Disciplina: ANÁLISE DE OBJETIVOS
Profª Phd Renata Guizzardi
Mestrado em Informática

Alunos:
Frederico Pinto de Souza
Marcos Rodrigues Saúde
Marcos Specimille

KAOS – **“Keep All Objectives Satisfied”**

1. Introdução

KAOS é uma metodologia para engenharia de requisitos que permite a analistas construir modelos de requisitos e derivar esses modelos para obter documentos de requisitos a partir dos modelos KAOS.

1.1 Metas

Tradicionalmente na engenharia de requisitos pergunta-se o que o sistema precisa fazer. Na orientação a metas pergunta-se **por quê** uma certa funcionalidade é necessária e **como** ela deve ser implementada.

Mas o que são metas?

“Uma meta é um objetivo não operacional que deve ser atingido a fim de desenvolver o sistema. Não operacional significa que os objetivos não são formulados em termos de objetos e ações disponíveis para algum agente do sistema; em outras palavras, uma meta é formulada para não ser estabelecida através de estados de transações adequadas sob o controle de algum agente.”

“Metas são objetivos de alto nível do negócio, organização ou sistema. Eles expressam a razão do sistema proposto e guia decisões em vários níveis dentro do projeto. Maximizar o lucro da empresa é um exemplo de uma meta de alto nível da corporação.”

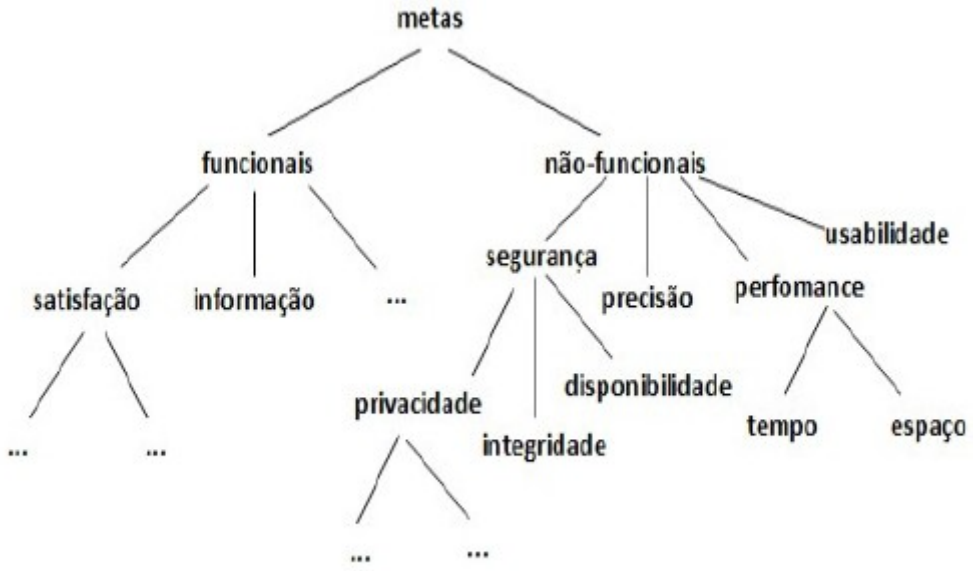
A engenharia de requisitos orientada a metas foca o sistema a ser construído e o seu ambiente como uma coleção de componentes ativos denominados agentes. Os componentes ativos podem restringir seu comportamento a fim de garantir restrições que fazem parte de sua responsabilidade. Esses elementos

são seres humanos desempenhando determinados papéis, artefatos ou software. Em oposição aos passivos, os agentes ativos têm a opção de possuir comportamento. Na Engenharia de Requisitos orientada a metas, aos agentes são atribuídas responsabilidades a fim de atingir as metas. Um requisito é uma meta cuja realização é da responsabilidade de um único agente de software, enquanto uma suposição é uma meta cuja realização é delegada a um único agente do ambiente

Ao contrário dos requisitos, as expectativas não podem ser executadas pelo software a ser desenvolvido sendo necessário aguardar obrigatoriamente a execução de normas ou regulamentação na organização. Na realidade os requisitos implementam metas da mesma maneira como programas implementam especificações de projeto.

Na abordagem KAOS, os objetivos podem ser decompostos através de ligações E (AND) e OU (OR), e estas denominam-se por “refinamento”. As ligações de refinamento AND relacionam um objetivo com um conjunto de sub-objetivos, isto significa que satisfazer todos os sub-objetivos ou submetas no refinamento é a condição para satisfazer o objetivo. As ligações de refinamento OR relacionam um objetivo com um conjunto alternativo de refinamentos, isto significa que satisfazer um dos refinamentos é a condição para satisfazer o
objetivo. A estrutura do refinamento AND/OR dos objetivos para um determinado sistema, pode ser representada através de um grafo acíclico dirigido.

Quando se fala de objetivos em KAOS, é necessário distingui-los de acordo com os diferentes tipos de categoria de requisitos, isto é, requisitos funcionais e os não-funcionais.



Exemplo de refinamento de metas funcionais e não funcionais.

Os Agentes representam um conceito importante na abordagem KAOS, pois são responsáveis em alcançar os requisitos e as expectativas. Os agentes podem ser humanos ou componentes automatizados. Um requisito é um objetivo colocado sob a responsabilidade de um agente de software e expectativa é um tipo de objetivo a ser atingido por um agente que é parte do ambiente do sistema.

Os conflitos estão focados em problemas entre pares de requisitos não-funcionais, o que significa que os dois objetivos não podem ser satisfeitos simultaneamente.

Os obstáculos são situações que violam um objetivo, um requisito ou uma expectativa. Neste caso, diz-se que o obstáculo “obstrui” o objetivo, requisito ou expectativa. O tratamento de obstáculos permite que os analistas identifiquem e localizem as circunstâncias excepcionais durante a engenharia de requisitos, a fim de produzir, por exemplo, requisitos robustos ou até mesmo novos requisitos para evitar ou reduzir o impacto dos obstáculos, resultando num software de confiança. Os obstáculos impedem com que os objetivos de um sistema sejam alcançados. Lidar com obstáculos é muito importante para a construção de sistemas seguros e robustos.

1.2 Modelos

Em geral, uma especificação KAOS é uma coleção de modelos que possuem como itens principais:

- **Modelo de metas**: reune um conjunto de metas cujo objetivo é representar a melhor forma de se resolver um problema. As metas genéricas podem ser decompostas em submetas, estas, por sua vez,

podem ser decompostas em requisitos (a execução dos requisitos são de responsabilidade dos agentes/atores). A relação entre metas, submetas e requisitos, proporciona o refinamento do modelo. Neste modelo também é possível mapear os obstáculos (representados por paralelogramo vermelho). Os obstáculos se relacionam diretamente com metas, submetas, requisitos e expectativas por meio de arestas direcionadas;

- **Modelo de objetos:** modelo UML que pode ser derivado de especificações formais de metas, uma vez que se referem a objetos e suas propriedades,

- **Modelo de operação**: define os vários serviços a serem prestados por agentes de software.

