Rational Unified Process for Systems Engineering

Parte 1: Introdução ao RUP SE Versão 2.0

por Murray Cantor
Principal Engineer
Rational Brand Services
IBM Software Group

Nota do Editor:
O IBM Rational Unified Process® ou RUP® oferece às organizações de desenvolvimento de software uma estrutura para simplificar todas as atividades relacionadas ao ciclo de vida de desenvolvimento. Desde sua apresentação formal em 1996, o RUP desenvolveu-se para suportar uma variedade de requisitos de desenvolvimento, incluindo "engenharia de sistemas", ou SE. Em 2001, o primeiro RUP Plug-In para suportar engenharia de sistemas foi proposto pela organização de serviços estratégicos do Rational Software. O RUP SE v1 Plug-In tornou-se amplamente disponível em 2002 e seu suporte continua sendo oferecido pela equipe de serviços da marca IBM Rational recém criada.

Ao descrever uma visão para a próxima geração do RUP SE, Murray Cantor inicia uma série de três artigos este mês; continuaremos essa série até os lançamentos de setembro e outubro do The Rational Edge. Embora esses artigos sejam consistentes com o RUP SE Plug-In atual, eles apresentam algumas extensões à estrutura do processo. Observe que o RUP SE Plug-In disponível no momento seja o RUP SE v1, transferível por download do Rational Developer Network (autorização requerida).

Introdução

Este artigo fornece uma visão geral da evolução mais recente do Rational Unified Process for

Systems Engineering® ou RUP SE®. Os usuários do RUP devem observar se o RUP Plug-In for SE disponível no momento é o RUP SE v1 Plug-In, que foi disponibilizado em 2002.

O RUP SE é um aplicativo da estrutura do processo de engenharia do software Rational Unified Process , ou RUP. O RUP SE suporta o desenvolvimento de sistemas em grande escala compostos por software, hardware, trabalhadores e componentes de informações. O RUP SE inclui uma estrutura do modelo de arquitetura que permite a consideração de um conjunto de diferentes perspectivas (lógicas, físicas, informação, etc.) para fornecer uma solução que determina os interesses de vários detentores de ações de desenvolvimento. Uma característica distinta do RUP SE é que os requisitos desses tipos diferentes de componentes são juntamente derivados da crescente especificidade dos requisitos gerais do sistema.

O RUP SE determina projetos que:

- Sejam grandes o suficiente para exigir várias equipes com desenvolvimento simultâneo.
- Tenham desenvolvimento simultâneo de hardware e software.
- Tenham problemas de implementação arquiteturalmente significativos.
  OU
- Incluam um novo projeto da infra-estrutura subjacente da tecnologia da informação para suportar processos de negócios em desenvolvimento.

O RUP SE é fornecido como um RUP Plug-In. Este artigo contém alguns conceitos que, nesta redação, ainda não foram incluídos no Plug-In. Apesar disso, esses conceitos são apresentados aqui para fornecer nossa mais recente e melhor compreensão de como atender às necessidades de desenvolvimento de sistemas.

Começaremos com uma discussão de sistemas e os desafios enfrentados pelo desenvolvedor de sistemas modernos. Esta discussão e seguida pelos pontos de projeto do RUP SE, isto é, como ele determina os desafios de desenvolvimento de sistemas. As duas seções seguintes apresentam as técnicas de especificação de requisitos e de modelagem baseadas na UML do RUP SE, bem como o uso de semântica UML (Linguagem de Modelagem Unificada). A primeira seção, Especificação do Sistema, fornece uma descrição completa destacada do sistema. Parte II, a ser publicada no próximo mês, focalizará a arquitetura do sistema e apresentará a estrutura da arquitetura do RUP SE, que descreve as partes internas do sistema com base em vários pontos de vista. Parte III, em outubro, abordará a análise de requisitos e fluxo, uma introdução ao método para derivar requisitos e a especificação para os elementos da estrutura do RUP SE. Isso incluirá uma descrição do Joint Realization Method, uma técnica nova para derivar juntamente a especificação de elementos de arquitetura em vários pontos de vista. A Parte III também incluirá uma discussão da programática do RUP SE.

Terminologia e Conceitos no Desenvolvimento de Sistemas

Por um sistema, entendemos que seja um conjunto de recursos que fornecem serviços utilizados por uma empresa para realizar um objetivo ou uma missão[1] de negócios. Os componentes do sistema geralmente consistem em hardware, software, dados e trabalhadores. Os sistemas são especificados pelos serviços que eles oferecem, juntamente com outros requisitos não comportamentais, como confiabilidade ou custo de propriedade. O projeto de um sistema consiste em especificar os componentes, seus atributos e seus relacionamentos.

