abstrações do *framework* i* e as abstrações do modelo BDI e as associações observadas entre as abstrações do modelo BDI e o código do SMA intencional para o *framework* JADEX. As heurísticas transformacionais de desenho e de implementação são apresentadas na Seção 3.2. O uso das heurísticas transformacionais como elos de rastreabilidade, ou rastros, é exposto na Seção 3.3. Os trabalhos relacionados são discutidos na Seção 3.4. Finalmente, a Seção 3.5 apresenta as considerações finais do capítulo.

3.1. Associações entre abstrações dos modelos i* e BDI e código JADEX

Os requisitos descritos em modelos intencionais são desenhados em agentes inteligentes que simulam o raciocínio humano através do modelo BDI. Optou-se pelo modelo BDI como a base para a arquitetura mental dos agentes, pois o modelo i* utiliza o mesmo paradigma de orientação, ou seja, a orientação a metas. Assim, evitam-se transições abruptas de nível de abstração entre os modelos de requisitos e de desenho. O objetivo desse processo é baixar o nível de abstração das especificações do software em desenvolvimento, obtendo-se o desenho do software.

Existem diferenças significativas entre um modelo i* e uma especificação BDI. As duas principais diferenças são: (i) os modelos i* representam uma rede de atores sociais, enquanto o modelo BDI visa representar a arquitetura mental interna de um único agente, e (ii) o modelo i* representa metas flexíveis como tarefas, e metas e metas flexíveis contribuem para as metas flexíveis de forma positiva ou negativa através de elos de contribuição.

Embora existam diferenças significativas entre a semântica dos dois modelos, os modelos compartilham muitas semelhanças, como os conceitos de ator/agente, metas/desejos, tarefas/intenções para atingir metas, crenças/crenças, recursos/crenças, entre outros. Assim, foi possível relacionar as abstrações dos dois modelos, como apresentado na Figura 3.1.

A abstração de ator do *framework* i* é representada no modelo BDI como um agente inteligente, pois um ator do *framework* i* possui as atitudes mentais informativas (visão do mundo e de seu estado interno), motivacionais (metas a serem atingidas) e deliberativas (ações para alterar o ambiente e atingir as metas).

As abstrações de agente, posição e papel do *framework* i* são representadas no modelo BDI também como agentes, pois todas essas abstrações são especializações da abstração de ator e, portanto, também possuem as mesmas atitudes mentais.

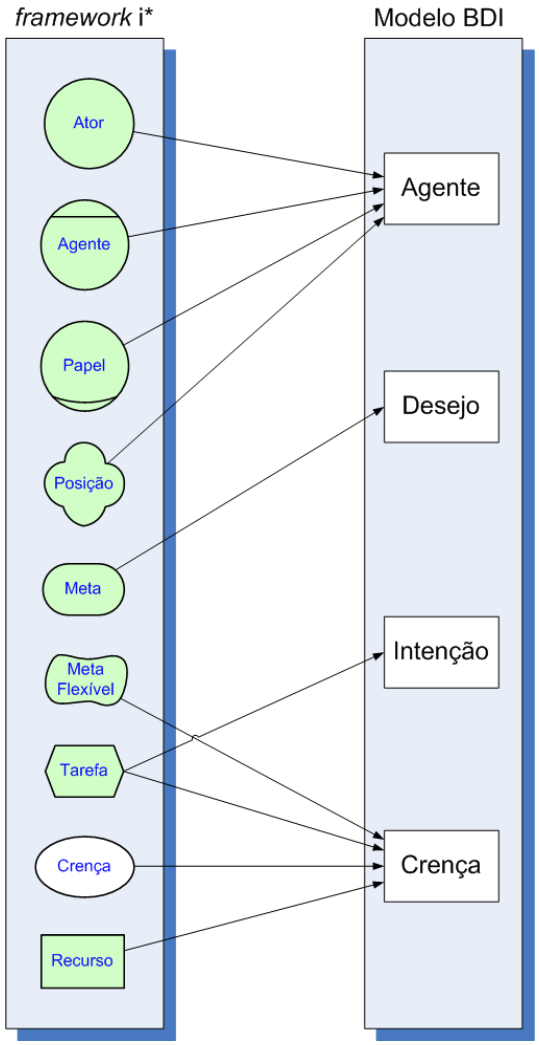

Figura 3.1 - Relações entre as abstrações do *framework* i* e do modelo BDI

A abstração de meta do *framework* i* está diretamente relacionada à abstração de desejo do modelo BDI, pois ambas representam estados desejados para o ambiente que envolve o ator/agente. A abstração de meta flexível do *framework* i* não possui um relacionamento direto com alguma abstração do modelo BDI. Porém, podemos relacioná-la com a abstração de crença, já que uma meta flexível representa como um ator enxerga um critério de qualidade do mundo real.

