 4.16 – Diagrama de *SDsituations* do SimulES.

4.2.3.Modelar as Metas dos Atores

Passo 1: Identificar Agentes, Posições e Papéis

Na Figura 4.17 estão identificados e apresentados sob a forma de um diagrama SA (*Strategic Actor*) os atores como agentes, assumindo posições e desempenhando papéis.

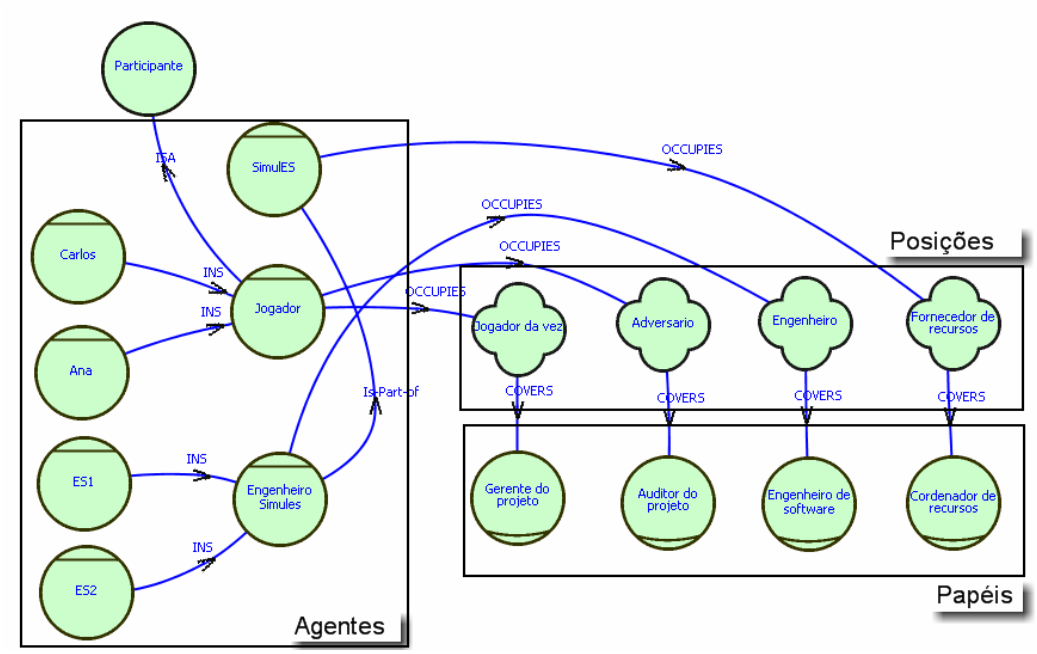


Figura 4.17 – O modelo SA do SimulES.

Passo 2: Criar os Painéis de Intencionalidade

As Figuras seguintes apresentam os painéis de intencionalidade (diagramas IP) do SimulES.

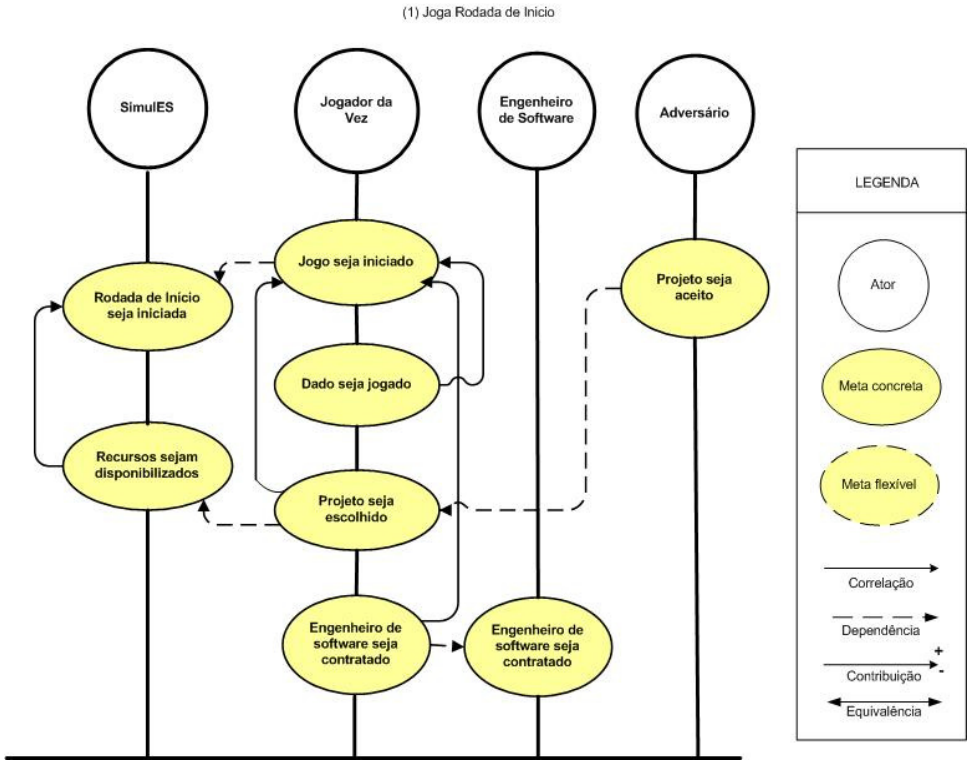


Figura 4.18 – Diagrama IP – Joga Rodada de Início.

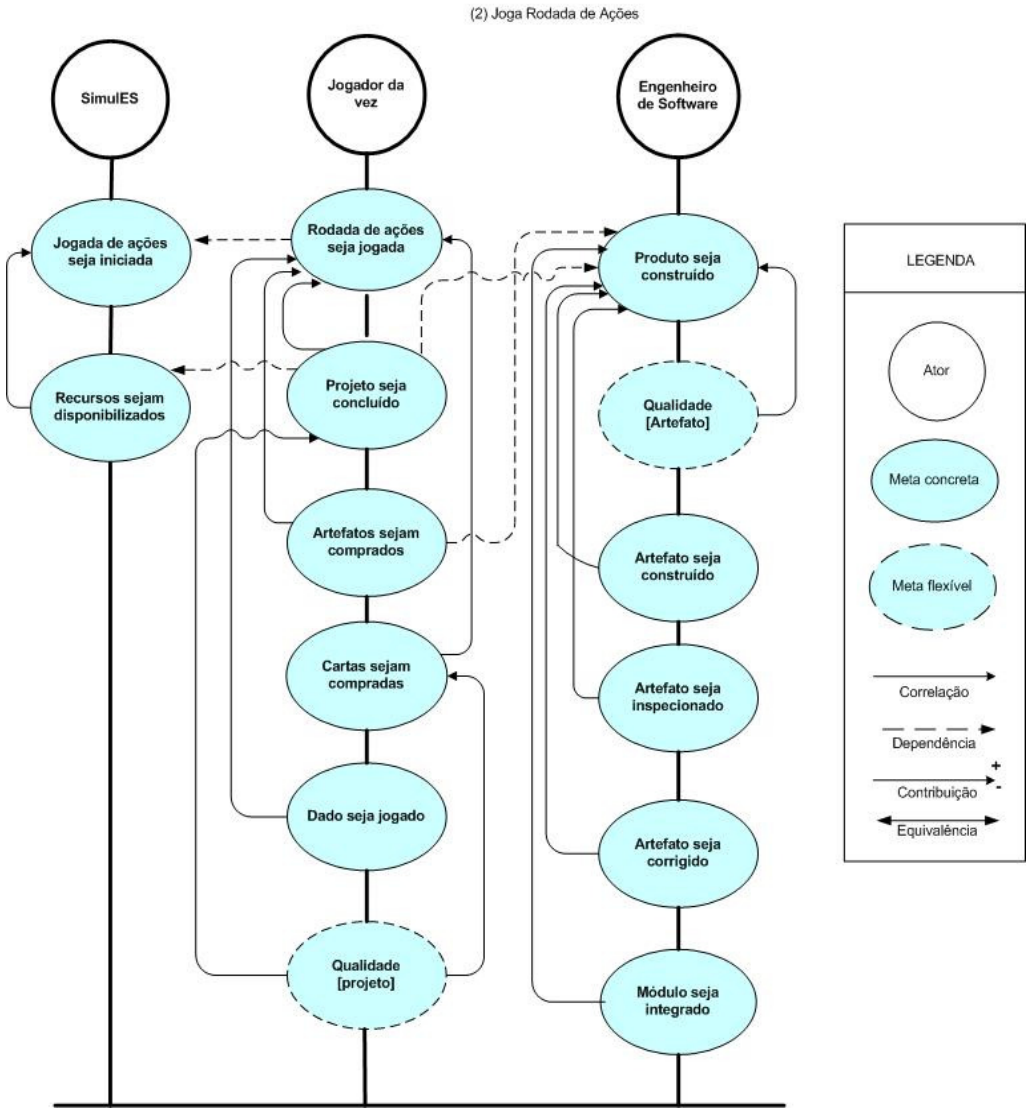
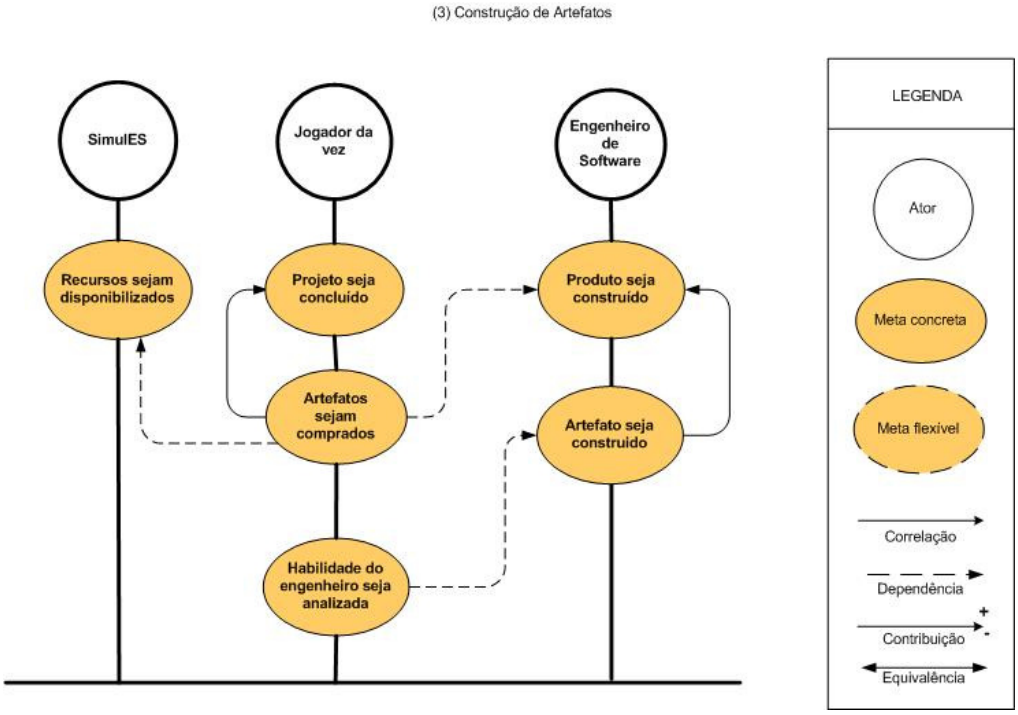


Figura 4.19 – Diagramas IP – Joga Rodada de Ações.



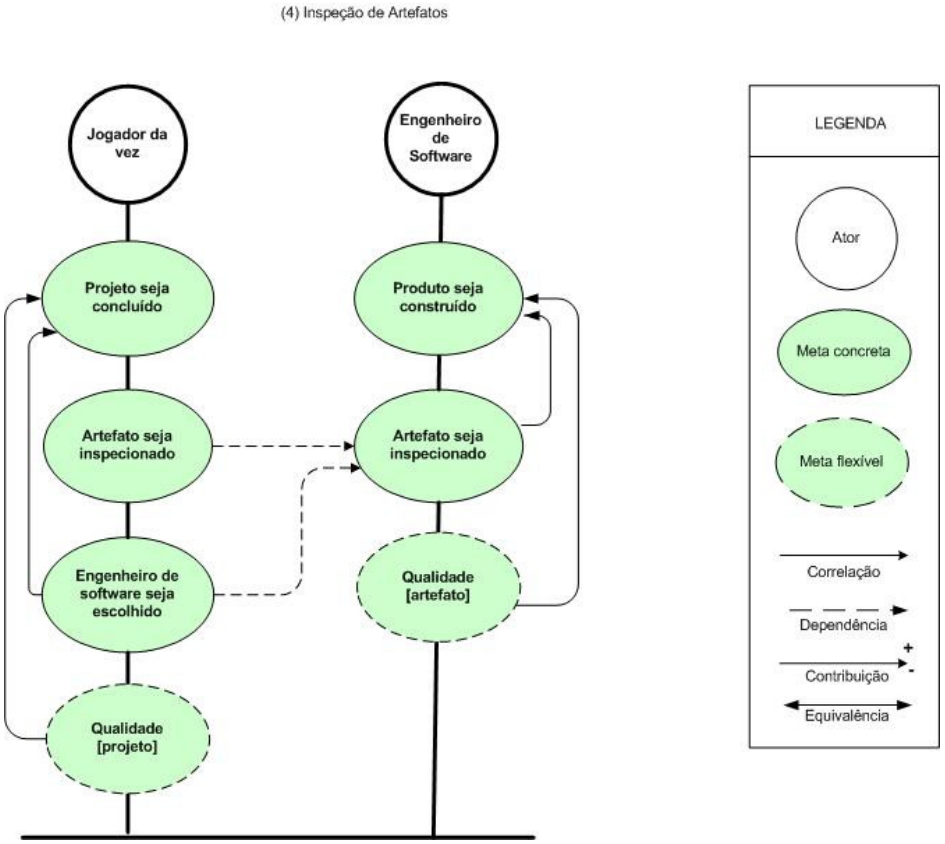


Figura 4.20 – Diagramas IP – Construção de Artefatos e Inspeção de Artefatos.

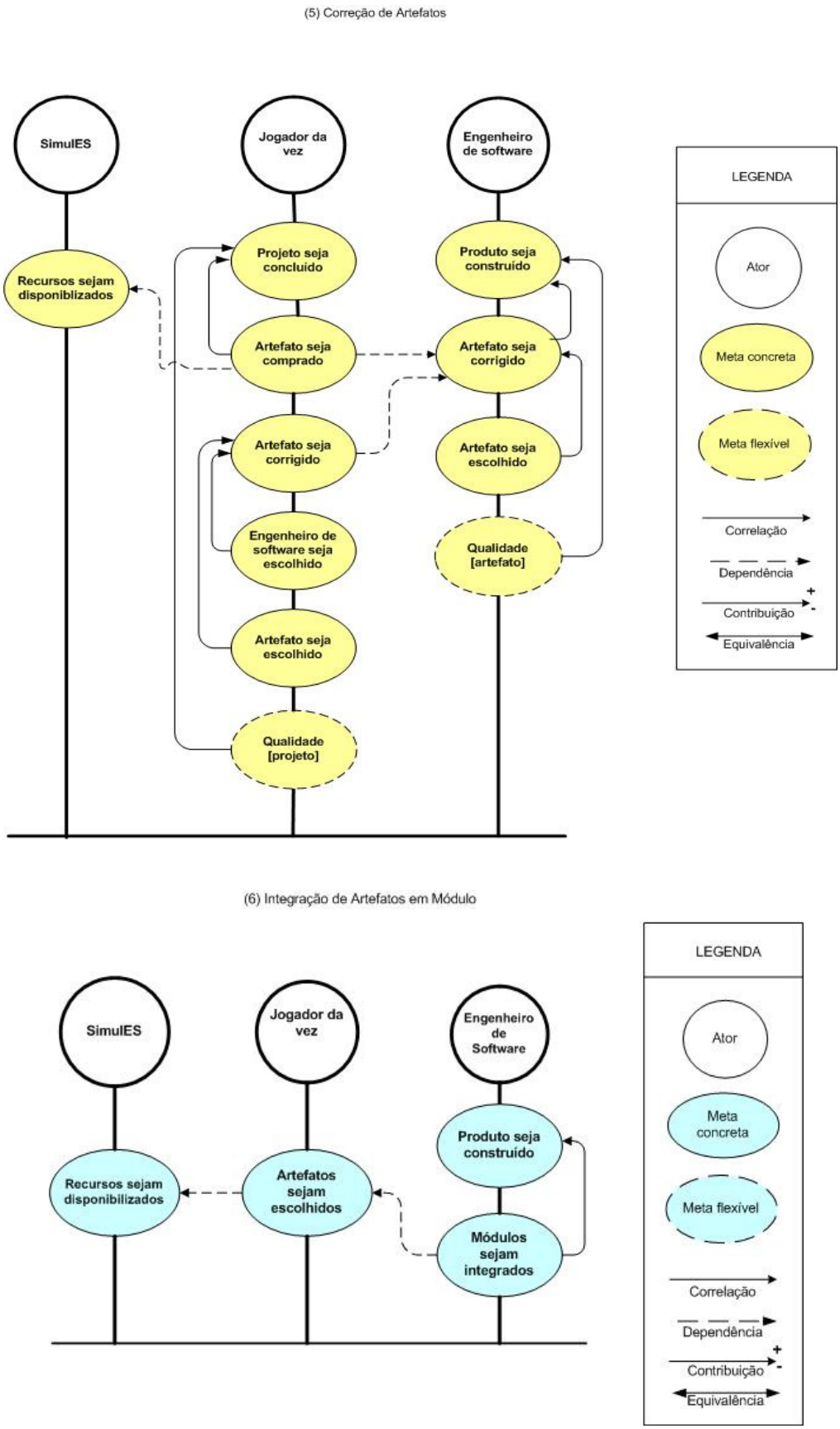


Figura 4.21 – Diagramas IP: Correção de Artefatos e Integração de Artefatos em Módulo.