- **Modelo de responsabilidade**: traduz o conceito de ator ou entidade externa. Contém todos os diagramas de responsabilidade de um sistema. Estes diagramas descrevem para cada agente, os requisitos e expectativas dos quais ele é responsável ou que tenha sido atribuído a ele. Além de se conectarem aos requisitos, os agentes/atores também se conectam às expectativas.

Relação entre meta, submeta, obstáculo, agente/ator e expectativa

1.3 Origem

KAOS tem como origem a cooperação entre a Universidade de Oregon e a Universidade de Louvain (Bélgica) em 1990. Pesquisas, ampliações e melhorias estão sendo feitas com a metodologia pela Universidade de Louvain.

2. Idéias subjacentes ao KAOS

A seguir são apresentadas as idéias-chave que devemos saber antes iniciar a engenharia de requisitos com uso do KAOS.

2.1. Construir um modelo de requisitos

Atualmente, equipes de desenvolvimento apreciam técnicas de modelagem para a especificação de soluções. A primeira idéia-chave por trás do KAOS é construir um modelo de requisitos, que é utilizado para descrever o problema a ser resolvido e as restrições que devem ser atendidas por qualquer fornecedor da solução. O KAOS foi concebido para:

• Providenciar descrições de problemas, permitindo definir e manipular conceitos relevantes para a descrição do problema;

• Melhorar o processo de análise do problema através de uma abordagem sistemática para a descoberta e estruturação de requisitos;

• Tornar clara as responsabilidades de todos os participantes do projeto (*stakeholders*);

• Permitir que os interessados se comuniquem facilmente e de forma eficiente sobre os requisitos.

2.2 Justificar os requisitos, ligando-os às metas de níveis superiores

As metas são propriedades desejadas do sistema que são manifestadas por alguns dos interessados. Como exemplo, uma meta extraída do estudo de caso Elevador:

"*Cada vez que um passageiro chama um elevador de um andar F1 para ir a um andar F2, o sistema do elevador, eventualmente, leva-o à F2."*

Fazendo uso do KAOS os analistas descobrem as metas do novo sistema entrevistando os atuais e os futuros usuários, analisando os sistemas existentes, a leitura dos documentos técnicos disponíveis, etc. KAOS permite

aos analistas estruturar as metas coletadas de forma direta em gráficos acíclicos de modo que:

- cada meta no modelo (exceto a raiz (*root*) - a mais alta meta estratégica) normalmente é **justificada** por, pelo menos, uma outra meta que explica **por que** a meta foi introduzida no modelo;

- cada meta (exceto as folhas, metas de mais baixo nível) é **refinada** como uma coleção de submetas descrevendo **como** a meta refinada pode ser alcançada.

Perto do topo do gráfico está o negócio ou as metas estratégicas. Nas partes inferiores do gráfico (folhas) estão os requisitos de sistema. O exemplo dado acima é uma meta de negócio: ela dá uma propriedade pertencente à atividade principal do sistema do elevador. Ela não pode ser considerada como um requisito: o qualificador "*eventualmente*" não é suficientemente preciso. A seguir é apresentado um melhor candidato para ser um requisito:

"Quando existe uma chamada para um elevador no andar F, o primeiro elevador passando pelo andar F e na direção solicitada parará no piso F, exceto quando estiver cheio."

Neste caso o conceito de um "elevador cheio" deve ser definido em outro lugar no documento de requisitos.

Negócios e metas estratégicas são expressos em termos do vocabulário das partes interessadas (*stakeholders*). Metas de nível inferior são normalmente expressas com palavras do vocabulário dos *stakeholders*, bem como termos específicos de técnicas introduzidas no modelo e, quando necessário.

Identificar as metas não é procedimento executado exclusivamente de cima para baixo (que vai das metas de negócios mais elevadas até reduzir aos requisitos técnicos) ou uma abordagem bottom-up. Na maioria dos casos as duas abordagens devem ser utilizadas ao mesmo tempo. Muitas vezes, os analistas começam a revelar primeiro as metas intermediárias. Em seguida, eles seguem procurando por razões, em um nível superior (estratégico), para cada nova meta (perguntando "**por que** queremos isso?"). Para descobrir também submetas mais específicas, os analistas têm que se perguntar "**como** vamos atingir esses objetivo?".

Em sistemas reais, algumas metas podem ser **conflitantes**. As metas são conflitantes, se o sistema alcançar um estado em que não é possível satisfazer as metas simultaneamente. Por exemplo, metas de desempenho podem ter conflitos com metas de segurança; metas de informação podem entrar em conflito com metas de segurança e privacidade. Desejos de diferentes papéis de usuários também podem entrar em conflito. É particularmente importante identificar as metas conflitantes o quanto mais cedo possível no ciclo de vida

do software (melhor ainda se isso ocorrer durante o levantamento dos requisitos). Lidar com conflitos ou, mais geralmente, com obstáculos que impedem o cumprimento das metas, permite construir um documento mais completo dos requisitos e um sistema mais robusto.

O **modelo de metas** KAOS é o conjunto de diagramas de metas inter-relacionadas utilizado para resolver um problema particular.

2.3 Construir um modelo do sistema como um todo, não apenas a parte de software desse sistema

Limitar o escopo da análise de requisitos para o sistema de software a ser construído não é suficiente. O sistema de software deve ser usado dentro de um ambiente específico; é muito importante identificar, registrar e levar em conta todos os requisitos e pressupostos sobre a parte do ambiente que interage com o sistema de software.

Requisitos sobre os agentes que interagem com o sistema de software a ser desenvolvido são conhecidos como as **expectativas** no modelo de meta. Eles são introduzidos para mostrar como o sistema de software e seu ambiente deve cooperar para atingir as metas do sistema de informação. Eles fornecem uma clara separação entre as responsabilidades definidas no sistema de software e as estabelecidas no ambiente. Os obstáculos podem ser introduzidos no modelo, a fim de permitir que ao sistema de software lidar com um ambiente de interação que não cumpra as expectativas alocadas para ele.

2.4 Construir um modelo de responsabilidade

Além de metas, os agentes representam um importante tipo de conceito no KAOS. Os agentes são tanto humanos ou componentes automatizados que são responsáveis pela realização dos requisitos e expectativas.

No caso do elevador, o passageiro e o controlador do elevador são dois exemplos de agentes.

O KAOS requer analistas para associar cada requisito ou expectativa com um agente responsável por esses itens. De forma mais precisa é possível definir esses termos no KAOS como:

- um requisito é uma meta do tipo baixo nível a ser alcançada por um agente de software. O agente de software é responsável por essa meta.

• uma expectativa é de um tipo de meta a ser alcançada por um agente que faz parte do ambiente do sistema.

Em muitos casos, metas são atribuídas a diversos agentes, em vez de um único. Com KAOS, uma distinção é feita entre estas duas possibilidades. **Atribuição** é utilizada quando vários agentes podem ser responsabilizados por algum requisito ou expectativa, considerando que a **responsabilidade** é usada quando há apenas um agente que é responsável por ela.