Sistema é uma daquelas palavras que têm um conjunto de diferentes, se relacionados, significados[2] que podem causar confusão quando utilizados para abordar o desenvolvimento técnico. Geralmente, um sistema é um conjunto ou uma junção de elementos que demonstram o comportamento coletivamente. Todas as definições encontradas na literatura de engenharia do sistema constroem essa idéia. Além da definição de sistema que mencionamos anteriormente, aqui são apresentadas algumas definições adicionais de sistemas e engenharia de sistemas.:

Do Conselho Internacional de Engenharia de Sistemas[3]

> Engenharia de Sistemas é uma abordagem interdisciplinar e os meios para permitir a realização de sistemas bem-sucedidos. Ela se concentra na definição das necessidades do cliente e da funcionalidade necessária anteriormente no ciclo de desenvolvimento, na documentação de requisitos e na continuidade da síntese do projeto e da validação do sistema ao considerar o problema completo:
>
> - Operações
> - Desempenho
> - Teste
> - Fabricação
> - Custo e Planejamento
> - Treinamento e Suporte
> - Descarte

Do Mil-STD-499A

> Gerenciamento de Engenharia -- O gerenciamento da engenharia e do resultado técnico requerido para transformar um requisito militar em um sistema operacional. Ele inclui a engenharia do sistema necessária para definir os parâmetros de desempenho do sistema e a configuração preferida do sistema para satisfazer o requisito, o planejamento e o controle de tarefas do programa técnico, a integração das especialidades de engenharia e o gerenciamento de um resultado totalmente integrado das engenharias de projeto, de especialidades, de teste, de logística e de produção para atender a objetivos de custo, desempenho técnico e planejamento.

Do The Art of Systems Architecting por Maier e Richtin[4]

> ...sistemas são coleções de itens diferentes que, juntos, produzem resultados inacessíveis pelos elementos isolados.

São Exemplos de Sistemas

- Os sistemas de tecnologia da informação que são constituídos principalmente de software, hardware e periféricos de computador e trabalhadores.
- Produtos como aviões, satélites, automóveis e comutadores telefônicos que consistem em hardware e software.
- Empresas que prestam ou realizam uma missão. Essas empresas podem ser constituídas de trabalhadores habilitados por hardware e software. Muitas vezes, essas empresas ultrapassam limites organizacionais.

Todas essas definições sugerem que os sistemas possam ser visualizados com base em duas perspectivas diferentes:

- Perspectiva da caixa preta: O sistema como um todo: os serviços que ele fornece e os requisitos a que ele atende.
- Perspectiva de caixa branca: Os elementos ou as partes que compõem o sistema.

Qual dessas duas perspectivas nós utilizamos na visualização, na compreensão e no uso de um sistema depende das necessidades envolvidas. O problema da engenharia de sistemas é projetar e implementar um sistema que atenda às necessidades de todos os envolvidos do sistema, incluindo

- Usuários quem está preocupado com a funcionalidade e o desempenho.
- Proprietários que estão preocupados com o custo de implantação e de propriedade.
- Investidores que estão preocupados com a vantagem competitiva no mercado ou no espaço da missão.

O RUP SE fornece mecanismos e uma estrutura de modelo baseada em UML para dar suporte a equipes de engenheiros de sistemas à medida que eles determinam a visualização da caixa preta do sistema e especificam um ótimo design de sistema de caixa branca que atenda a todas as necessidades dos envolvidos.

Desafios do Sistema Moderno

A engenharia de sistemas[5] foi identificada como uma disciplina na metade do séculoXX.[6] Antigamente, os engenheiros de sistemas receberam o desafio de projetar e especificar entidades complexas, independentes que forneceram recursos sem precedentes. Alguns sucessos anteriores da abordagem dos sistemas incluem Athena Rocket, NORAD e o programa Apollo. Atualmente, não se espera apenas que os sistemas forneçam o conjunto correto de recursos, mas também espera-se que eles atendam a um novo conjunto de desafios.

Obviamente, os projetos antigos de engenharia de sistemas apresentaram desafios significativos de seus próprios projetos. Por exemplo, o lançamento do primeiro satélite de exibição de imagens representou uma grande realização técnica. Primeiramente, a equipe de desenvolvimento desse projeto teve de analisar quais recursos eram necessários para o sistema sem precedentes atender às necessidades do usuário. Em seguida, a equipe teve de entender como utilizar a tecnologia existente para representar por imagem uma estrutura no solo, armazenar a imagem por um período suficiente para fazer download dela para uma estação terrestre e, em seguida, torná-la acessível a partir da estação de trabalho de um analista. Naquele momento, não era muito preocupante que a solução exigisse recursos dedicados. Fornecer a capacidade em si foi suficiente para atender às necessidades do depositário.

Atualmente, os sistemas de projeto de imagens por satélite estão menos preocupados em fornecer o recurso básico e mais preocupados em otimizar o sistema para atender a um conjunto mais abrangente de necessidades dos envolvidos. Por exemplo, além de desejarem obter as informações que o sistema reúne o mais rápido possível, os analistas obter o recurso para integrar essas informações aos dados provenientes de outras origens. Os proprietários de sistema desejam reduzir o uso de hardware (e software) dedicado, pois eles não podem arcar com os custos para manter um sistema por recurso e desejam reutilizar muito mais os recursos existentes. Além disso, eles têm grande interesse em reduzir os custos contínuos de manutenção e operação.

Além disso, é certo que a missão e as tecnologias de ativação do sistema serão alteradas diversas vezes durante o ciclo de vida do programa. Aqueles que investem no sistema desejam que ele se desenvolva em resposta a essas mudanças no custo mínimo e com o mínimo de interrupções. Eles também esperam que seus investimentos resultem em propriedade intelectual reutilizável.