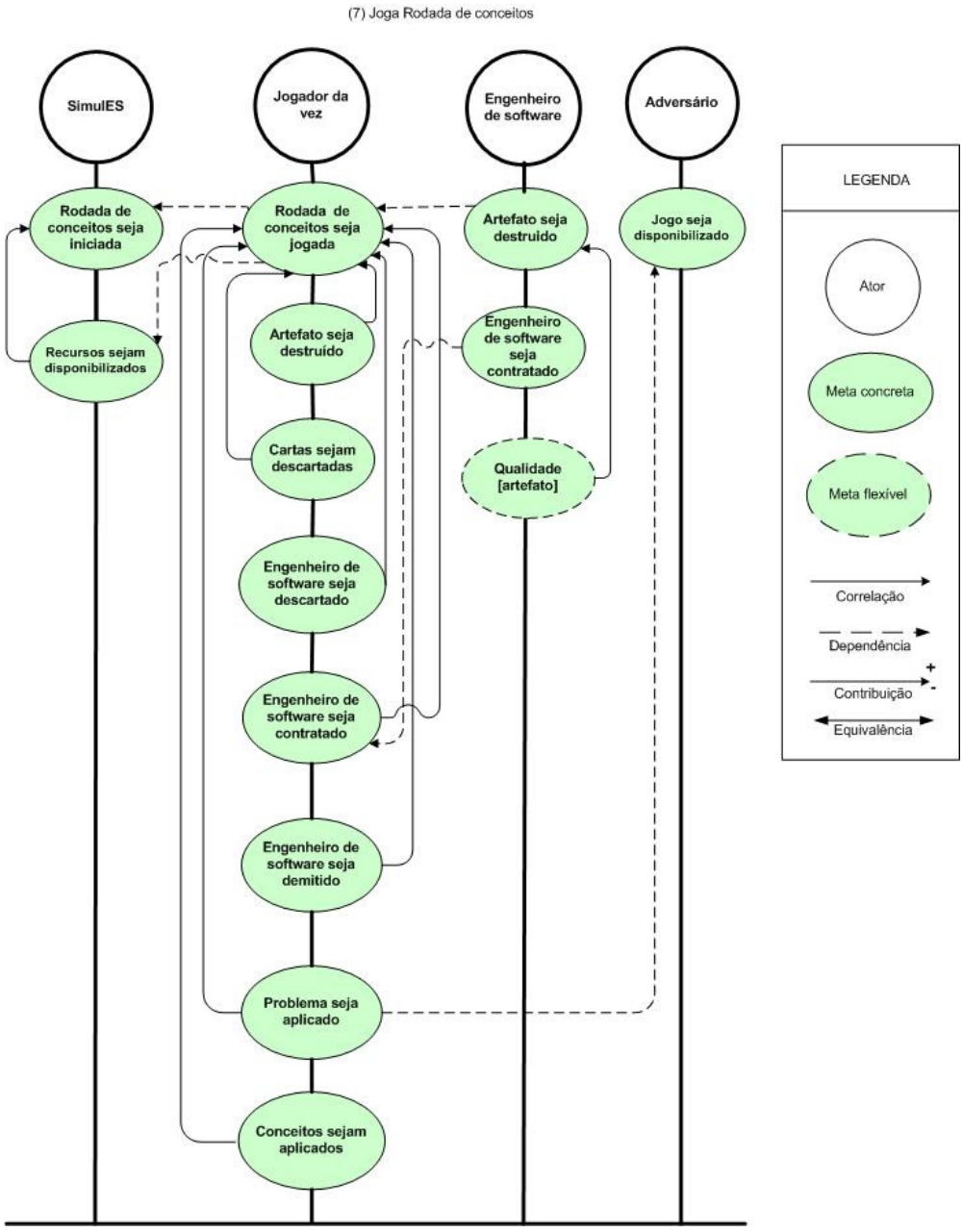


Figura 4.22 – Diagrama IP: Joga Rodada de Conceitos.

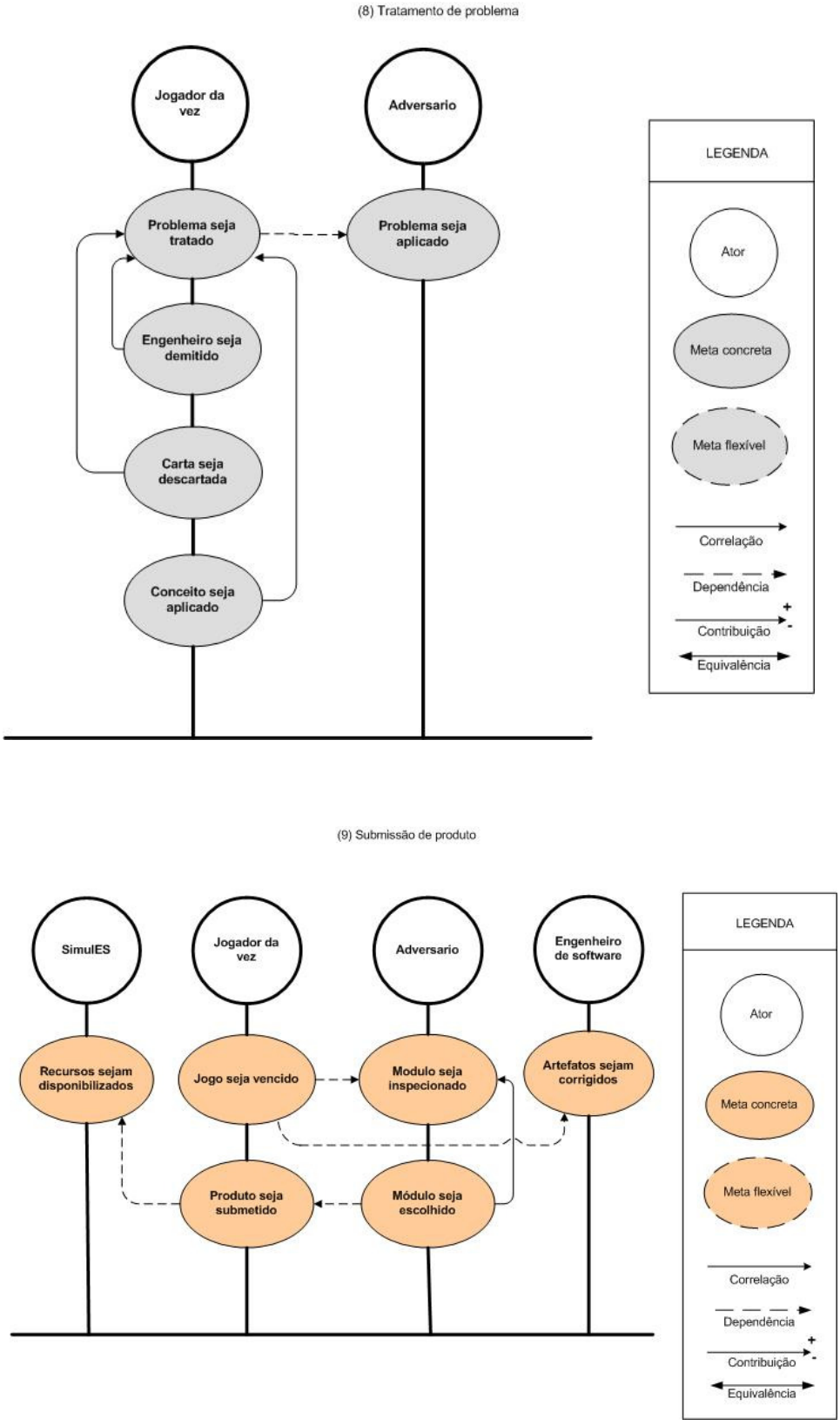


Figura 4.23 – Diagramas IP – Tratamento de Problema e Submissão de Produto.

4.2.4.Modelar a Racionalização das Metas dos Agentes

Passo 1: Construir Modelos SD

Passo 2: Construir Modelos SR

A seguir estão representados os diagramas SD e SR para cada *SDsituation* descritas anteriormente.

SDsituation: Joga Rodada de Início

O diagrama IP desta situação mostra as seguintes dependências estratégicas: uma entre jogador da vez / adversário, uma entre jogador da vez / engenheiro de software e duas entre jogador da vez / SimulES. Baseando–se na análise de grau de liberdade das dependências, chegou–se à conclusão que todas eram por metas, ou seja, o *dependee* tem plena liberdade para tomar decisões.

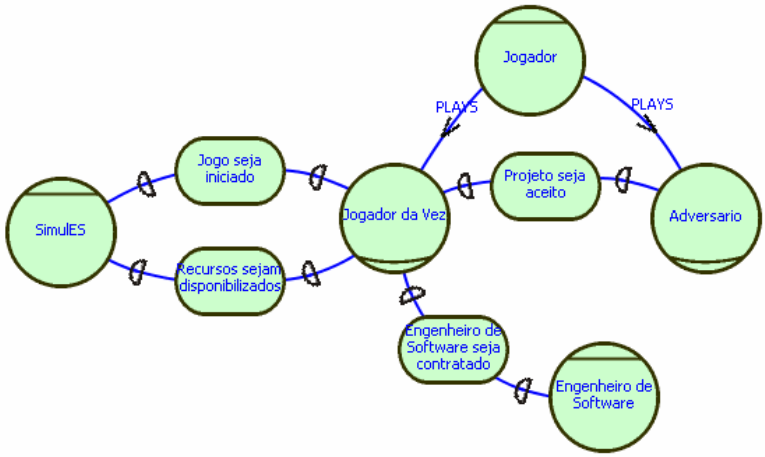


Figura 4.24 – Modelo SD – Joga Rodada de Início.

A Figura 4.25 mostra os elementos e ligações provenientes do *rationale* da situação de dependência estratégica Joga Rodada de Início.

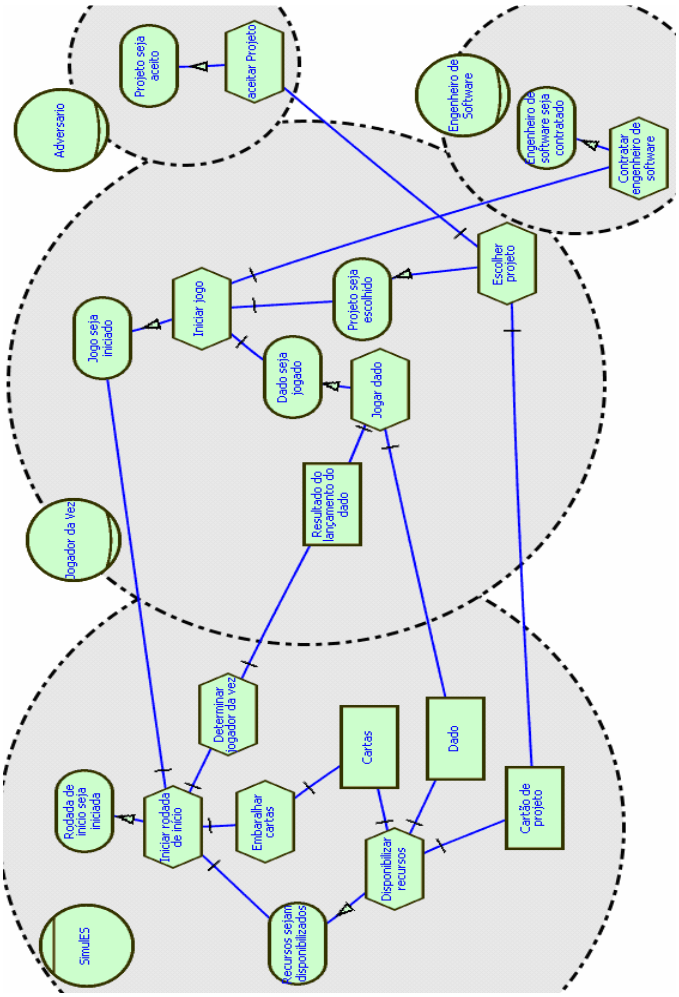


Figura 4.25 – Modelo SR – Joga Rodada de Início.

SDsituation: Joga Rodada de Ações

O diagrama IP desta situação apresenta as seguintes dependências estratégicas: duas entre SimulES / jogador da vez e duas entre jogador da vez / engenheiro de software. Nesta ocasião, todas as dependências com exceção de uma são metas. A restante é a meta flexível "qualidade [artefato]".

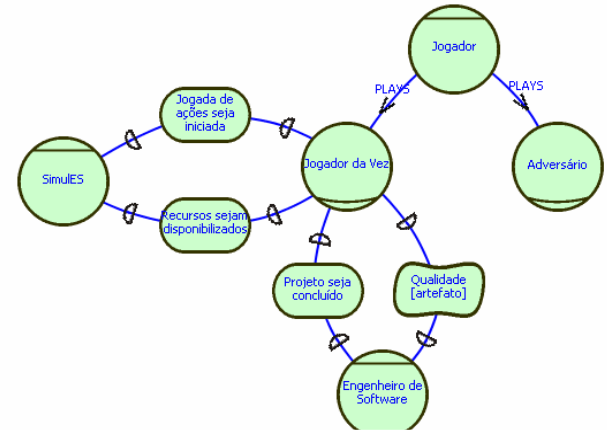


Figura 4.26 – Modelo SD – Joga Rodada de Ações.

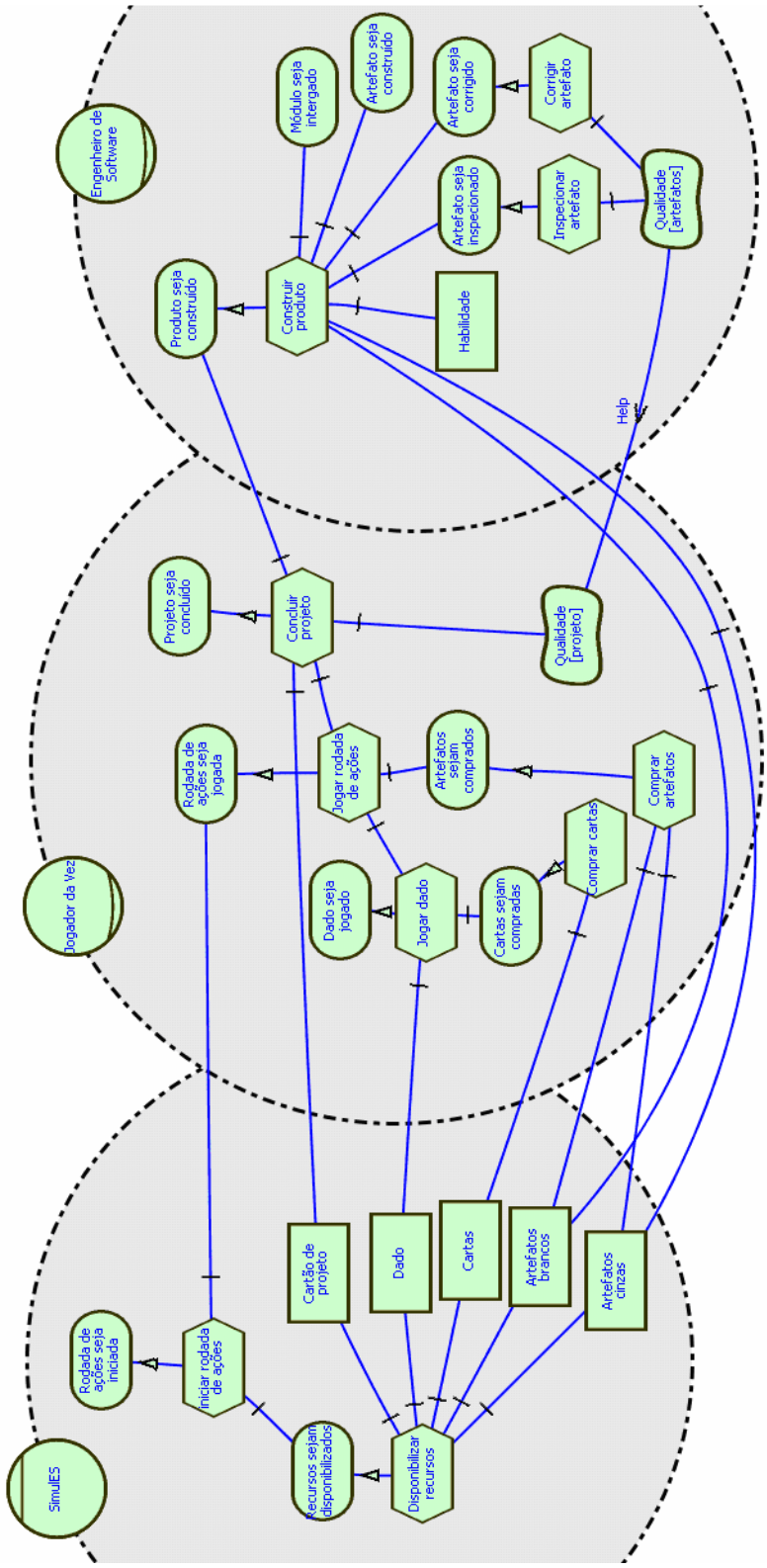


Figura 4.27 – Modelo SR – Joga Rodada de Ações.

SDsituation: Construção de Artefato

A Figura 4.28 mostra as dependências: uma entre SimulES / jogador da vez e duas entre jogador da vez / engenheiro de software. Uma das dependências é representada por uma tarefa, pois o *dependee* deve obedecer a certos critérios.

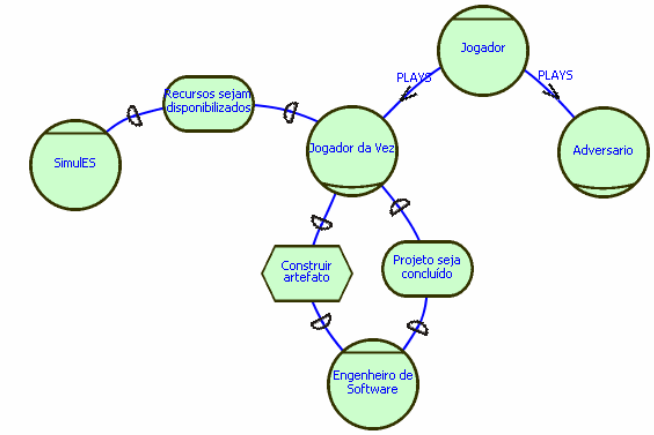


Figura 4.28 – Modelo SD – Construção de Artefato.

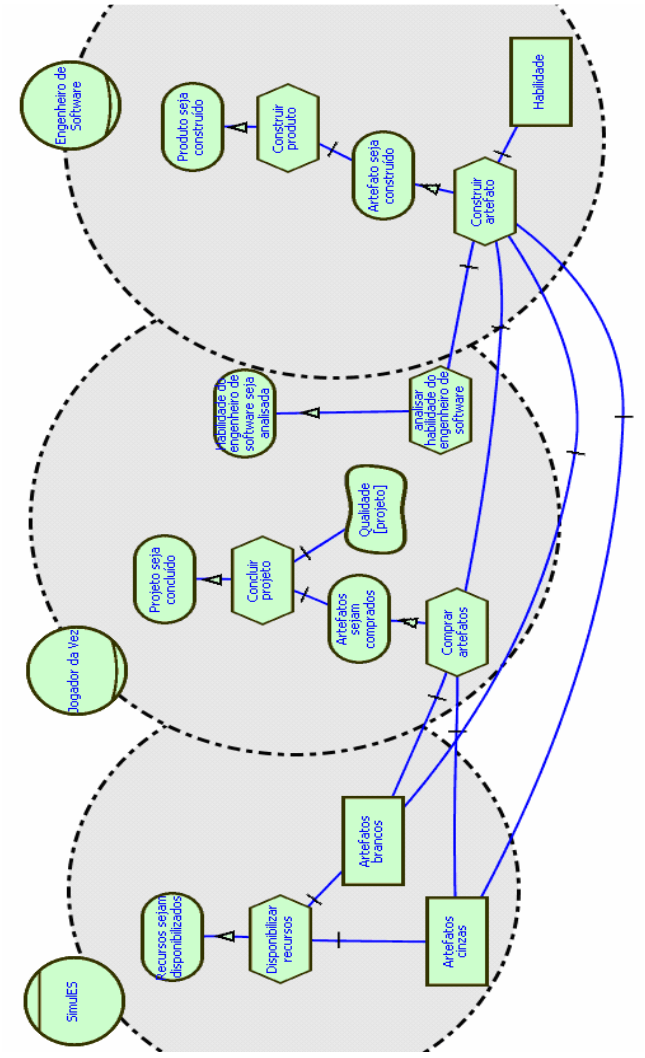


Figura 4.29 – Modelo SR – Construção de Artefato.