Basicamente, esta diferença dá ao analista um critério para parar o refino de metas em submetas, ou seja, o refinamento não é mais necessário, logo que a meta foi colocada sob a responsabilidade de um único agente.

KAOS permite que diagramas de responsabilidade sejam derivados do modelo, em que cada diagrama exibe todos os requisitos/expectativas os quais um agente é responsável.

O **modelo de responsabilidade** do KAOS é o conjunto de diagramas de responsabilidade derivados.

2.5 Construir um glossário consistente e completo de todos os termos relacionados a problemas usados na descrição de requisitos

O KAOS permite aos analistas trabalharem no glossário progressivamente e simultaneamente durante a definição das metas de requisitos através da construção de um **modelo de objetos** KAOS. Um modelo de objeto contém objetos, como: **agentes**, **entidades** e **relacionamentos** entre esses itens. A notação usada no modelo de objeto é compatível com a utilizada na UML para os diagramas de classe.

Exemplos de entidades do problema do elevador são: elevador e andar.

Alguns exemplos de relacionamentos:

- *em(l, f)* reter o elevador l se este está localizado no andar f.

- *chamar(p, f)* reter um elevador se este foi chamado no piso *f* por um passageiro *p* e a chamada ainda não foi autorizada pelo sistema.

O Glossário de documentos de requisitos é construído para percorrer o modelo de objeto e listar todos os conceitos que ele contém.

2.6 Descrever como os agentes devem se comportar ordenadamente para satisfazer os requisitos pelos quais eles são responsáveis.

Os agentes de software são responsáveis pelos requisitos, tendo também capacidades. O **modelo de operação** do KAOS soma todos os comportamentos que os agentes precisam ter para cumprir seus requisitos. Os comportamentos são expressos em termos de operações realizadas pelos agentes. Essas operações funcionam sobre objetos descritos no modelo de objetos: elas podem criar objetos, provocar transições de estado do objeto ou desencadear outras operações através de **eventos** enviados e recebidos.

Exemplos de operações executadas pelos passageiros no problema do elevador são: *apertar botão*, *entrar* e *sair*, exemplos de operações executadas pelo elevador são: *abrir portas*, *fechar portas*, *mover-se para cima* e *mover-se para baixo*.

Um diagrama de operação do KAOS normalmente compõe operações realizadas por um ou vários agentes para atingir um requisito. As composições são feitas através de fluxos de dados (a saída de uma operação de saída se torna a entrada de outra operação) ou de controle de fluxo (um evento enviado por uma operação desencadeia ou para outra operação). Um diagrama de operação descreve como os agentes devem cooperar a fim de fazer funcionar o sistema. Com o KAOS o modelo de operação está ligado ao modelo de metas: os analistas justificam as operações pelas metas que eles "operacionalizaram". Uma operação sem justificação significa que ou ainda está faltando metas no modelo ou que a operação não é necessária. Inversamente, se alguns requisitos são deixados sem "operacionalização", eles podem ser apenas ilusões.

2.7 Basear o documento de requisitos sobre o modelo de requisitos

No final de um processo de elicitação de requisitos, normalmente não precisamos de um modelo, mas de um documento de requisitos. Com KAOS, o documento de requisitos é construído com base em um modelo e documento (*template*). A informação é extraída do modelo para preencher *template*. Por exemplo, a seção Glossário é construída por meio do modelo de objetos. Os requisitos são inseridos no documento, que cruzando o modelo de metas de cima (Negócio/objetivos estratégicos) para baixo (Requisitos). Os Requisitos sobre a arquitetura do sistema são derivados do modelo de responsabilidade e

os requisitos do comportamento do sistema são derivados a partir do modelo de operação.

O resultado é um documento consistente, não ambíguo e completo. A coerência e integridade do documento são garantidas pelos modelos subjacentes do KAOS. As alterações são feitas nos modelos do KAOS, e não nos documentos que são à saída dos modelos. Os documentos podem ser re-gerados, conforme a necessidade a fim de refletir as últimas alterações no modelo. Dentro de um modelo KAOS, cada conceito é definido apenas uma vez (sem ambigüidade e qualquer contradição). Se o analista segue a metodologia KAOS (cada meta é justificada por metas de alto nível e refinadas em requisitos, os requisitos são colocados sob a responsabilidade dos agentes e operacionalizados no modelo de operação), o documento de requisitos será completo no que diz respeito as metas identificadas. A completude do modelo de metas é garantida pela revisão de diferentes diagramas durante seções de validação com os *stakeholders* (especialistas de domínio, usuários, etc.).

2.8 Validar requisitos através da primeira revisão de modelo

A proposta do KAOS apresenta que é mais eficiente validar os requisitos organizando opiniões coletivas dos modelos do que recorrer às pessoas para lerem um documento muito técnico, e em seguida realizar reuniões para discutir suas observações. Reuniões à distância usando recursos de vídeo conferência entre as partes também podem ser usadas.

2.9 Usar uma abordagem defensiva para a construção de um modelo de requisitos

A idéia é similar a "*programação defensiva*". Ela consiste da investigação de forma sistemática o que poderia dar de errado no sistema, isto é, como e por que algum requisito pode não ser satisfeitos. Circunstâncias em que tais casos podem ocorrer são denominadas **obstáculos**.

Quando obstáculos são descobertos, o analista pode seguir estratégias diferentes para fazer face deles, como exemplo: adicionar novos requisitos que poderiam prevenir o obstáculo de ocorrer, mitigando o impacto de um obstáculo inevitável no sistema, etc.

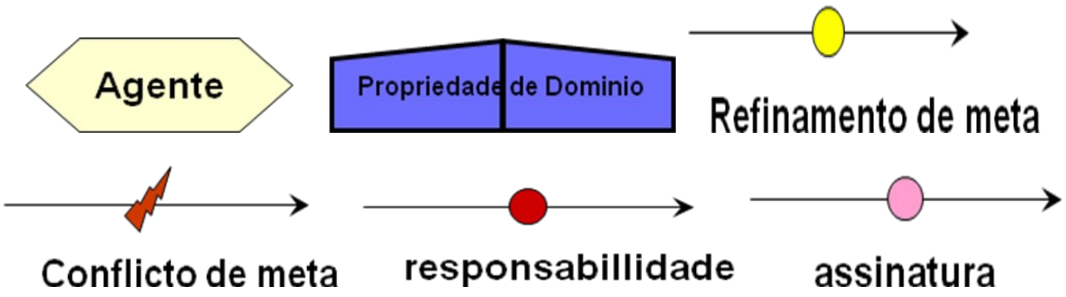
Exceções ou modos de operação degradada podem ser especificados desta maneira.

