1
Introdução

O objetivo deste capítulo é estabelecer o contexto da pesquisa realizada. Ao longo deste capítulo serão apresentadas: a motivação para a pesquisa, os objetivos do trabalho, os conceitos utilizados e a organização desta dissertação.

Motivação

Jogos educacionais vêm sendo propostos no ensino de ciências da computação, e também no ensino de engenharia de software [1], [2], [3]. Nesta pesquisa abordaremos um destes jogos educacionais, o SimulES [4], [5] [6]. Ele é um jogo de cartas educacional concebido como uma evolução do jogo *Problems and Programmers* [1], tendo origem no grupo de engenharia de requisitos da PUC-Rio. SimulES permite a simulação do processo de desenvolvimento de software, onde o estudante assume diferentes papéis em uma equipe de software e desta forma enfrenta problemas típicos da construção de software.

SimulES será implementado usando uma modelagem que utiliza a perspectiva intencional[7]. Este tipo de modelagem é geralmente usado para modelar contextos organizacionais com base nos relacionamento entre atores e suas dependências. Um de nossos objetivos é demonstrar que modelos iniciais em jogos devem levar em conta a interação entre atores (jogadores) e, portanto, um modelo intencional é apropriado para este fim. Acreditamos que os modelos intencionais, além de proporcionar uma visão mais ampla da colaboração entre atores, também podem ser apropriados para apoiar conceitos de transparência [8], sendo a transparência uma qualidade que se torna importante no momento em que queremos descrever o que o software está fazendo para aqueles que usam ou são afetados por ele, em outras palavras, a transparência de software está relacionada com características de qualidade que o software deve ter para que seja de fácil compreensão para estes usuários.

Objetivos

Com nosso trabalho queremos mostrar que modelos iniciais de jogos devem levar em conta a interação entre atores. Para tal interação, o modelo intencional é apropriado, como também incorpora elementos de qualidade próprios da transparência [9]. Queremos mostrar o enfoque do desenvolvimento de software fundamentado nestes elementos, pois acreditamos que artefatos de software baseados no modelo intencional (*i**) torna o modelo mais perto da implementação e assim, mais transparente. Nosso principal objetivo foi o de automatizar o jogo SimulES partindo de uma modelagem intencional. Com isso estaremos demonstrando através de um exemplo (SimulES-W), como desenvolver um software de maneira mais transparente. Além da visão dos modelos, utilizamos estratégias de elicitação de requisitos para melhorar o entendimento do jogo e sua interação com os jogadores.

Conceitos

No decorrer deste trabalho serão utilizados alguns conceitos importantes, que serão descritos a seguir:

Léxico Ampliado da Linguagem (LAL)

O LAL (Léxico Ampliado da Linguagem) [10] é uma técnica que permite descrever os símbolos de uma linguagem. Sua idéia central é permitir que sejam identificadas as palavras ou frases do entorno social da aplicação em estudo, também conhecido como Universo de Informação (UdI)[1].

No LAL do UdI, cada entrada está associada a um símbolo (palavra ou frase do UdI), e pode ser classificada como sujeito, objeto, verbo ou estado. Cada símbolo pode possuir sinônimos e é descrito por noções e impactos [13]. Noção é a descrição do significado do símbolo. Impacto descreve os efeitos do uso do símbolo ou sua ocorrência no UdI. A utilização do LAL será apresentada no Capítulo 3.

[1] O Universo de Informações (ou Universo de Discurso – UofD) inclui todas as fontes de informação e todas as pessoas relacionadas ao software.

Cenários

Os cenários [18] são descrições de situações em linguagem natural semi-estruturada que permitem entender, unificar, analisar, e rastrear relacionamentos do sistema a serem construídos. Eles também permitem descrever as interações entre componentes de um sistema. O uso de cenários é uma boa prática, pois permite elicitar e especificar o comportamento do sistema como se fossem descrições de situações em um ambiente particular. É importante ressaltar que os cenários estão associados diretamente com o LAL [10], já que usam os símbolos ali definidos.

