Modelação das interações completas da rede social profissional LinkedIn usando a ontologia DEMO

Licenciatura em Engenharia Informática
Trabalho Final de Curso
Orientador: Mestre Sérgio Guerreiro
Discente: Ana Martins: nº20072779
Setembro de 2011

Índice

Índice 2
Lista de figuras 3
Lista de abreviaturas e síglas 4
Resumo 5
Abstract 5
1.Introdução 6
2.Enquadramento Teórico 7
Redes sociais 7
Redes sociais e ontologia 7
DEMO: Design & Engineering Methodology of Organizations 7
3.Método de modelação do DEMO 10
4. Resultados 12
4.1 Interacções da rede social LinkedIn 12
4.1.2 The Coordination-Actors-Production Analysis 12
4.2 Modelação 14
4.2.1 Transaction Result Table –TRT do LinkedIn 14
4. 2.2 Complete detailed Actor Transaction Diagram –ATD 15
4.2.2.1 Global ATD 16
4.3 Process Model: Process Structure Diagram – PSD do LinkedIn 18
4.3.1 PSD A01 18
4.3.2 PSD A03 18
4.3.3 PSD A07 18
4.3.4 PSD A10 19
4.4 Information Use Table –IUT do LinkedIn 20
4.5 O Action Model –AM do LinkedIn 21
4.6 State Model do LinkedIn 39
4.6.1 Object Property List – OPL 39
4.6.2 Object Fact Diagram –OFD 40
4.7 The Interstriction Model do LinkedIn 41
4.7.1 Bank Contents Table –BCT 41
4.7.2 The Actor Bank Diagram –ABD 42
4.7.3 The Organization Construction Diagram –OCD 43
5.Conclusões e trabalho futuro 44
Bibliografia 45
Anexos
4.6.2.1 OFD: people, membership, profile, upgrade account 46
4.6.2.2 OFD: membership, recommendation, update 46
4.6.2.3 OFD: memberships, connection and invitation 47
4.6.2.4 OFD: memberships, control, group and job 47

Lista de figuras

Figura 1. The B- I- and D-Organization 8
Figura 2. The ontological aspect models 10
Figura 3. DEMO methodology 10
Figura 4. Complete detailed Actor Transaction Diagram- ATD 15
Figura 5. Global ATD 16
Figura 6. PSD A01 18
Figura 7. PSD A03 18
Figura 8. PSD A07 18
Figura 9. PSD A10 19
Figura 10. Object Fact Diagram –OFD 40
Figura 11. The Actor Bank Diagram –ABD 42
Figura 12. The Organization Construction Diagram –OCD 43

Lista de abreviaturas e siglas

DEMO - *Design Engineering Methodology for Organizations*

DSI - *Desenvolvimento de Sistemas de Informação*

CM - *Construction Model*

PM- *Process Model*

AM- *Action Model*

ST- *State Model*

IAM -*the Interaction Model*

ISM -*The Interstriction Model*

OCD – *Organization Construction Diagram*

TRT- *Transaction Result Table*

BCT - *Bank Contents Table*

PSD- *Process Structure Diagram*

ATD -*Actor Transaction Diagram*

ARS- *Action Rules Specifications*

OFD -*Object Fact Diagram*

OPL -*Object Property List*

IUT- *Information Use Table*

Resumo

Actualmente as relações sociais mostram-se cada vez mais complexas, o que torna a modelação das mesmas um grande desafio. Neste trabalho é proposto uma modelação da rede social profissional LinkedIn para tal, utiliza-se a metodologia DEMO (Design Engineering Methodology for Organizations) com o objectivo de modelar as principais transacções da rede social LinkedIn de maneira a simplificar a complexidade da mesma.

Abstract

Currently social relations appear to be increasingly complex, which makes modeling of such a challenge. This paper proposes a modeling professional social network LinkedIn for such uses to DEMO methodology (Design Engineering Methodology for Organizations) in order to model the main transactions of the LinkedIn social network in order to simplify the complexity of it.

1. Introdução

Actualmente as relações sociais mostram-se cada vez mais complexas, tornando a sua modelação um grande desafio.

Escolhemos a rede social LinkedIn para a realização do nosso trabalho, por esta ser considerada a maior rede social profissional, por se focalizar na carreira e emprego do indivíduo.

Para modelar uma rede social existe várias metodologias, neste trabalho é proposto modelar a rede social LinkedIn de acordo com o standard Design Engineering Methodology for Organizations (DEMO).

A DEMO foi criada e explicada por Jan Dietz. Evoluiu para uma metodologia que pode representar de forma coerente, abrangente, consistente, concisa um modelo conceptual da organização (ou empresa).

De acordo com o Enterprise Engineering Institute," DEMO is a methodology for the design, engineering, and implementation of organizations and networks of organizations. The entering into and complying with commitments is the operational principle for each organization. These commitments are established in the communication between social individuals, i.e. human beings. " [9]

Segundo Saman Sattari Khavas, A DEMO "é uma metodologia para a concepção e engenharia de organizações, usada principalmente para o Desenvolvimento de Sistemas de Informação (DSI) e para redesenhar os processos do negócio". [6]

Tem como capacidade reduzir a complexidade de uma organização, facultando um modelo conceptual da organização perceptível por todos na organização.

O objectivo principal da metodologia DEMO é o de alinhar os processos de design e desenvolvimento com os principais processos de uma organização. Para isso, a metodologia abstrai-se da descrição detalhada de cada processo e tem ênfase nos conceitos genéricos e nos roles.

O trabalho está organizado da seguinte forma: o enquadramento teórico é apresentado na secção 2 no qual enumeramos definições de redes sociais, ontologia e DEMO; O método de modelação da DEMO, os resultados são apresentados respectivamente nas seções 3, 4; na secção 5 estão enunciadas as conclusões e o trabalho futuro.

2. Enquadramento teórico

Redes sociais

Segundo Wasserman "Uma rede social consiste num conjunto finito de actores e nas relações definidas entre eles". [7]
As Redes Sociais na Internet são formadas por indivíduos com interesses, valores e objectivos comuns, que visam a partilha de informações.

De acordo com Laura Garton et al. (2006) "social network is a set of people (or organizations or other social entities) connected by a set of social relationships, such as friendship, co-working or information exchange." [3]

Segundo André Desessards Jardim, (2010) uma Rede Social é considerada como "uma das formas de representação dos relacionamentos afectivos ou profissionais entre os seres humanos. A rede é responsável pelo compartilhamento de ideias entre as pessoas que possuem interesses e objetivos em comum." [5]

A rede social LinkedIn é a maior rede social profissional, com mais de cem milhões de membros a nível mundial. Os utilizadores focalizam-se na carreira e no emprego. No LinkedIn proliferam o empreendedorismo, o marketing e a comunicação. É utilizada por muitos para partilhar o seu curriculum vitae, procurar emprego ou iniciar e/ou manter contactos profissionais. Partilham-se ideias, iniciam-se discussões e oferecem-se oportunidade de carreira e emprego.

Redes sociais e ontologia

Hendler (2001), citado por André Jardim em 2010, apresenta a seguinte definição: "Uma Ontologia é uma especificação formal e explícita de uma conceptualização compartilhada. Onde: formal significa legível para computadores*;* especificação explícita está relacionado com conceitos, propriedades, relações, funções, restrições, axiomas; compartilhado significa conhecimento consensual*;* e conceptualização é uma visão abstracta e simplificada do mundo que se deseja representar." [5]

As ontologias são utilizadas, em áreas como a Inteligência Artificial, Web Semântica, Engenharia de Software, Arquitetura da Informação, .., como uma forma de representação do conhecimento sobre o mundo. "A sua utilização tem como objetivo estruturar de forma organizada as informações de um determinado domínio de conhecimento e reflectir um entendimento semântico de situações do mundo real." [5]

Portanto a ontologia possibilita uma representação formal e unificada que permite mapear elementos de uma Rede Social e para tal utilizaremos a ontologia DEMO.

DEMO: Design & Engineering Methodology of Organizations

DEMO foi criada e explicada por Jan Dietz, na década de noventa, lentamente evoluiu para uma metodologia que pode representar de forma coerente, completa, consistente, concisa um modelo conceptual da organização (ou empresa).

"Demo Engineering Methodology for Organizations (DEMO) is a method for organization engineering, an emerging discipline concerning the design and implementation of organizations "[1].

A DEMO permite que os processos de negócios das organizações sejam modelados a nível conceptual, que se traduz num resumo da organização e na forma como trabalha. DEMO concentra-se sobre os actos de comunicação que ocorrem entre os actores humanos na organização.

A metodologia abstrai-se da implementação, os modelos finais apresentados por esta metodologia pretendem mostrar a essência da organização. Desta forma, a metodologia tenta

simplificar a complexidade dos processos das organizações, de modo a tornar a sua gestão mais fácil.

Os benefícios da metodologia DEMO

Os benefícios da metodologia DEMO, segundo Enterprise Engineering Institute são a completude, concisão, coerência e abrangência. [8]
Estes benefícios são alcançados principalmente porque DEMO traduz totalmente a essência de uma organização, independente da maneira da implementação. "Esta essência é altamente estável e sempre up-to-date, uma vez que só mostra os produtos (serviços) que são entregues e a estrutura dos seus processos de negócios, mas não como estes são implementados.
Conciso:essencialmente é um 'raio – X" da organização de forma muito compacta e resumida. Pode ser considerado como o "ponto de partida ideal para os gestores, com base no qual, os gestores poderão tomar decisões bem informadas sobre as mudanças desejadas. Os gestores têm um modelo na linguagem que percebem."

Coerente: para o DEMO "uma organização consiste numa integração coerente em três camadas organizacioanais: a B-organização (empresa), a I-organização (informações) e a D-organização (documento)[1]. Estes constituem uma hierarquia coerente, em que a I-organização suporta a B-organização e a D-organização suporta a I-organização."
"A integração coerente dos três aspectos das organizações faz (re) criação e (re) engenharia administrável. Com um (re) design da B-organização, as consequências dessa acção para os sistemas de informação existentes, e para a infra-estrutura de informação, são fáceis de monitorizar e proteger.".

Consistente: "O impacto das mudanças num dos pontos de vista sobre as perspectivas é sempre plena e directamente visível, não há surpresas, nem consequências imprevistas, durante o processo de implementação."

Completo: "para o DEMO a implementação organizacional fornece uma definição completa e clara de competências, autoridades e responsabilidades, a estrutura de informação fornece todas as informações relevantes que os actores precisam".

1 original: the B-organization (business), the I-organization (information) and the D-organization (document).

Lógica da DEMO

De acordo com Saman Sattari Khavas, DEMO baseia-se numa teoria que identifica os princípios e definições do sistema e entidades dentro desse sistema. Esta teoria define o mundo, as entidades existentes na mesma, o comportamento dessas entidades e suas interdependências. [6]
Segundo esta teoria, cada sistema é identificado por um conjunto de elementos que interagem uns com os outros e com os elementos no meio ambiente. O próprio ambiente é composto pelo mesmo tipo de elementos . Os elementos em DEMO são seres humanos que realizam tarefas específicas no sistema e têm responsabilidades específicas. Estes seres humanos são conhecidos pelo papel que desempenham. Assim, os elementos centrais que formam o sistema e o ambiente são os actor roles.