SDsituation: Inspeção de Artefato

O modelo apresenta duas dependências entre jogador da vez / engenheiro de software: por tarefa e por meta flexível. Ambos já foram discutidos anteriormente.

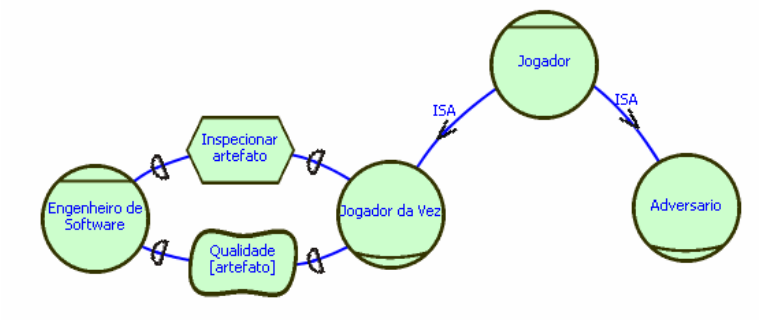


Figura 4.30 – Modelo SD – Inspeção de Artefato.

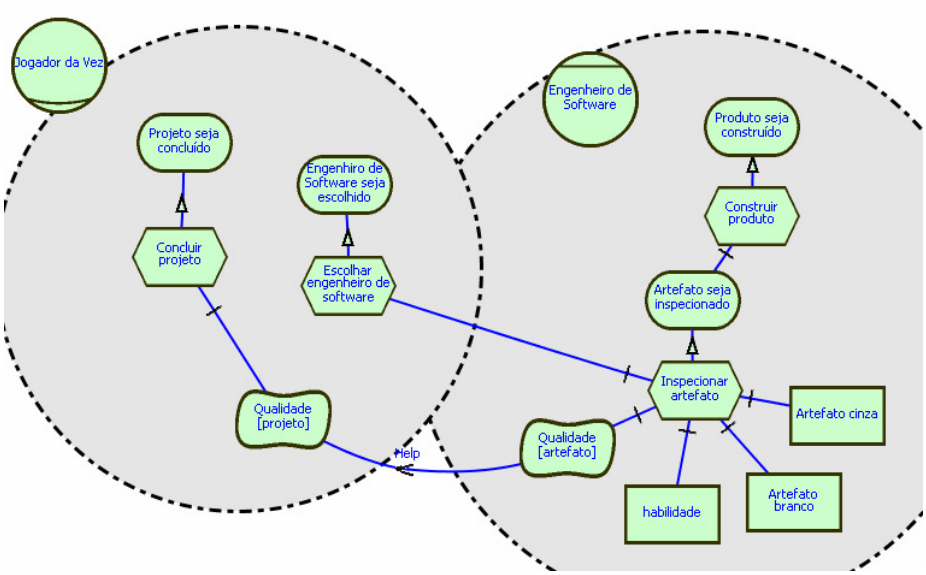


Figura 4.31 – Modelo SR – Inspeção de Artefato.

SDsituation: Correção de Artefatos

O diagrama IP desta situação mostra as seguintes dependências estratégicas: uma entre jogador da vez / SimulES e duas entre jogador da vez / engenheiro de software. Baseando–se na análise de grau de liberdade das dependências,

observa–se que temos novamente uma dependência por tarefa, além da presença de uma meta flexível.

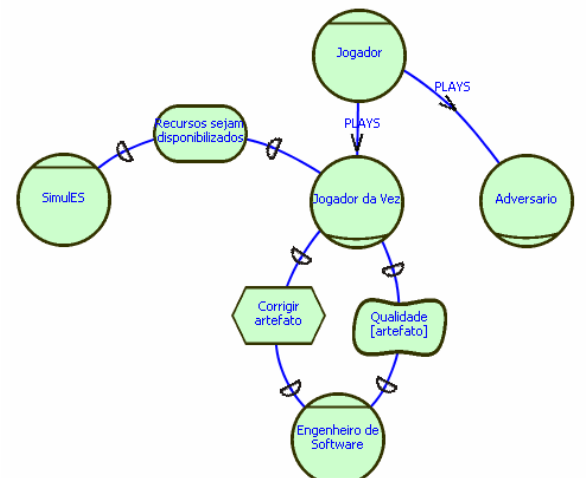


Figura 4.32 – Modelo SD – Correção de Artefato.

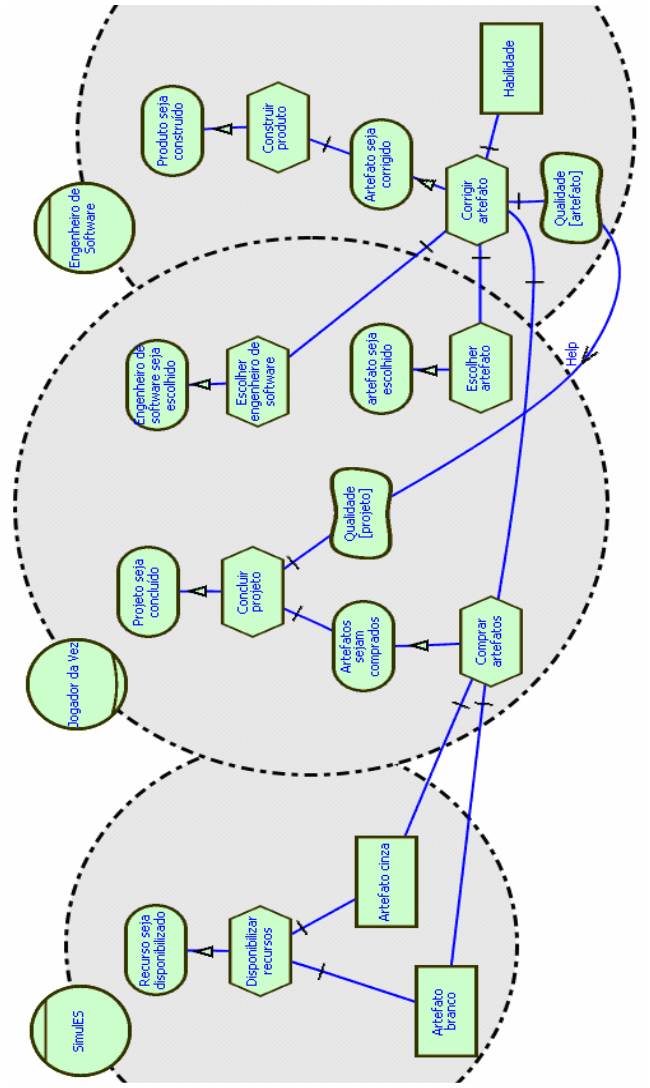


Figura 4.33 – Modelo SR – Correção de Artefato.

SDsituation: Integração de Artefatos em Módulo

Esta situação apresenta apenas duas dependências: uma entre SimulES / jogador da vez e uma entre jogador da vez / engenheiro de software.

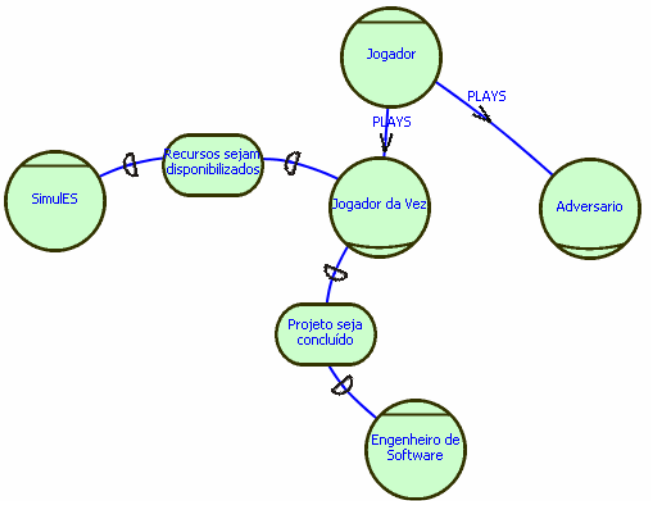


Figura 4.34 – Modelo SD – Integração de Artefato em Módulo.

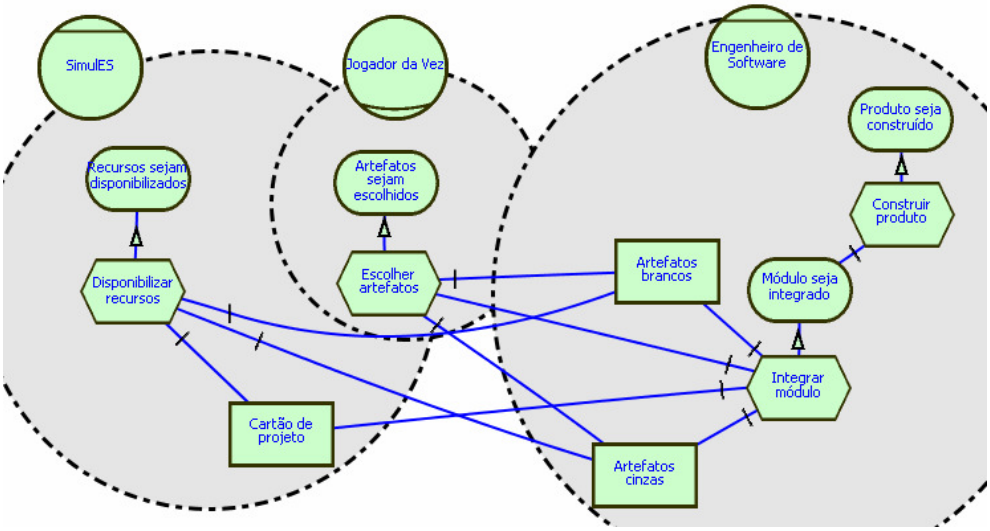


Figura 4.35 – Modelo SR – Integração de Artefato em Módulo.

SDsituation: Joga Rodada de Conceitos

O diagrama IP dessa situação apresenta cinco dependências estratégicas: duas entre jogador da vez / SimulES, uma entre jogador da vez / adversário e duas entre jogador da vez / engenheiro de software. Apenas uma delas é por tarefa. Todas as demais são por metas.

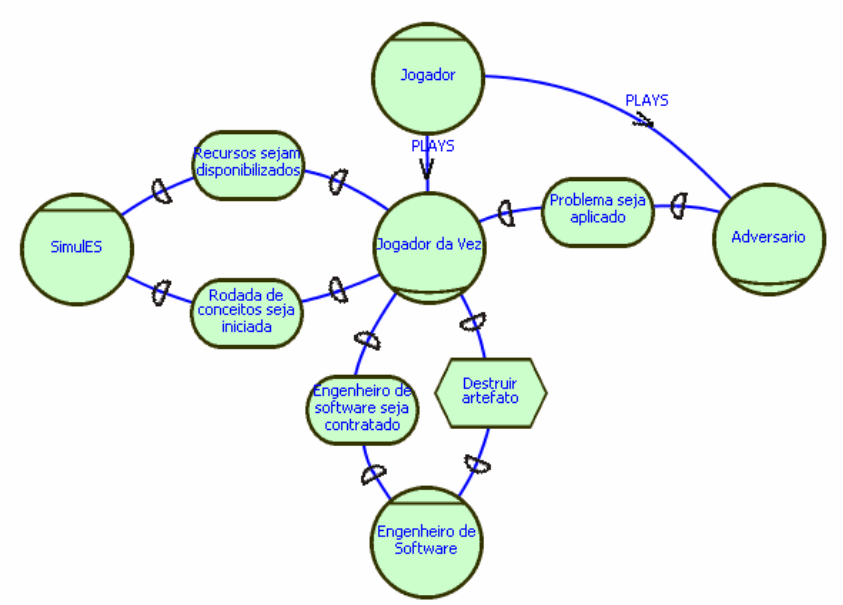


Figura 4.36 – Modelo SD – Joga Rodada de Conceitos.

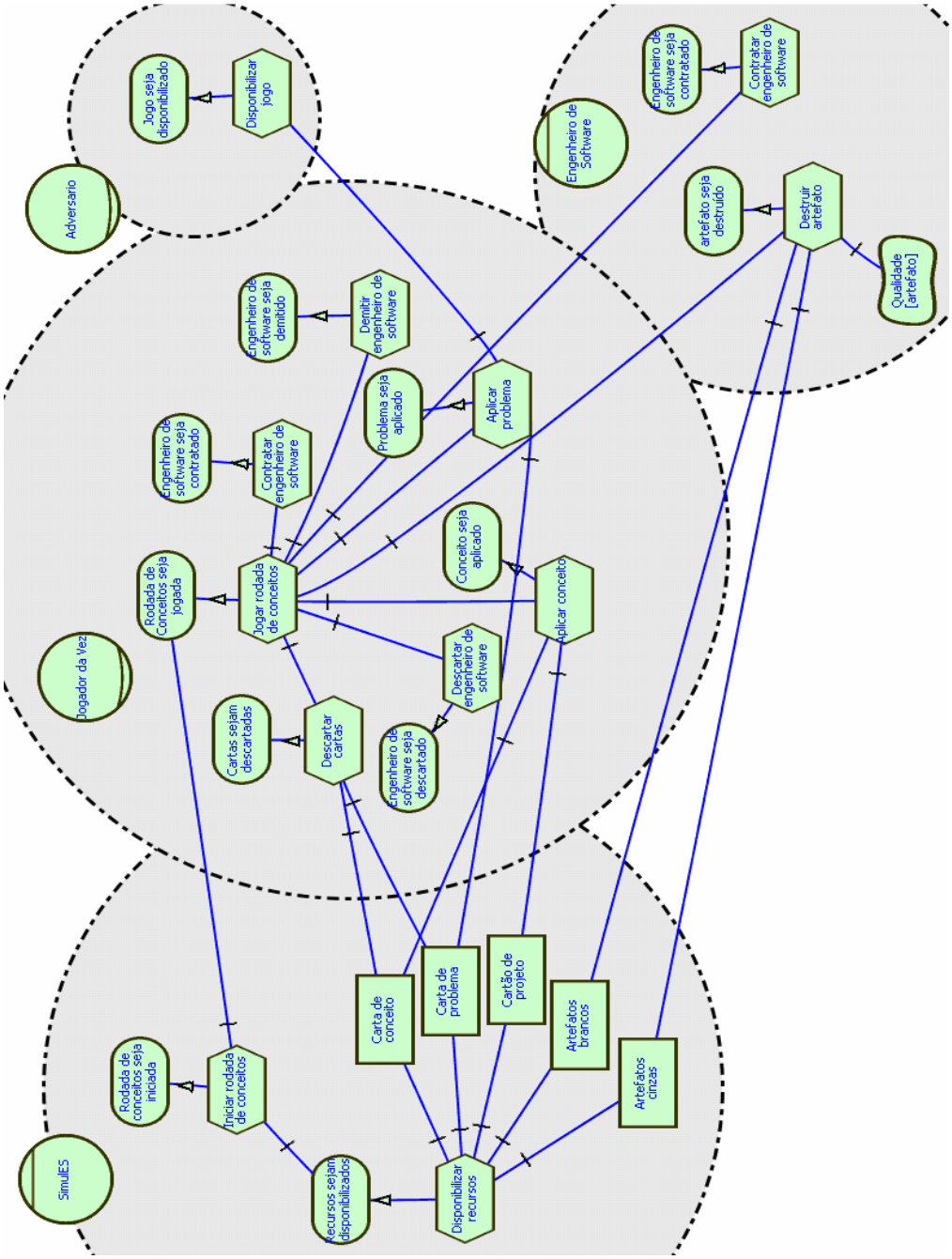


Figura 4.37 – Modelo SR – Joga Rodada de Conceitos.

SDsituation: Tratamento de Problema

Esta situação apresenta apenas uma dependência entre jogador da vez / adversário, representada por uma meta concreta.

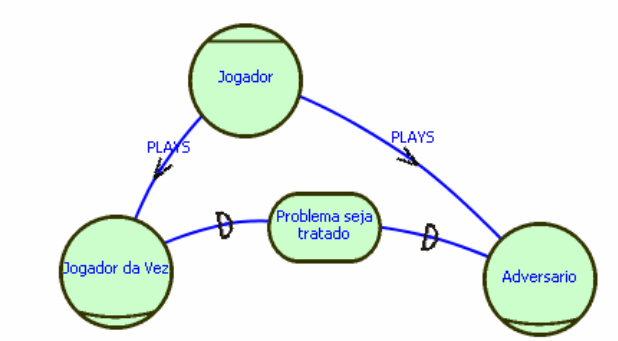


Figura 4.38 – Modelo SD – Tratamento de Problema.

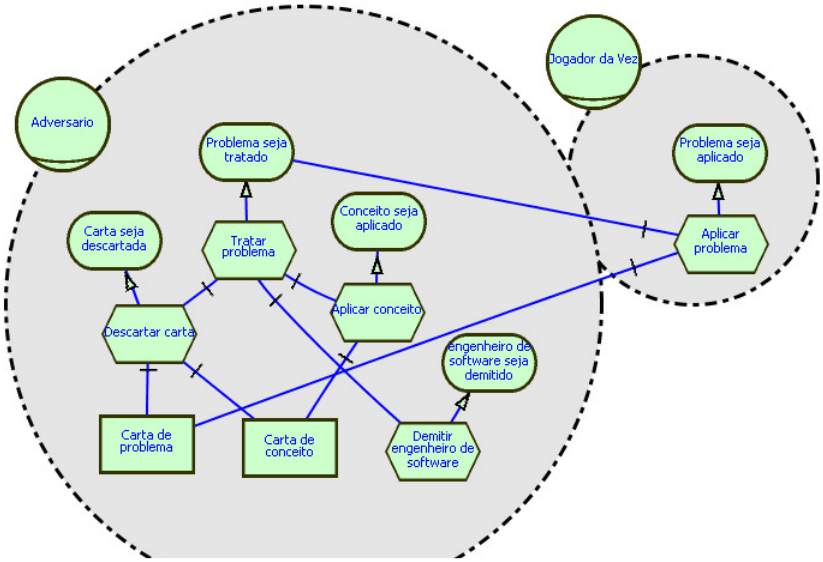


Figura 4.39 – Modelo SR – Tratamento de Problema.

SDsituation Submissão de Produto

O diagrama IP desta situação mostra as seguintes dependências estratégicas: uma entre jogador da vez / SimulES, uma entre jogador da vez / engenheiro de software e duas entre jogador da vez / adversário. Baseando–se na análise de grau de liberdade das dependências, chegou–se à conclusão que uma das dependências era por recurso, pois o *depender* (adversário) necessita da

liberação de um recurso (módulos) para que o mesmo seja inspecionado. Além disso, apresentam–se dependências por metas e uma por tarefa.