Muitos Tipos de Requisitos

Para atender às necessidades dos envolvidos, os engenheiros de sistemas precisam considerar um amplo conjunto de requisitos. A seguir está uma lista parcial de considerações:

- Funcionalidade: O recurso fornecido a usuários e outros sistemas para atender à necessidade dos negócios. Os requisitos funcionais devem incluir o comportamento que o sistema exibe à medida que ele oferece a funcionalidade.
- Usabilidade: Facilidade de acesso à função do sistema.
- Capacidade de Manutenção: Facilidade de descoberta, isolamento e remoção de defeitos.
- Capacidade de Extensão: Facilidade de incluir funcionalidade.
- Escalabilidade: A capacidade de suportar números crescentes de usuários, itens de dados, e assim por diante, à medida que os requisitos aumentam com o tempo.

- Confiabilidade: A probabilidade de uma resposta correta do sistema, possivelmente incluindo preocupações com segurança.
- Desempenho: O tempo esperado de resposta do sistema para uma etapa em um caso de uso sob cargas de capacidade.
- Capacidade: O número esperado de usuários e itens de dados.
- Capacidade de Suporte: A facilidade do serviço no campo, incluindo o tempo de inatividade aceitável.
- Fabricação
- Custo de implantação
- Custo operacional

Dependendo das circunstâncias, pode haver outros requisitos do sistema, como suporte de logística, segurança e necessidades de treinamento remoto.

Alguns desses requisitos são familiares para as equipes de desenvolvimento de software. Alguns não podem ser determinados sem considerações de hardware, software e trabalhador, as quais devem ser especificadas concorrentemente em uma disciplina de projeto do sistema.

Outro recurso de desenvolvimento de sistemas é a possível necessidade de manter um número de configurações de sistema. Por exemplo, um indivíduo pode desejar manter especificações do sistema para produtos com arquiteturas comuns, mas com implementações diferentes de hardware e/ou software que atendam a pontos distintos de custo/desempenho.

Mudança Rápida

Grande parte da metodologia original de engenharia de sistemas foi desenvolvida no início da revolução tecnológica (~1955-1980), quando a velocidade e o impacto da mudança tecnológica atual não foram previstos. Os métodos originais da engenharia de sistema se concentravam em garantir que os requisitos fossemcuidadosamente especificados e, em seguida, atendidos da melhor maneira possível.

Atualmente, sabemos que um sistema pode ter uma vida útil de 30 anos. Entretanto, como nenhum engenheiro pode saber exatamente qual será a necessidade do sistema em cinco anos, quanto mais em 30, há uma maior recompensa estabelecida em sistemas que podem se adaptar às necessidades de desenvolvimento. O designer de sistema atual também deve levar em consideração que o ambiente ou o contexto em que o sistema opera também se desenvolverão. Além de seu ciclo de vida, é bem provável que qualquer sistema estabelecido em campo hoje interaja com sistemas imprevistos.

Por exemplo, os sistemas comerciais continuarão a se tornar mais integrados, estabelecendo novos requisitos em sistemas legados; novos sistemas de defesa ficarão on-line, estabelecendo novos requisitos em sistemas existentes.

Também saber que a tecnologia de ativação será alterada para qualquer sistema. Com o tempo, os sistemas existentes deixarão de ser competitivos ou não serão tão econômicos para manter. A estruturas de arquitetura de desenvolvimento dos sistemas modernos precisam fornecer um meio independente de tecnologia para considerar sobre nova hospedagem ou implementação.

Um Maior Espaço de Soluções

As tecnologias atuais podem ajudar designers a atender a esses complexos desafios, pois eles fornecem muitos maneiras de abordar o design do sistema. Por exemplo, embora os designers do primeiro sistema de imagem por satélite tenha limitado opções e precisem ser engenhosos sobre a superação de restrições técnicas, os designers de sistema atuais, na verdade, têm capacidade e processamento excessiva e podem realocar responsabilidades do subsistema para otimizar o sistema. Dessa forma, às vezes, eles precisam fazer opções difíceis. Este fenômeno é real para uma grande quantidade de domínios, como telecomunicações, aviônicas, tecnologia da informação, e assim por diante.

Balanceando Considerações Físicas e Lógicas

Há muitas maneiras de conceitualizar um sistema. Duas delas são:

- Como uma entidade física, controlada pelas leis da física e por disciplinas clássicas, como engenharia mecânica, elétrica e civil.
- Como uma grande máquina de estado, controlada pelas orientações aprimoradas de engenharia de software e da ciência de computação.

Alguns sistemas, como mecanismos de foguetes, talvez sejam melhores analisados como uma entidade física. Outros, como a maioria dos sistemas de tecnologia da informação, são grandes máquinas de estado. Na verdade, todos os sistemas se ajustam a alguns aspectos de ambos os conceitos. Os mecanismos dos foguetes possuem controladores internos acionados pelo software. Os sistemas de informação são controlados pelas leis físicas que restringem o desempenho, a latência e a confiabilidade do sistema.

Antigamente, os líderes de projeto geralmente adotariam um desses pontos de vista, dependendo do tipo de sistema, e gerenciariam de maneira adequada. Se eles pensassem no sistema principalmente de uma perspectiva de hardware -- como algo físico -- veriam o software como um "mal necessário" que permitiu que o hardware fizesse seu trabalho. Se eles pensassem no sistema de uma perspectiva de software -- o hardware como um mecanismo de hosting -- geralmente tratariam o hardware como uma novidade.