Os Actor roles são capazes de realizar certos tipos de actos, têm a capacidade de interagir uns com os outros através da realização de actos de coordenação (*coordination acts*). Cada acto de coordenação é realizado por dois actores, do iniciante e do destinatário (the

performer and the addressee). Através da realização de um acto de coordenação o actor informa a outra parte sobre sua intenção de uma produção (*production*). Nos *coordination acts* os *actor roles* podem solicitar, a promessa de entregar, pergunta ou declarar uma produção. Produção (*production*) é o resultado de um production act que é realizada por um *actor roles*.

Produção (*production*) é categorizada em três camadas diferentes. Cada camada é diferente das outras pelo nível de inteligência utilizada para a produção. A camada mais alta é chamada camada ontológica (*ontological layer*) na qual a produção é uma inovação ou uma decisão. Na segunda camada ao nível do intelecto é reduzido, apenas ocorre a interpretação dos dados e produção de informações a partir de dados. Esta camada é chamada de *Infological Layer*. A camada mais baixa é chamada de *Datalogical layer*. A produção desta camada é apresentada na secção 4 (resultados,) noemadamente, nas interacções da rede social LinkedIn.

Toda acção é feita com base num acordo entre dois *actor roles* através de uma série de negociações. Este processo é chamado de uma transação (*transaction*). Basicamente, a transação (*transaction*) é composta por vários *coordination acts* em torno de um *production act*. As Transações são desta forma identificáveis pelas suas produções. Uma vez que as produções são associados em as camadas, cada transação também pode ser associado a camadas. Portanto, uma transação pode ser uma *ontological, infological or a datalogical transaction*.

A DEMO caracteriza uma organização como uma rede de actores, cada *actor roles* tem funções específicas. Esses *actor roles* podem originar transações. Uma vez que as transações pertencem a três diferentes camadas, a organização terá 3 camadas. Estas camadas estão representados na figura1. O B-organização separa as transacções pertencentes à *ontological layer*, I-organização é associada com as operações pertencentes à *infological layer* e D-organization inclui as transações na camada *datalogical layer*. Este tipo de representação pode abstrair a essência da organização da implementação, separando a camada de negócios de outras camadas da organização.

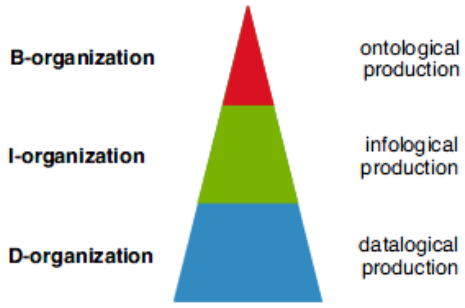


Figura 1. The B- I- and D-Organization, em "Enterprise Ontology: Theory and Methodology", 2006

3. Método de modelação do DEMO

Os modelos em DEMO são na sua maioria uma representação dos conceitos discutidos na secção anterior (lógica DEMO). A DEMO, como mencionado anteriormente, representa cada "organização em quatro modelos parciais: *construction Model, Process Model, Action Model e State Model"*. [2] Ver também figura 2.

O *Modelo de Construção* (CM) específica as transações identificadas, os tipos e as funções que estão associados ao actor roles e as suas ligações com o information banks , (designa o nome colectivo para production bank e coordination bank), em suma, o "CM específica a construção da organização".[2] Ele ocupa o topo do triângulo presente na figura 3.
Significa que o CM é o modelo mais conciso, e que não há nada acima dele. Juntamente com a PM, AM e SM, o CM constitui o completo conhecimento ontológico de uma organização. A linha tracejada separa o triângulo CM em duas partes. A parte esquerda é the Interaction Model (IAM), que expõem as influências activas entre funções de actor, isto é, a execução de transações. Enquanto que a parte direita é The Interstriction Model (ISM), que ostenta as influências passivas entre os actor roles.
O CM é representado por meio do Organization Construction Diagram (OCD), the Transaction Result Table (TRT), e pela Bank Contents Table (BCT).

O *Modelo de Processo (PM)* contém, para cada tipo de transação no CM, o padrão de transações específicas para cada tipo de transacção. O PM também contém as relações causais e condicionais entre transações. Essas relações determinam, além dos padrões da operação, as trajetórias possíveis no C- world. Por outras palavras, a PM específica o espaço de estados (*state space*) e o espaço de transição (*transition space*) do C-world. O PM é colocado logo abaixo do CM no triângulo, porque é o primeiro nível de detalhe da CM, ou seja, compreende o detalhe dos tipos de transação identificada na TRT.
O PM é representado por meio do Process Structure Diagram (PSD).

O *Modelo de Acção (AM)* específica as regras de acção que servem como diretrizes para os actores, contém uma ou mais regras para cada tipo. Estas regras são agrupadas de acordo com os actor roles que se distinguem. A AM é colocado logo abaixo da PM no triângulo, porque é o segundo nível de detalhe do CM, agrega, o detalhe dos passos identificados na PM dos tipos de transação no CM. No nível ontológico de abstração não há nada abaixo do AM.

O *Modelo de Estado (SM)* específica o *state space* do P-mundo: o objecto e tipos de facto, os tipos de resultado, e a coexistência ontológica de regras. O transition space do P-world não está contida no SM e é totalmente derivável a partir do transition space do C- world. A SM é colocado por cima do AM, porque é directamente baseado na AM, que específica todas as classes de objectos, fact types, e as regras de coexistência ontológica que estão contidos na AM. Por outro lado, o SM é colocado logo abaixo do CM (e, portanto, no mesmo nível como o PM), porque também pode ser visto como o detalhe de uma parte da CM, ou seja, the contents of the information banks (coordination and production banks).

O primeiro resultado do método é uma lista dos tipos de transação identificadas e o actor participante, bem como a identificação dos limites da organização, que neste caso é uma rede social profissional, nomeadamente o Linkedin.

Tendo como base este conhecimento, o IAM pode ser feito de imediato. O qual é expresso num Actor Transaction Diagram (ATD) e numa Transaction Result Table- (TRT). De seguida, o Process Structure Diagram-(PSD) é produzido. Seguido da Action Rules Specifications (ARS). As regras de acção são expressas numa linguagem pseudo-

algorítmica/códiga. Posteriormente, o SM é produzido, expresso num Object Fact Diagram- (OFD) e numa Object Property List- (OPL).
Neste ponto somos capazes de completar o PM com a Information Use Table (IUT). Por fim, o ISM é produzido, constituido por um Actor Bank Diagram (ABD) e por uma Bank Contents Table (BCT). Normalmente, o ABD é desenhado como uma extensão do ATD; juntos constituem o Organization Constrution Diagram (OCD).

Como poderá ver ao longo deste documento na seccção 4.2 (resultados) .

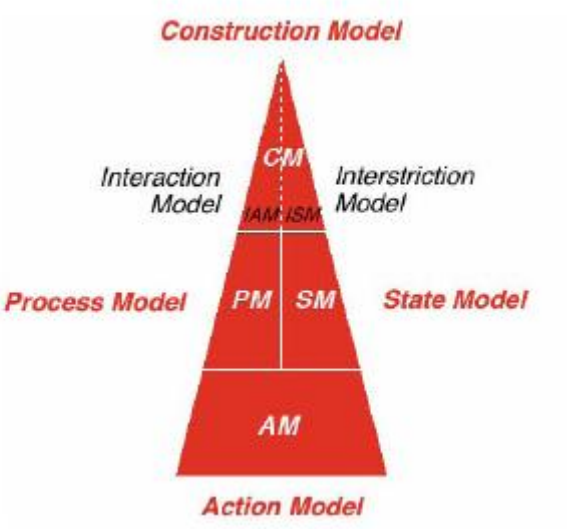


Figura 2. ” The ontological aspect models”, Dietz Jan L.G., em “Enterprise Ontology: Theory and Methodology”, 2006, pág140

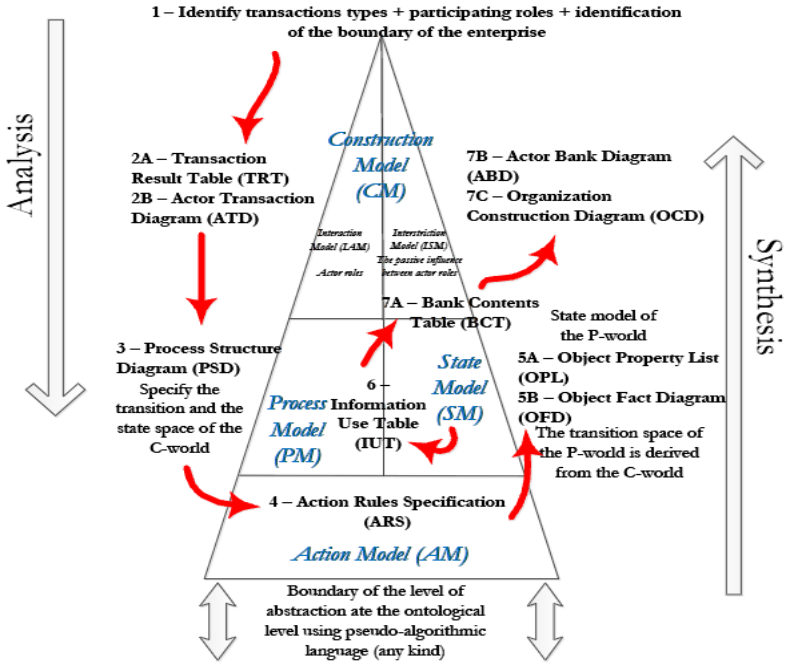


Figura 3. “DEMO methodology”, Sérgio Guerreiro, 2011 [4]

4. Resultados

4.1 Interacções da rede social LinkedIn

4.1.1 Análise ontológicas, infológicas e datalógicas da descrição

A análise ontológica será feita utilizando a coloração das partes apropriadas das descrições, assim, utilizaremos o vermelho para os itens ontológicos (ontological items) , o verde para itens infológicos (infological items), e o azul para itens datalógicos (datalogical items).

Na rede social profissional *LinkedIn* existem diversas funcionalidades disponíveis.
Das várias possibilidades, destacaremos algumas que representam os diferentes casos possíveis no desenvolvimento da modelação de uma rede social.
"Como qualquer rede social, a mesma só faz sentido se existirem [utilizadores] ligados." Como tal, qualquer pessoa [potencial membro] pode <efectuar o registo> no Linkedin. Se todos dados obrigatórios forem enviados (primeiro nome, último nome, nome de utilizador, cidade onde reside, génro, data de nascimento, email, password, número de telemóvel), o potencial membro é automaticamente aceite e registado. Depois o membro pode actualizar o seu perfil, com dados pessoais, e dados literários/escolares, dados profissionais.
O passo seguinte é um membro ''ligar-se'' a outros membros ou convidar outros utilizadores para o seu grupo de pessoas conhecidas / amigas ou que partilhem do mesmos interesses. Para isso, tem de enviar o convite de ligação (conexão) . Para a transacção ser finalizada com sucesso, o destinatário do convite deve aceitar o pedido de ligação do outro utilizador. No caso de o convite ser aceite, os dois utilizadores ficam ligados.
Da mesma forma tem também a possibilidade de remover os amigos do seu grupo. Esta é uma transacção que não carece de aprovação do amigo 'desligado'.
Qualquer membro tem a possibilidade de criar e gerir grupos de interesses. (Group) No caso de um utilizador tentar aderir a um grupo, esta adesão fica sujeita à aprovação do gestor do grupo. Desse grupo já podem fazer parte outros membros.
Também a qualquer momento podem sair de um grupo a que pertençam. Para deixar de estar ligado a um grupo, um membro não tem que esperar qualquer tipo de confirmação, apenas escolher sair.
Qualquer membro tem ainda a possibilidade de fazer recomendações (recommendation) a outros membros, tem também a possibilidade de pedir uma recomendação a outro membro.
Cada membro pode disponibilizar propostas de trabalho (job) . Essas propostas estarão visíveis para membros que se queiram candidatar. Um membro também pode propor-se a uma determinada oferta de trabalho. Terá de escolher a oferta e candidatar-se.
Cada membro pode disponibilizar qualquer tipo de informação (update) em qualquer altura, pode disponibilizar uma noticia, um link, uma frase,… (funcionalidade da rede social Twitter.)
Ao qual outro membro, ou o próprio pode comentar, gostar (Like, funcionalidade do Facebook).
Dadas as funcionalidades dos diferentes tipos de conta, existe a possibilidade dos utilizadores associarem-se a outro tipo de conta. O upgrade de conta implica um pagamento e só depois são disponibilizadas as novas funcionalidades que a conta permite.