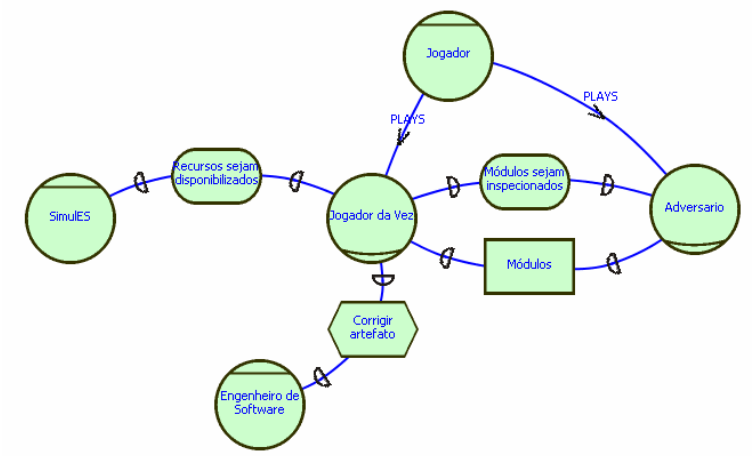


Figura 4.40 – Modelo SD – Submissão de Produto.

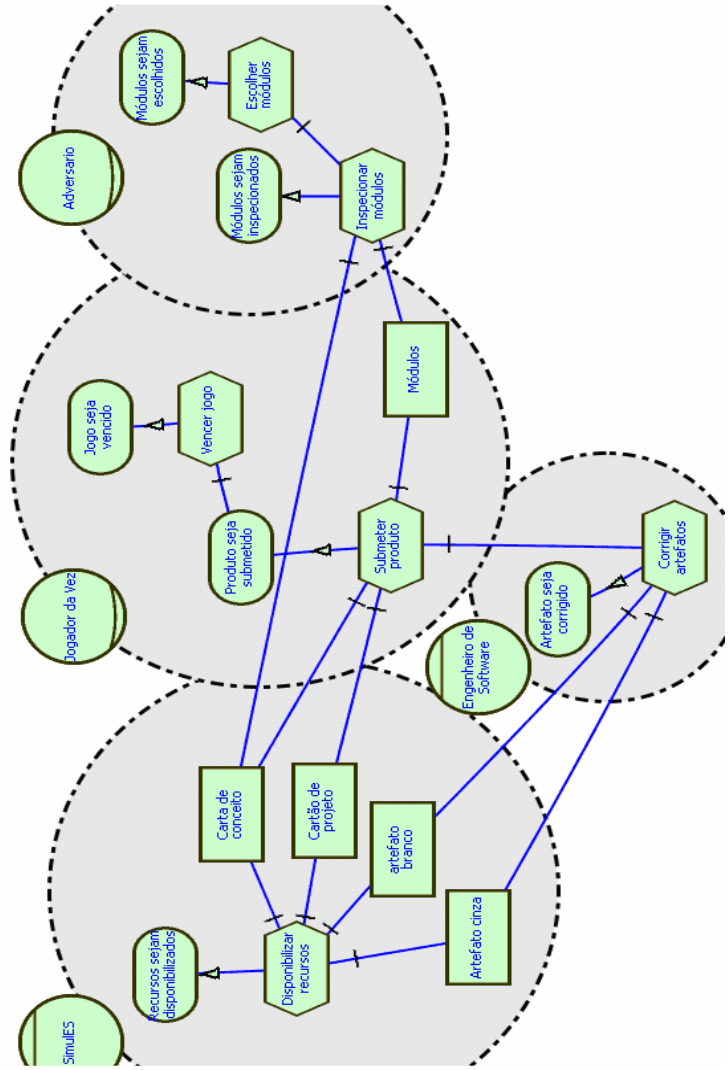


Figura 4.41 – Modelo SR – Submissão de Produto.

4.2.5.Especificar as SDsituations

A seguir, serão descritas as *SDsituations* utilizando a técnica de cenários [13]. Neste passo é importante que a descrição em cenários reflita exatamente os modelos construídos, pois sobre essa descrição será realizada a transformação para validar os modelos i* obtidos.

Titulo:	**Joga Rodada de Início**
Objetivo:	Jogo seja iniciado
Contexto:	Tabuleiro individual na mesa, cartas embaralhadas sobre a mesa e
Localização Geográfica:	Sala de jogo
Localização Temporal:	Início de jogo
Restrição:	
Pré-condição:	O sentido do jogo é horário
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, adversário, engenheiro de software, SimulES
Episódios:	1– SimulES inicia rodada de início.
	2– Jogador da vez lança o dado. **Restrição:** Jogador que tirar o maior número no dado, inicia o jogo.
	3– SimulES determina jogador da vez.
	4– Jogador da vez escolhe projeto.
	5– Adversário concorda com o projeto.
	6– Jogador da vez contrata engenheiro de software.
Exceções:	

Figura 4.42 – SDsituation Joga Rodada de Início (descrição).

Titulo:	**Joga Rodada de Ações**
Objetivo:	Rodada de ações seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Antes da rodada de conceitos
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES inicia rodada de ações.
	2– Jogador da vez lança dado.
	3– Se o resultado do lançamento do dado for igual a 1, 2 ou 3, jogador da vez compra 1,2 ou 3 cartas do monte de problemas e conceitos.
	4– Se o resultado do lançamento do dado for igual a 4, 5 ou 6, jogador da vez compra 3 cartas do monte de problemas e conceitos e a diferença de engenheiros de software.
	5– Engenheiro de software pode construir, inspecionar ou corrigir artefato.
	6– Engenheiro de software pode integrar artefatos em um módulo.
Exceções:	

Figura 4.43 – SDsituation Joga Rodada de Ações (descrição).

Titulo:	**Construção de Artefato**
Objetivo:	Produto seja construído
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez possui engenheiro de software
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez, para cada engenheiro de software, compra artefatos.
	3– Engenheiro de software constrói artefatos.
	4– Jogador da vez coloca artefatos no tabuleiro.
Exceções:	

Figura 4.44 – SDsituation Construção de Artefato (descrição).

Titulo:	**Inspeção de Artefato**
Objetivo:	Artefatos sejam inspecionados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, engenheiro de software
Episódios:	1– Jogador da vez escolhe artefato do que será inspecionado.
	2– Engenheiro de software inspeciona artefato. **Restrição:** Engenheiro de software gasta dois pontos de tempo para inspecionar artefatos de outro engenheiro.
	3– Jogador da vez desvira artefatos.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto]
Exceções:	

Figura 4.45 – SDsituation Inspeção de Artefato (descrição).

Titulo:	**Correção de Artefato**
Objetivo:	Artefatos sejam corrigidos
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos defeituosos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez escolhe engenheiro que terá artefato corrigido.
	3– Jogador da vez escolhe artefato defeituoso.
	4– Engenheiro de software corrige artefato. **Restrição:** Engenheiro de software gasta dois pontos de tempo para corrigir artefatos de outro engenheiro.
	5– Jogador da vez substitui artefato corrigido por outro da mesma cor.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto]
Exceções:	

Figura 4.46 – SDsituation Correção de Artefato (descrição).

Titulo:	**Integração de Artefatos em Um Módulo**
Objetivo:	Módulos sejam integrados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez escolhe módulo do projeto
	3– Jogador da vez seleciona os artefatos do tabuleiro.
	4– Engenheiro de software integra o módulo em um monte fora do tabuleiro
Exceções:	

Figura 4.47 – SDsituation Integração de Artefatos em Módulo (descrição).

Titulo:	**Joga Rodada de Conceitos**
Objetivo:	Rodada de conceitos e problemas seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido Todos os jogadores terminaram a rodada de ações
Recursos:	cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software, adversário
Episódios:	1– SimulES inicia rodada de conceitos.
	2– Jogador da vez descarta cartas, caso possua mais de 6 cartas nas mãos.
	3– Jogador da vez aplica conceitos caso sejam permanentes. **Restrição:** não pode exceder o orçamento do projeto.
	4– Jogador da vez pode contratar engenheiro de software.
	5– Jogador da vez pode demitir engenheiro de software.
	6– Engenheiro de software pode destruir artefato.
	7– Jogador da vez pode escolher até 3 adversários para submeter problema.
	8– Jogador da vez lê em voz alta o problema aplicado ao adversário.
Restrição:	RNF: Qualidade [artefato]
Exceções:	

Figura 4.48 – SDsituation Joga Rodada de Conceitos (descrição).

Titulo:	**Tratamento de Problema**
Objetivo:	Problemas sejam tratados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de conceitos
Restrição:	
Pré-condição:	Adversário recebeu problema do jogador da vez
Recursos:	cartas, cartão de projeto, tabuleiro individual, tabuleiro central
Restrição:	
Atores:	Jogador da vez, adversário
Episódios:	1– Jogador da vez aplica problema.
	2– Adversário atende a demanda do problema.
	3– Adversário pode aplicar conceito.
	4– Adversário mantém problema permanente.
	5– Adversário descarta problema temporário.
Exceções:	

Figura 4.49 – SDsituation Tratamento de Problema (descrição).

Titulo:	**Submissão de Produto**	
Objetivo:	Jogo seja vencido	
Contexto:	Cartão de projeto no centro da mesa	
Localização Geográfica:		Sala de jogo
Localização Temporal:		Após a rodada de conceitos
Restrição:		
Pré-condição:		Jogador da vez integrou último módulo
Recursos:	cartas, cartão de projeto, módulos	
Restrição:		
Atores:	SimulES, jogador da vez, engenheiro de software, adversário	
Episódios:	1– SimulES disponibiliza recursos.	
	2– Jogador da vez disponibiliza seus módulos produzidos.	
	3– Adversário escolhe módulos. **Restrição:** adversários não podem escolher módulo protegido por carta de conceito.	
	4– Adversário inspeciona módulos.	
	5– Engenheiro de software corrige um artefato por turno, se necessário.	
	6– Jogador da vez vence o jogo.	
Exceções:		

Figura 4.50 – SDsituation Submissão de Produto (descrição).

4.3.Transformar

Após descrever as situações de dependência estratégica utilizando a técnica de cenários [13], nosso trabalho irá aplicar uma transformação para que os cenários fiquem adequados a ferramenta de simulação *UCEd* [16]. Em seguida, os cenários serão simulados e validados com os interessados. As heurísticas de transformação de cenários estão presentes na seção 3.3.1.

Inicialmente, para fins didáticos, iremos tratar as metas flexíveis que foram eliciadas anteriormente (Qualidade [artefato] e Qualidade [projeto]). A Figura 4.51 mostra o grafo NFR com suas respectivas operacionalizações, que serão inseridas posteriormente nos cenários transformados.

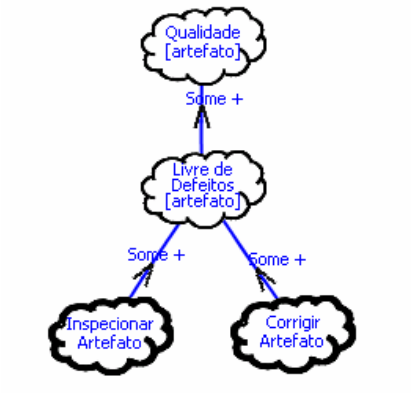


Figura 4.51 – Grafo NFR da meta flexível Qualidade [artefato].

Podemos observar na Figura 4.51 que as ações de inspecionar artefatos e corrigir artefatos que apresentem defeitos contribuem positivamente para a qualidade do artefato. A Figura 4.52 apresenta o grafo NFR de Qualidade [projeto].

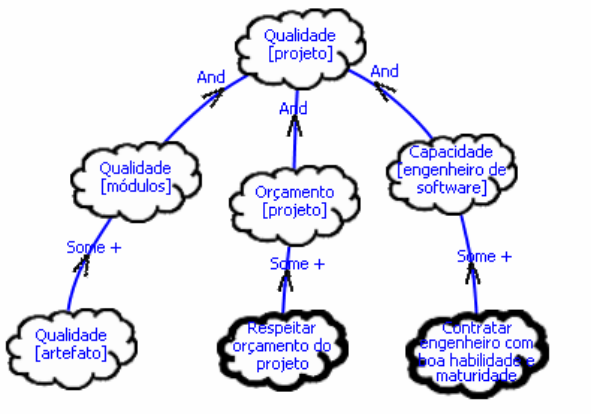


Figura 4.52 – Grafo NFR da meta flexível Qualidade [projeto].

Pode–se perceber na Figura 4.52 que a qualidade do projeto está relacionada à qualidade dos módulos que compõem este projeto, a não ultrapassar o orçamento do projeto e a capacidade (habilidade e maturidade) de cada engenheiro de software que participa do projeto. As operacionalizações ocorrem de maneira natural, ou seja, para se ter módulos com qualidade, os artefatos que compões estes módulos também devem ser de qualidade. Já para o orçamento, o respeito em não ultrapassá–lo durante o projeto também influencia em sua qualidade. Por fim, a capacidade de cada engenheiro de software determina o ritmo do projeto construído, a minimização dos riscos de receber um problema e a melhoria da taxa de construção dos artefatos.

Neste momento, tendo em mãos os grafos NFRs das metas flexíveis, podemos integrar suas operacionalizações das mesmas nos cenários [14], para inseri–los como entrada para a ferramenta de simulação. Em seguida, serão apresentados os cenários transformados.

Title: Joga Rodada Inicio
Description:
CONTEXTO: Tabuleiro individual e cartas embaralhadas sobre a mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: inicio de jogo

RECURSOS: dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, adversario , engenheiro de software
Goal: Descrever os preparativos de inicio de jogo
Follows Use Cases:
Invariant:
Precondition: O sentido de jogo e horario **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND**
LOCALIZACAO TEMPORAL **is** inicio de Jogo
STEPS
1.SimulES inicia rodada de inicio.
2.Jogador da vez lanca dado.
3.SimulES determina jogador da vez.
4.Jogador da vez inicia o jogo.
5.Jogador da vez escolhe projeto.
6.Adversario concorda com o projeto.
7.Jogador da vez contrata engenheiro de software
ALTERNATIVES
2.a.IF Jogador da vez nao tirou o maior numero **no** dado /* RESTRICAO*/
2.a.1. SimulES determina adversario como jogador da vez.
2.a.2. Adversario inicia o jogo.
2.a.3. Goto Step 5.
Success Postcondition:

Figura 4.53 – Cenário Joga Rodada de Início (transformado).

Title: Joga Rodada Acoes
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: antes da rodada de conceitos

RECURSOS: dado, cartas, cartao de projeto, tabuleiro central, tabuleiro individual

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, engenheiro de software
Goal: Descrever as etapas da rodada de acoes
Follows Use Cases:
Invariant:
Precondition: jogador da vez ja foi escolhido **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** antes da rodada de conceitos
STEPS
1.SimulES inicia rodada de acoes.
2.Jogador da vez lanca dado.
3.Jogador da vez compra 1, 2 ou 3 cartas se resultado do lancamento igual a 1, 2 ou 3.
4.Jogador da vez compra 3 cartas e a diferenca de engenheiro de sw se resultado igual 4, 5 ou 6.
5.Engenheiro de software pode construir, inspecionar ou corrigir artefato. // Qualidade [artefato]
6.Engenheiro de software pode integrar artefatos em um modulo.
ALTERNATIVES
5.a. Jogador da vez contrata engenheiro de software // Qualidade [projeto]
5.a.1. Jogador da vez deve respeitar orcamento do projeto
Success Postcondition:

Figura 4.54 – Cenário Joga Rodada de Ações (transformado).

Title: Construcao de Artefato
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: durante rodada de acoes

RECURSOS: cartas, cartao de projeto, artefatos, tabuleiro central, tabuleiro individual

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, engenheiro de software
Goal: Descrever as etapas da construcao de artefatos
Follows Use Cases:
Invariant:
Precondition: jogador da vez possui engenheiro de software **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** durante rodada de acoes
STEPS
1.SimulES disponibiliza recursos.
2.Jogador da vez, para cada engenheiro de software, compra artefatos.
3.Engenheiro de software constroi artefatos.
4.Jogador da vez coloca artefatos **no** tabuleiro.
Success Postcondition:

Figura 4.55 – Cenário Construção de Artefato (transformado).

Title: Inspecionar Artefato
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: durante rodada de acoes

RECURSOS: cartas, cartao de projeto, artefatos, tabuleiro central, tabuleiro individual

System Under Design: SimulES
Primary Actor: jogador da vez
Participants: engenheiro de software
Goal: Descrever as etapas de inspecao de artefatos
Follows Use Cases:
Invariant:
Precondition: jogador da vez tem artefatos no tabuleiro AND LOCALIZACAO GEOGRAFICA is sala de jogo AND LOCALIZACAO TEMPORAL is durante rodada de acoes
STEPS
1.Jogador da vez escolhe artefato que sera inspecionado.// Qualidade [projeto]
2.Engenheiro de software inspeciona artefato. // Qualidade [artefato]
3.Jogador da vez desvira artefato.
ALTERNATIVES
1.a. If Engenheiro de software inspeciona artefato de outro engenheiro
1.a.1. Engenheiro de software gasta dois pontos de tempo na inspecao.
1.a.2. Goto Step 1.
Success Postcondition:

Figura 4.56 – Cenário Inspeção de Artefato (transformado).

Title: Corrigir Artefato
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: durante rodada de acoes

RECURSOS: cartas, cartao de projeto, artefatos, tabuleiro individual, tabuleiro central

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, engenheiro de software
Goal: Descrever as etapas da correcao de artefatos
Follows Use Cases:
Invariant:
Precondition: jogador da vez tem artefatos defeituosos no tabuleiro AND LOCALIZACAO GEOGRAFICA is sala de jogo AND LOCALIZACAO TEMPORAL is durante rodada de acoes
STEPS
1.SimulES disponibiliza recursos.
2.Jogador da vez escolhe engenheiro que tera artefato corrigido. // Qualidade[projeto]
3.Jogador da vez escolhe artefato defeituoso.
4.Engenheiro de software corrige artefato. //Qualidade[artefato]
5.Jogador da vez substitui artefato corrigido por outro da mesma cor
ALTERNATIVES
4.a. Engenheiro de software corrige artefato de outro engenheiro
4.a.1. Engenheiro de software gasta dois pontos de tempo.
4.a.2. Goto Step 4.
Success Postcondition:

Figura 4.57 – Cenário Correção de Artefato (transformado).

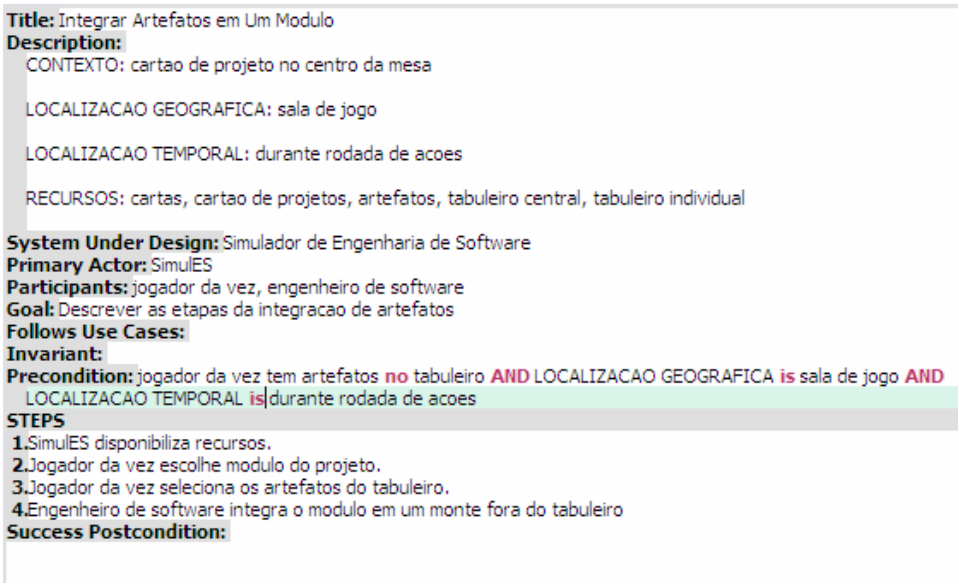


Figura 4.58 – Cenário Integração de Artefato em Módulo (transformado).