2.10 Considerar o documento de requisitos como uma referência que deve ser atualizada durante o projeto

Um erro comum é considerar o documento de requisitos como um documento congelado utilizado para negociar com fornecedores de soluções e não atualizável após esse período. Normalmente os requisitos sofrem alterações durante a fase de desenvolvimento e, por vezes, em outras fases também. O documento de requisitos deve ser atualizado conforme necessário durante todo o ciclo de vida de software. Uma das causas mais frequentes de falhas em projetos é a incapacidade de resolver o problema da evolução dos requisitos. O modelo de requisitos do KAOS ajuda a analisar o impacto das mudanças em requisitos, ou seja, ajuda a determinar outras alterações que são implícitas em uma nova solicitação de modificação. O KAOS fornece uma exibição de alto nível sobre o sistema a ser criado: o que ele faz, sobre o que, por quê, como, por quem e quando.

O KAOS pode ser aplicado independentemente do modelo de desenvolvimento utilizado (cascata, iterativo, incremental, etc.). No modelo em cascata, por exemplo, uma grande quantidade de tempo é gasto no início para elicitar todos os requisitos e escrever um documento de requisitos que torna referência para o desenvolvimento. Às vezes, prazos curtos, restrições de mercado ou a inexistência de requisitos específicos no início do projeto obrigam as equipes de desenvolvimento a seguir um processo de desenvolvimento iterativo e incremental. Os modelos do KAOS podem ser desenvolvidos e atualizados de forma a refletir os requisitos coletados em instâncias, por exemplo, através de protótipos, técnicas de simulação ou outra forma de aquisição.

Padrão dos diagramas



Comparativo entre KAOS e Tropos

1. Abrangência.

O principal objetivo por trás do KAOS é a construção de um modelo de requisitos, isto é, um modelo que descreva o problema e as restrições que precisam ser atendidos pela solução que será desenvolvida. A proposta do Tropos é um pouco mais abrangente, pois visa cobrir um amplo espectro do processo de desenvolvimento de software.

A metodologia proposta pelo Tropos aborda as etapas de *early requirements*, *late requirements*, *architectural design* e *detailed design,* abaixo temos um comparativo entre a abrangência do KAOS e do Tropos sobre o processo de desenvolvimento de software:

Processo de Desenvolvimento de Software					
Metodologias	***Early Requirements***	***Late Requirements***	***Architectural Design***	***Detailed Design***	***Implementation***
Tropos					
Kaos					

Por ter uma abrangência maior, o Tropos se preocupa, durante todo o processo de modelagem, em fornecer métodos, com um certo grau de formalismo, visando garantir que o desenvolvimento foi guiado pelos requisitos. Em contrapartida, o principal objetivo em Kaos é gerar uma documentação de requisitos mais rica, baseada nos modelos desenvolvidos.

2. Conceitos chave

- Agentes x Atores

Um conceito chave similar em ambas metodologias é a da existência de entidades autônomas que interagem com o sistema, podendo estas entidades serem representadas por pessoas ou sistemas. No entanto, em Tropos, temos que um ator pode ser um papel ou uma posição. Este conceito não está explicitado em KAOS, mas pode facilmente ser notado quando da criação dos modelos.

- Objetivos

Na metodologia KAOS, objetivos são propriedades de sistema desejadas expressadas por algum stakeholder. Similarmente, a metodologia Tropos conceitua objetivo como um interesse estratégico de um ator. Embora ambos os conceitos estejam próximos, existe uma diferença em relação a classificação destes objetivos.

O Tropos subdivide seus objetivos em duas classes distintas: *Hard Goals* e *Soft Goals.* Os *Hard Goals* são os objetivos para os quais podemos estabelecer critérios que nos permitam afirmar se estes foram atingidos ou não. De forma oposta, para os *Soft Goals* isto não é possível, por ser uma espécie de objetivo qualitativo, geralmente expressando requisitos não funcionais.

Na metodologia KAOS, os objetivos também são divididos em duas categorias distintas: *requirements,* um objetivo de baixo nível que é atribuído a um agente de software*,* e *expectations,* um objetivo que será atingido por outro agente que é parte do ambiente.

As classificações, em ambas as metodologias, não guardam quaisquer relações e não devem ser confundidas. Note-se que o critério de classificação é distinto. Na primeira, Tropos, o critério baseia-se no fato de poder ou não ser medido o alcance do um determinado objetivo. Na segunda, KAOS, o critério é se o objetivo será atribuído a um agente de software ou a outro agente que compõem o ambiente do sistema.

- Assignment e Responsibility x Dependency e Delegation

Os objetivos identificado em KAOS, são, conforme visto acima subdivididos em *requirements* e *expectations.* Os *requirements* são atribuídos a um agente de software, diz-se então que este agente é responsável por aquele *requirement*, este é o conceito de *responsability*. Já as *expectations* identificadas, quando são atribuídas a um agente, dá-se nome deste relacionamento de *assignment*.

As relações de dependência e de delegação que constam em Tropos surgem da interação entre os agentes, ou seja, na primeira um agente depende que o outro atinja determinado objetivo para ter sucesso, na segunda um agente transfere a responsabilidade por determinado objetivo para um terceiro.

Percebe-se que os conceitos em ambas as metodologias não guardam relação forte entre si, mas tentam exprimir um conceito de responsabilidade e dependência de algo. Quando comparamos o *assignment* do KAOS com o *dependency* do Tropos, percebemos que um determinado objetivo pode não ser cumprido, se o agente do qual ele depende não cumprir com a sua parte ou, se for externo aos sistema, não interagir.

- Conflict x Contribution

Ambas as metodologias reconhecem que um objetivo pode afetar outro. Em KAOS, este tipo de relação é denominado *conflict* e, em Tropos, *Contribution.* Como o próprio nome indica, em KAOS, só são modeladas relações negativas, ou seja, quando um objetivo influencia negativamente o outro. Em Tropos, as relações entre objetivos podem ser positivas ou negativas.

3. Modelagem de objetivos.

O processo de modelagem em ambas as metodologia guarda certa similaridade, mas os modelos obtidos são diferentes.

Em KAOS, objetivos são identificados por meio de entrevistas aos *stakeholders* e progressivamente decompostos. No modelo, não são identificados os stakeholders que originaram os objetivos, somente a decomposição hierárquica deles. Durante o processo de refinamento, os objetivos são atribuídos aos agentes, ou seja, os agentes precisam atingir determinados objetivos para que os objetivos de mais alto nível sejam satisfeitos.

Diferentemente do que é feito em KAOS, em Tropos, inicialmente os atores são identificados e a partir deles os objetivos. A identificação de quem será responsável por determinado objetivo só é alcançada após o refinamento do modelo, mediante os processos de delegação e dependência.

4. Outros modelos e conceitos.

A KAOS apresenta um modelo específico chamado de Responsability Model, este modelo mostra todos os objetivos pelo qual um agente é responsável. Em Tropos, isto é mostrado no próprio modelo de objetivos através de uma linha tracejada partindo do agente e envolvendo todos os seus objetivos dentro de um círculo.

A metodologia KAOS apresenta ainda outros modelos: Object Model e Operation Model, para os quais não achamos um correspondente imediato e Tropos.

A KAOS trás ainda a possibilidade de modelagem de obstáculos para um determinado objetivo, possibilidade esta que não encontra similar em Tropos.