4.1.2 The Coordination-Actors-Production Analysis

"A forma adequada de realizar esta análise é o de desenhar uma pequena caixa ou disco ou diamante sobre os pedaços de texto que estão marcados a vermelho no passo anterior."
Utiliza-se este tipo de análise para indicar um actor role, um C-act/result, ou P-act/result.
Para indicar o actor role utiliza-se "[" e "]", para indicar um c-act (coordenação) utiliza-se os parênteses "(" e ")" e para indicar um p-act (produção) utiliza-se s "<" e ">"

Na rede social profissional *LinkedIn* existe diversas funcionalidades disponíveis.

Das várias possibilidades, destaco algumas que representam os diferentes casos possíveis no desenvolvimento da modelação de uma rede social.
"Como qualquer rede social, a mesma só faz sentido se existirem [utilizadores] ligados." Como tal, [qualquer pessoa]/ [potencial membro] pode <efectuar o registo> no Linkedin. Se todos dados obrigatórios forem enviados (primeiro nome, último nome, nome de utilizador, cidade onde reside, génro, data de nascimento, email, password, número de telemóvel), o [potencial membro] é automaticamente aceite e registado. Depois de <realizar o login> [o membro] (pode) (actualizar/modificar o seu perfil), com dados pessoais, e dados literários/escolares, dados profissionais.
O passo seguinte é um [membro] ''<ligar-se>'' a outros [membros] ou <convidar> outros [utilizadores] para o seu grupo de [pessoas conhecidas / amigas] ou que partilhem do mesmos interesses. Para isso, tem de (enviar) o convite de ligação (conexão) . Para a transacção ser finalizada com sucesso, o destinatário do convite deve aceitar o pedido de ligação do outro [utilizador]. No caso de o convite ser aceite, os dois [utilizadores] ficam ligados.
Da mesma forma tem também a possibilidade de <remover> os [amigos] do seu grupo. Esta é uma transacção que não carece de aprovação do amigo 'desligado'.
Qualquer [membro] tem a possibilidade de <criar> e (gerir) grupos de interesses. (Group) No caso de um utilizador tentar <aderir> a um grupo, esta adesão fica sujeita à aprovação do gestor do grupo. Desse grupo já podem fazer parte outros membros.
Também a qualquer momento podem (sair) de um grupo a que pertençam. Para deixar de estar ligado a um grupo, um membro não tem que esperar qualquer tipo de confirmação, apenas escolher sair.
Qualquer membro tem ainda a possibilidade de <fazer recomendações> (recommendation) a outros membros, tem também a possibilidade de (pedir) uma recomendação a outro [membro].
Cada membro pode <disponibilizar> propostas de trabalho (job) . Essas propostas estarão visíveis para membros que se queiram candidatar. Um [membro] também pode (propor-se) a uma determinada oferta de trabalho. Terá de escolher a oferta e candidatar-se.
Cada [membro] pode <disponibilizar> qualquer tipo de informação (update) em qualquer altura, pode disponibilizar uma noticia, um link, uma frase,… (funcionalidade da rede social Twitter.) Ao qual outro [membro], ou o próprio pode (comentar, gostar) (Like, funcionalidade do Facebook).
Dadas as funcionalidades dos diferentes tipos de conta, existe a possibilidade dos [utilizadores] <associarem-se> a outro tipo de conta. O upgrade de conta implica um pagamento e só depois são disponibilizadas as novas funcionalidades que a conta permite.

4.2 Modelação

4.2.1 Transaction Result Table – TRT do LinkedIn

Na seguinte tabela estão presentes as diversas transações e os correspondentes resultados

Transaction Type	**Result Type**
T01 Request Membership	R01 *membership M has been started*
T02 Connection start	R02 *the connection C has been started*
T03 Invitation	R03 *the invitation I has been sent*
T04 Group	R04 *the group G has been done*
T05 Job	R05 *the job J has been done*
T06 Upgrade Account	R06 *the account for membership M was upgraded*
T07 Modify a profile	R07 *the profile P has been modified*
T08 Recommendation	R08 *the recommendation R has been done*
T09 Update status	R09 *the update U has been done*
T10 Control (Add + comment+ change + delete +like)	R10 *the control CT has been done*

4.2.2 Complete detailed Actor Transaction Diagram –ATD

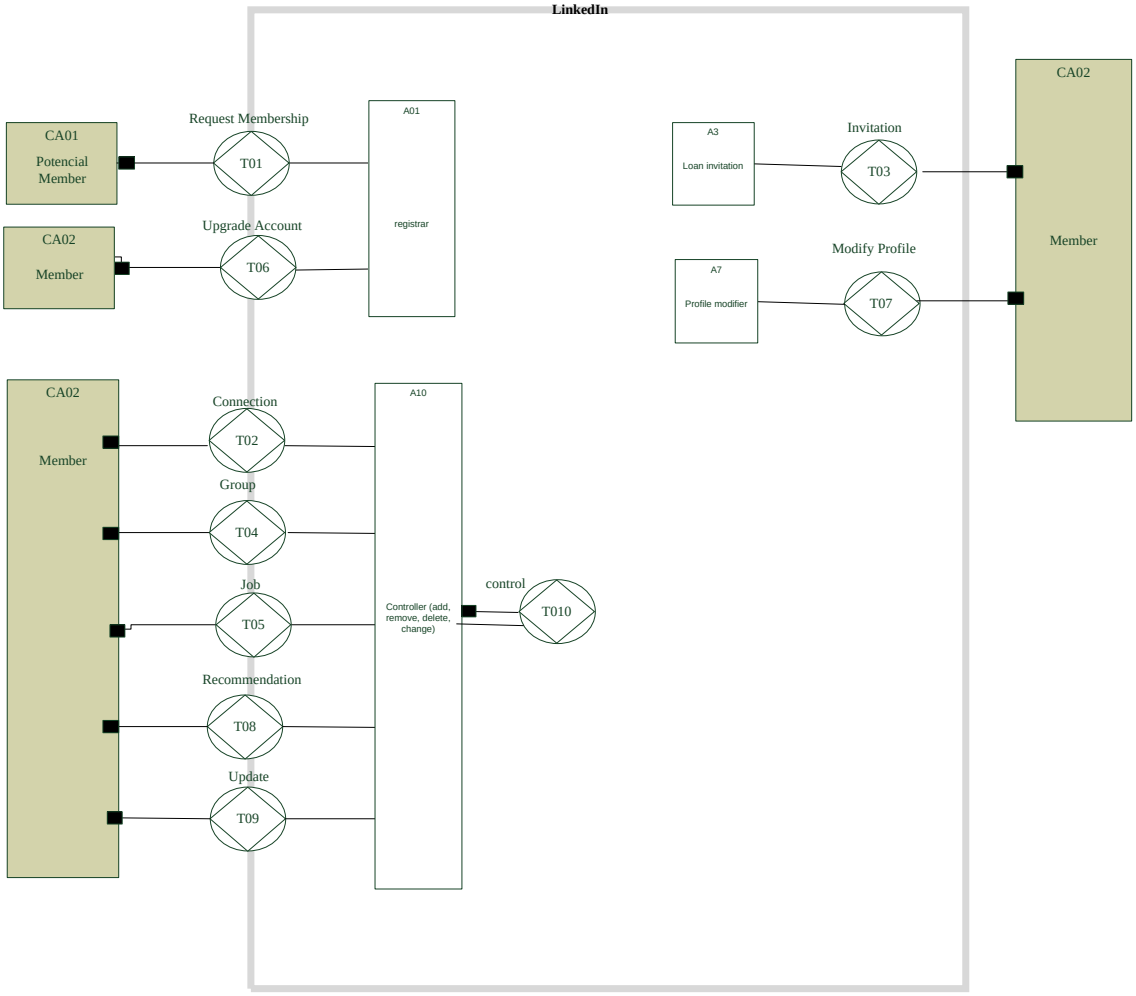


Figura 4. Complete detailed Actor Transaction Diagram –ATD

4.2.2.1 Global ATD

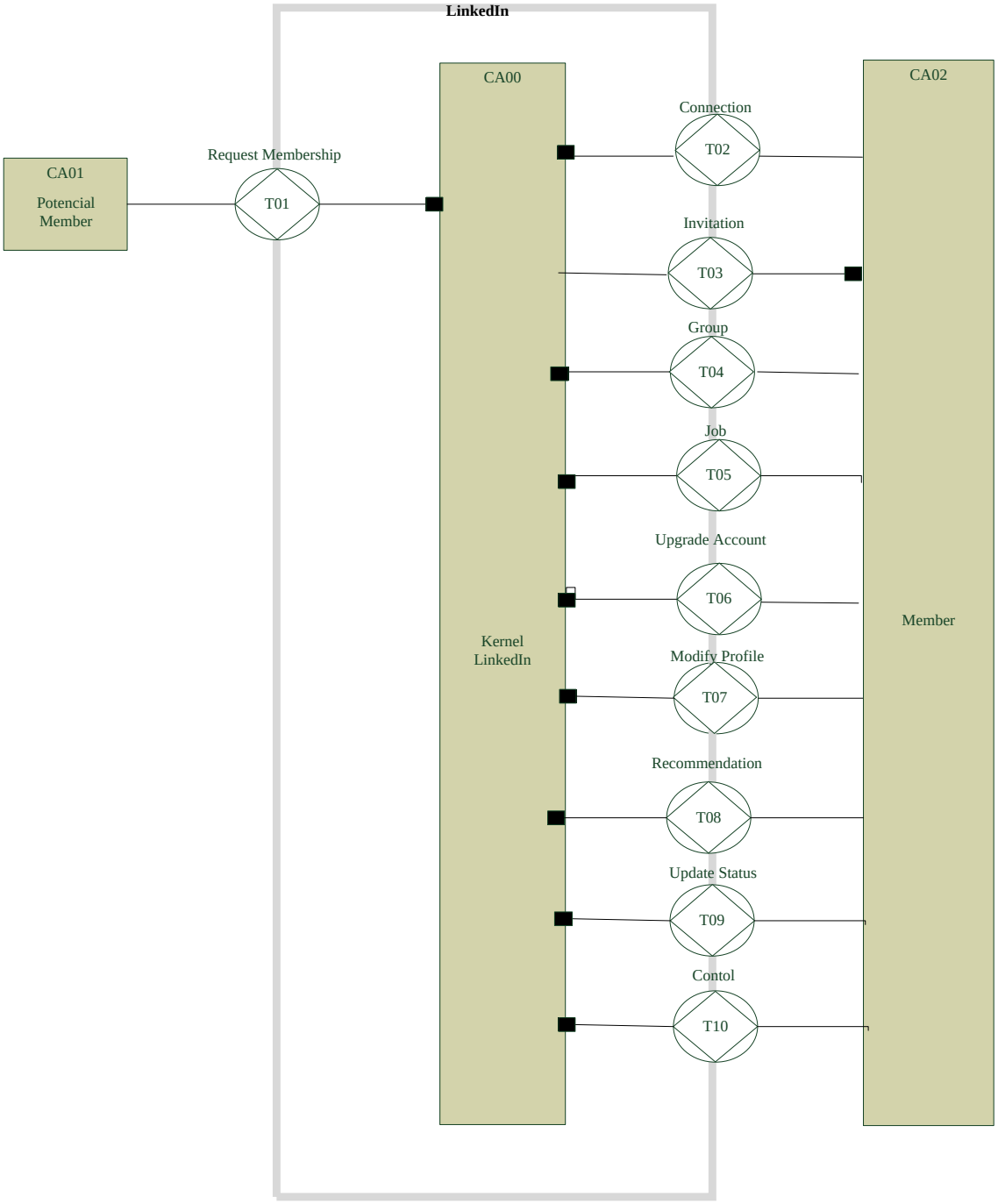


Figura 5. Global ATD

Breve explicação do ATD.