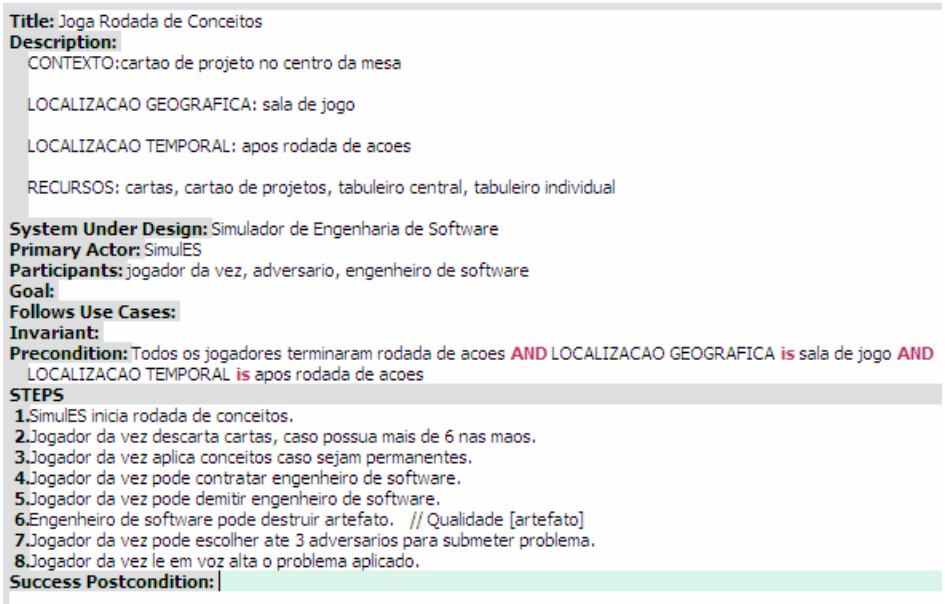


Figura 4.59 – Cenário Joga Rodada de Conceitos (transformado).

Title: Tratamento de Problema
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: apos rodada de conceitos

RECURSOS: cartas, cartao de projeto, tabuleiro central, tabuleiro individual

System Under Design: Simulador de Engenharia de Software
Primary Actor: Jogador da vez
Participants: adversario
Goal:
Follows Use Cases:
Invariant:
Precondition: Adversario recebe problema de jogador da vez **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** apos rodada de conceitos
STEPS
1.Jogador da vez aplica problema.
2.Adversario atende a demanda do problema.
3.Adversario pode aplicar conceito.
4.Adversario mantem problema permanente.
5.Adversario descarta problema temporario.
Success Postcondition:

Figura 4.60 – Cenário Tratamento de Problema (transformado).

Title: Submissao de Produto
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: apos rodada de conceitos

RECURSOS: cartas, cartao de projeto, modulos

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, adversario, engenheiro de software
Goal: descrever as etapas de submissao de produto
Follows Use Cases:
Invariant:
Precondition: jogador da vez integrou ultimo modulo **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** apos rodada de conceitos
STEPS
1.SimulES disponibiliza recursos.
2.Jogador da vez disponibiliza seus modulos produzidos.
3.Adversario escolhe modulos.
4.Adversario inspeciona modulos.
5.Engenheiro de software corrige um artefato por turno, se necessario.
6.Jogador da vez vence o jogo.
ALTERNATIVES
3.a.IF Modulo esta protegido por carta de conceito
3.a.1. Adversario escolhe outro modulo para inspecionar.
3.a.2. Goto Step 2.
Success Postcondition:

Figura 4.61 – Cenário Submissão de Produto (transformado).

Ao término desta etapa, o engenheiro de requisitos estará pronto para iniciar a etapa de validação utilizando a ferramenta de simulação *UCEd*.

4.4. Validar por Simulação

Esta atividade visa validar os modelos i* propostos pelo engenheiro de requisitos. Para isso, foi realizada a simulação no laboratório do *Semiotic Engineering Research Group* – SERG, da PUC–Rio. Contou–se ainda com a participação de três voluntários com conhecimento do domínio do problema, fazendo o papel dos interessados.

O SimulES foi descrito através de nove situações de dependência estratégica, representadas pelos seguintes cenários: (C1) Joga Rodada de Início; (C2) Joga Rodada de Ações; (C3) Construção de Artefatos; (C4) Inspeção de Artefatos; (C5) Correção de Artefatos; (C6) Integração de Artefatos em Módulo; (C7) Joga Rodada de Conceitos; (C8) Tratamento de Problemas e (C9) Submissão de Produtos. Como mencionado anteriormente, a realização da validação foi centrada no léxico, na ordem dos episódios e na completude e corretude dos elementos que formam os cenários.

A simulação foi realizada individualmente com os interessados em um ambiente controlado no SERG. A ferramenta de simulação *UCEd* foi utilizada em conjunto com um software de captura de tela e áudio. Cabe ressaltar que a validação foi mediada pelo engenheiro de requisitos, onde o mesmo procurou sempre não influenciar a opinião dos interessados.

A análise dos resultados leva em consideração as alterações realizadas pelos interessados nos cenários validados. Considera–se alteração qualquer uma das seguintes alternativas:

- Criação de um novo episódio
- Correção de um episódio existente
- Ordenação entre episódios (sem envolver mudança no léxico)
- Eliminação de algum episódio
- Fusão entre episódios

Os gráficos a seguir representam as alterações encontradas pelos participantes do estudo durante a simulação:

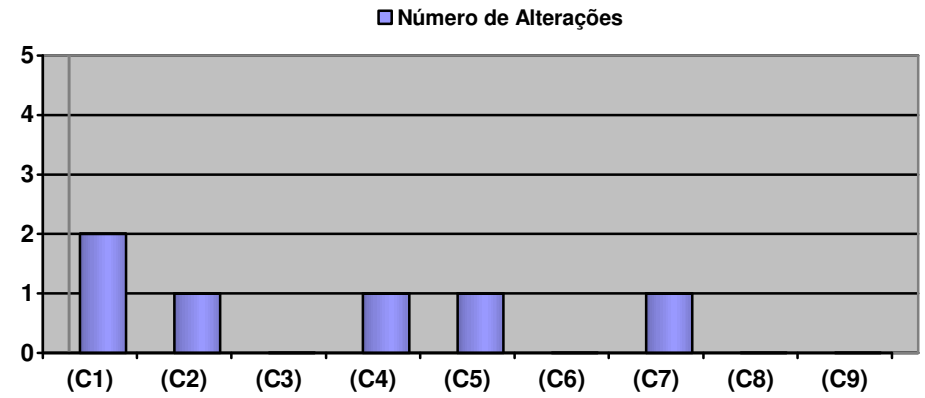


Figura 4.62 – Alterações realizadas pelo Interessado 1 durante a simulação.

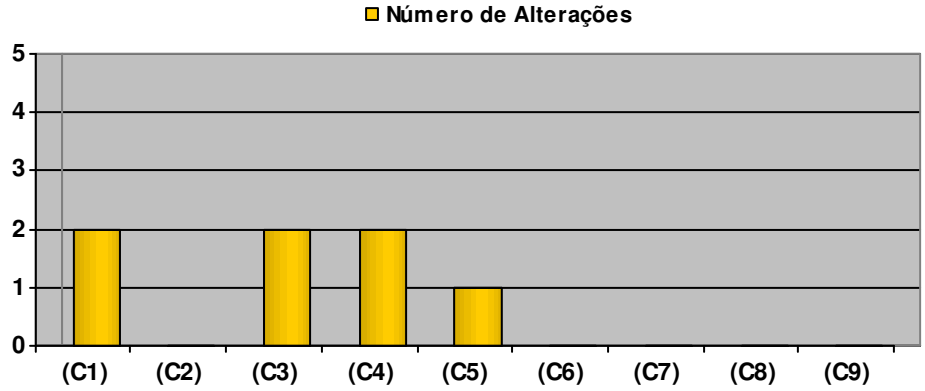


Figura 4.63 – Alterações realizadas pelo Interessado 2 durante a simulação.

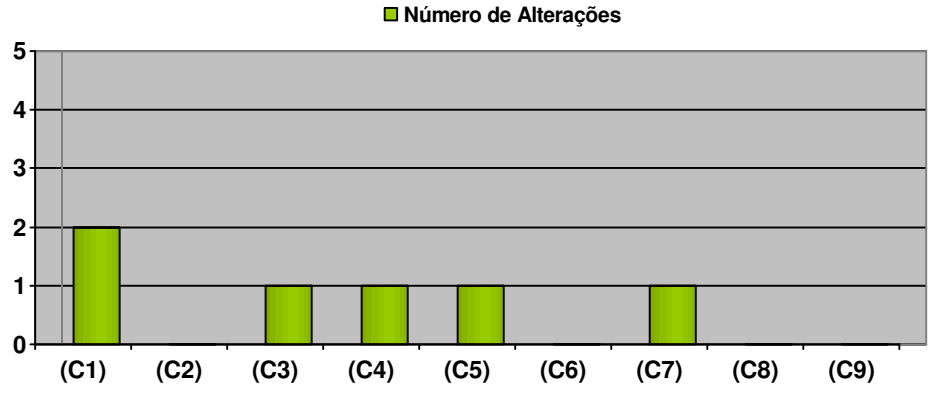


Figura 4.64 – Alterações realizadas pelo Interessado 3 durante a simulação.

Combinando os gráficos individuais supracitados obtemos o gráfico resultante com todas as alterações realizadas em cada cenário:

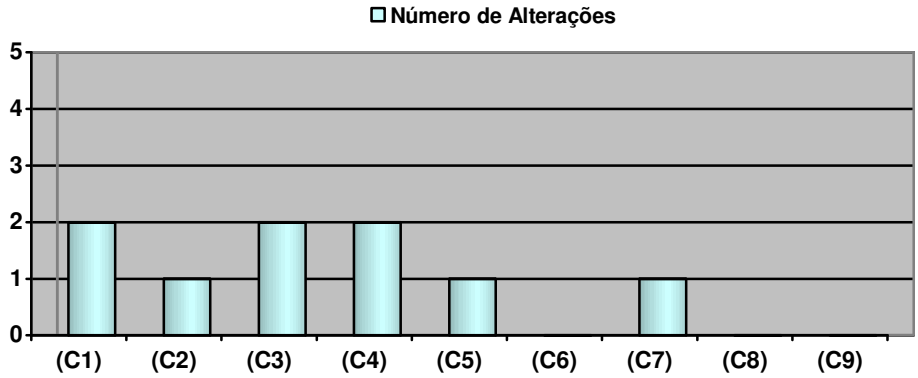


Figura 4.65 – Alterações resultantes realizadas pelos Interessados.

Durante a utilização da ferramenta de simulação, não há uma ordenação dos episódios definida, ou seja, o interessado é de certa forma obrigado a propor uma ordenação que será confrontada com o raciocínio inicial do engenheiro de requisitos. Neste caso, o interessado age ativamente na validação dos episódios, colocando–se em problemas e situações que não foram percebidas durante construção dos modelos.

Com isso, os interessados tornam–se os principais responsáveis pela elaboração dos cenários. Além disso, através das gravações, o engenheiro de requisitos consegue captar mínimos detalhes durante a validação, permitindo uma análise mais rica e detalhada.

A seção a seguir irá tratar as discrepâncias, os erros ou omissões obtidas através do *feedback* dos interessados, presentes nos gráficos supracitados, durante a etapa de validação.

4.4.1.Tratando os Resultados Obtidos

Durante a etapa de validação com os três interessados, foram encontrados todos os tipos de alterações citadas anteriormente (criação, reordenação, correção, eliminação e fusão de episódios).

Pode–se perceber através dos gráficos de alterações dos interessados presentes na seção anterior que os cenários (C6) Integração de Artefatos em Módulo, (C8) Tratamento de Problema e (C9) Submissão de Produto não sofreram quaisquer tipo de alteração durante a validação. Porém, isso não quer dizer que tais cenários não serão afetados, pois pode ocorrer de novos símbolos

criados impactarem estes cenários de alguma forma. Portanto, todos os cenários serão analisados à luz da simulação realizada.

As alterações do tipo **eliminação** e **reordenação** foram estudadas inicialmente, e percebemos que as mesmas não apresentaram alterações ao léxico, base de nossa estratégia. Baseando–se na heurística (4) de validação, observamos que todos os símbolos pertencentes aos episódios eliminados possuíam redundância no léxico, originários de outros impactos determinados na primeira fase da estratégia (Construir Modelo i*). Podemos assim perceber que ao retirarmos "Jogador da vez inicia jogo" e "Adversário inicia jogo" do cenário *Joga Rodada de Início* não iremos impactar os modelos construídos.

Com relação à **reordenação** dos episódios, foi observado que nada foi mudado com relação ao léxico, de acordo com a heurística (2) de validação. A simples troca de episódio não afetou o léxico nem a intencionalidade dos atores participantes da *SDsituation* correspondente. A ordem final dos episódios será exibida posteriormente.

Correções nos episódios foram sugeridas pelos interessados 1 e 3. De acordo com a heurística (3) de validação, deve–se observar durante as correções a ocorrência de novos símbolos no léxico. Durante a validação do SimulES, as correções solicitadas foram mudar "Engenheiro de software gasta dois pontos de tempo na inspeção" para "Engenheiro de software gasta dois pontos de tempo na inspeção de artefatos de outros engenheiros". O mesmo ocorreu para o episódio que envolvia correção de artefatos. A última alteração solicitada foi de "Jogador da vez descarta cartas caso possua mais de seis nas mãos" para "Jogador da vez descarta cartas excedentes caso possua mais de seis cartas nas mãos". Com base nestes dados, podemos concluir que não houve impacto no léxico e portanto, os modelos não foram afetados pelas correções.

O segundo interessado foi o único a sugerir a **fusão** de alguns episódios. Seguindo a heurística (5) de validação, devemos determinar se a fusão destes episódios ocasionou a criação ou eliminação de algum símbolo do léxico. Durante a validação, o interessado 2 entendeu que "colocar artefatos no tabuleiro" é parte de "Engenheiro de software constrói artefato", não sendo necessário, portanto, a separação dos episódios. O mesmo ocorre com "desvira artefato", que faria parte de "Engenheiro de software inspeciona artefato". Com base nesses episódios,

pudemos perceber que a fusão não adiciona ou elimina símbolos do léxico, o que indica que neste caso não é necessário alterar os modelos.

Por fim, iremos analisar as **criações ou inserções** de episódios nos cenários. Durante a simulação com o primeiro interessado, ocorreu a criação do léxico "partida" e a criação do episódio "Jogador da vez analisa as cartas em suas mãos". Obedecendo a heurística (1) da validação, devemos estudar a relação do novo símbolo "partida" no léxico atual.

Observou–se que "partida" é sinônimo do símbolo "jogo" no léxico. Ou seja, podemos considerar a mesma noção e impactos para ambos os símbolos. Sendo assim, poderíamos ter "Jogador vence partida" no lugar de "Jogador da vez vence o jogo". Continuando o raciocínio, a meta "Jogo seja iniciado" na situação de dependência estratégica *Joga Rodada de Início*, poderia ser substituída por "Partida seja iniciada", ficando a cargo dos interessados escolherem o termo mais adequado.

Os interessados 2 e 3 solicitaram a criação de dois episódios. "Engenheiro de software gasta dois pontos de tempo na inspeção" e "Engenheiro de software gasta dois pontos de tempo na correção" para os seus respectivos cenários (*Inspeção de Artefato* e *Correção de Artefato*). Essa solicitação é amparada no fato de atribuir melhor entendimento desses cenários, pois inicialmente estava descrita somente a quantidade de pontos gastos na inspeção/correção de artefatos produzidos por outros engenheiros. Cabe ressaltar que a criação de ambos episódios não afeta de forma alguma o léxico do SimulES, e consequentemente, os modelos produzidos.

Em sequência, temos a criação do episódio "Jogador da vez analisa as cartas em suas mãos" durante a simulação do cenário *Joga Rodada de Conceitos*. Podemos observar que o episódio supracitado faz parte dos impactos do sujeito jogador da vez, e deixou de ser elicitado pelo engenheiro de software por algum motivo.

Cabe ressaltar que, como foi definido anteriormente, analisar é um verbo pouco preciso, sendo assim uma ação flexível (como visto na seção 1.3.3). Pelo método Eri*c, ações flexíveis dão origem a metas flexíveis do modelo, o que nos leva a concluir que foi esquecida uma meta flexível em algum (ou alguns) modelo(s) propostos.

Com base neste raciocínio, observou–se que uma meta flexível relacionada a estratégia do jogador deveria fazer parte do modelo. A meta flexível Boa[estratégia] deve ser inicialmente inserida em seu respectivo *template* (meta flexível para impactos de símbolos do tipo sujeito).

	<meta flexível>		
-- impacto resposta ao por quê?	**<TIPO atributo de qualidade**	**[TOPICO]> sujeito/objeto LAL**	**<meta concreta associada> <ator>**
-- Jogador da vez analisa as cartas em suas mãos	**ação flexível**		
Porque	**Boa**	**[estratégia]**	**Rodada de conceitos seja jogada Jogador da vez**

Figura 4.66 – *Template* da meta flexível Boa [estratégia] (parcial).

Continuando a análise, de acordo com a heurística (1) de validação, devemos observar se a meta flexível em questão está relacionada a algum outro impacto, de acordo com as metas elicitadas anteriormente. Com isso, apresenta–se os demais símbolos relacionados à referida meta flexível na Figura 4.66:

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
APLICAR CONCEITO			
-- jogador da vez neutraliza problema	ação concreta		Conceito seja aplicado Jogador da vez
-- jogador da vez melhora seu jogo			
Porque	Boa	[estratégia]	Rodada de conceitos seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
APLICAR PROBLEMA			
-- adversário recebe problema	ação concreta		Problema seja aplicado Jogador da vez
-- adversário trata problema			
Porque	Boa	[estratégia]	Problema seja aplicado Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
INSPECIONAR ARTEFATO			
-- jogador da vez conhece o artefato inspecionado			
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
Porque	Boa	[estratégia]	Artefato seja corrigido Engenheiro de Software
-- jogador da vez pode corrigir artefato caso seja defeituoso			
Porque	qualidade	[artefato]	Artefato seja inspecionado Engenheiro de Software
Porque	Boa	[estratégia]	Artefato seja corrigido Engenheiro de Software

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
CORRIGIR ARTEFATO			
-- jogador da vez possui um novo artefato em seu tabuleiro.	ação concreta		Artefato seja corrigido Engenheiro de Software
Porque	qualidade	[artefato]	Artefato seja corrigido Engenheiro de Software
Porque	Boa	[estratégia]	Artefato seja corrigido Engenheiro de Software

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
DEMITIR ENGENHEIRO DE SOFTWARE			
-- Jogador da vez pode usar orçamento novamente para a contratação de um novo engenheiro			
Porque	Boa	[estratégia]	Rodada de conceitos seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
JOGAR RODADA DE AÇÕES			
-- simules inicia rodada de ações	ação concreta		Rodada de ações seja iniciada por SimulES
-- jogador da vez joga rodada de ações	ação concreta		Projeto seja concluído Jogador da vez
Porque	Boa	[estratégia]	Rodada de conceitos seja jogada Jogador da vez
-- jogador da vez obtém resultado do lançamento do dado	ação concreta		Rodada de conceitos seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
RECEBER PROBLEMA			
jogador da vez trata problema			
Porque	Boa	[estratégia]	Rodada de conceitos seja jogada Jogador da vez

TIPO: VERBO	<meta flexível>		
— impacto resposta ao por quê?	<TIPO atributo de qualidade	[TOPICO]> sujeito/objeto LAL	<meta concreta associada> <ator>
TRATAR PROBLEMA			
-- jogador da vez pode contrapor problema aplicado			
Porque	Boa	[estratégia]	Conceito seja aplicado Jogador da vez
-- jogador da vez atende demanda do problema			
Porque	Boa	[estratégia]	Problema seja tratado Jogador da vez

Figura 4.67 – Símbolos impactados pela meta flexível Boa [estratégia].