Principais benefícios da metodologia KAOS

O maior benefício de KAOS é permitir que os requisitos sejam descritos e desenvolvidos desde o problema até a solução. Este rastreamento entre o problema e o espaço de soluções é fundamental, não só para se ter certeza de que o sistema desenvolvido estará correto do ponto de vista dos requisitos, mas também para permitir que as decisões arquiteturais sejam as mais adequadas.

Os documentos de requisitos elaborados com KAOS, tendem possuir mais detalhes e também contribui para uma menor ambiguidade, uma vez que fica claro quem é responsável por qual atividade.

ESTUDO DE CASO

DESCRIÇÃO DO PROBLEMA:

O propósito é criar para uma empresa um novo software que gerencia elevadores. Prover melhorias de desempenho e qualidade no desenvolvimento são os principais atributos do software. O software usado pela empresa no momento foi desenvolvido sem nenhuma metodologia para a criação dos requisitos. Para a construção do novo software será utilizado a metodologia KAOS.

1. Modelo de Objetivos:

O primeiro diagrama de objetivos mostra a visão geral do que o sistema deve atender. O objetivo principal do sistema é garantir a solicitação de transporte segura, ser eficiente, usável e barato. Esse objetivo principal pode ser refinado em outros cinco objetivos que são: solicitação de transporte satisfeita, sistema do elevador seguro, sistema do elevador eficiente, sistema do elevador usável e por fim, sistema de elevador barato. A Figura 1 mostra como fica o modelo de objetivos geral do Sistema Elevador.

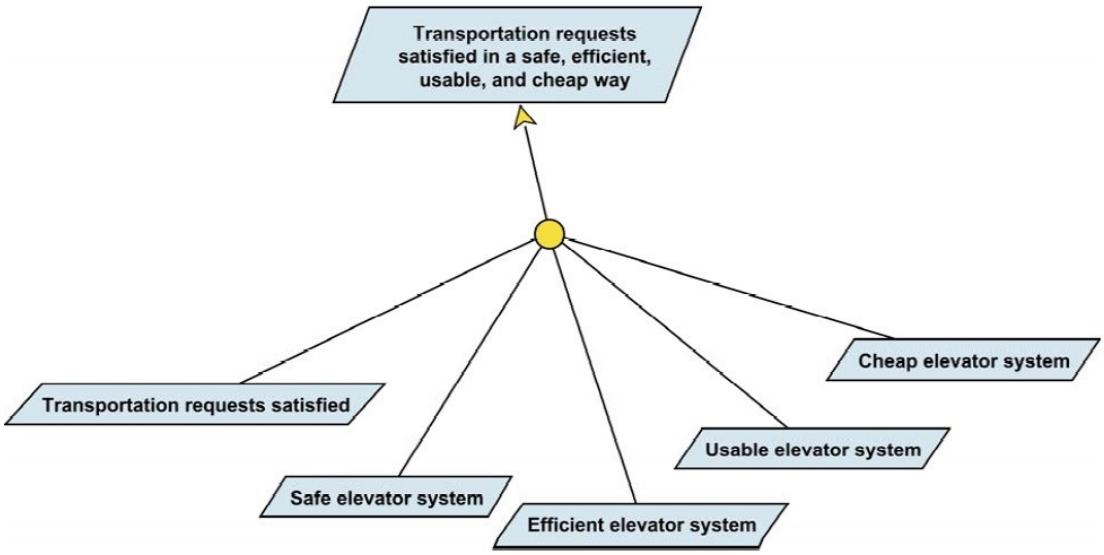


Figura 1

O objetivo agora é refinar o modelo acima, garantindo que todos os objetivos citados sejam satisfeitos. Para representar o refinamento dos objetivos, iremos refinar somente o objetivo *"Transportation request satisfied"* (Solicitação de transporte satisfeita).

A Figura 2, apresenta o diagrama de objetivos do objetivo *"Transportation request satisfied".* Este objetivo mostra o conceito de propriedades de domínio. Propriedades de domínio são propriedades

relevantes para o domínio do projeto. Elas são usadas nos refinamentos para provar que os refinamentos estão completos.

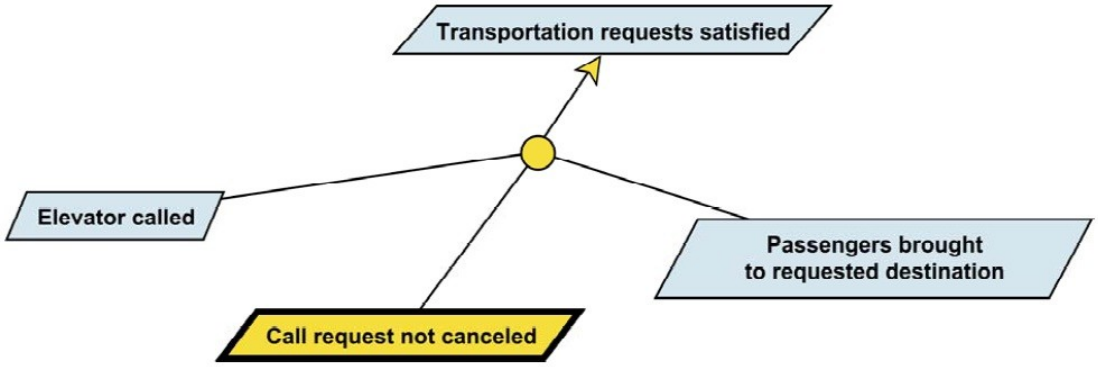


Figura 2

Na Figura 2, se o objetivo “Call request not canceled” (Ligação não cancelada) não for satisfeito, então o objetivo superior *“Transportation request satisfied”* nunca será satisfeito.

Refinando agora o objetivo *“Elevator called”* (Elevador ligado), temos uma camada de interface com botões e seus respectivos atores. Os paralelogramos que estão contornados com sombra mostram a expectativa do seu respectivo agente. Por exemplo, a expectativa do agente “Passenger” (Passageiro) é “Button depressed” (Botão acionado), ou seja, a expectativa do agente passageiro é pressionar o botão. A Figura 3 abaixo mostra exatamente isso.