A numeração (CA00, CA01 e CA02) dos composite actors é arbitrária.

Os nomes das actor roles não fazem parte formal de um CM, no entanto, o uso apropriado de nomes podem melhorar considerevelmente a legibilidade do diagrama.

O composite actor CA01 (potencial membro) representa a pessoa que quer tornar-se membro do Linkedin. Obviamente, o iniciador do processo é T01 (registro no Linkedin). CA02 (membro) representa os membros reais do Linkedin. Ele é o iniciador de trasações como por exemplo criar conexões (T02) com outros membros ou potenciais membros, criar um grupo (T04); criar/publicar (T05) propostas de trabalho, mudar o seu perfil, acrescentando informações pessoais e ou profissionais(T07), pedir ou fazer recomendações (T08) a outros membros facilitando e dando a credibilidade desses membros no mercado de trabalho; é iniciador de publicar o estado do perfil (T09), o membro pode ainda solicitar a alteração do tipo de conta (T06), e por fim pode solicitar transacções de controlo, tais como, adicionar, remover, modificar,apagar e like. E o composite Actor CA00 representa o admin da plataforma Linkedin

4.3 Process Model: Process Structure Diagram - PSD

4.3.1 PSD A01

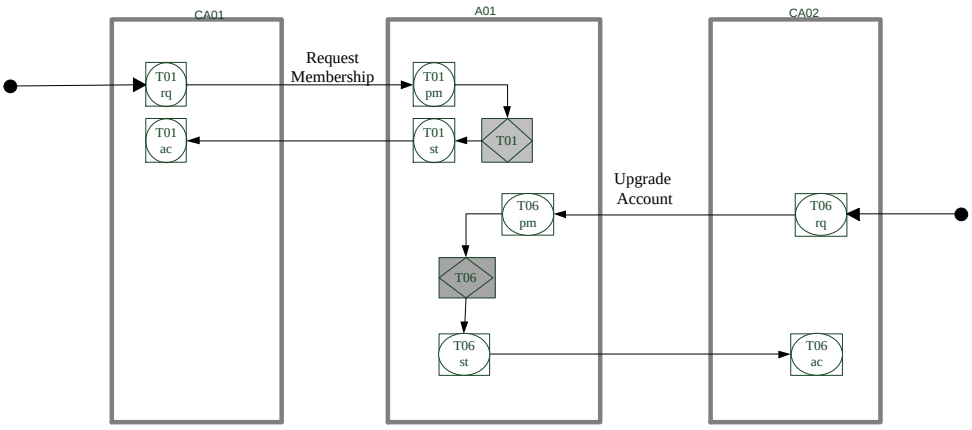


Figura 6: PSD A01

4.3.2 PSD do A03

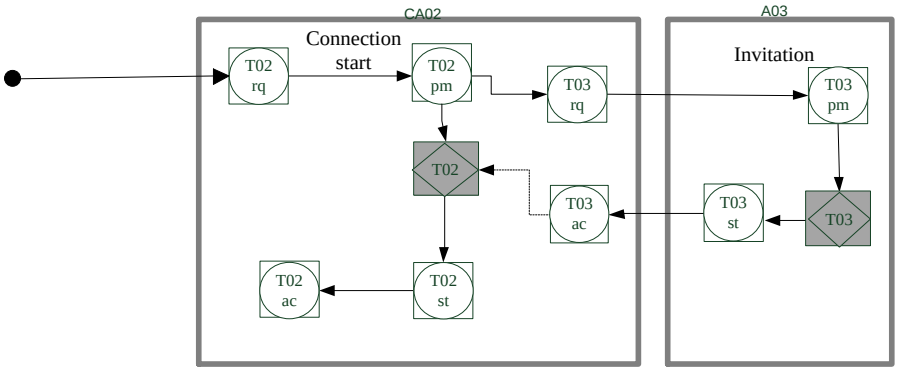


Figura 7: PSD A03

4.3.3 PSD do A07

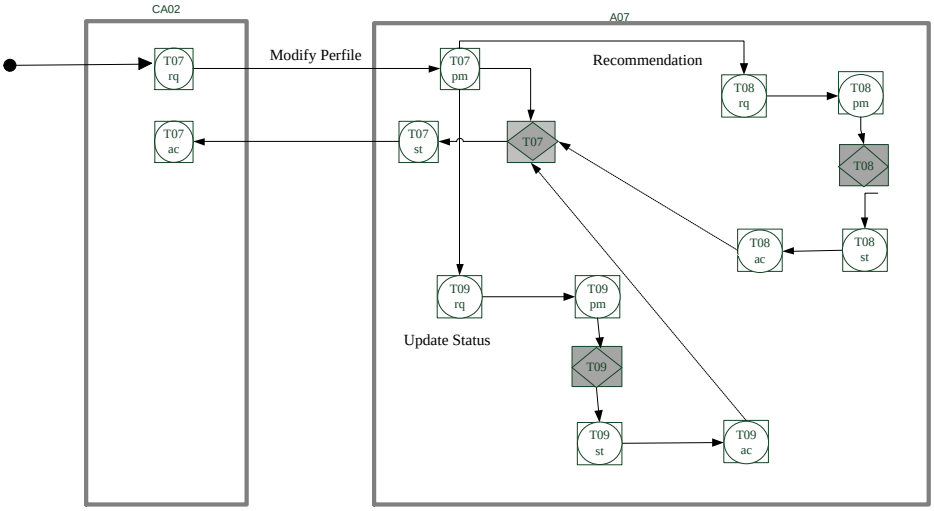


Figura 8: PSD A07

4.3.4 PSD do A10

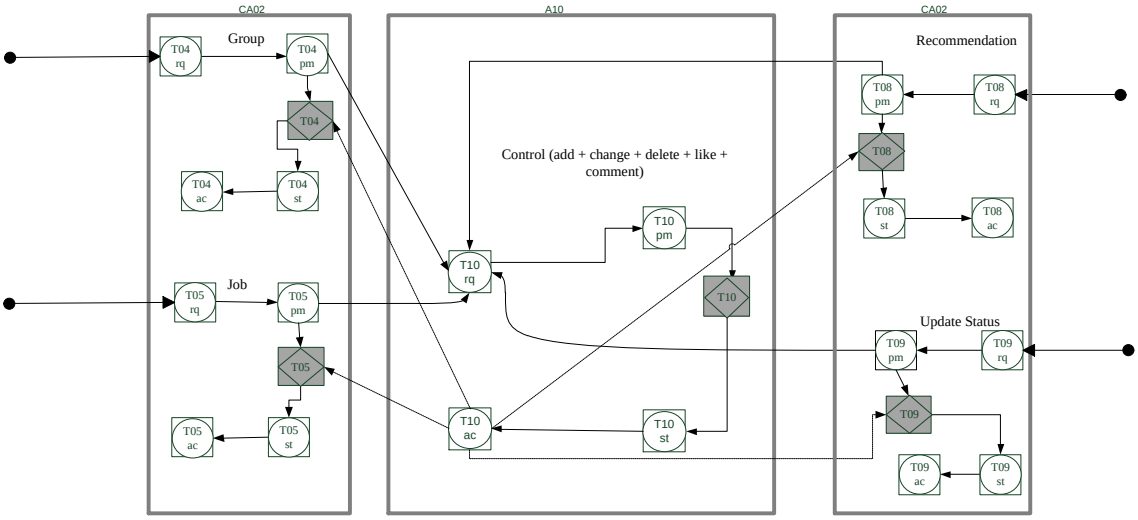


Figura 9: PSD A10

Breve explicação dos PSD's

Começamos por considerar todas as transações representadas no CM, como o suporte do processo de "negócio". O iniciador é mais uma vez o CA01- Potencial membro com a transação T01/rq.

Apartir do momento em que um indíviduo é membro Linkedin, está autorizado a cumprir o composite actor CA02, e assim iniciar as transações do tipo T02 à T10.

Por exemplo para o membro criar uma conexão (T02/rq) com outro membro desencadeia a transação de convidar o membro (T03/rq).

No caso do update status do perfil, o membro pode partilhar uma publicação, isto é, uma alteração do estado (T09/rq) e outro membro gostar (like) ou querer comentar,o que resulta numa actualização da publicação (T10/ac- T09 execute). De igual modo permite ao membro após publicar a alteração do estado apagá-la, modificá-la e remover os comentários (T10/rq).

No caso do upgrade de conta (T06/rq) o membro pode desejar alterar o tipo de conta e para tal tem que pagar uma taxa anual, o upgrade só é concedido após o pagamento da taxa.

No caso de um membro pretender criar, apagar, modificar, o remover um grupo (T04/rq e T10/ac) o mesmo termina após o "control"(T10/rq) tenha sido realizado .

O caso do membro solicitar uma oferta de trabalho(T05/rq) pode desencadear a execução do control, no caso de adicionar nova oferta, alterar algum parâmetro da mesma, ou mesmo apagar a oferta (T10).

Os casosT04/rq, T08/rq, T15/rq são transações identicas ao caso da oferta(T05) de trabalho bem como ao caso da publicação do estado (T09)

No caso T07/rq, o membro poderá pretender apenas alterar ou acrescentar algum conteúdo do seu perfil, como pode alterar o estado (T09) ou adicionar uma recomendação no perfil de outro membro.