Tabela 1 – Estrutura de cenário adaptada de [59]

Título	Identificador do cenário. Deve ser único.
Objetivo	Descrição da finalidade do cenário. Deve ser descrito também como este objetivo é alcançado.
Contexto	Descrição do estado inicial do cenário. Deve ser descrito através de pré-condições, localização geográfica e/ou temporal.
Atores	São entidades envolvidas diretamente com a situação. Um ator, para ser válido, deve aparecer em pelo menos um dos episódios.
Recursos	São entidades passivas utilizadas na situação. Um Recurso, para ser válido, deve aparecer em pelo menos um dos episódios.
Episódios	Sentenças (seqüência / paralelismo / seleção) que correspondem a ações e decisões com participação dos atores e utilização de recursos. A estrutura de controle básica é a seqüência, mas pode-se utilizar seleção opcional idade e paralelismo
Restrições	São aspectos não funcionais que qualificam/restringem o que está sendo descrito pelo cenário. Estes aspectos podem estar relacionados ao contexto, recursos ou episódios.
Exceções	Situações que impedem que o objetivo do cenário seja alcançado. O tratamento para tais situações deve ser descrito por outro cenário.

Segundo [18] os cenários são descrições de situações comuns ou cotidianas. Eles devem ter aspetos de usabilidade e permitir aprofundar no conhecimento do problema e unificar critérios, além de explicitar a obtenção de compromissos entre os clientes e usuários como também os detalhes da organização.

Neste trabalho, será utilizada a abordagem apresentada em [24], onde sua estrutura está composta pelas entidades: *título, objetivo, contexto, recursos, atores, episódios, exceções* e *restrições* conforme [59]. A Tabela 1 apresenta as entidades e suas descrições.

Visão Geral do *Framework i**

O *framework i** (i de intencional- estrela de distribuída) [7] permite modelar contextos organizacionais baseado nos relacionamentos de dependência intencional entre os atores. Este *framework* é usado para obter um melhor entendimento dos relacionamentos organizacionais com base na interação entre atores.

Identificar estes relacionamentos entre os atores ajuda para que engenheiros de requisitos possam elicitar metas nas fases iniciais. Estas metas serão identificadas respondendo a questionamentos sobre *por que os atores têm estas dependências organizacionais.* O foco do *framework i** é possibilitar a compreensão das razões tanto internas dos atores como aquelas que são compartilhadas entre eles, uma vez que as mesmas são expressas explicitamente, auxiliando na escolha de alternativas durante a etapa de modelagem do software. Neste *framework* os participantes do contexto organizacional são chamados de atores. Conforme Yu [7], atores são entidades ativas que efetuam ações para alcançar metas através do exercício de suas habilidades e conhecimentos, podendo ter dependências intencionais entre eles. Uma *dependência intencional* ocorre quando dois ou mais atores compartilham algum elemento de dependência definido no *framework i**. A seguir uma descrição geral dos elementos de dependência:

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

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

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

(iv) **Meta flexível:** condição ou estado no mundo que o ator deseja alcançar. É diferente de uma meta concreta, pois a condição para ser alcançada não é definida desde o inicio, além disso, ela pode estar sujeita a interpretação. Conforme [12] quando uma meta flexível é um componente em uma decomposição de tarefa, ela serve como uma meta de qualidade para aquela tarefa, guiando (ou restringindo) a seleção entre as alternativas para a decomposição da tarefa.

Conforme [7], Yu usou dois modelos dentro do *Framework i**: o modelo SD (siglas em inglês de *Strategic Dependency*) e o modelo SR (siglas em inglês de *Strategic Rationale*), descritos a seguir.