Aumento de Tamanho e Complexidade do Software

Com o surgimento da tecnologia de objetos, das estruturas de componentes e da automação de desenvolvimento do software, a produtividade do desenvolvimento do software tem aumentado triplamente desde 1970,[7] e o tamanho e a complexidade dos aplicativos de software continuam a crescer. Ao mesmo tempo, têm ocorrido grandes ganhos em potência de processamento, memória de computador, armazenamento de dados disponíveis e largura da banda da rede. Essas mudanças, por sua vez, têm levado aos mais sofisticados ambientes operacionais já existentes. Os segmentos de mercado de software e de sistemas têm utilizado todos esses recursos aprimorados para desenvolver programas cada vez maiores e mais complexos. Na realidade, as pressões competitivas têm levado a um aumento dez vezes maior em relação ao tamanho de uma média de aplicativo de software, como medido por pontos funcionais.[8] E, presumidamente, como a tecnologia continua a melhorar, essa tendência continuará.

Muitas Disciplinas

No desenvolvimento de sistemas modernos, uma equipe comum de desenvolvimento é formada por trabalhadores que reproduzem funções, como arquitetos, desenvolvedores, designers, testadores, entre outros. Tanto no RUP como no RUP SE, todas essas pessoas trabalham concorrentemente para desenvolver seus artefatos específicos em todo o ciclo de vida. Esses trabalhadores não distribuem trabalhos uns para os outros de modo serial. Em vez disso, eles trabalham juntos em toda a operação envolvida, desenvolvendo níveis de detalhe para determinar suas áreas de interesse. Uma das principais metas -- e desafios -- de um processo de engenharia e de uma estrutura de arquitetura é fornecer uma maneira para os vários envolvidos de desenvolvimento se comunicarem e alinharem as decisões de seus projetos. Considerando a complexidade do design do sistema moderno, a equipe de desenvolvimento do sistema deve determinar uma ampla variedade de interesses:

- Construções e integração
- Modelagem de negócio
- Modelagem de dados
- Problemas de domínio
- Desenvolvimento de hardware
- Fatores humanos
- Tecnologia da Informação
- Logística e suporte de campo
- Desenvolvimento de software
- Especificação e design do sistema geral

Membros diferentes da equipe de desenvolvimento são responsáveis por diferentes, mas sobrepostos, conjuntos de interesses. Por exemplo, além de garantir a adequação da arquitetura de software para atender aos requisitos funcionais, os arquitetos de software geralmente asseguram

- Capacidade de Uso: facilidade de acessar a funcionalidade do sistema.
- Capacidade de Manutenção: facilidade de isolamento e remoção de defeitos, sem introduzir outros.
- Capacidade de Extensão: facilidade de incluir nova funcionalidade em um produto de software existente.

Embora os engenheiros de sistemas geralmente determinem problemas de

- Disponibilidade/confiabilidade: a probabilidade de que o sistema estará disponível e responderá corretamente à entrada.
- Desempenho: pronto atendimento do sistema para a entrada.
- Capacidade: o número de itens, como usuários ou registros de dados, que o sistema pode manipular.
- Escalabilidade: a facilidade de aumento da capacidade.
- Capacidade de Suporte: a facilidade de fornecer suporte no campo. A Capacidade de Suporte pode incluir a instalação do sistema e a aplicação de correções.

Outros interesses da engenharia de sistemas específicos do domínio incluem: segurança, facilidade de treinamento e suporte a logística.

Encontrando os Desafios da Complexidade

O RUP SE fornece os artefatos para determinar os interesses que temos descrito e os fluxos de trabalho para o desenvolvimento de suas especificações detalhadas.

Embora esta série de artigos não determine todos os interesses de todos os envolvidos, ela descreve uma estrutura de arquitetura que fornece pontos de vista de modelos para permitir uma separação de interesses em todos os níveis de especificação do sistema. O mecanismo de fluxo que descreveremos fornece uma maneira de manter a consistência dos modelos nos pontos de vista. O uso desses mecanismos permite que a equipe siga juntamente a prática do RUP de evolução contínua de artefatos em todo o ciclo de vida de desenvolvimento.

Pontos de Design do RUP SE

O problema de desenvolvimento de sistemas é diferente do problema de

desenvolvimento apenas de software, em que o desenvolvimento de sistemas determina um conjunto mais amplo de requisitos do que aquele normalmente determinada nos resultados do software. Mesmo assim, é importante observar que quase todos os esforços de desenvolvimento do software contêm alguns elementos do problema de sistemas. Exemplos de esforços de desenvolvimento de software que apresentam interesses do sistema incluem aplicativos baseados na Web, aplicativos de negócios, integrações da tecnologia da informação e software incorporado, bem como sistemas de defesa e de inteligência.

O ponto importante aqui é que os projetos de desenvolvimentos dos sistemas devem determinar os muitos desafios que os engenheiros de sistemas enfrentam, conforme discutido na seção anterior. Para endereçar esses problemas, o RUP SE adota os seguintes pontos de design:

- Seguir a definição padrão de segmento de mercado dos sistemas.

  Aplicar a estrutura RUP ao desenvolvimento de sistemas.

- Estender o modelo de arquitetura do RUP 4+1 para a estrutura do modelo RUP SE e estender ou modificar as funções, as atividades, os artefatos e as disciplinas de RUP para levar em consideração novas visualizações.
- Utilizar UML como a linguagem de modelagem.
- Fornecer recursos de ferramentas.
- Manter todos os níveis de modelo como um recurso do programa.