4.4 Information Use Table –IUT do LinkedIn

Object class, fact type or result type	**Process Steps**
Person	T01/ rq
P é membro da rede social	T01 /rq
Member	T01/ ac
Relationship	T03 /rq,
Group	T04/rq, T10/ac
M add a group G	T04/ac, T10/ac
M out a G	T04/rq, T10/ac
Jobs	T05/rq, T10/ac
M post a J	T05/rq, T10/ac
Profile	T07/rq
M modify P	T07/rq, T10/ac
Recommendation	T08/rq, T10/ac
M sent a R for other M	T08/rq, T10/ac
Update status	T09/rq, T10/ac
M post a u	T09/rq, T10/ac
M commented on the update of the other M	T09/rq, T10/ac
M like on the update of the other M	T09/rq, T10/ac
M delete U	T09/rq, T10/ac
Account	T06/rq

4.5 O Action Model –AM do LinkedIn

Action Rules for action role A01

T01- Request Membership

on requested T01(M) **with** member (new M) = P
if(email(P) <email_exit > && user(P)<user_nexist> ==0) **then** promise T01 (M)
else decline T01 (M)
end if
no

Específica o que deve ser feito quando a T01 é solicitada. A primeira coisa a fazer é gerar uma entidade do tipo de associação. A entidade é indicada por M. O potencial membro será o membro desta associação que é indicada por P. Aplicações de adesão em que o email actual do membro já existe é lhe recusada a adesão. Todos as outras aplicações são seguidas por uma promessa.

on promised T01(M)
execute T01 (M)
state T01 (M)
no

se os dados não se encontram na base de dados users segue-se o state.

on stated T01(M)
if <decision of the LinkedIn Kernel is acceptable> t**hen** accept T01 (M)
else reject T01 (M)
end if
no

Se os dados não se encontram na base de dados e cumprem os requesitos obrigatórios o potencial membro é aceite, se não é recusado. Após o potencial membro ter sido aceite fica com a sessão activa onde pode ter acesso a todas as funcionalidades do Linkedin.

T06-Upgrade account

on requested T06(A) **with** member (M)
if (member(M).upgrade_account()) **then** promise T0.6(A)
end if
no

O membro pretende fazer um upgrade de conta é desencadeado a promessa,

on promised T06 (A)
if (#Account (A).upgrade_accountPS() ==0) **then**
while{ (fee == pfee)
Notifier.upgrade_account(account_id,upgrade_id.deliver);
} **end while**
execute T06 (A)
state T06 (A)
end if
no

Se pagou a taxa a alteração é aceite, se pagou o valor errado é rejeitado.

on stated T06(A)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T06 (A)
end if
no

como a alteração de conta foi aceite envia um accept para a transação 06, confirmação do estado da alteração

on accepted T06(A)
execute T06(A)
state T06 (A)
no

Action Rules for action role A03

T02- Add connection e T03 -Invitation
on requested T02(R) **with** member (M) = P
if (member(P).add_con ()) **then** promise T02(R)
end if
no

quando o membro pretende ligar-se a outro membro desencadeia-se a promessa.

on promised T02(R)
request T03 (R)
execute T02(R)
no

Se a conexão não existir desecnadeia o convite de conexão

on requested T03(R)
if (member(P).add_rel()) **then** promise T03(R)
else decline T03(R)
end if
no

é solicitado ao membro 2 o pedido de conexão ao membro passando à promessa, se o membro cancelar o pedido o processo é negado

on promised T03(R)
while (#member(P).add_rel ()) {
invite = Invitation.find(invite_id)
invite.send!
} **end while** execute T03(R)
state T03 (R)
no

Se o relacionamento não existir a transação é executada

on stated T03(R)
if <decision of the member 2 is acceptable > **then** accept T03 (R)
else reject T03(R)
end if

no

o membro 2 pode aceitar ou rejeitar o relacionamento com o outro membro.

on accepted T03 (R)
    if <decision of the member 2 with LinkedIn Kernel is acceptable> **then**
    execute T02 (R)
    state T02 (R)
    end if
no

No caso de aceitar o relacionamento essa informação faz executar a T02

on stated T02(R)
    if <decision of the LinkedIn Kernel is acceptable > **then** accept T02(R)
    end if
no

Action rules for action role A10

T04- Group (option add group) e T10- Control (add + delete + comment + like + change)

on requested T04(G) **with** member (M)
    if (#member(M) .add_group() ==0) **then** promise T04(G)
    else reject T04(G)
    end if
no

Quando o membro pretende criar um grupo e já existe um grupo com o nome que ele escolheu a trasação é rejeitada, se não é processada a promessa

on rejected T04(G)
    stop T04(G)
no

on promised T04(G)
    request T10(G)
    state T04(G)
no

on requested T10(CT)
    if (#Group(G).add_group() ==0) **then** promise T10(CT)
    else reject T10(CT)
    end if
no

Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
    while (#Group(G) .add_group();{
        validates group_name;
        validate group_describe;
        creategroup();
    }**end while**
    execute T10(CT)

state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T04(G)
end if
no

on stated T04(G)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T04 (G)
end if
no

T04- Group (option delete group) e T10- Control (add + delete + comment + like + change)

on requested T04(G) **with** member (M)
if (member(M) .del_group() ==0) **then** promise T04(G)
else reject T04
end if
no

Quando o membro pretende apagar um grupo e esse grupo não existe a transação é rejeitada, se não é processada a promessa

on rejected T04(G)
stop T04(G)
no

on promised T04(G)
request T10(CT)
state T04(G)
state T04(G)
end if
no

on requested T10(CT)
if (#Group(G). del_group() ==0) **then** promise T10(CT)
else decline T10(CT) (caso em que o grupo já existe, evitar duplicados)
end if
no

Quando o membro pretende apagar um grupo é processada a promessa, se

on promised T10(CT)
while (#Group(G) .del_group()){
group = group.find(group _id);
group.remove_all_traces;
group.destroy;
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T04(G)
end if
no

on stated T04(G)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T04 (G)
end if
no

T05- Job (option add job) e T10- Control (add + delete + comment + like + change)

on requested T05(J) **with** member (M)
if (member(M).add_job() ==0) **then** promise T05(J)
else reject T05(J)
end if
no

Quando o membro pretende criar uma proposta de trabalho e já existe uma proposta com o nome que ele escolheu a trasação é rejeitada, se não, é processada a promessa

on rejected T05(J)
stop T05(J)
no

on promised T05(J)
request T10(J)
state T05(J)
no

on requested T10(CT)
if (#Job(J) .add_job () ==0) **then** promise T10(CT)
else decline T10(CT) (caso em que o job já existe, evitar duplicados)
end if
no
Quando o membro pretende criar uma proposta é processada a promessa, se

on promised T10(CT)
while (#Job(J) .add_job()){
validates job_name;
validate job_describe;
createjob= new job(job_id ++,job_name, job_describe);
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T05(J)
end if

no

on stated T05(G)
if <decision of the LinkedIn Kernel is acceptable ≥ accept T05 (J)
end if
no

T05- Job (option add comment job) e T10- Control (add + delete + comment + like + change)

on requested T05(J) **with** member (M)
if (member(M). add_comment_job() ==0) **then** promise T05(J)
else reject T05
end if
no
Quando o membro pretende adicionar um comentário uma proposta de trabalho e essa proposta já não existe a transação é rejeitada, se não, é processada a promessa

on rejected T05(J)
stop T05(J)
no

on promised T05(J)
request T10(J)
state T05
no

on requested T10(CT)
if (#Job(J) .add_ comment _job() ==0) **then** promise T10(CT)
else decline T10(CT) (caso em que o job já existe, evitar duplicados)
end if
no
Quando o membro pretende adicionar um comentário a uma proposta de trabalho é processada a promessa, se

on promised T10(CT)
while (#Job(J) .add_job()){
validates job_name;
validate job_describe;
def self.perform(recipient_id, sender_id, comment_id)
Notifier.comment_on_job(recipient_id, sender_id, comment_id).deliver);
end
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T05(J)
end if
no

on stated T05(G)

if <decision of the LinkedIn Kernel is acceptable > **then** accept T05 (J)
end if
no

T05- Job (option delete job) e T10- Control (add + delete + comment + like + change)

on requested T05(J) **with** member (M)
if (member(M) .del_job() ==0) **then** promise T05(J)
else reject T05(J)
end if
no
Quando o membro pretende apagar uma proposta de trabalho é processada a promessa, se

on promised T05(J)
request T10(J)
state T05
no
on requested T10(CT) **with** member (M)
if (#Job(J).del_job() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no
Quando o membro pretende apagar a proposta é processada a promessa, se

on promised T10(CT)
while (#Job(J) .del_job()){
job = job.find(job_id);
job.remove_all_traces;
job.destroy;
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T05(J)
end if
no

on stated T05(J)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T05 (J)
end if
no

T05- Job (option change job) e T10- Control (add + delete + comment + like + change)

on requested T05(J) **with** member (M)
if (member(M).change_job()==0) **then** promise T05(J)
else reject T05
end if
no

quando o membro pretende modificar algum parametro da proposta de trabalho publicada é processada a promessa, se

on promised T05(J)
	if(#Job(J) .change_job() !0) **then** request T10(J)
	state T05
	end if
no

on requested T10(CT)
	if (#Job(J) .change_job() ==0) **then** promise T10(CT)
	else decline T10(CT)
	end if
no
Quando o membro modifcar a proposta publicada é processada a promessa, se

on promised T10(CT)
	while (#Job(J) .change_job()){
		mofify job_name;
		modify job_describe;
		modify job_date;
		modify job_date_end;
	}**end while**
	execute T10(CT)
	state T10(CT)
no

on stated T10(CT)
	if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
	execute T05(J)
	end if
no

on stated T05(J)
	if <decision of the LinkedIn Kernel is acceptable > **then** accept T05 (J)
	end if
no

T08- Recommendation (option add) e T10- Control (add + delete + comment + like + change)

on requested T08(RE) **with** member (M)
	if (member(RE) .add_recommendation()) **then** promise T08 (RE)
	else decline T08(RE)
	end if
no
O membro pretende fazer uma recomendação a outro membro é desencadeado a promessa,

on promised T08 (RE)
	request T10(CT)
	execute T08(RE)
	state T08 (RE)
no

on requested T10(CT)
if (#Recommendation(RE) .add_recommendation() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no
Quando o membro pretende adiconar uma recomendação no perfil de outro membro é processada a promessa, se

on promised T10(CT)
while (# Recommendation(RE) .add_recommendation())
{validates_name_friend;
validates_profile_friend;
validates_idfriend;
add_data;
sentrecommendation;
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T08(RE)
end if
no

on stated T08(RE)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T08 (RE)
else decline T08 (RE)
end if
no

Em caso de sucesso o a transação é aceite, caso contrário é recusada

T08- Recommendation (option add comment on recommendation) e T10- Control (add + delete + comment + like + change)

on requested T08(RE) **with** member (M)
if (member(RE).add_ comment_recommendation() ==0) **then** promise T08 (RE)
else decline T08(RE)
end if
no
O membro pretende comentar uma recomendação a outro membro é desencadeado a promessa,

on promised T08 (RE)
if (#Recommendation (RE.add_ comment_recommendation() !0) **then** request T10(CT)
execute T08(RE)
state T08 (RE)
end if
no

on requested T10(CT)
if (#Recommendation(RE).add_ comment_recommendation() ==0) **then** promise T10(CT)
else decline T10(CT)

end if
no
Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
while (# Recommendation(RE).add_ comment_recommendation(){
validates_name_friend;
validates_profile_friend;
validates_idfriend;
add_data;
def self.perform(recipient_id, sender_id, comment_id)
Notifier.comment_on_post(recipient_id, sender_id, comment_id).deliver;
}**end while**
execute T10(CT)
state T10(CT)

no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT)
execute T08(RE)
end if
no

on stated T08(RE)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T08 (RE)
else decline T08 (RE)
end if
no

Em caso de sucesso o a transação é aceite, caso contrário é recusada

T08- Recommendation (option delete) e T10- Control (add + delete + comment + like + change)

on requested T08(RE) **with** member (M)
if (member(RE). del_recommendation() ==0) **then** promise T08 (RE)
else decline T08(RE)
end if
no