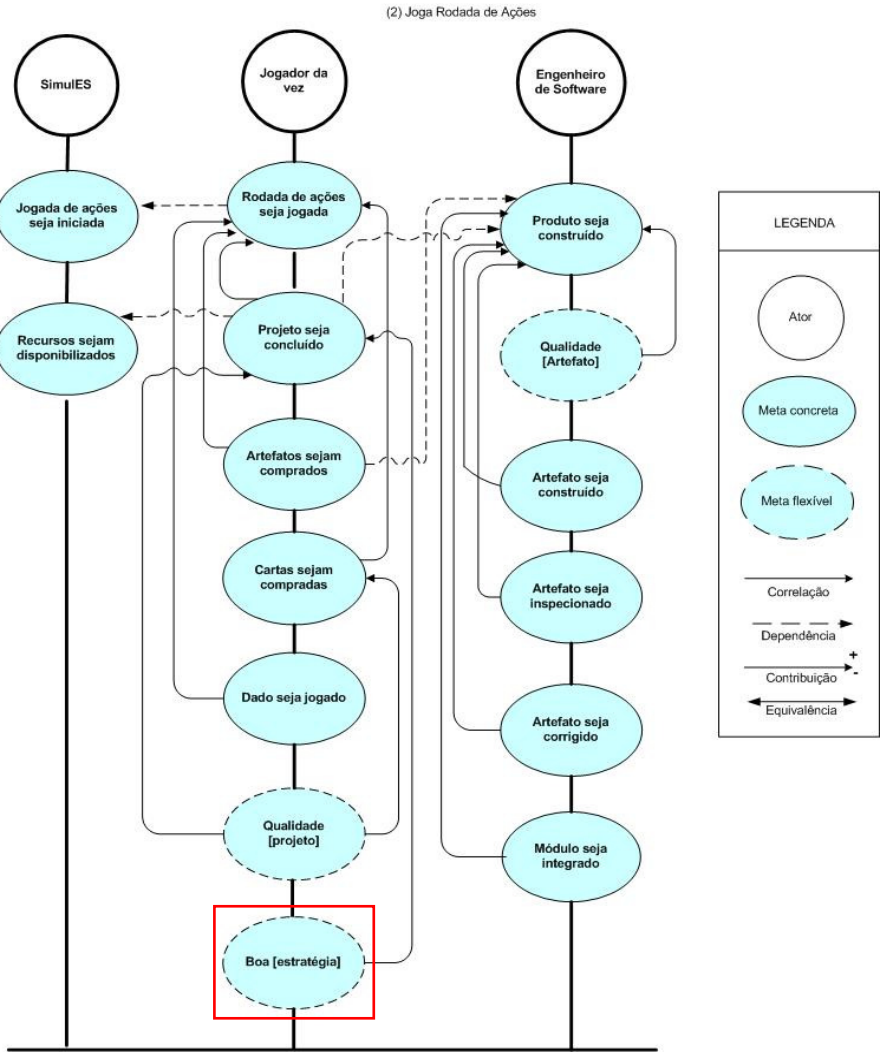
Após definir as metas impactadas por Boa [estratégia], de acordo com a seção 3.2.1.3 deste trabalho, torna–se necessário refiná–las e organizá–las novamente por situações de dependência estratégica para facilitar o entendimento do engenheiro de requisitos. A Figura a seguir apresenta a nova tabela para o Jogador da vez, já com a referida meta flexível incluída (destacada em azul).

DEPENDER	SITU					DEPENDEE
JOGADOR DA VEZ						
	1	jogo	seja	iniciado		
	1	projeto	seja	escolhido		
	2	jogada de ações	seja	iniciada	Por	**SimulES**
	2	rodada de ações	seja	jogada		
	7	rodada de conceitos	Seja	iniciada	Por	**SimulES**
	7	rodada de conceitos	seja	jogada		
	2,3,4,5	projeto	seja	concluído		
	7	jogo	seja	disponibilizado		
	1,2	dado	seja	jogado		
	3	artefato	seja	construído	por	**engenheiro de software**
	4	artefato	seja	inspecionado	por	**engenheiro de software**
	5,9	artefato	seja	corrigido	por	**engenheiro de software**
	7,8	cartas	sejam	descartadas		
	7,8	engenheiro de software	seja	demitido		
	7	engenheiro de software	seja	descartado		
	1,7	engenheiro de software	seja	contratado		
qualidade [artefato]	2,4,5,7	artefato	seja	livre de defeitos		
qualidade [projeto]	2,4,5,7	projeto	seja	concluído	por	**Engenheiro de software**
	9	jogo	seja	vencido		
	3	habilidade do engenheiro	Seja	analisada		
	4,5	engenheiro de software	Seja	escolhido		
	7	artefato	seja	destruído	por	**engenheiro de software**
	7	problema	seja	aplicado		
	7,8	conceito	seja	aplicado		
	9	produto	seja	submetido		
	9	modulo	seja	escolhido	Por	**adversário**
	9	modulo	seja	inspecionado	Por	**adversário**
Boa [estratégia]	**2**	**rodada de ações**	**seja**	**jogada**		
Boa [estratégia]	**4**	**artefato**	**seja**	**inspecionado**		
Boa [estratégia]	**5**	**artefato**	**seja**	**corrigido**		
Boa [estratégia]	**7**	**rodada de conceitos**	**seja**	**jogada**		
Boa [estratégia]	**8**	**problema**	**seja**	**tratado**		

Figura 4.68. Metas refinadas com a adição da meta flexível Boa [estratégia].

Uma vez determinado em que situações de dependência estratégica Boa[estratégia] se encontra, devemos refazer os painéis de intencionalidade dessas situações, a fim de incluir a meta flexível descoberta.

As Figuras a seguir exibem os novos painéis de intencionalidade (diagramas IP) com a meta flexível Boa [estratégia] inserida. Usaremos retângulos na cor vermelha para dar maior destaque à meta flexível inserida.



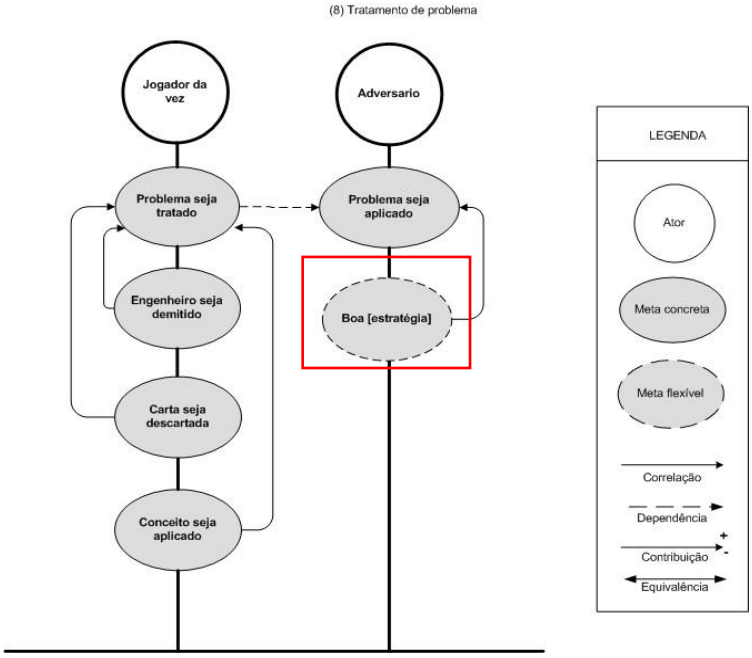
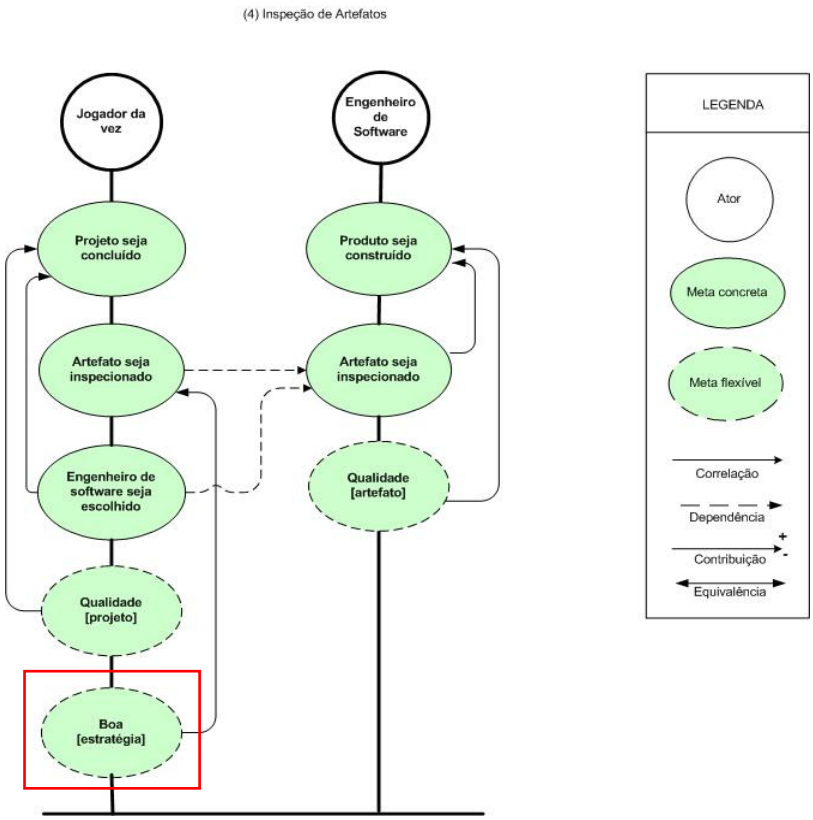


Figura 4.69. Diagramas IP Joga Rodada de Ações e Tratamento de Problema com a adição da meta flexível Boa [estratégia].



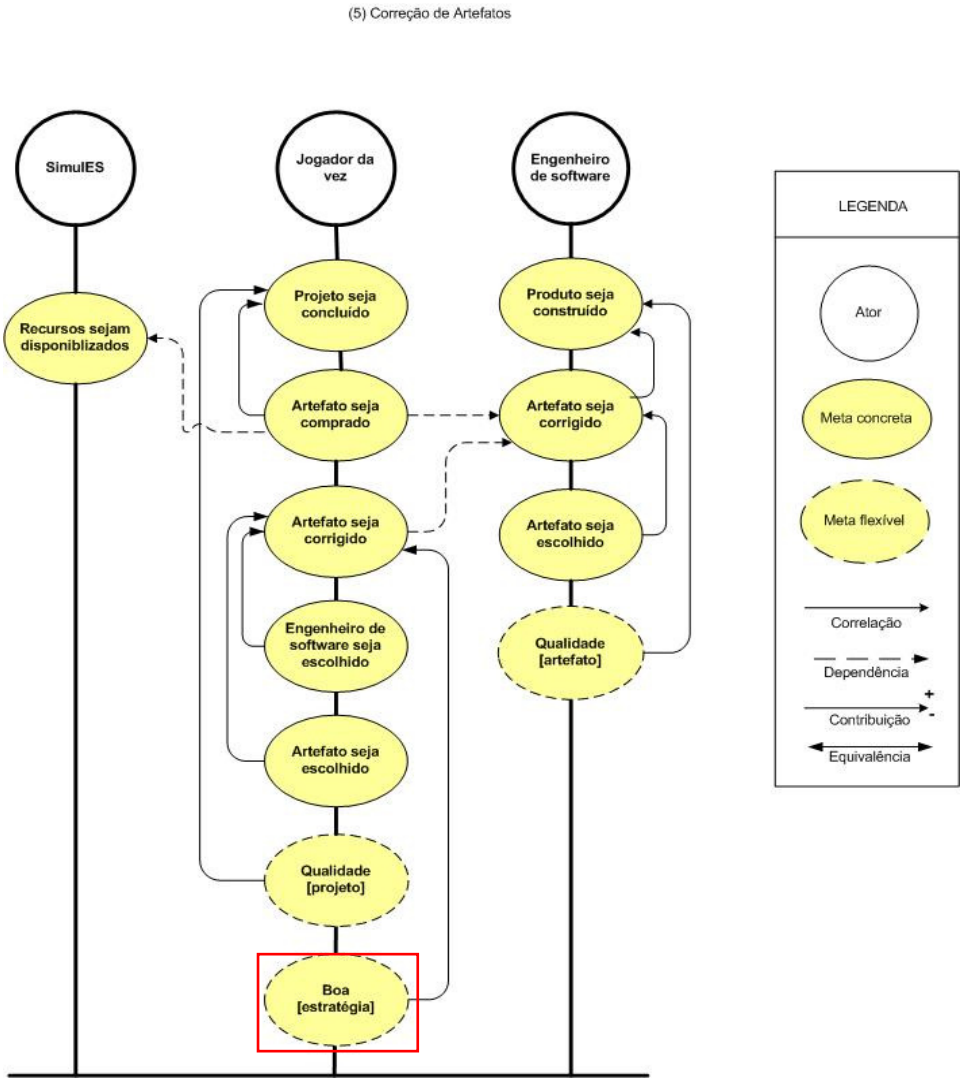


Figura 4.70. Diagramas IP Inspeção de Artefato e Correção de Artefato com a adição da meta flexível Boa [estratégia].

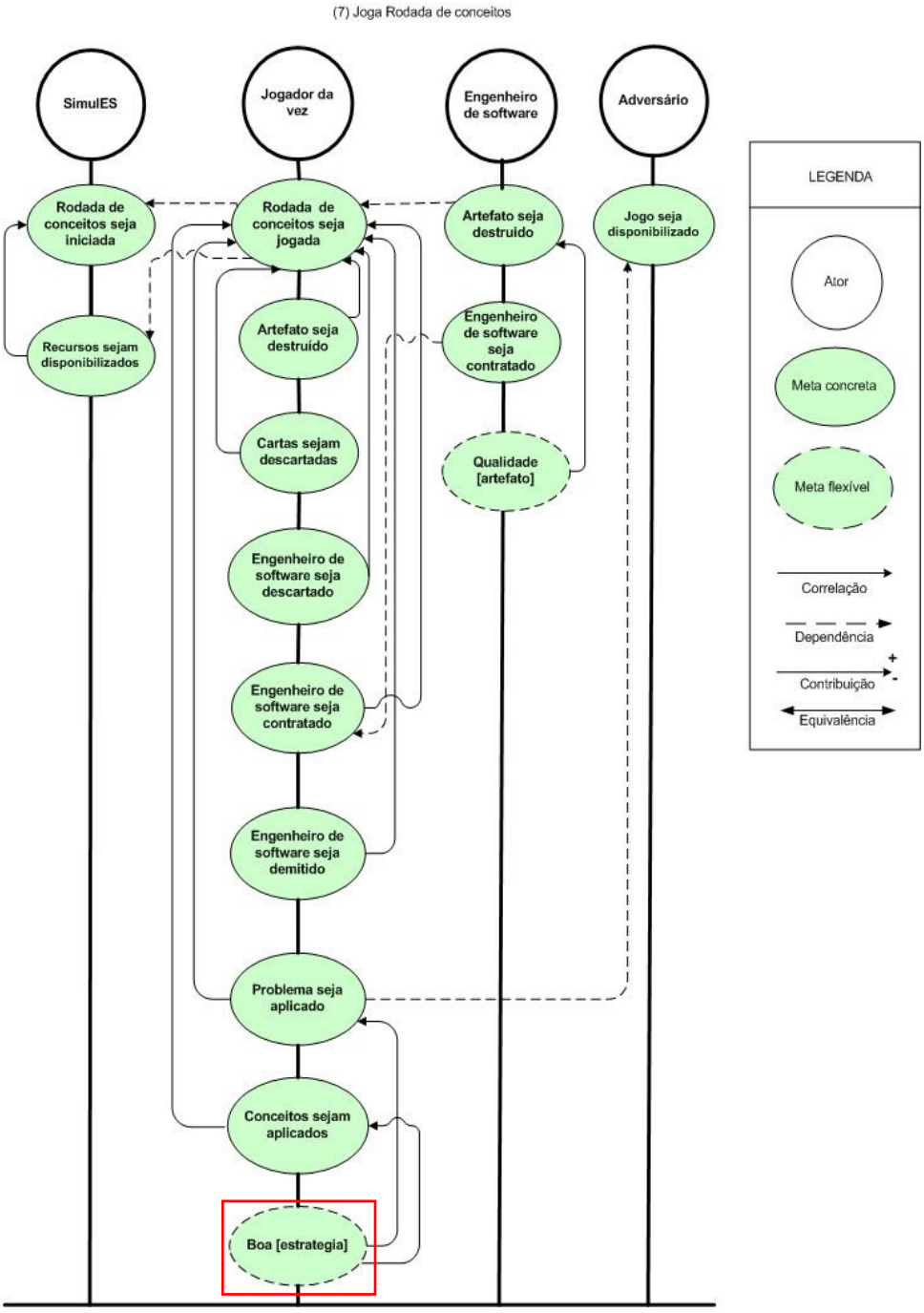


Figura 4.71. Diagramas IP Joga Rodada de Conceitos com a adição da meta flexível Boa [estratégia].

Podemos perceber nos diagramas IP apresentados anteriormente que a inclusão da meta flexível Boa [estratégia] não causou o aparecimento de relacionamentos de dependência, apenas de relacionamentos de correlação. Isso quer dizer que não irá surgir nenhum elemento de dependência alterando o diagrama SD proposto.

Porém, a referida meta flexível irá impactar metas nos cinco modelos SR das situações de dependência estratégica supracitadas. Em sequência, serão apresentados os diagramas SR que foram alterados pela inserção de Boa[estratégia]. As alterações foram destacadas utilizando retângulos vermelhos.