Vamos explorar cada um desses pontos de design em mais detalhes.

Seguir a definição padrão de segmento de mercado (de facto) dos sistemas

Como observamos na seção Terminologia e Conceitos anteriormente, os sistemas podem ser visualizados das perspectivas de caixa preta e caixa branca. O RUP SE segue este princípio. A perspectiva de caixa preta é descrita na seção Especificação do Sistema (veja a seguir). A perspectiva de caixa branca é descrita na seção Arquitetura do Sistema, incluída na continuação do próximo mês. Observe que os elementos descritos pelo RUP SE incluem hardware, software, trabalhadores e dados.

Aplicar a estrutura RUP ao desenvolvimento de sistemas

O ciclo de vida e as disciplinas do RUP são mostrados na Figura 1. O RUP SE segue o RUP de duas maneiras:

- Ciclo de Vida: Focalizando a remoção de riscos, o RUP SE segue as quatro fases do RUP alavancando a compreensão em desenvolvimento da equipe dos detalhes do projeto.

- Iterações: O RUP SE defende uma série de construções do sistema com base em identificação de risco e mitigação; uma iteração geralmente incluirá pelo menos uma construção do sistema. Em particular, todos os artefatos, incluindo os planos de projeto detalhados, desenvolvem iterações. Um recurso-chave que o RUP SE herda do RUP é uma rejeição de desenvolvimento em cascata e o uso de desenvolvimento iterativo.

- Disciplinas: O RUP SE segue a área em foco, ou as "disciplinas" mostradas na figura 1, que fornece um número de visualizações na definição de processo subjacente e o esforço que será realizado pela equipe no desenvolvimento do sistema. Embora a equipe do projeto RUP contenha engenheiros de sistema, não há disciplina separada de de engenharia de sistemas. Em vez dos engenheiros de sistema reproduzirem uma ou mais funções RUP e participar de uma ou mais disciplinas do RUP. Observe que os fluxos de trabalho e as atividades das disciplinas são modificadas para determinar problemas mais amplos do sistema. Essas modificações são descritas nas seções a seguir.

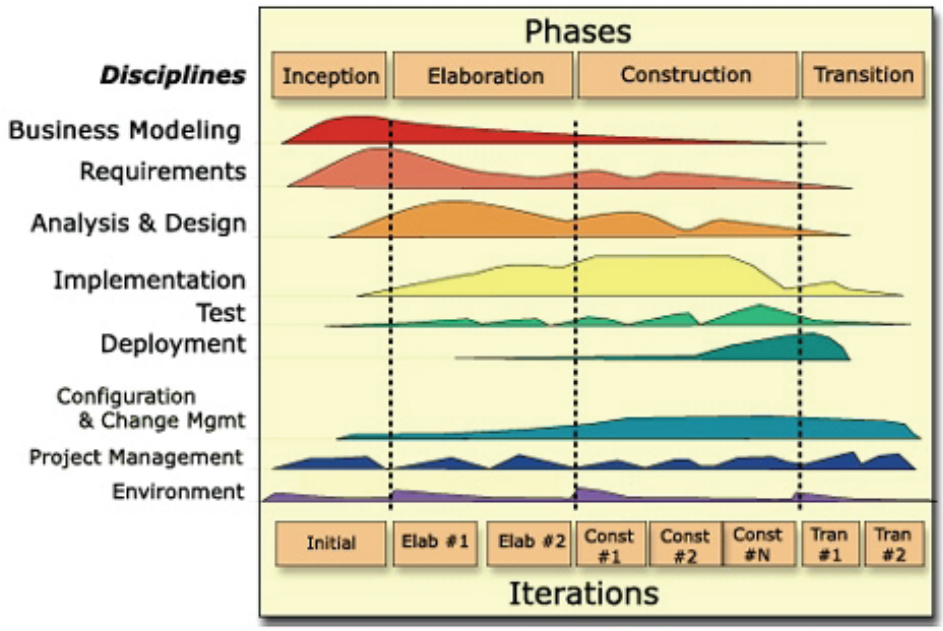

Figura 1: Estrutura do Processo de RUP (adotado pelo RUP SE)

Conforme descrito a seguir, o RUP SE complementa o RUP com artefatos adicionais, juntamente com atividades e funções para suportar a criação desses artefatos. Esses artefatos são descritos no RUP SE Plug-In mais detalhadamente.

Além disso, como um plug-in de aplicativo de estrutura do RUP, o RUP SE oferece a oportunidade de utilizar esses princípio subjacentes de gerenciamento de RUP para desenvolvimento de sistemas :

- Gerenciamento com base em resultados
- Desenvolvimento central da arquitetura

Estender o modelo de arquitetura do RUP 4+1 para a estrutura do modelo RUP SE

As estruturas de arquitetura permitem que os desenvolvedores considerem os diferentes interesses de especificação e design e, em seguida, documentam os resultados dessa consideração de maneira padrão e consistente. O RUP original é um processo de desenvolvimento de software; sua estrutura de arquitetura 4 + 1 é inadequada para determinar todos os interesses de desenvolvimento de sistemas, pois mais envolvidos requerem mais visualizações. Descreveremos a nova estrutura na Parte II dessa série, na arquitetura.