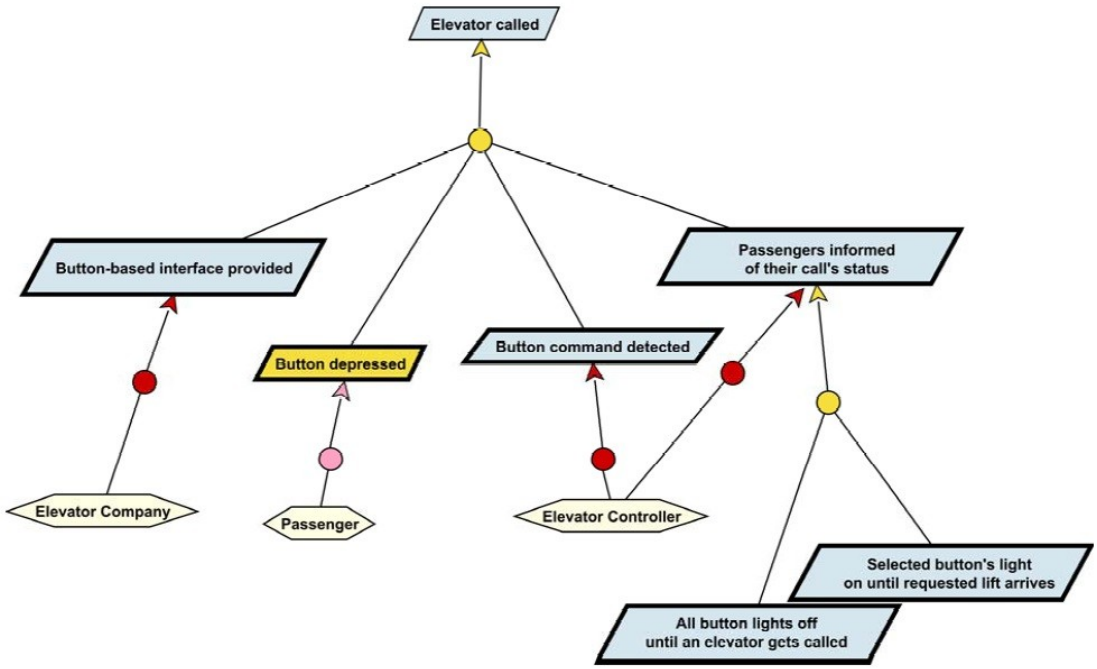


Figura 3

Por fim, está representado na Figura 4, o refinamento do objetivo *“Button depressed”.* Este objetivo é refinado em mais dois objetivos. “*Up or down floor button pushed*” (Botão subir ou descer da porta pressionado) e “*Destination floor Button pushed inside cage*” (Botão andar de destino pressionado).

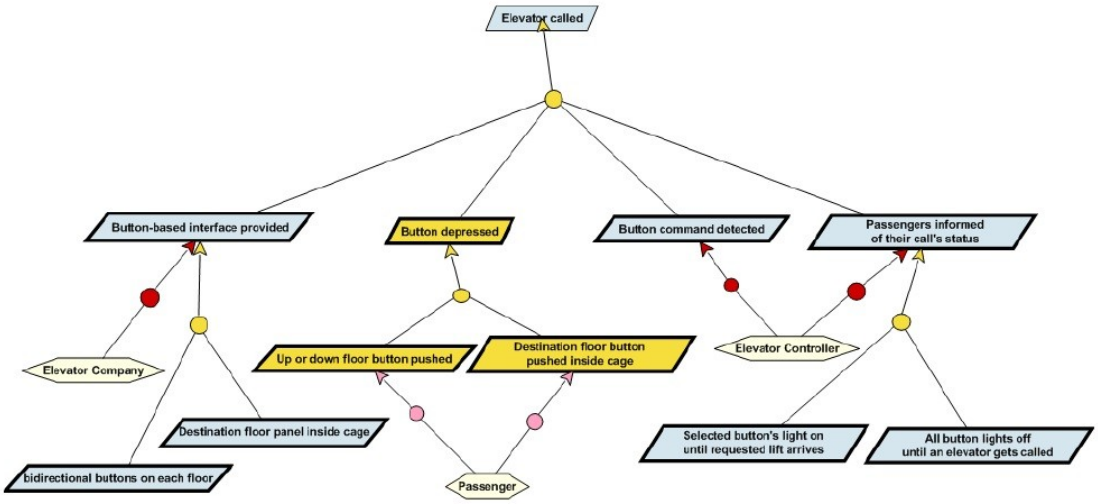


Figura 4

2. Modelo de Responsabilidades:

O modelo de responsabilidades contém todos os diagramas de responsabilidades. O diagrama de responsabilidade descreve para cada agente as necessidades e as suas expectativas.

Para construir um diagrama de responsabilidade, o analista revisa as diferentes necessidades e expectativas do modelo de objetivo e atribui um agente para cada um deles. A Figura 5 mostra abaixo o diagrama de responsabilidade do agente "*Elevator Controller*" (Controlador Elevador). Algumas das principais necessidades e expectativas do "*Elevator Controller*" é "*Elevator stopped*" (Parar elevador), "*Emergency light on when needed*" (Luzes de emergência quando necessário) dentre outros.