O Modelo SD – Strategic Dependency

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

No modelo SD, um relacionamento baseado em cooperação entre dois atores representa a dependência que existe entre eles, onde o primeiro é designado "*depender*" ou "dependente" e o segundo "*dependee*" ou "de quem se depende", unidos por um elo de dependência, designado como "*dependum*", está relação de dependência ou *dependum* pode representar uma meta a ser atingida, uma tarefa a

ser executada, um recurso a ser fornecido ou uma meta flexível a ser satisfeita. A seguir as definições do *dependum* segundo [7]:

(i) Dependência por **meta**: acontece quando o *depender* depende do *dependee* para que certo estado do mundo seja alcançado. Como acontece na Figura 1, ao *dependee* é dada a liberdade de execução da meta. Por tanto, com uma dependência por meta, o *depender* ganha a habilidade de assumir que a condição ou estado do mundo será alcançado, é assim que o *depender* se torna vulnerável, pois o *dependee* pode falhar ou não realizar tal condição;

(ii) Dependência por **tarefa**: acontece quando o *depender* depende de do *dependee* para que este último realize uma tarefa. Este tipo de relação especifica "como" a tarefa deve ser realizada, mas não menciona "porquê". Como na Figura 1, o Jogador vai depender do SimulES para que os recursos sejam disponibilizados. Isso faz com que o depender (Jogador) seja vulnerável, pois o *dependee* (SimulES) pode falhar ou não executar a tarefa.

(iii) Dependência por **recurso**: acontece quando *depender* depende de outro, o *dependee,* para que uma entidade (física ou computacional) seja disponibilizada. Como mostrado na Figura 1 o depender (jogador) ganha a habilidade de usar a entidade (Cartão do projeto) como um recurso, este fato faz que este fique vulnerável, pois a entidade ou recurso pode tornar-se indisponível.

(iv) Dependência por **meta flexível**: acontece quando o *depender* depende do *dependee* para que certo estado do mundo seja alcançado, porém diferentemente de uma meta (rígida ou concreta), pois o critério para a condição de ser alcançada não é definido a princípio, estando sujeito à interpretação. Quer dizer que para metas flexíveis, são usados os termos "*satisfeita a contento*" conforme [48] ou "*razoavelmente satisfeita*". Esta subjetividade faz com que a satisfação de metas flexíveis seja inerente a requisitos não funcionais, por se tratar de aspectos de qualidade [12]; como é ilustrado na Figura 1 não se possuem critérios claramente definidos para sua satisfação, além disso, o depender (Jogador) será

vulnerável, pois o *dependee* (SimulES) pode falhar em realizar tal condição.

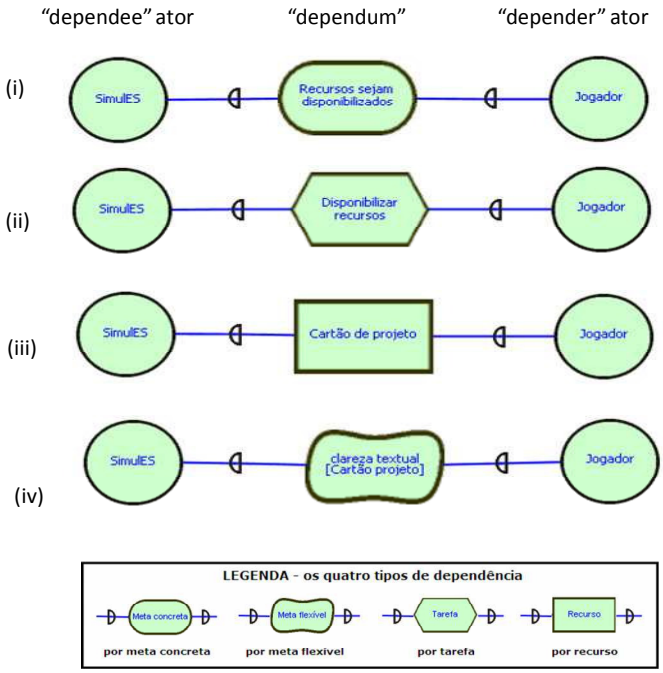

Figura 1 – Ilustração da dependência entre atores utilizada em modelos SD.

O Modelo SR – Strategic Rationale

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

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

possui duas classes principais de elos: elos de decomposição de tarefa e elos "meios-fim".