Utilizar UML como a linguagem de modelagem

A estrutura de modelo RUP SE utiliza o UML para expressar os vários diagramas que compõem as visualizações de estrutura de modelo de arquitetura. A versão atual adota a semântica, UML 1.4, incluindo esteriótipos. Na versão seguinte, moveremos para a semântica UML 2.0, juntamente com o perfil de engenharia de sistemas no processo quando adotado.

Fornecer recursos de ferramenta

Para suportar o RUP SE, o IBM Rational Software fornece um RUP Plug-In que descreve a extensão do RUP em detalhes, juntamente com os complementos da ferramenta IBM Rational Rose® e IBM Rational RequisitePro®. O Plug-In atualmente disponível foi liberado em 2002.

Manter todos os níveis de modelo como um recurso do programa

Como mencionado anteriormente, o ciclo de vida do sistema muitas vezes ultrapassa os requisitos iniciais e as tecnologias de ativação, levando, com o passar do tempo, à funcionalidade desatualizada ou, de outra forma, insuficiente, ou com custo inaceitável de propriedade. Entretanto, ele segue que uma estrutura de arquitetura disponível deva manter as visualizações de modelo em níveis crescentes de especificidade: Os níveis superiores estabelecem contexto e especificação; os níveis inferiores estabelecem componentes e listas de materiais. A rastreabilidade deve ser mantida em todos o processo. A manutenção desses níveis fornece a configuração para considerar sobre o impacto das alterações. As alterações na missão geralmente resultam em alterações no núvel superior no modelo que flui para os níveis inferiores. As alterações em tecnologia permitem diferentes negociações de design ou diferentes realizações do design atual. O RUP SE fornece os níveis de modelo e a rastreabilidade necessários.

Especificação do Sistema

Especificação do sistema é o processo de mostrar os recursos da baixa preta do sistema: sua funcionalidade visível externamente, quais serviços ela fornece e quais medidas de eficácia espera-se que ela atenda. No RUP SE, a especificação do sistema consiste em estudar como se espera que o sistema execute no contexto. Isto é, o sistema é considerado um participante em uma empresa mais ampla. A especificação do sistema é seguida de uma análise da empresa e a função que o sistema reproduz na ativação da empresa mais ampla, para atender ao objeto ou à missão de seus negócios. Este processo é descrito a seguir.

Empresas

Observe que uma empresa, como um sistema, também é um conjunto de recursos (trabalhadores, hardware, software e dados) utilizado para atender a um objetivo de negócios ou realizar uma missão. Como um sistema, uma empresa tem agentes -- entidades que utilizam ou colaboram com a empresa. Na realidade, muitas vezes é importante considerar uma empresa como um sistema de sistemas.[9]

Embora um produto, como um automóvel ou um aeroplano, faça parte de uma colaboração mais ampla da empresa. O avião deve interagir com o piloto e todos os sistemas de controle de tráfego aéreo.

Análise do Contexto

O sistema que está sendo desenvolvimento é sempre parte de uma empresa maior. Para especificar o sistema, o indivíduo precisa compreender as atividades da empresa mais ampla, particionar essa empresa no sistema e em outras entidades, analisar como o sistema participa na ajuda para que a empresa forneça seus serviços e, em seguida, capturar os resultados dessa análise como a especificação do sistema. Nesta seção, exploramos este fluxo de trabalho mais detalhadamente.

Observe que os sistemas têm características de entidades físicas e lógicas. Eles são entidades lógicas porque fornecem serviços, transmitem mensagens e interagem com outras entidades lógicas; os sistemas são físicos porque eles têm limites finitos em seus recursos e esses limites devem ser considerados. Exemplos dessas limitações são o pronto atendimento e a capacidade. Atualmente, a UML não tem semântica para essa entidade. Como abordado a seguir, no RUP SE modelamos sistemas como classificadores estereotipados, mas incluem as características físicas relevantes como atributos de classe.

Uma especificação do sistema captura uma descrição de caixa preta do sistema. Ela estabelece o escopo e os limites do sistema, os serviços que ele fornece, seus outros atributos e os itens que ele troca com outras entidades. Essas informações são capturadas em um diagrama de contexto do sistema,[10] como mostrado na Figura 2.

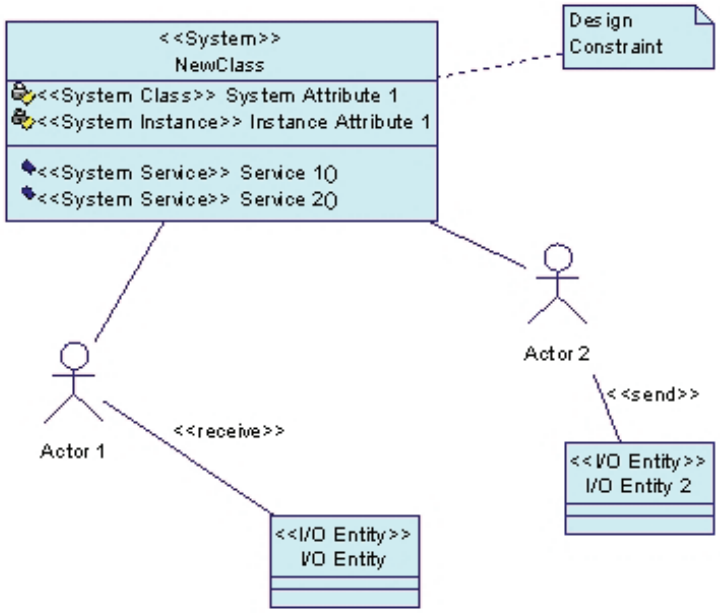

Figura 2: Diagrama de Contexto do Sistema RUP SE

Como mostrado na Figura 2, um diagrama de contexto consiste em

- Classificadores de sistema
- Agentes de sistema (sistemas e usuários externos)
- Relações de agentes do sistema
- Entradas de Entrada/Saída
- Relacionamentos de entidades de Entrada/saída
- Restrições de design

Vamos explorar cada um desses elementos.