O membro pretende fazer/pedir uma recomendação a outro membro é desencadeado a promessa,

on promised T08 (RE)
if (#Recommendation (RE). del_recommendation() !0) **then** request T10(CT)
state T08 (RE)
end if
no

Na promessa é executada a transação

on requested T10(CT)
if (#Recommendation(RE). del_recommendation() ==0) **then** promise T10(CT)
else decline T10(CT)

```
        end if
no
```

Quando o membro pretende criar um grupo é processada a promessa, se

```
on promised T10(CT)
        while (# Recommendation(RE). del_recommendation()[
                recommendation = recommendation.find(recommendation_id);
                recommendation.remove_all_traces;
                recommendation.destroy;
        }end while
        execute T10(CT)
        state T10(CT)
no

on stated T10(CT)
        if <decision of the LinkedIn Kernel  is acceptable > then accept T10(CT)
        execute T08
        end if
no

on stated T08(RE)
        if <decision of the LinkedIn Kernel  is acceptable > then accept T08 (RE)
        else decline T08 (RE)
        end if
no
```

Em caso de sucesso o a transação é aceite, caso contrário é recusada

T09- Update status (option add) e T10- Control (add + delete + comment + like + change)

```
on requested T09 (U) with member (M)
        if (member(M).add_post() ==0) promise T09U)
        else decline T09 (U)
        end if
no
```

O membro pretende publicar algo (opinião, um link, uma notícia) é desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

```
on declined T09 (U)
        quit T09 ( U)
no
```

o processo é eliminado

```
on promised T09 (U)
        if( #Update (U). add_post() !0) then request T10 (CT)
        state T09 ( U)
        end if
no
```

na promessa é executada a intenção de publicar algo (opinião, um link, uma notícia)

```
on requested T10(CT)
```

```
        if ( #Update (U).add_post() = =0) then promise T10(CT)
        else decline T10(CT)
        end if
no
```

Quando o membro pretende criar um grupo é processada a promessa, se

```
on promised T10(CT)
        while (#Update(U).add_post(){
                validates_length_of :post,  :maximum => 2000;
                post_id => @status.id
                add_data;
                sentpost;

//notificar todos os amigos do membro
   def self.perform(user_ids, object_klass, object_id, person_id)
          members= Member.where(:id => member_ids)
          object = object_klass.constantize.find_by_id(object_id)
          person = Person.find_by_id(person_id)
          members.each{|user| Notification.notify(member, object, person) }
        }end while
        execute T10(CT)
        state T10(CT)
no

on stated T10(CT)
        if <decision of the LinkedIn Kernel  is acceptable > then accept T10(CT)
        execute T09
        end if
no

on stated T09(U)
        if <decision of the LinkedIn Kernel  is acceptable> then accept T09 (U)
        else decline T09 (U)
        end if
no
```

T09- Update status (option add comment on post) e T10- Control (add + delete + comment + like + change)

```
on requested T09 (U) with member (M)
        if (member(M). add_ comment_ post() ==0) then promise T09U)
        else decline T09 (U)
        end if
no
```

O membro pretende publicar algo (opinião, um link, uma notícia) é desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

```
on declined T09 (U)
        quit T09 ( U)
no
```

o processo é eliminado

```
on promised T09 (U)
```

request T10 (CT)
state T09 (U)
no

na promessa é executada a intenção de publicar algo (opinião, um link, uma notícia)

on requested T10(CT)
if (#Update (U). add_ comment_ post() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no

Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
while (#Update(U) .add_ comment_ post(){
validates_length_of :post, :maximum => 2000;
post_id => @status.id
add_data;
def self.perform(recipient_id, sender_id, comment_id)
Notifier.comment_on_post(recipient_id, sender_id, comment_id).deliver
sentpost;

//notificar todos os amigos do membro
def self.perform(user_ids, object_klass, object_id, person_id)
members= Member.where(:id => member_ids)
object = object_klass.constantize.find_by_id(object_id)
person = Person.find_by_id(person_id)
members.each{|user| Notification.notify(member, object, person) }
end
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT) execute T09
end if
no

on stated T09(U)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T09 (U)
else decline T09 (U)
end if
no

T09- Update status (option like on post) e T10- Control (add + delete + comment + like + change)

on requested T09 (U) **with** member (M)
if (member(M).add_ like_ post() ==0) **then** promise T09U)
else decline T09 (U)
end if
no

O membro pretende publicar algo (opinião, um link, uma notícia) é desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

on declined T09 (U)
quit T09 (U)
no

o processo é eliminado

on promised T09 (U)
request T10 (CT)
state T09 (U)
no

na promessa é executada a intenção de publicar algo (opinião, um link, uma notícia)

on requested T10(CT)
if (#Update (U).add_ like_ post() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no

Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
while (#Update(U).add_ like_ post(){
validates_post;
def self.perform(recipient_id, sender_id, like_id)
Notifier.liked(recipient_id, sender_id, like_id).deliver
end
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT) execute T09
end if
no

on stated T09(U)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T09 (U)
else decline T09 (U)
end if
no

T09- Update status (option delete on post) e T10- Control (add + delete + comment + like + change)

on requested T09 (U) **with** member (M)
if (member(M). del_post() ==0) **then** promise T09U)
else decline T09 (U)
end if
no

O membro pretende publicar algo (opinião, um link, uma notícia) é desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

on declined T09 (U)
quit T09 (U)
no
o processo é eliminado

on promised T09 (U)
request T10 (CT)
state T09 (U)
end if
no
na promessa é executada a intenção de publicar algo (opinião, um link, uma notícia)

on requested T10(CT)
if (#Update (U). del_ post() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no
Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
while (#Update(U).del_ post(){
post = post.find post _id);
post.remove_all_traces;
post.destroy;
}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT) execute T09
end if
no

on stated T09(U)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T09 (U)
else decline T09 (U)
end if
no

Action Rules for actor role A07

on requested T07 (MP) **with** member (M)
if (member(M). modify_perfile() ==0) **then** promise T07(MP)
else decline T07 (MP)
end if
no
O membro poderá pretender modificar o seu perfil, adicionar alguma informação ou publicar algo (opinião, um link, uma notícia) ou a opção de adicionar, apagar uma recomendação é

desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

on declined T07 (MP)
quit T07 (MP)
no
o processo é eliminado

on promised T07 (MP)
Select case Perfile
Case modifyperfile
execute T07(MP)
state T07(MP)
Case mperfile_post
request T09 (MP)
state T07(MP)
Case mperfile_recommendation
request T08 (MP)
state T07(MP)
end Select
no

Na promessa é executada a modificação de algum parâmetro como opcção por defeito, o membro pode pretender alterar o seu estado de perfil, por exemplo, tem intenção de publicar algo (opinião, um link, uma notícia) ou ainda pode apagar uma recomendação que lhe tenham feito, ou adicionar uma recomendação no perfil de outro membro.

Case mperfile_post

on requested T09 (U) **with** member (M)
if (member(M).add_post() ==0) **then** promise T09U)
else decline T09 (U)
end if
no

O membro pretende publicar algo (opinião, um link, uma notícia) é desencadeado a promessa, caso o membro veja que um membro já publicou algo semelhante cancela a transação.

on declined T09 (U)
quit T09 (U)
no
o processo é eliminado

on promised T09 (U)
request T10 (CT)
state T09 (U)
end if
no
na promessa é executada a intenção de publicar algo (opinião, um link, uma notícia)

on requested T10(CT)
if (#Update (U). add_post() ==0) **then** promise T10(CT)
else decline T10(CT)
end if
no

Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
while (#Update(U). add_post(){
validates_length_of :post, :maximum => 2000;
post_id => @status.id
add_data;
sentpost;

//notificar todos os amigos do membro
def self.perform(user_ids, object_klass, object_id, person_id)
members= Member.where(:id => member_ids)
object = object_klass.constantize.find_by_id(object_id)
person = Person.find_by_id(person_id)
members.each{|user| Notification.notify(member, object, person) }

}**end while**
execute T10(CT)
state T10(CT)
no

on stated T10(CT)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT) execute T09
end if
no

on stated T09(U)
if <decision of the LinkedIn Kernel is acceptable > **then** accept T09 (U) execute T07
else decline T09 (U)
end if
no

Case mperfile_recommendation

on requested T08(RE) **with** member (M)
if (member(RE). del_recommendation() ==0) **then** promise T08 (RE)
else decline T08 (RE)
end if
no

O membro pretende fazer/pedir uma recomendação a outro membro é desencadeado a promessa,

on promised T08 (RE)
request T10(CT)
state T08 (RE)
no

Na promessa é executada a transação

on requested T10(CT)
if (#Recommendation(RE) .del_recommendation( ) ==0) **then** promise T10(CT)
else decline T10(CT)
end if

no

Quando o membro pretende criar um grupo é processada a promessa, se

on promised T10(CT)
    while (# Recommendation(RE) .del_recommendation(){
        recommendation = recommendation.find(recommendation_id);
        recommendation.remove_all_traces;
        recommendation.destroy;
    }**end while**
    execute T10(CT)
    state T10(CT)
no

on stated T10(CT)
    if <decision of the LinkedIn Kernel is acceptable > **then** accept T10(CT) execute T08
    end if
no

on stated T08(RE)
    if <decision of the LinkedIn Kernel is acceptable > **then** accept T08 (RE) execute T07
    end if
no

Em caso de sucesso o a transação é aceite, caso contrário é recusada

on stated T07(U)
    if <decision of the LinkedIn Kernel is acceptable > **then** accept T07 (MP)
    end if
no

4.6 State Model

4.6.1 Object Property List - OPL do LinkedIn

Property type	Object class	Scale
Name	PERSON	STRING
Email	PERSON	STRING
person (*)	PERSON	NUMBER
User	MEMBER	STRING
Login	PERSON	STRING
Password	MEMBER	NUMBER
Members (*)	MEMBER	NUMBER
Account (*)	ACCOUNT	NUMBER
Fee	ACCOUNT	NUMBER
Relationship	RELATIONSHIP	NUMBER
Invite(*)	RELATIONSHIP	NUMBER
#Group (*)	GROUP	STRING
Group(*)	GROUP	NUMBER
#Job (*)	JOB	STRING
Job(*)	JOB	NUMBER
Createjob (*)	JOB	STRING
#Recommendation	RECOMMENDATION	STRING
Recommendation(*)	RECOMMENDATION	NUMBER
Comment	CONTROL	STRING
#Update (*)	UPDATE	STRING
post (*)	UPDATE	NUMBER
Like	CONTROL	NUMBER
#Perfile (*)	PERFILE	STRING

person (P)= Person.find_by_id(person_id)
members= Member.where(:id => member_ids)
Account (A) -> upgrade_accountPS =0
invite = Invitation.find(invite_id)
#Group(G) -> add_group , #Group(G) -> del_group =0
Group=group.find(group_id);
#Job(J) -> add_job =0, #job(J) -> del_job =0, #Job(J) -> add_ comment _job =0, **(** #Job(J) -> change_job =0
job = job.find(job_id);
createjob= new job(job_id +job_name+job_describe);
#Recommendation (RE) - > add_recommendation =0, #Recommendation (RE) - > del_recommendation =0, #Recommendation (RE) -> add_ comment_recommendation =0, recommendation = recommendation.find(recommendation _id);
#Update (U) -> add_post =0, #Update (U) -> add_ comment_ post =0, **(** #Update (U) -> add_ like_ post =0
post = post.find post _id
#Perfile -> modify_perfile =0