(i) **Elos de decomposição de tarefa**: Uma tarefa (cuja noção está associada a 'como fazer alguma coisa') é modelada em termos de sua decomposição em suas subcomponentes, que podem ser: metas, tarefas, recursos e/ou metas flexíveis (os quatro tipos de nós). Na representação com elos de decomposição de tarefa apresentado na Figura 2, a tarefa "B" é decomposta em uma (sub) meta: "submeta M"; uma (sub) tarefa: "subtarefa S"; e uma meta flexível: "meta flexível F".

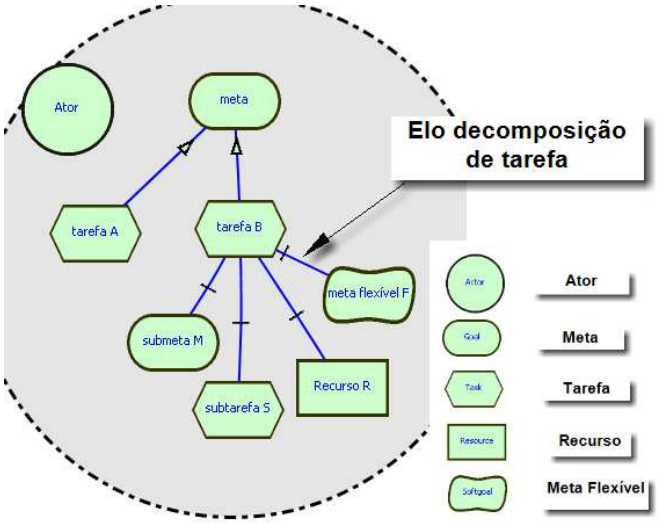

Figura 2 – Decomposição de tarefas.

(ii) **Elos *Meios-Fim*:** os elos do tipo *meios-fim* segundo [12] representam relacionamentos do tipo uma meta a ser alcançada, uma tarefa a ser feita, um recurso a ser produzido, ou uma meta flexível a ser razoavelmente satisfeita (*satisficed*) nomeado como "fim" e os "meios" alternativos para alcançá-los. Na Figura 2 a "Tarefa B" representa o "meio" pois a noção de tarefa está associada a 'uma atividade que tem que ser feita'. Além disso, a notação gráfica do *i** apresentada na Figura 3 indica-nos que uma cabeça de seta aponta dos meios para o fim. Os tipos de elos meios-fim estão descritos a seguir:

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

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

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

Elo Meta Flexível – Meta Flexível – nos elos deste tipo, tanto o "fim" quanto o "meio" são metas flexíveis, permitindo o desenvolvimento de uma hierarquia *meios–fim* de metas flexíveis. Isso permite que as metas flexíveis sejam cada vez mais refinadas, até chegar a metas flexíveis mais fáceis de operacionalizar [12].

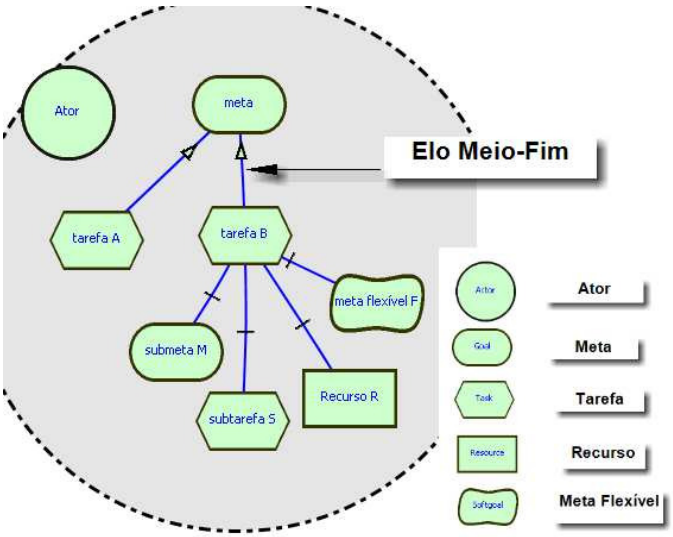

Figura 3 – Elos meios-fim.

Na Figura 3 se representa o elo do tipo Meta–Tarefa, onde a meta "fim" tem duas tarefas "meio" "A" e "B".