Classificadores de Sistema

Sistemas são entidades. Um conjunto de sistemas, cada um atendendo à mesma especificação, é uma instância de uma classe de sistema. Por exemplo, automóvel Saab de 93 forma uma classe e o Saab 93 com VIN 12345678 é uma instância ou objeto nessa classe. Portanto, os sistemas podem ser descritos pelo objeto de UML e a semântica da classe. Embora em alguns casos, como em sistemas de TI, seja esperado que haja apenas uma instância do sistema, estas semânticas ainda são úteis.

Seguindo a semântica de UML, os objetos do sistema podem ter três tipos de atributos:

- Atributos de classe que têm o mesmo valor para todos os objetos do sistema.
  -- por exemplo, a capacidade total de combustível no Saab 93 ou o número total de usuários simultâneos em um sistema de TI.

- Atributos de instância cujo valor pode variar entre objetos do sistema

  -- por exemplo, volume de combustível atual no VIN 123456789 ou número atual de usuários em um sistema de TI.

- Medidas de mérito que são metas de design mais gerais, como tempo médio para falha ou risco técnico de desenvolvimento. Essas medidas podem ser capturadas pelos valores marcados.

Observe que os atributos de instância fornecem parâmetros para desenvolvimento e simulações de etapas de teste. Os atributos de classe fornecem intervalos para as etapas de teste e simulações, embora as medidas de mérito sejam usadas para determinar a adequação do design do sistema e para configurar os critérios de decisão para comercializações de design.

As operações de classe do sistema são chamadas de serviços do sistema. Nos termos da UML, essas classes são operações. Lembre-se que, em UML, as operações são classes e as instâncias dessas classes são mensagens encontradas em diagramas de seqüência ou diagramas de interação que fazem parte da realização de casos de uso. No desenvolvimento de sistemas, elas são encontradas estudando como o sistema interage com seus agentes para atender às necessidades ou à missão da empresa.

Além disso, seguindo a semântica usual de UML, os serviços podem ser agregados às interfaces.

Agentes do Sistema

Lembre-se que em UML, agentes são entidades que interagem com o sistema, geralmente usuários ou outros sistemas. Muitos desenvolvedores de sistemas acharam-no útil para incluir elementos de ambiente como horário ou condições atmosféricas como tipos de agentes.

Geralmente, há dois tipos de agentes de sistema:

- Agentes corporativos são externos à empresa que interagem diretamente com o sistema.
- Agentes internos fazem parte da empresa, não do sistema. Esses agentes geralmente são trabalhadores da empresa ou de outros sistemas empresariais.

Além disso, os agentes de sistema não devem ser confundidos com trabalhadores de sistema. Os trabalhadores que fazem parte do sistema não são agentes. Como veremos a seguir, eles podem ser agentes para alguns dos subsistemas do sistema.

Relacionamentos de Agentes do Sistema

As colaborações do sistema e dos agentes são similares às colaborações do subsistema e do sistema UML1.x: As instâncias do agente chamam os serviços do sistema para preencher suas funções na colaboração, e vice-versa. Ele segue o princípio de que a semântica que especifica que um agente usa um sistema é uma dependência de UML. A direcionalidade da dependência observa se o sistema chama os serviços de agente, ou vice-versa, ou ambos.

Às vezes, o agente e o sistema podem estar firmemente acoplados, de forma que os os atributos de uns são afetados pelos atributos do outros, e o relacionamento do agente e do sistema é promovido por uma associação. Por exemplo, considere um motorista (o agente) e um automóvel (o sistema). Quando o automóvel estiver em movimento, provavelmente o motorista estará em um estado superior de prontidão quando o automóvel dor estacionado.

Os serviços do sistema podem ser agregados a interfaces. As interfaces podem ser utilizadas para especificar qual conjunto de operações é usado por quais erros ou simplesmente para criar categorias de serviços. Observe que um determinado serviço pode ser incluído em mais de uma interface.

Entidades de Entrada/Saída

As entidades de Entrada/Saída (E/S) incluem tudo o que é gerado e/ou recebido pelo sistema, como informações ou itens físicos. Um sistema de varejo pode fornecer informações do cartão de crédito a outro sistema de informações de cartão de crédito. Um sistema de ar condicionado pode alternar entre ar quente e ar frio com a atmosfera. As entidades comuns de entrada incluem consultas de banco de dados, atualizações de arquivo, entradas de sensor e entradas de controle. As entidades comuns de saída incluem a consulta de resultados e saídas de sensor.

As entidades de E/S são classes de UML com atributos, mas sem numerações.

Relacionamentos de Entidades de Entrada/Saída

Observe, conforme mostrado na Figura 2, que as entidades de E/S têm relacionamentos de agentes com estereótipos especiais. A associação captura se a entidade é enviada ou recebida pelo agente. A semântica mantém o relacionamento agente-sistema no modelo e não sugere que um diagrama de contexto também seja um diagrama de fluxo de dados.