4.6.2 Object Fact Diagram –OFD complete (with result types)

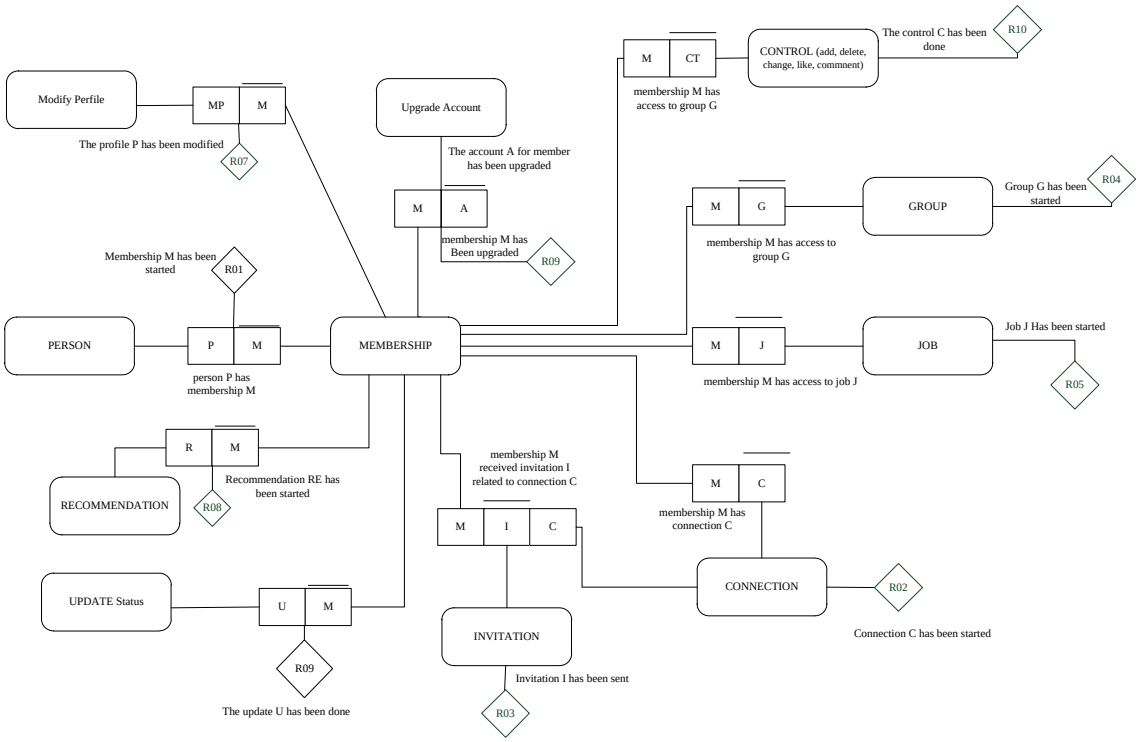


Figura 10: Object Fact Diagram OFD

Poderá consultar o OFD dividido em partes, em anexo, de modo a facilitar a legibilidade do diagrama

4.7 The Interstriction Model

4.7.1 Bank Contents Table –BCT do LinkedIn

Object class, fact type or result type	**Process Steps**
MEMBERSHIP	CBP1
P é membro da Rede Social	
membership M has been started	
PROFILE	CBP1
M modify perfile (MP)	
the profile P has been modified	
CONNECTION	CBP2
the connection C has been started	
M add connection with other M	
M remove a connection with other M	
INVITATION	CBP2
the invitation I has been sent	
GROUP	PB3
the group G has been done	
M add a group G	
M out a G	
JOB	PB4
the job J has been done	
M post a J	
ACCOUNT	PB5
the account for M has been upgraded	
RECOMMENDATION	CBP7
the recommendation RE has been done	
M sent a R for other M	
UPDATE	CBP6
the update U has been done	
M post a u	
M commented on the update of the other M	
M like on the update of the other M	
M delete U	
CONTROL (add, delete, change, like, comment)	CBP8
the control CT has been done	

Uma BCT específica o tipo de factos, os objectos, e os de tipos de resultado da SM.

Para começar, os dados do Linkedin são necessários pelo CA00. O CA01 tem acesso ao CBP1. O CA02 precisa de saber as conexões (CBP2) que tem, a que grupos (CBP3) pertence, as alterações de estado que publicou (CBP6) que fez, as recomendações (CBP7) que tem, as propostas (CBP4) que publicou ou a(s) proposta(s) (CBP4) que tem no seu perfil e o estado da conta (PB5) e por fim pode também pretender saber as acções de control que fez ou que fizeram nas suas publicações (CBP8).

4.7.2 The Actor Bank Diagram –ABD

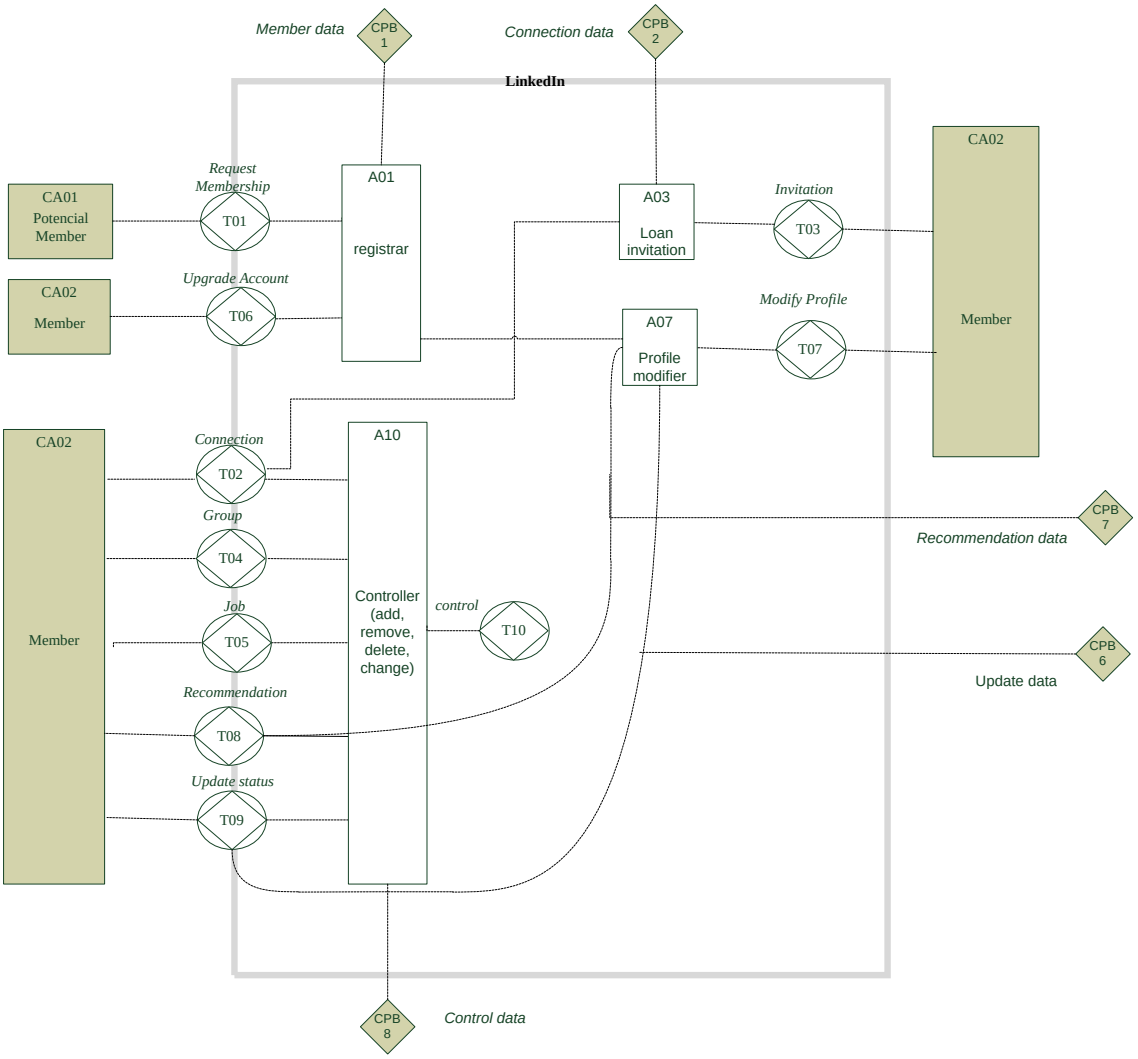


Figura 11: The Actor Bank Diagram –ABD do LinkedIn

4.7.3 The Organization Construction Diagram –OCD

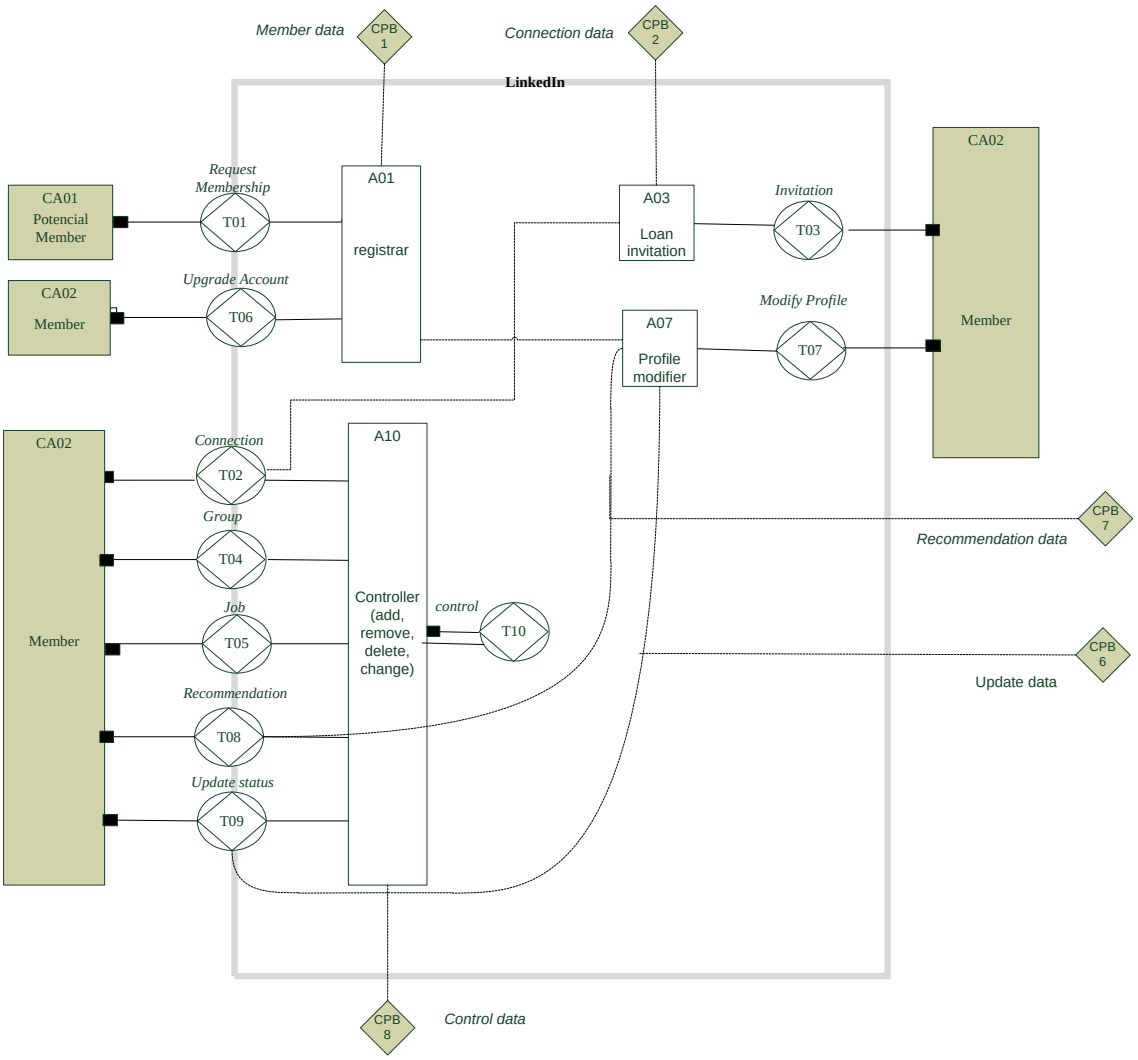


Figura 12: The Organization Construction Diagram –OCD

5. Conclusões e trabalho futuro

A DEMO é uma metodologia para conceber, engenharia, implementar e redesenhar processos de negócio das organizações. É usada sobretudo no desenvolvimento de sistemas de informação.