Extensões do *Framework i**

Este trabalho fará referência a algumas extensões sobre o *framework i** original: o modelo SA (*Strategic Actor*) [15]; situações de dependências estratégicas (*SDsituations*) [16]; e painéis de intencionalidade (*Diagramas IP*) [11]. Essas extensões serão detalhadas a seguir.

O Modelo SA – *Strategic Actor*

O objetivo do modelo SA [6] é modelar atores em *i**. Ele auxilia no entendimento dos atores e seus relacionamentos, do ponto de vista estrutural [3].

O modelo SA usa os conceitos de agente, posição e papel, definidos em [7], para refinamento de atores e seus relacionamentos. A seguir serão descritos alguns conceitos conforme [7]:

Ator - um ator é uma entidade ativa que executa atividades para atingir suas metas conforme o exercício de seu conhecimento (*know-how*) e habilidades [7]. O termo ator é usado para fazer referência a qualquer unidade à qual se possa atribuir dependências intencionais [7], [12].

Posição - uma posição é formada por um conjunto de papéis desempenhados por um agente [11]. Ele se localiza em um nível intermediário de abstração entre um papel e um agente.

Papel: um papel é uma caracterização abstrata do comportamento de um ator social em algum contexto especializado [7]. Onde suas características podem eventualmente ser transferidas para outros atores sociais.

Agente - um agente é um ator com manifestações físicas concretas, tal qual um ser humano [7]. O termo agente pode fazer referência tanto para humanos quanto para agentes artificiais (hardware/software). Suas características geralmente são intransferíveis para outros indivíduos, como habilidades, limitações e experiências [7].

Conforme [15] um novo tipo de ator foi proposto para o relacionamento de instanciação entre agentes: o agente real. Assim, é mais específico que o agente, um agente genérico, e pode ser identificado, como por exemplo, uma pessoa específica ou um software específico.

Um resumo dos relacionamentos descritos por Leite et al. em [15] os quais fazem uso das definições e exemplos de [7] são descritos a seguir:

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

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

Ocupa (OCCUPIES): relacionamento de agente para posição. Um agente pode ocupar mais de uma posição e uma posição pode ser ocupada por mais de um agente [12].

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

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

Parte de (is–Part–of): relacionamento de agente para agente, posição para posição e de papel para papel. Agentes, posições e papéis podem ser ou podem ter mais de uma sub-parte. Conforme [12] há uma restrição para este relacionamento entre posições. Uma posição é parte de outra se todos os papéis desta posição também são cobertos pela outra posição.

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

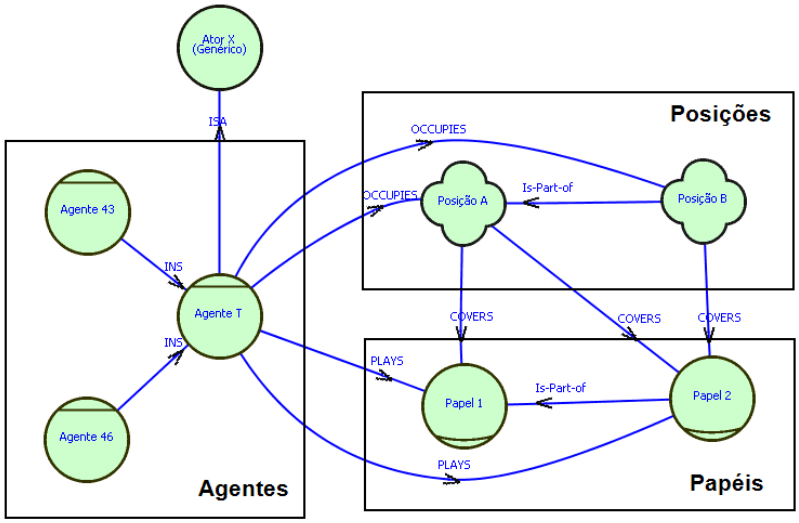

Figura 4 – Ilustração de um modelo SA [11].