A abstração de tarefa do *framework* i* está diretamente relacionada à abstração de intenção do modelo BDI, uma vez que ambas descrevem uma ação ou sequencia de ações para tentar atingir uma meta. Entretanto os impactos que uma tarefa exerce sobre metas flexíveis não fazem parte da abstração de intenção do modelo BDI e devem ser representados como crenças do agente.

Originalmente, o *framework* i* não incluía a abstração de crença, o que já foi corrigido. A abstração de crença do *framework* i* é a mesma abstração de crença utilizada no modelo BDI. Recursos, entretanto, exigem uma maior atenção. Recursos que representam informações podem ser representados integralmente como uma crença no modelo BDI. Já os recursos que representam objetos do mundo real precisam antes ser abstraídos, selecionando apenas as características relevantes ao agente.

A Figura 3.2 mostra as relações entre as abstrações da especificação BDI e do código de SMAs em JADEX. Agentes executáveis são implementados como agentes JADEX. Agentes não executáveis (originados a partir de papéis e posições) tornam-se capacidades que devem ser assimiladas por agentes executáveis. Desejos são traduzidos como metas atingíveis, mantidas ou realizadas de acordo com a *tag* "type" da especificação BDI. As intenções são traduzidas como planos do agente – classes Java que estendem a classe "Plan" do JADEX. As crenças com cardinalidade 0..1 ou 1..1 são traduzidas como crenças do agente, enquanto as crenças com cardinalidade 0..n ou 1..n são traduzidas como um conjunto de crenças.

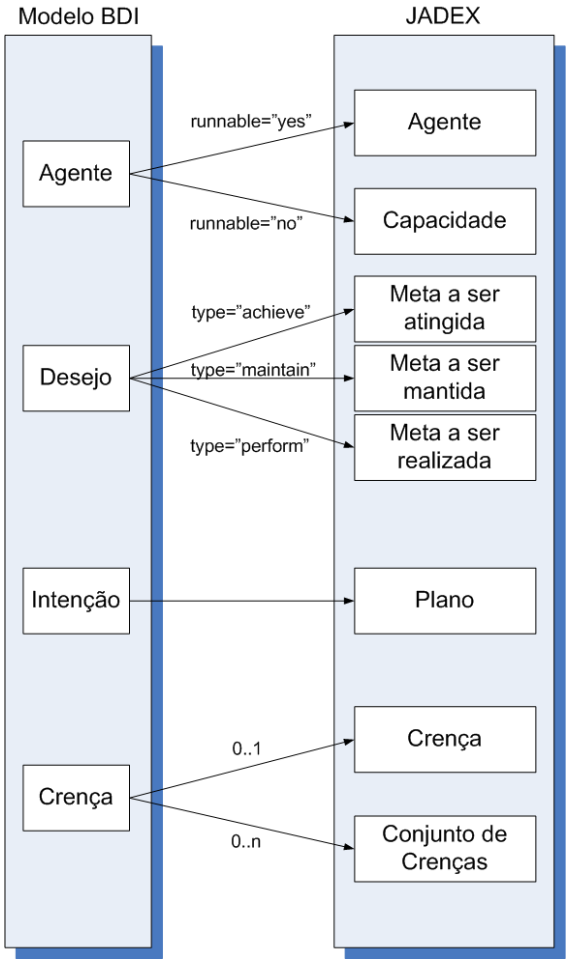

Figura 3.2 - Relações entre as abstrações do modelo BDI e código JADEX

3.2. Heurísticas Transformacionais de Desenho e de Implementação

Com base nos relacionamentos entre as abstrações dos modelos i* e BDI, apresentados na Figura 3.1, foi possível produzir heurísticas que facilitam a obtenção da especificação BDI para o software (Serrano e Leite 2011c; Serrano et al. 2009). Essa especificação é um documento XML que possui uma estrutura definida através de um documento XML Schema. A Tabela 3.1 mostra as principais heurísticas transformacionais de desenho que produzem a especificação BDI a partir dos modelos Dependência Estratégica e *Rationale* Estratégico do *framework* i*.

Tabela 3.1 - Heurísticas transformacionais de desenho: do *framework* i* para a especificação BDI