Esta metogologia permitiu representar de forma coerente, completa, consistente e concisa um modelo conceptual da rede social. Uma vez que através dos diagramas, das tabelas e das regras é possível ter a noção do sistema completo com maior ou menor detalhe. É possivel fazer alterações sem que isso altere o restante trabalho e não há situações imprevistas.

Neste trabalho apresentamos DEMO como uma metodologia de modelação das interações completas na rede social LinkedIn.
O principal objectivo deste trabalho foi modelar as principais interacções que ocorrem numa rede social .
Espera-se com este trabalho contribuir para a divulgação da metodologia DEMO, aumentar a usabilidade da mesma em trabalhos futuros por outros profissionais da área e por fim pretende-se que este trabalho possa auxiliar potencias utilizadores da metodologia DEMO.

Como trabalho futuro, uma melhoria pode ser feita, a demonstração da aplicabilidade da metodologia.

Bibliografia

Batista, André Filipe de Moraes *Um Estudo sobre a Perspectiva da Modelagemde Sistemas Multiagentes via a Teoria das Redes Sociais,* Monografia apresentada ao Centro de Matemática, Computação e Cognição, Universidade Federal do ABC, Brasil, 2009

Dietz, Jan L.G. Dietz, *DEMO: towards a discipline of Organization Engineering,* [online], Available: http://www.demo.nl/publications/, 2001

Dietz, Jan L.G. Dietz, *The atoms, molecules and fibers of organizations*, [online], Available: http://www.demo.nl/publications, 2003

[1] Dietz, Jan L.G. & Halpin, Terry - Using DEMO and ORM in Concert - a Case Study, [online], Available: http://www.demo.nl/publications/, 2004

Jan L. G. Dietz. Enterprise Ontology: Theory and Methodology. Springer, May 2006. ISBN 3540291695.

Dietz, Jan L.G. Dietz, *DEMO-3 Way of Working*, [online], Available: http://www.demo.nl/publications/, 2009

[2] Dietz, Jan L.G. Dietz, *DEMO-3 Models and Representations*, [online], Available: http://www.demo.nl/publications/, 2009

Fidom, Michael *Van DEMO naar een Relationele database*, [online], Available: http://www.demo.nl/publications/, 2009

[3] Garton, Laura; Haythornthwaite, Caroline and Wellman, Barry, *Studying Online Social Networks*, [Online], Available: http://onlinelibrary.wiley.com/doi/10.1111/j.1083-6101.1997.tb00062.x/full, 2006

Guerreiro, Sérgio *DEMO chapter fo MEISI*, apresentação das aulas do MEISI, complementos de engenharia de software, 2010/2011

[4] Guerreiro, Sérgio *Enterprise Governance enforcement in the operation of the run-time business transactions with DEMO and access control,* presentation given at Msc thesis lectures, LEIC/IST/UTL, Lisbon, September 2011.

[5] Jardim, André Desessards *Aplicações de Modelos Semânticos em Redes Sociais*, Dissertação de mestrado, Universidade Católica de Pelotas, Rio Grande do Sul, Brasil. 2010

[6] Khavas, Saman Sattari *The Adoption of DEMO in Practice*, Thesis Master of Science in Computer Science, University of Technology, Netherlands, 2010

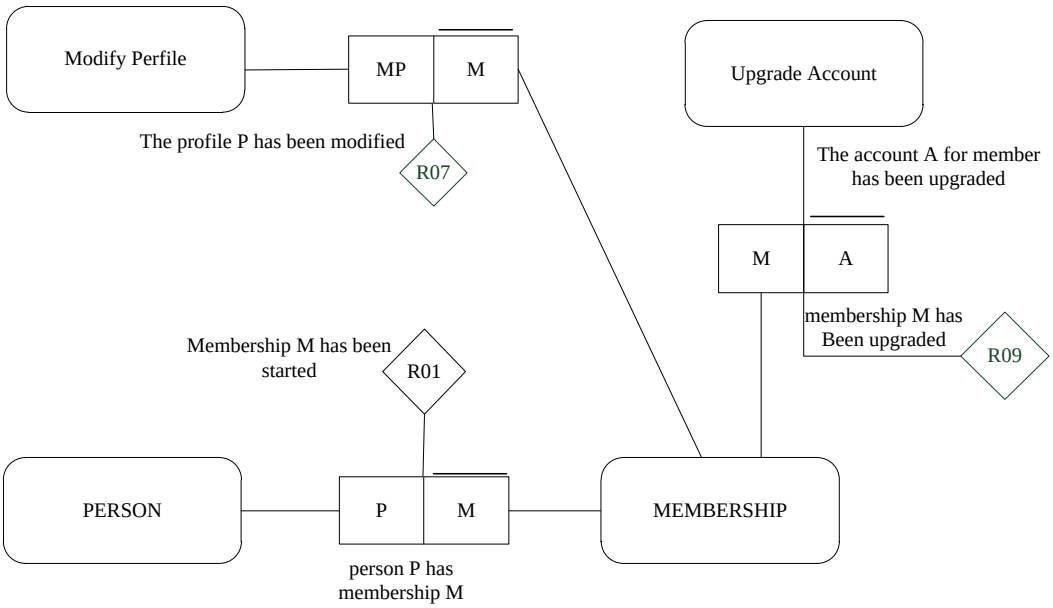
[7] Wasserman, S.; Faust, K. *Social Network Analysis: methods and applications*. [S.l.]: Cambridge University Press, 1994.

[8] Enterprise Engineering Institute, *The Benefits of DEMO*, [Online], Available: http://www.demo.nl/methodology

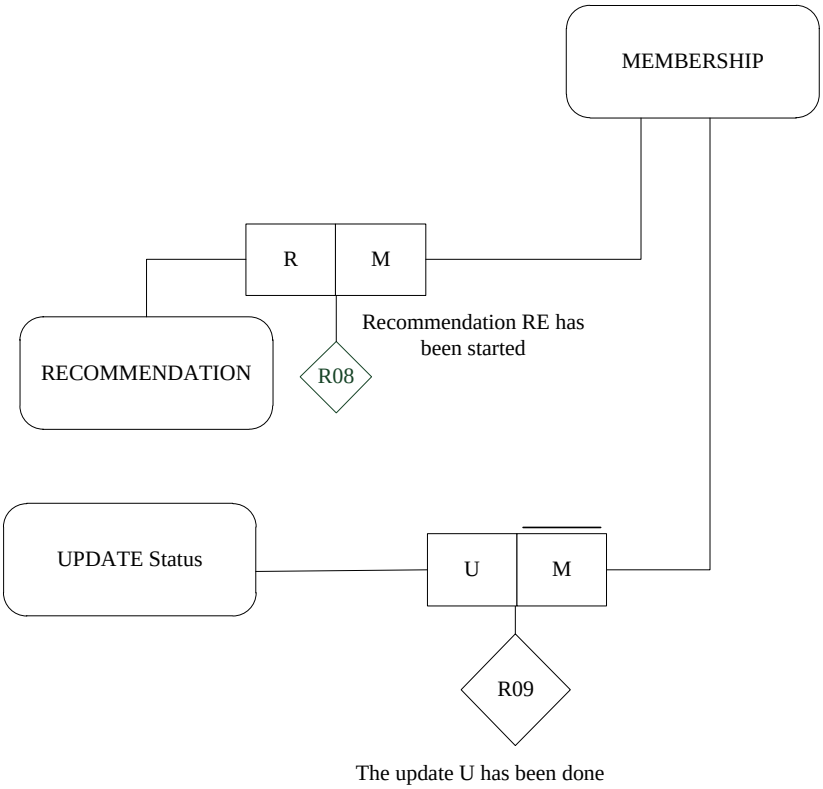
[9] http://www.demo.nl/

Anexo

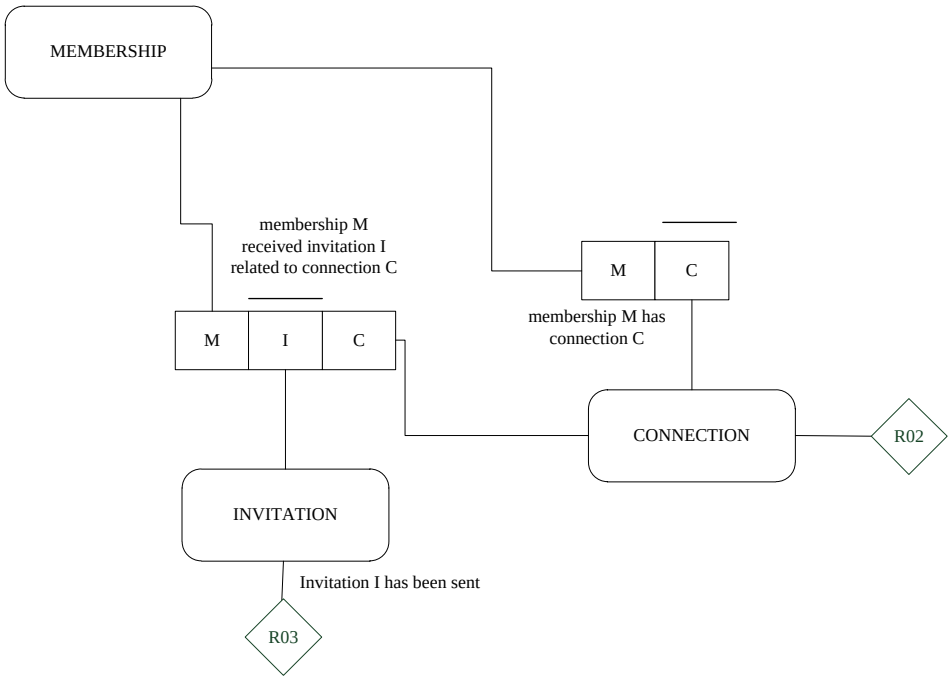
4.6.2.1 OFD: people, membership, profile, upgrade account



4.6.2.2 OFD: membership, recommendation, update



4.6.2.3 OFD: memberships, connection and invitation



4.6.2.4 OFD: memberships, control, group and job