Situação de Dependência Estratégica – *SDsituations*

As *SDsituations* foram propostas por Oliveira et al. em [16] como uma representação estruturada de situações de dependência estratégica entre atores dentro de um contexto organizacional. Conforme [11] uma *SDsituation* reúne um bloco de situações que compartilham um objetivo em comum.

Segundo [16] elementos de dependência como meta, tarefa, recurso ou meta flexível os quais envolvem atores não estão isolados, por tanto, é uma situação bem definida de colaboração que é conhecida como "situação de dependência estratégica". Além disso, uma *SDsituation* pode ser identificada separadamente das outras *SDsituations*, formando uma cadeia de interdependências entre elas.

Conforme por Oliveira et al. em [16] existem três tipos de interdependências entre *SDsituations*: física, lógica e temporal. Entretanto mais de um tipo de interdependência pode ocorrer simultaneamente. Os três tipos de interdependências serão descritos a seguir [11]:

Física – ocorre quando um recurso é preparado por uma *SDsituation* e é solicitado por outra *SDsituation.*

Lógica – ocorre quando uma ou mais *SDsituations* necessita da conclusão de outra *SDsituation* para sua iniciação, ou quando uma ou mais *SDsituations* dependem da conclusão de outra *SDsituation* para sua conclusão.

Temporal – ocorre quando uma ou mais *SDsituations* necessitam esperar algum tempo após o início de outra *SDsituation* ou quando uma ou mais *SDsituations* necessitam esperar algum tempo após a conclusão de outra *SDsituation*.

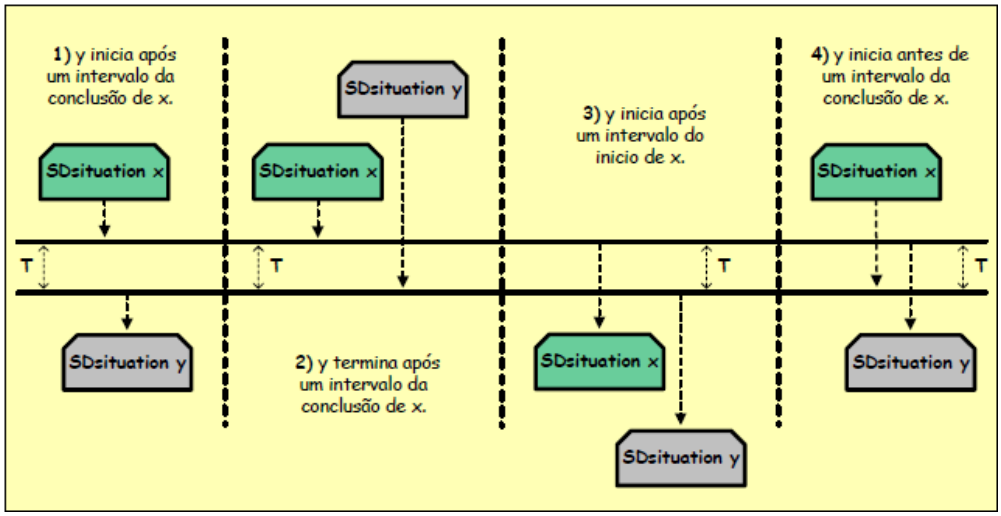

Figura 5 – Variantes de interdependência lógica das SDsituations [11].

Painel de Intencionalidade – Diagrama IP

Conforme [11], um "Painel de Intencionalidade" ou diagrama IP é um modelo SR reduzido, onde são considerados os atores somente como proprietários das metas, sendo que estas metas podem ser concretas e flexíveis e tem relações entre elas. Toda a intencionalidade é representada em um único diagrama que é guiado na sua construção pelas *SDsituations*, visando principalmente diminuir a dificuldade de entendimento do diagrama SR, que é complexo por natureza.

Um diagrama IP possui nós e arestas. Os nós representam as metas concretas ou as metas flexíveis, enquanto as arestas representam os tipos de relações entre metas (Correlação, Contribuição, Equivalência, Dependência). Os

atores estão presentes no diagrama para indicar os responsáveis pelas metas. A seguir, serão descritos os tipos de relações [2]:

Correlação: esta relação ocorre entre duas metas de um mesmo ator. Como na Figura 6 a "meta w" é uma meta inicial que pode ser um subcomponente de uma tarefa, sendo esta o meio para alcançar a meta principal "meta t". Ou seja, o sucesso da meta principal é determinado pelo alcance da meta inicial.