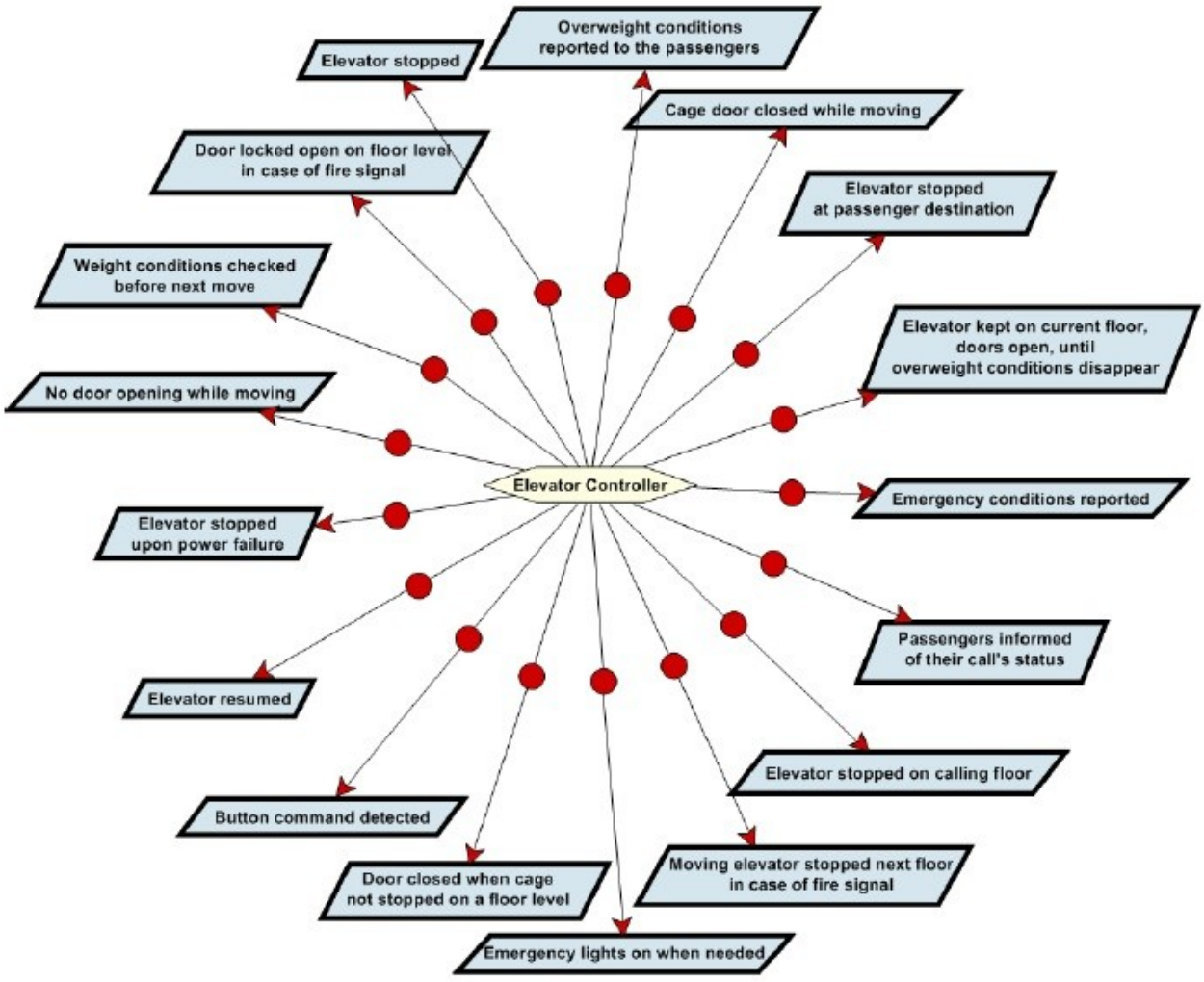


Figura 5

3. Modelo de Objetos:

O modelo de objetos é usado para definir e documentar os conceitos do domínio da aplicação, que são relevantes aos conhecimentos dos requisitos e para prover restrições estáticas sobre o sistema operacional que irá satisfazer os requisitos. Parte do modelo de objetos encontra-se relacionado ao domínio das partes interessadas, (stakeholders's), a outra parte apresenta os requisitos funcionais e requisitos do sistema operacional.

A Figura 6 mostra os componentes do Sistema Elevador.

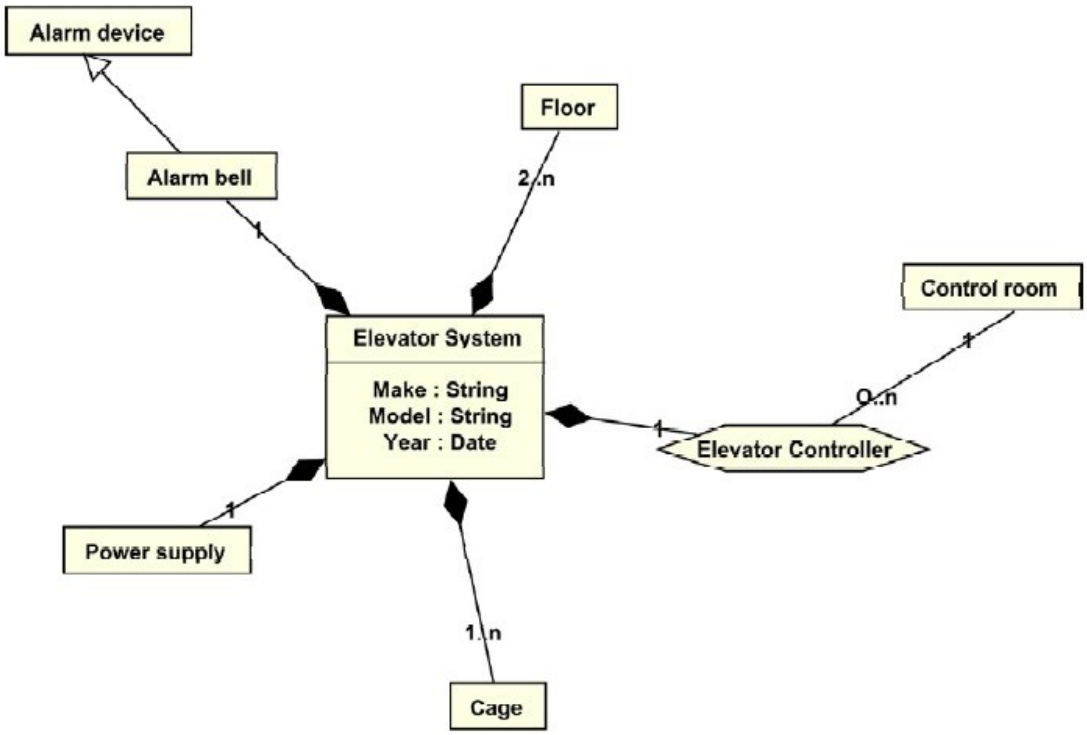


Figura 6

O "*Elevator System*" contém os seguintes componentes:

- Uma ou mais "*Cage*" (Gaiolas).
- No mínimo dois "*Floors*" (Andares).
- Uma campainha.
- Um controlador de elevador, situado na sala de controle.
- Fonte de alimentação.

As Figuras 7 e 8 apresentam respectivamente, os objetos "*Cage*" e "*Floors*".