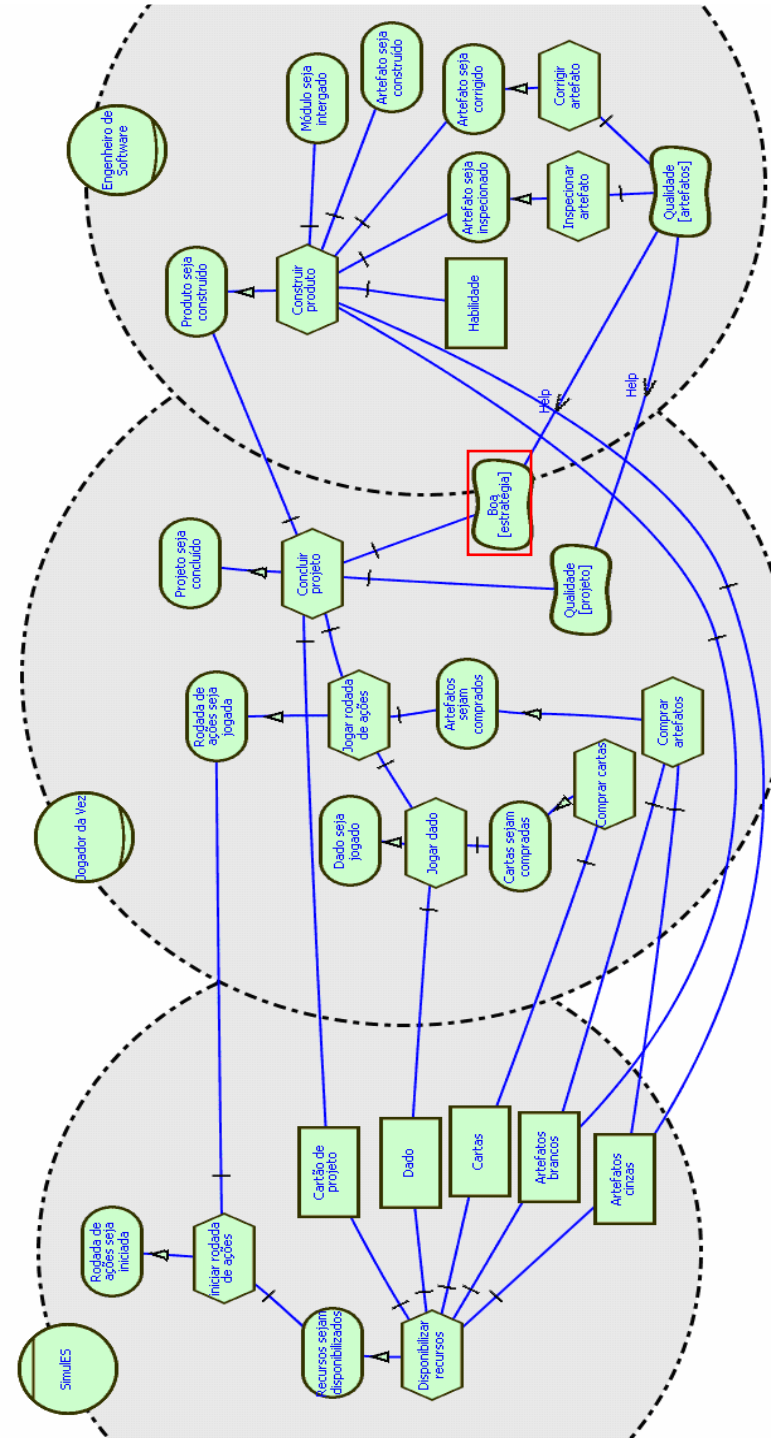


Figura 4.72. Diagrama SR Joga Rodada de Ações modificado.

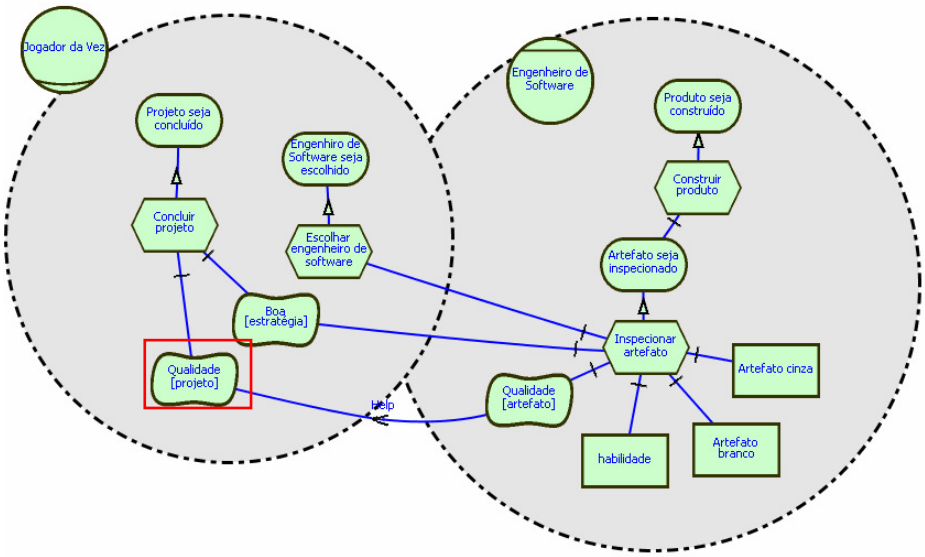


Figura 4.73. Diagrama SR Inspeção de Artefato modificado.

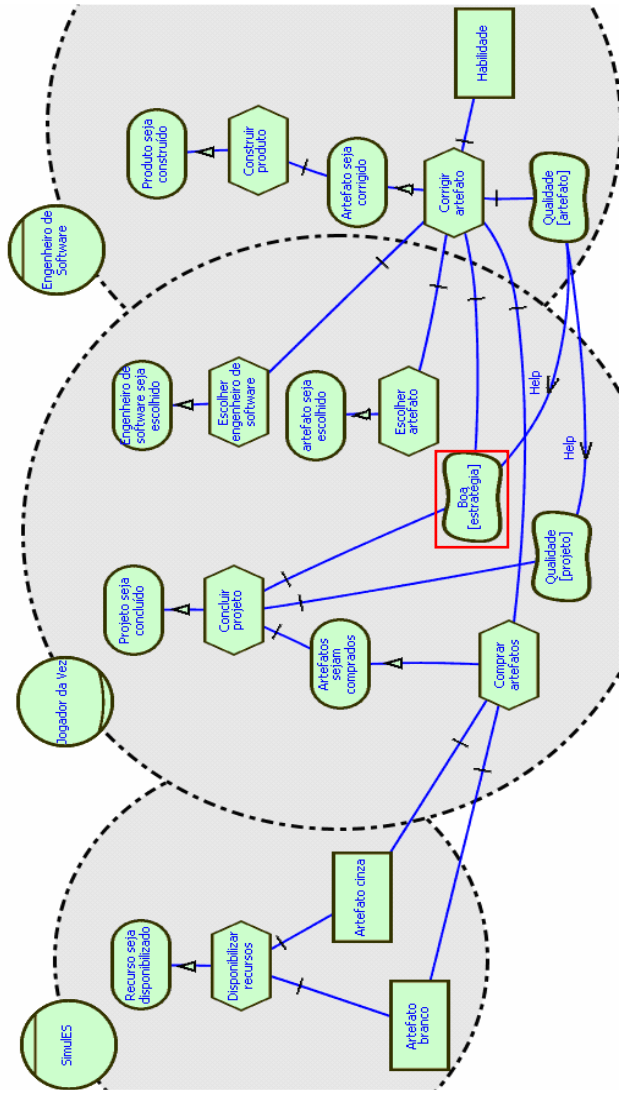


Figura 4.74. Diagrama SR Correção de Artefato modificado.

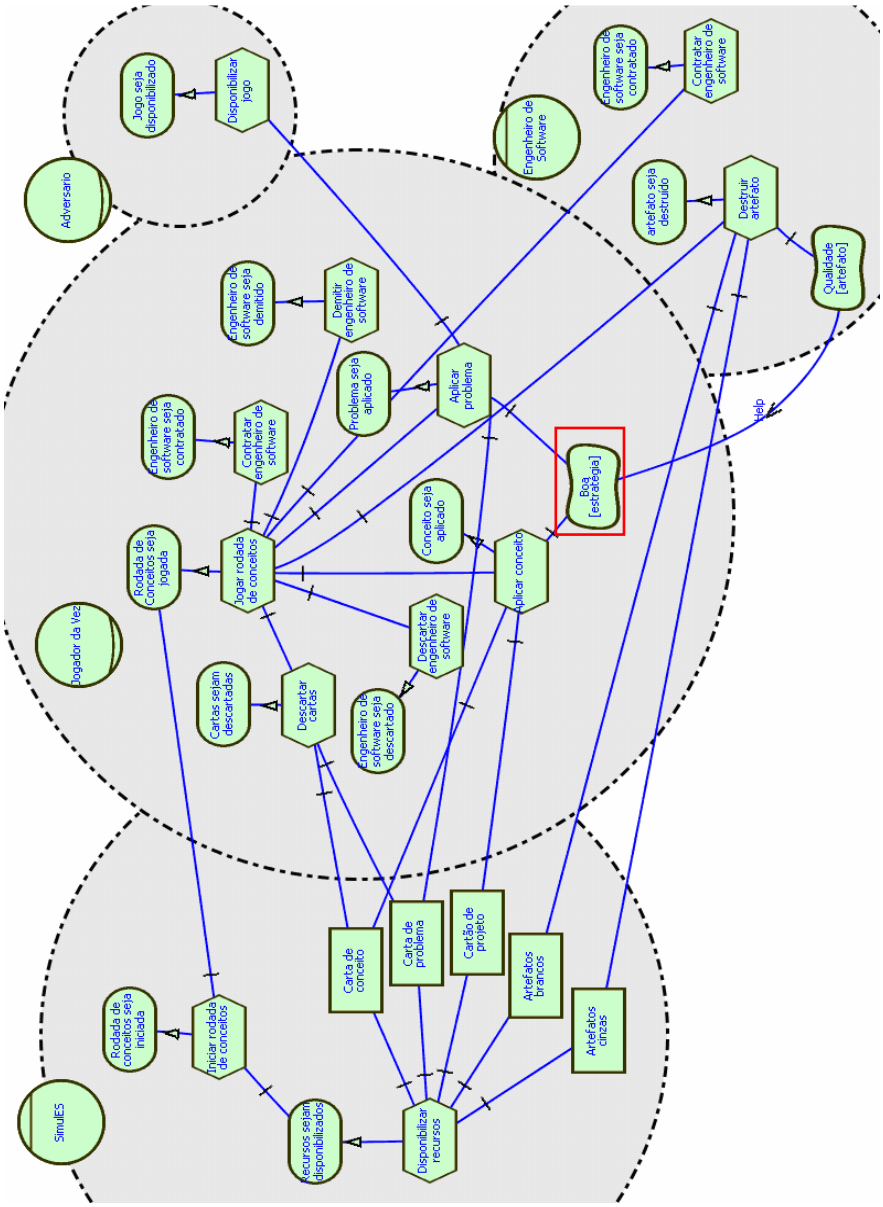


Figura 4.75. Diagrama SR Joga Rodada de Conceitos modificado.

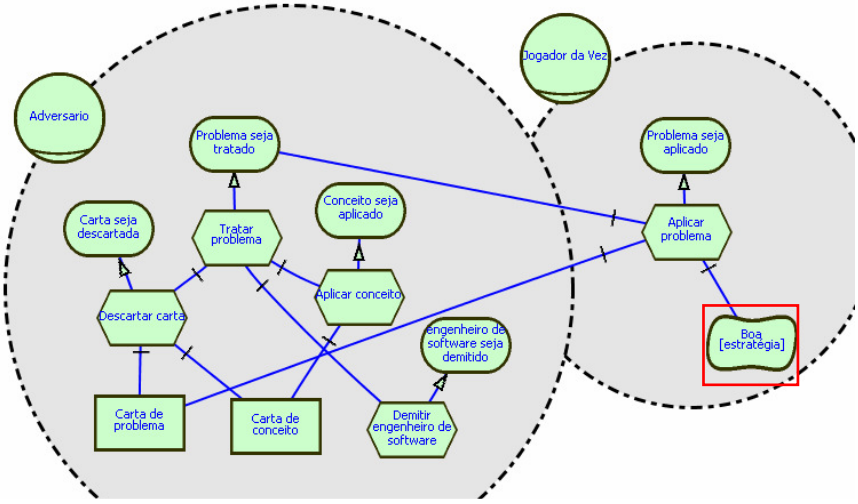


Figura 4.76. Diagrama SR Tratamento de Problema modificado.

O passo seguinte será descrever as situações de dependência estratégica que sofreram alterações durante o processo de validação. Cabe ressaltar que as *SDsituations Integração de Artefatos em Módulo* e *Submissão de Produto* não sofreram alterações. Serão exibidos os cenários antes das correções e após as correções, para que o resultado da simulação realizada fique mais claro.

Titulo:	**Joga Rodada de Início (antes da correção)**
Objetivo:	Jogo seja iniciado
Contexto:	Tabuleiro individual na mesa, cartas embaralhadas sobre a mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Início de jogo
Restrição:	
Pré-condição:	O sentido do jogo é horário
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, adversário, engenheiro de software, SimulES
Episódios:	1– SimulES inicia rodada de início.
	2– Jogador da vez lança o dado. **Restrição:** Jogador que tirar o maior número no dado inicia o jogo.
	3– SimulES determina jogador da vez.
	4– Jogador da vez escolhe projeto.
	5– Adversário concorda com o projeto.
	6– Jogador da vez contrata engenheiro de software.
Exceções:	

Titulo:	**Joga Rodada de Início (depois da correção)**
Objetivo:	Jogo seja iniciado
Contexto:	Tabuleiro individual na mesa, cartas embaralhadas sobre a mesa e
Localização Geográfica:	Sala de jogo
Localização Temporal:	Início de jogo
Restrição:	
Pré-condição:	O sentido do jogo é horário
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, adversário, engenheiro de software, SimulES
Episódios:	1– SimulES inicia rodada de início.
	2– Jogador da vez lança o dado. **Restrição:** Jogador que tirar o maior número no dado inicia o jogo.
	3– Jogador da vez escolhe projeto.
	4– Adversário concorda com o projeto.
	5– Jogador da vez contrata engenheiro de software.
Exceções:	

Figura 4.77 – Descrição da SDsituation Joga Rodada de Início antes e depois da correção.

Titulo:	**Joga Rodada de Ações (antes da correção)**
Objetivo:	Rodada de ações seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Antes da rodada de conceitos
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES inicia rodada de ações.
	2– Jogador da vez lança dado.
	3– Se o resultado do lançamento do dado for igual a 1, 2 ou 3, jogador da vez compra 1,2 ou 3 cartas do monte de problemas e conceitos.
	4– Se o resultado do lançamento do dado for igual a 4, 5 ou 6, jogador da vez compra 3 cartas do monte de problemas e conceitos e a diferença de engenheiros de software.
	5– Engenheiro de software pode construir, inspecionar ou corrigir artefato.
	6– Engenheiro de software pode integrar artefatos em um módulo.
Exceções:	

Titulo:	**Joga Rodada de Ações (depois da correção)**
Objetivo:	Rodada de ações seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Antes da rodada de conceitos
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido
Recursos:	Dado, cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES inicia rodada de ações.
	2– Jogador da vez lança dado.
	3– Se o resultado do lançamento do dado for igual a 1, 2 ou 3, jogador da vez compra 1,2 ou 3 cartas do monte de problemas e conceitos.
	4– Se o resultado do lançamento do dado for igual a 4, 5 ou 6, jogador da vez compra 3 cartas do monte de problemas e conceitos e a diferença de engenheiros de software.
	5– Engenheiro de software pode construir, inspecionar ou corrigir artefato.
	6– Engenheiro de software pode integrar artefatos em um módulo.
Restrição:	RNF: Qualidade [artefato], Qualidade [artefato], Boa[estratégia]
Exceções:	

Figura 4.78 – Descrição da SDsituation Joga Rodada de Ações antes e depois da correção.

Titulo:	**Construção de Artefato (antes da correção)**
Objetivo:	Produto seja construído
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez possui engenheiro de software
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez, para cada engenheiro de software, compra artefatos.
	3– Engenheiro de software constrói artefatos.
	4– Jogador da vez coloca artefatos no tabuleiro.
Exceções:	

Titulo:	**Construção de Artefato (depois da correção)**
Objetivo:	Produto seja construído
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez possui engenheiro de software
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez, para cada engenheiro de software, compra artefatos.
	3– Engenheiro de software constrói artefatos.
Exceções:	

Figura 4.79 – Descrição da SDsituation Construção de Artefato antes e depois da correção.

Titulo:	**Inspeção de Artefato (antes da correção)**
Objetivo:	Artefatos sejam inspecionados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, engenheiro de software
Episódios:	1– Jogador da vez escolhe artefato do que será inspecionado.
	2– Engenheiro de software inspeciona artefato. **Restrição:** Engenheiro de software gasta dois pontos de tempo para inspecionar artefatos de outro engenheiro.
	3– Jogador da vez desvira artefatos.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto]
Exceções:	

Titulo:	**Inspeção de Artefato (depois da correção)**
Objetivo:	Artefatos sejam inspecionados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, engenheiro de software
Episódios:	1– Jogador da vez escolhe artefato do que será inspecionado.
	2– Engenheiro de software inspeciona artefato.
	3– Engenheiro de software gasta 1 ponto de tempo na inspeção. **Restrição:** Engenheiro de software gasta dois pontos de tempo para inspecionar artefatos de outro engenheiro.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto], Boa[estratégia]
Exceções:	

Figura 4.80 – Descrição da SDsituation Inspeção de Artefato antes e depois da correção.

Titulo:	**Correção de Artefato (antes da correção)**
Objetivo:	Artefatos sejam corrigidos
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos defeituosos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez escolhe engenheiro que terá artefato corrigido.
	3– Jogador da vez escolhe artefato defeituoso.
	4– Engenheiro de software corrige artefato. **Restrição:** Engenheiro de software gasta dois pontos de tempo para corrigir artefatos de outro engenheiro.
	5– Jogador da vez substitui artefato corrigido por outro da mesma cor.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto]
Exceções:	

Titulo:	**Correção de Artefato (depois da correção)**
Objetivo:	Artefatos sejam corrigidos
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Durante a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez tem artefatos defeituosos no tabuleiro
Recursos:	cartas, cartão de projeto, artefatos, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software
Episódios:	1– SimulES disponibiliza recursos.
	2– Jogador da vez escolhe engenheiro que terá artefato corrigido.
	3– Jogador da vez escolhe artefato defeituoso.
	4– Engenheiro de software corrige artefato. **Restrição:** Engenheiro de software gasta dois pontos de tempo para corrigir artefatos de outro engenheiro.
	5– Engenheiro de software gasta 1 ponto de tempo na correção. **Restrição:** Engenheiro de software gasta dois pontos de tempo para corrigir artefatos de outro engenheiro.
	6– Jogador da vez substitui artefato corrigido por outro da mesma cor.
Restrição:	RNF: Qualidade [artefato], Qualidade [projeto]. Boa[estratégia]
Exceções:	

Figura 4.81 – Descrição da SDsituation Correção de Artefato antes e depois da correção.