#	Quando aplicar	Ação	Saída (trechos em XML)
01	Para cada ator (ou agente, papel ou posição) pertinente à aplicação.	Criar um agente na especificação BDI. Atores e agentes possuem a *tag* runnable="yes". Papéis e posições possuem a *tag* runnable="no".	<agent name="[actor_name]" runnable="[yes|no]"> <beliefs/> <desires/> <intentions/> </agent>
02	Para cada meta de um ator.	Criar um desejo na especificação do agente.	<desire name="[goal_name]" type=""/>
03	Para cada meta flexível de um ator.	Criar uma crença com o tipo "Softgoal".	<belief name="[softgoal_name]" type="Softgoal" />
04	Para cada tarefa que visa atingir uma meta de um ator.	Criar uma intenção e associá-la ao desejo correspondente à meta.	<intention name="[task_name]"> <desire>[goal_name] </desire> <script lang="" /> </intention>
05	Para cada tarefa que visa atingir uma meta de um ator.	Criar uma crença com o tipo "Task".	<belief name="[task_name]" type="Task" />
06	Para cada tarefa que visa atingir uma meta de um ator.	Criar um cenário para descrever a tarefa e incluí-lo na *tag* "script" da intenção.	<script lang="scenarios"> [scenario] </script>
07	Para cada tarefa decomposta em subtarefas.	Criar um cenário que utiliza subcenários para descrever a tarefa e incluí-los na *tag* "script" da intenção.	<script lang="scenarios"> [scenario] [sub-scenario1] ... </script>
08	Para cada crença do ator com cardinalidade 0..1 ou 1..1.	Criar uma crença na especificação do agente.	<belief name="[belief_name]" type="[belief_type]" />
09	Para cada crença do ator com cardinalidade 0..n ou 1..n.	Criar um conjunto de crenças na especificação do agente.	<beliefset name="[beliefset_name]" type="[beliefset_type]" />
10	Para cada recurso que representa uma informação.	Criar uma crença na especificação do agente.	<belief name="[belief_name]" type="[belief_type]" />

#	Quando aplicar	Ação	Saída (trechos em XML)
11	Para cada recurso que representa um conjunto de informações	Criar um conjunto de crenças na especificação do agente.	<beliefset name="[beliefset_name]" type="[beliefset_type]" />
12	Para cada recurso que representa um recurso físico.	Abstrair do recurso físico as informações relevantes para o agente e criar uma crença.	<belief name="[belief_name]" type="[belief_type]" />
13	Para cada dependência por meta entre dois agentes.	Criar um desejo nas especificações dos dois agentes envolvidos e iniciar um protocolo de requisição (episódio de um cenário).	Heurística 02 <script lang="scenarios"> [scenario] [request episode] </script>
14	Para cada dependência por meta flexível entre dois agentes.	Acessar o grau de satisfação da meta flexível. Não é necessário estabelecer uma comunicação entre os agentes, pois impactos são sempre divulgados para todos os agentes.	<script lang="scenarios"> [scenario] [access episode] </script>
15	Para cada dependência por tarefa entre dois agentes.	Criar a tarefa na especificação de ambos os agentes e iniciar um protocolo de requisição.	Heurística 04 <script lang="scenarios"> [scenario] [request episode] </script>
16	Para cada dependência por recurso entre dois agentes.	Criar a crença em ambos os agentes e iniciar um protocolo de requisição.	Heurística 09, 10 ou 11 <script lang="scenarios"> [scenario] [request episode] </script>
17	Para cada dependência por meta entre vários agentes.	Criar a meta na especificação de todos os agentes envolvidos. Requisitar a todos os agentes ou responder a todos.	Heurística 02 <script lang="scenarios"> [scenario] [request_all episode] [inform_all episode] </script>

#	Quando aplicar	Ação	Saída (trechos em XML)
18	Para cada dependência por meta flexível entre vários agentes.	Acessar o grau de satisfação da meta flexível. Não é necessário estabelecer uma comunicação entre os agentes, pois impactos são sempre divulgados para todos os agentes.	<script Lang="scenarios"> [scenario] [access episode] </script>
19	Para cada dependência por tarefa entre vários agentes.	Criar a tarefa na especificação de todos os agentes envolvidos. Requisitar a todos ou responder a todos os agentes.	Idem à heurística 05 <script lang="scenarios"> [scenario] [request_all episode] [inform_all episode] </script>
20	Para cada dependência por recurso entre vários agentes.	Criar a crença na especificação de todos os agentes envolvidos. Requisitar a todos ou responder a todos os agentes.	Heurística 09, 10 ou 11 <script lang="scenarios"> [scenario] [request_all episode] [inform_all episode] </script>

A Figura 3.3 mostra o modelo arquitetural parcial no *framework* i* do estudo de caso Lattes-Scholar (mais detalhes no Capítulo 7). Para ilustrar o processo de aplicação das heurísticas, foi incluída uma numeração ao lado das abstrações do modelo. Cada número indica a heurística transformacional de desenho a ser aplicada para se obter a especificação do software no modelo BDI.

Os agentes “Página Web do Lattes-Scholar”, “Google” e “Lattes” são agentes externos ao software e, portanto, não são desenhados ou implementados. O agente “Gerente do SMA do Lattes-Scholar” é um agente interno desenhado como um agente executável do SMA. Os papéis “Pesquisador de Currículos” e “Repositório de Currículos” são desenhados como agentes não executáveis e serão, posteriormente, implementados como capacidades a serem utilizadas por agentes executáveis. As metas “URLs dos Currículos dos Pesquisadores sejam Recuperadas” e “Fotos dos Pesquisadores sejam Recuperadas” são desenhadas