Contribuição: esta relação que ocorre ente duas metas flexíveis de um mesmo ator. Na Figura 6 a meta flexível "Tipo 1 [tópico 2]" e chamada de meta inicial, e contribui de forma negativa ("–") para a outra meta flexível "Tipo 3 [tópico 1]", em contraste, a meta flexível "Tipo 2 [tópico 2]" e uma meta inicial, e contribui de forma positiva ("+") para a outra meta flexível "Tipo 1 [tópico 2]" .

Dependência: esta relação ocorre entre duas metas de atores diferentes**.** Ela representa a necessidade de satisfazer uma dependência entre dois atores, fazendo uso da mesma semântica do diagrama SD. As dependências podem representar uma tarefa, um recurso, uma meta flexível ou uma meta concreta que devem ser descobertas no diagrama SD, ilustrado na Figura 6 por a "meta concreta w" e "meta concreta r".

Equivalência: relação entre metas concretas ou metas flexíveis. Representam simplesmente que o relacionamento existe, as metas concretas ou flexíveis são equivalentes para atores diferentes, na Figura 6 é representada esta relação entre "meta concreta t" e "meta concreta y".

Conforme [11] é recomendado que cada diagrama IP retrate apenas uma *SDsituation* para manter o controle da complexidade.

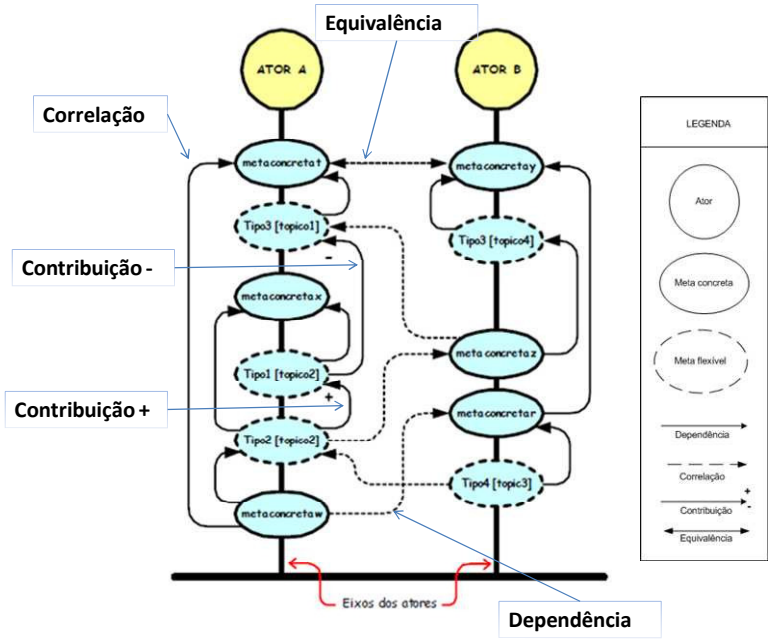

Figura 6 – Ilustração de um painel de intencionalidade adaptado de [11].

Organização da Dissertação

Esta dissertação apresenta a construção de um jogo educacional com modelagem intencional apoiado em conceitos de transparência. Seus capítulos estão organizados da seguinte forma:

O Capítulo 2 apresenta um resumo dos jogos educacionais e principalmente aqueles jogos para ensino na engenharia de software O Capítulo 3 apresenta a descrição dos requisitos para a evolução do SimulES. O Capítulo 4 apresenta as representações utilizadas para a modelagem do jogo e a nossa proposta de modelagem intencional. O Capítulo 5 apresenta os trabalhos relacionados com a modelagem intencional.

O Capítulo 6 detalha passo a passo a aplicação do mapeamento do modelo até o código.

O Capítulo 7 apresenta as conclusões e contribuições deste trabalho, além de sugestões para trabalhos futuros.