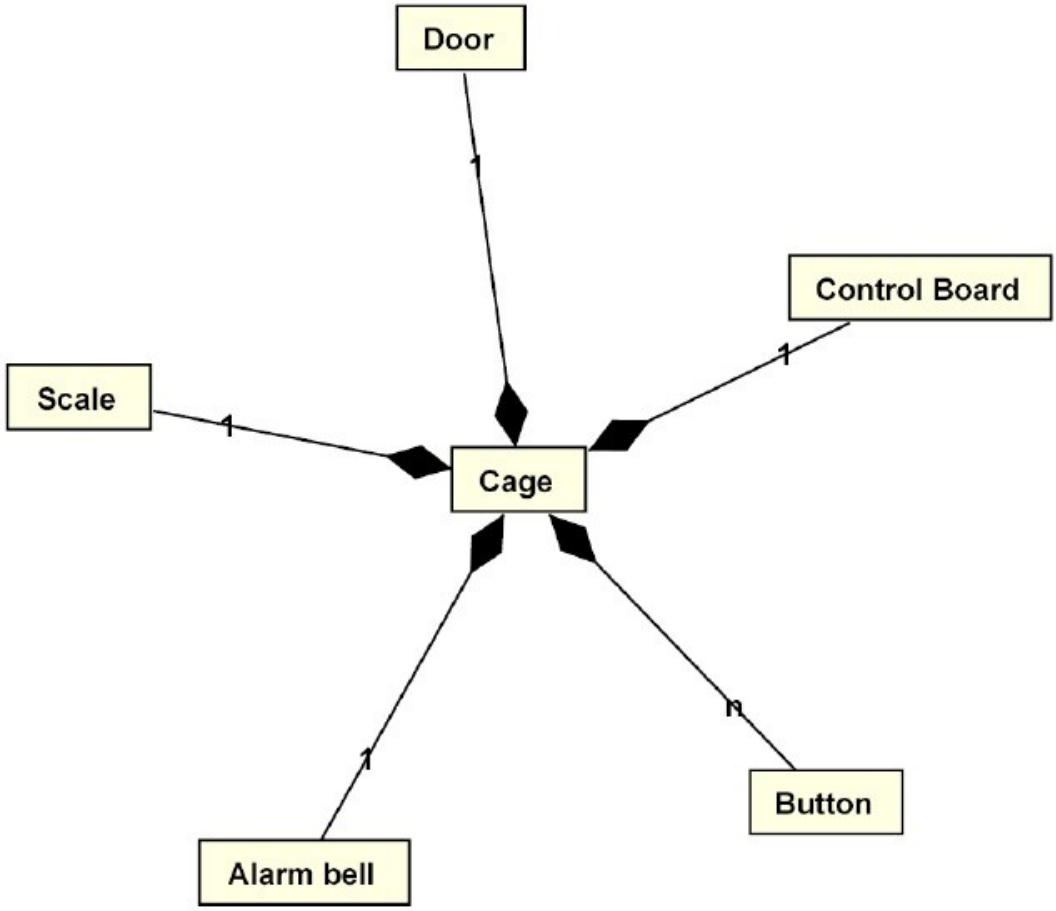


Figura 7

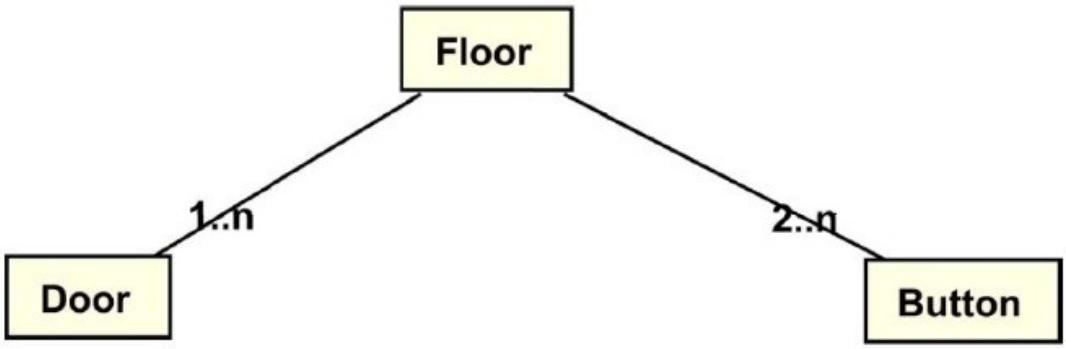


Figura 8

O modelo de objetos KAOS é compatível com o diagrama de classes UML. As entidades em KAOS correspondem às classes em UML e as associações em KAOS correspondem às associações em UML.

4. Modelo de Operação:

O Modelo de operação KAOS descreve todos os comportamentos dos agentes a fim de cumprir suas exigências. Comportamentos são expressos em termos de operações realizadas pelos agentes.

Na Figura 9, é apresentado um diagrama de operação. Nele, o evento "*Refresh*" é produzido pela operação "*Reschedule*" que é executada pela operação "*Execute schedule*".

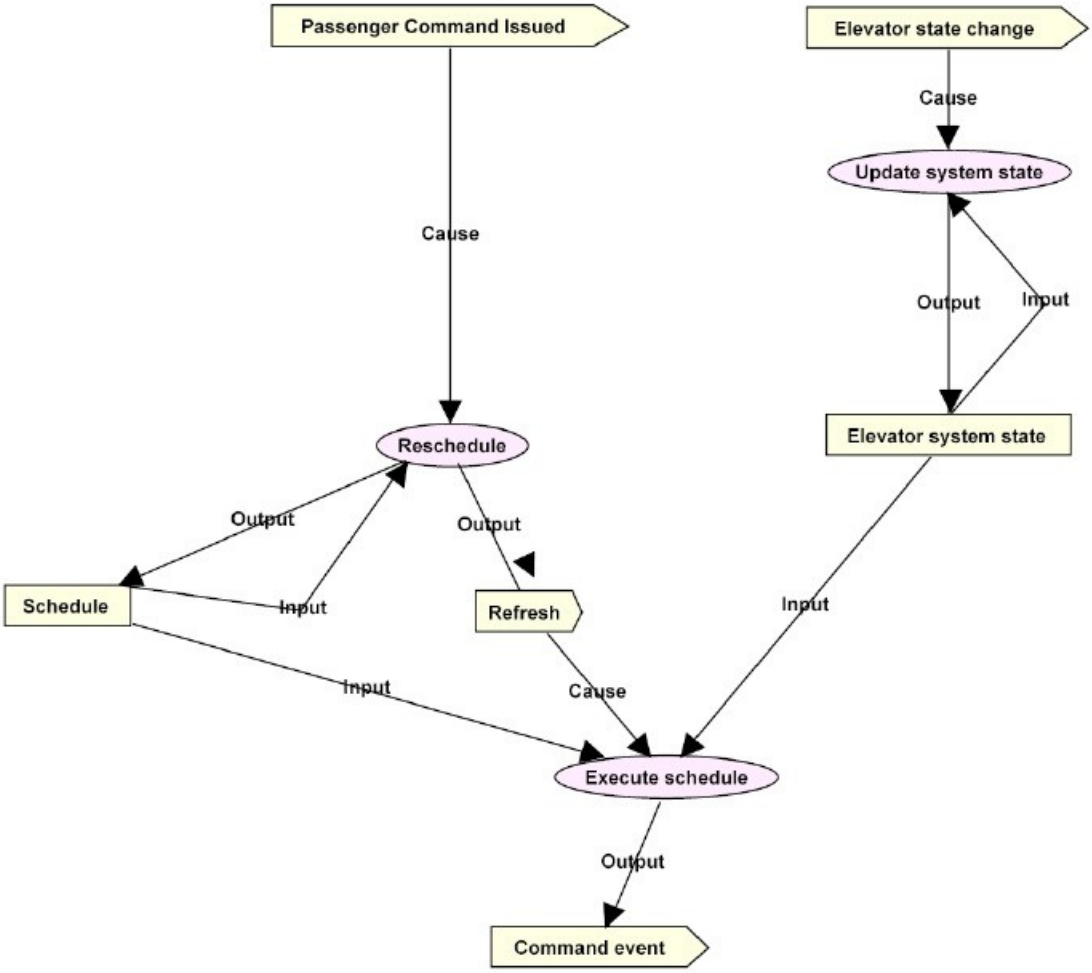


Figura 9

Ferramentas para a modelagem KAOS

- *Objectiver*: é uma ferramenta de metodologia que visa apoiar o trabalho e organizar os requisitos do projeto. É uma ferramenta comercial e não possui nenhuma versão free para download. A página na internet que apresenta a metodologia é “http://www.objectiver.com/”.

- RE-*Tools*: possui um conjunto de ferramentas de modelagem de requisitos. Ela permite desenvolver pelo padrão KAOS, porém é bastante limitada. Só é possível modelar os objetivos. A página na internet que apresenta a metodologia é “http://www.utdallas.edu/~supakkul/tools/RE-Tools/kaos.html”

Referência Bibliográfica

[1] **Tutorial. A KAOS Tutorial. Objectiver**. Tutorial elaborado pela Objectiver, França, Outubro de 2007.