Titulo:	**Joga Rodada de Conceitos (antes da correção)**
Objetivo:	Rodada de conceitos e problemas seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido Todos os jogadores terminaram a rodada de ações
Recursos:	cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software, adversário
Episódios:	1– SimulES inicia rodada de conceitos.
	2– Jogador da vez descarta cartas, caso possua mais de 6 cartas nas mãos.
	3– Jogador da vez aplica conceitos caso sejam permanentes. **Restrição:** não pode exceder o orçamento do projeto.
	4– Jogador da vez pode contratar engenheiro de software.
	5– Jogador da vez pode demitir engenheiro de software.
	6– Engenheiro de software pode destruir artefato.
	7– Jogador da vez pode escolher até 3 adversários para submeter problema.
	8– Jogador da vez lê em voz alta o problema aplicado ao adversário.
Restrição:	RNF: Qualidade [artefato]
Exceções:	

Titulo:	**Joga Rodada de Conceitos (depois da correção)**
Objetivo:	Rodada de conceitos e problemas seja jogada
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de ações
Restrição:	
Pré-condição:	Jogador da vez já foi escolhido Todos os jogadores terminaram a rodada de ações
Recursos:	cartas, cartão de projeto, tabuleiro central, tabuleiro individual
Restrição:	
Atores:	Jogador da vez, SimulES, engenheiro de software, adversário
Episódios:	1– SimulES inicia rodada de conceitos.
	2– Jogador da vez descarta cartas excedentes, caso possua mais de 6 nas mãos.
	3– Jogador da vez analisa as cartas que tem nas mãos.
	4– Jogador da vez aplica conceitos caso sejam permanentes. **Restrição:** não pode exceder o orçamento do projeto.
	5– Jogador da vez pode contratar engenheiro de software.
	6– Jogador da vez pode demitir engenheiro de software.
	7– Engenheiro de software pode destruir artefato.
	8– Jogador da vez pode escolher até 3 adversários para submeter problema.
	9– Jogador da vez lê em voz alta o problema aplicado ao adversário.
Restrição:	RNF: Qualidade [artefato], Boa [estratégia]
Exceções:	

Figura 4.82 – Descrição da SDsituation Joga Rodada de Conceitos antes e depois da correção.

Titulo:	**Tratamento de Problema (antes da correção)**
Objetivo:	Problemas sejam tratados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de conceitos
Restrição:	
Pré-condição:	Adversário recebeu problema do jogador da vez
Recursos:	cartas, cartão de projeto, tabuleiro individual, tabuleiro central
Restrição:	
Atores:	Jogador da vez, adversário
Episódios:	1– Jogador da vez aplica problema.
	2– Adversário atende a demanda do problema.
	3– Adversário pode aplicar conceito.
	4– Adversário mantém problema permanente.
	5– Adversário descarta problema temporário.
Exceções:	

Titulo:	**Tratamento de Problemas (depois da correção)**
Objetivo:	Problemas sejam tratados
Contexto:	Cartão de projeto no centro da mesa
Localização Geográfica:	Sala de jogo
Localização Temporal:	Após a rodada de conceitos
Restrição:	
Pré-condição:	Adversário recebeu problema do jogador da vez
Recursos:	cartas, cartão de projeto, tabuleiro individual, tabuleiro central
Restrição:	
Atores:	Jogador da vez, adversário
Episódios:	1– Jogador da vez aplica problema.
	2– Adversário atende a demanda do problema.
	3– Adversário pode aplicar conceito.
	4– Adversário mantém problema permanente.
	5– Adversário descarta problema temporário.
Restrição:	RNF: Boa [estratégia]
Exceções:	

Figura 4.83 – Descrição da SDsituation Tratamento de Problemas antes e depois da correção.

Uma vez realizadas as modificações propostas pelos interessados nas descrições em cenários, devemos agora buscar operacionalizações da meta flexível Boa [estratégia] para que a referida meta flexível seja inserida nos cenários transformados da ferramenta de simulação *UCEd.* A Figura a seguir exibe o grafo NFR da meta flexível supracitada.

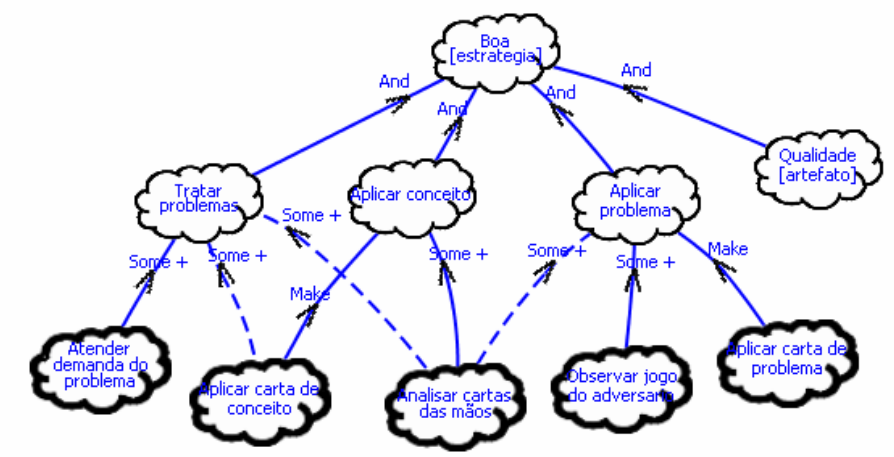


Figura 4.84 – Grafo NFR para Boa [estratégia].

A Figura 4.83 mostra a decomposição de uma boa estratégia durante o jogo em aplicar os conceitos e problemas de forma oportuna, tratar os problemas que forem recebidos e manter os artefatos com qualidade.

Sendo assim, com base no grafo da Figura anterior, iremos integrar as operacionalizações aos cenários transformados da ferramenta [14], com o objetivo de validar junto aos interessados as alterações realizadas nos modelos. A estratégia é bastante interativa. Isso quer dizer que em poucas interações teremos os modelos muito próximos do desejável pelos interessados. As Figuras subsequentes exibem os cenários transformados que foram alterados de acordo com a validação ocorrida anteriormente:

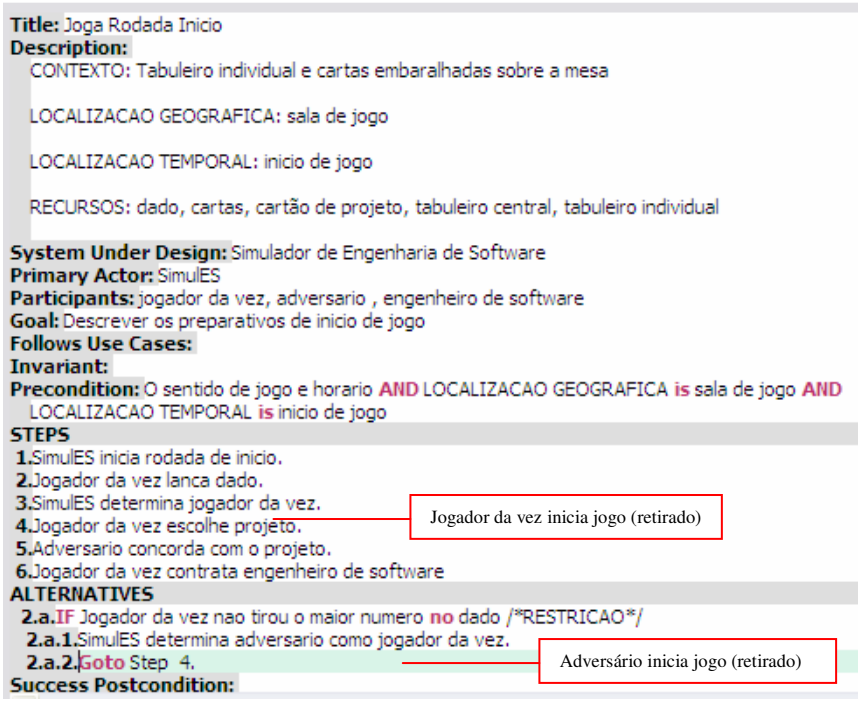


Figura 4.85 – Novo cenário Joga Rodada de Inicio (transformado).

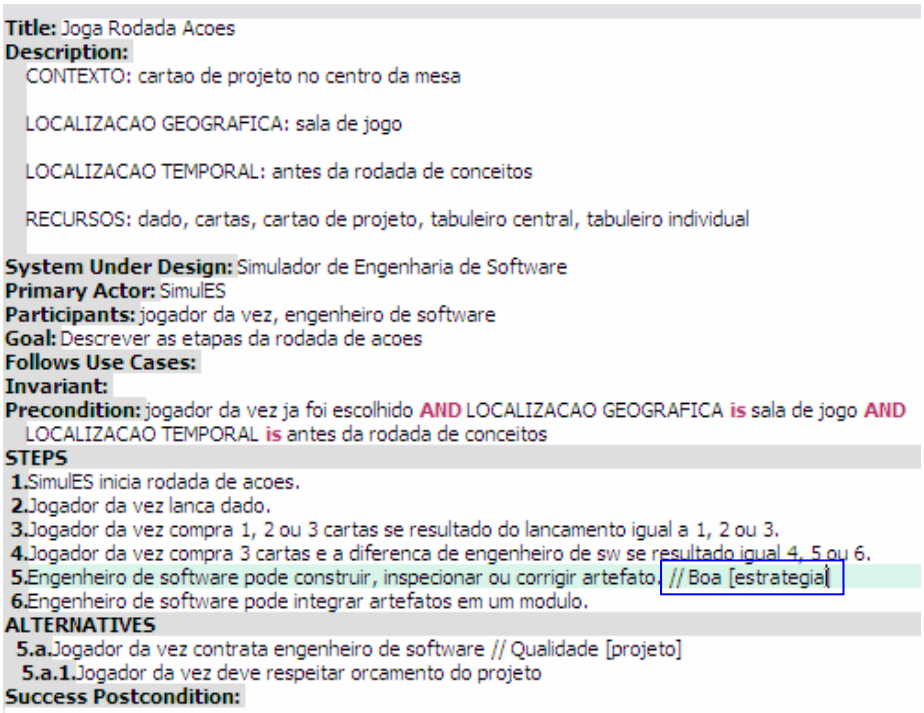


Figura 4.86 – Novo cenário Joga Rodada de Ações (transformado).

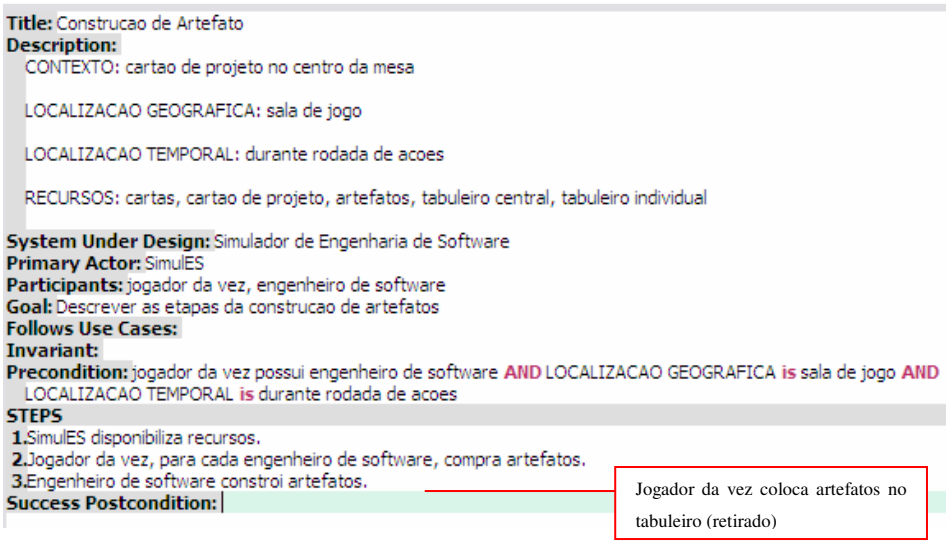


Figura 4.87 – Novo cenário Construção de Artefato (transformado).

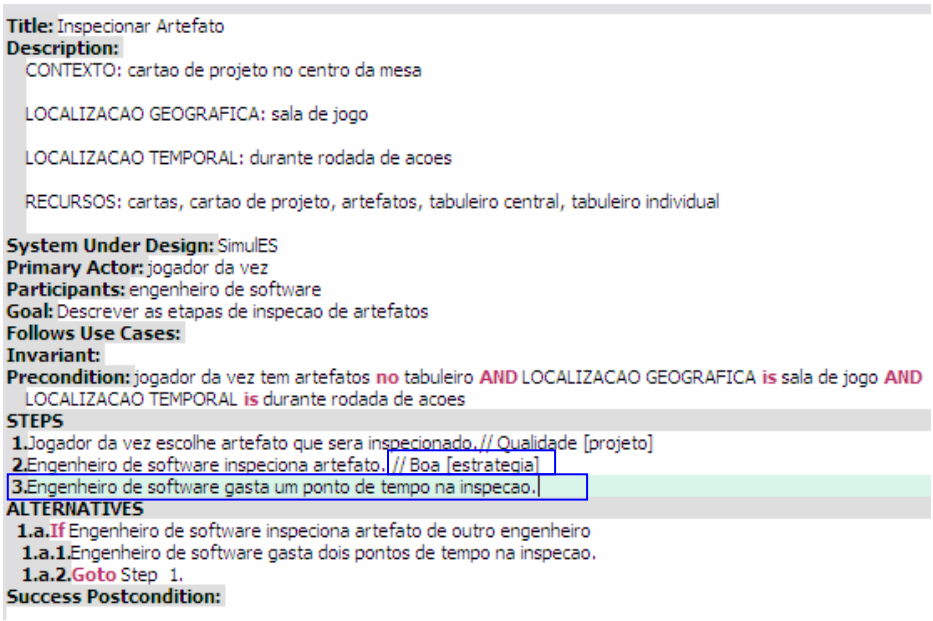


Figura 4.88 – Novo cenário Inspeção de Artefato (transformado).

Title: Corrigir Artefato
Description:
CONTEXTO: cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: durante rodada de acoes

RECURSOS: cartas, cartao de projeto, artefatos, tabuleiro individual, tabuleiro central

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, engenheiro de software
Goal: Descrever as etapas da correcao de artefatos
Follows Use Cases:
Invariant:
Precondition: jogador da vez tem artefatos defeituosos **no** tabuleiro **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** durante rodada de acoes
STEPS
1.SimulES disponibiliza recursos.
2.Jogador da vez escolhe engenheiro que tera artefato corrigido. // Qualidade[projeto]
3.Jogador da vez escolhe artefato defeituoso.
4.Engenheiro de software corrige artefato. // Boa[estrategia]
5.Engenheiro de software gasta 1 ponto de tempo na correcao.
6.Jogador da vez substitui artefato corrigido por outro da mesma cor
ALTERNATIVES
4.a. Engenheiro de software corrige artefato de outro engenheiro
4.a.1. Engenheiro de software gasta dois pontos de tempo.
4.a.2. Goto Step 4.
Success Postcondition:

Figura 4.89 – Novo cenário Correção de Artefato (transformado).

Title: Joga Rodada de Conceitos
Description:
CONTEXTO:cartao de projeto no centro da mesa

LOCALIZACAO GEOGRAFICA: sala de jogo

LOCALIZACAO TEMPORAL: apos rodada de acoes

RECURSOS: cartas, cartao de projetos, tabuleiro central, tabuleiro individual

System Under Design: Simulador de Engenharia de Software
Primary Actor: SimulES
Participants: jogador da vez, adversario, engenheiro de software
Goal:
Follows Use Cases:
Invariant:
Precondition: Todos os jogadores terminaram rodada de acoes **AND** LOCALIZACAO GEOGRAFICA **is** sala de jogo **AND** LOCALIZACAO TEMPORAL **is** apos rodada de acoes
STEPS
1.SimulES inicia rodada de conceitos.
2.Jogador da vez descarta cartas, caso possua mais de 6 nas maos.
3.Jogador da vez analisa as cartas em suas maos. // Boa [estrategia]
4.Jogador da vez aplica conceitos caso sejam permanentes. // Boa [estrategia]
5.Jogador da vez pode contratar engenheiro de software.
6.Jogador da vez pode demitir engenheiro de software.
7.Engenheiro de software pode destruir artefato. // Qualidade [artefato]
8.Jogador da vez pode escolher ate 3 adversarios para submeter problema. // Boa [estrategia]
9.Jogador da vez le em voz alta o problema aplicado.
Success Postcondition:

Figura 4.90 – Novo cenário Joga Rodada de Conceitos (transformado).

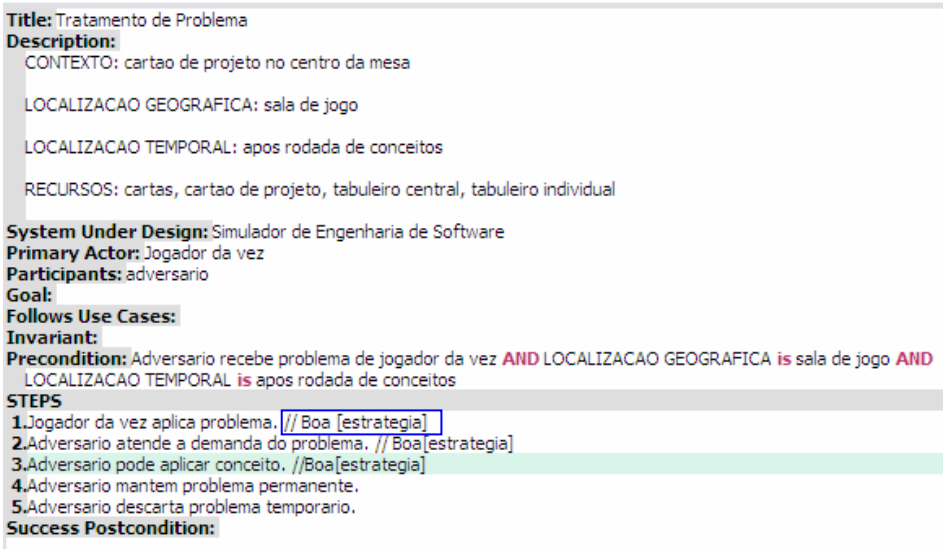


Figura 4.91 – Novo cenário Tratamento de Problema (transformado).

A partir deste momento, o engenheiro de requisitos poderá retomar o contato com os interessados para reafirmar se as alterações foram bem conduzidas e se o modelo está refletindo o desejo destes interessados. Para isso, o engenheiro de requisitos poderá realizar uma nova simulação, aos moldes da primeira, até ter certeza de que os desejos dos interessados foram satisfeitos a contento.

4.5.Considerações Sobre o Estudo de Caso

Ao término deste estudo de caso, podemos considerar seu resultado bastante positivo. Através da aplicação detalhada da estratégia de validação proposta utilizando caso SimulES, foi possível perceber de uma maneira mais clara e concisa a contribuição que a estratégia pode oferecer ao engenheiro de requisitos.

Pode–se perceber também que o uso de um método confiável e eficaz na construção dos modelos i* (o método Eri*c), mostrou–se uma boa base para que a

estratégia fosse desenvolvida. Espera–se que quanto melhor for a elicitação e construção dos modelos, menos erros, discrepâncias ou omissões sejam encontrados.

A descoberta de novos símbolos e episódios impactou os modelos i* propostos e suas respectivas descrições em cenários. Nesta ocasião, através do uso de heurísticas para transformação e validação fomos capazes de inserir as alterações nesses modelos. Essas heurísticas se mostraram bastante eficazes durante o processo.