Restrições de Design

Freqüentemente, as especificações do sistema não são estritamente considerações de caixa preta. As especificações incluem restrições nas partes internas do sistema. Exemplos dessas restrições de design incluem componentes que devem ser utilizados e algoritmos que devem ser seguidos. À medida que o design é desenvolvido e que os elementos de caixa branca são especificados, essas restrições localizarão uma expressão natural em uma das visualizações do RUP SE descritas na seguinte seção.

Não há semântica de UML específica para expressar restrições de design. No RUP SE, essa semântica pode ser recuperadas de uma das duas maneiras: como notas no diagrama de contexto ou, preferencialmente, como um documento de requisitos suplementares associado à classe do sistema.

Casos de Uso e Serviços do Sistema

O RUP SE utiliza casos de uso e serviços [11] para capturar o comportamento do sistema. Em ambos os casos, a semântica padrão de UML [12] é utilizada:

- Os casos de uso descrevem como o sistema é utilizado por seus agentes – em outras palavras, o conjunto de colaborações/cenários antecipados entre os agentes e o sistema. Os casos de uso indicam como o sistema é utilizado para atender ao objetivo ou à missão mais ampla dos negócios corporativos. No RUP SE, as Descrições de Casos de Uso são idênticas àquelas utilizadas no RUP padrão. Especificamente, elas apenas descrevem as interações da baixa preta entre o sistema e seus agentes.
- Serviços são as operações (possivelmente resumidas) fornecidas pelo sistema para que ele possa preencher sua função nos cenários de caso de uso. Instâncias de serviços são as mensagens encontradas nos diagramas de seqüência de UML que consideram os cenários do caso de uso. Geralmente, há um mapeamento n-para-m entre o caso de uso e os serviços.

A Figura 3 é um exemplo de um diagrama de contexto parcial para um sistema de varejo.

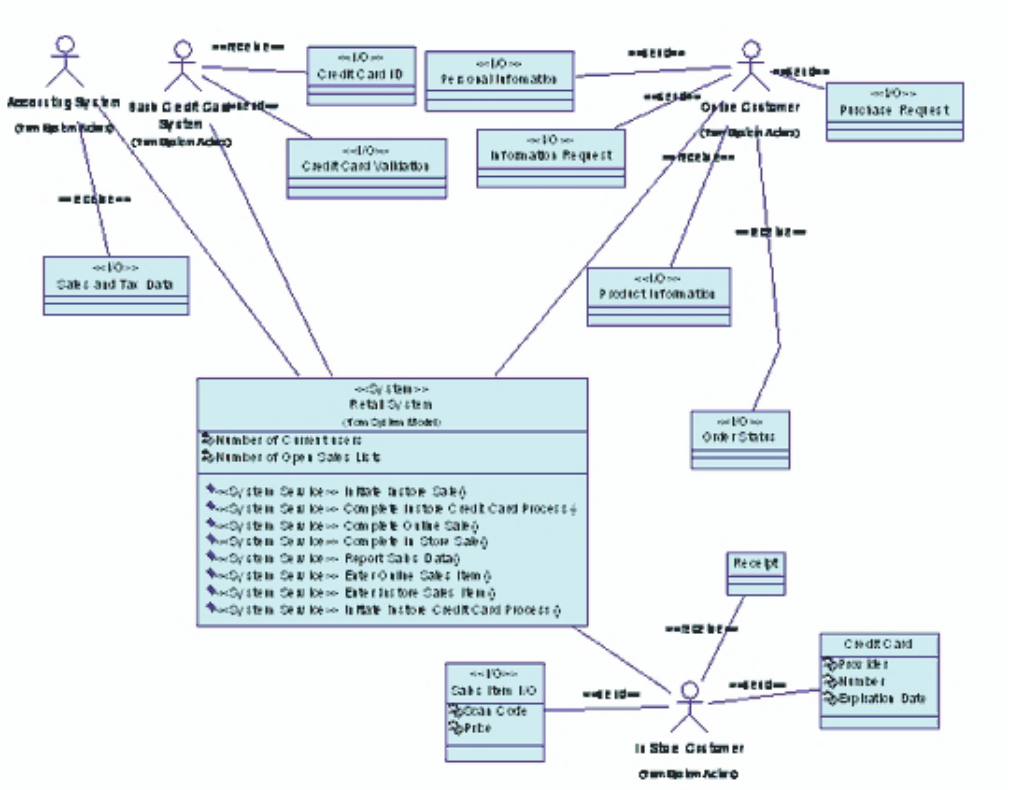

Figura 3: Diagrama de Contexto do Sistema de Varejo

Clique para expandir

Próximo Mês:

Esta série de artigos sobre o RUP SE continuará na publicação de setembro do The Rational Edge, com uma discussão de Arquitetura do Sistema, pois ela pertence ao desenvolvimento de sistemas. A terceira parte e a parte final aparecerão na publicação de outubro, como uma discussão de análise de requisitos, incluindo o fluxo de requisitos funcionais e suplementares para elementos de arquitetura -- organização do projeto, problemas de desenvolvimento interativo e integração.

Notas

[1] Blanchard and Fabrycky, Systems Engineering and Analysis (third edition), Prentice Hall, 1998.

[2] Por exemplo, o Oxford English Dictionary lista 11 definições.

[3] http://www.incose.org/whatis.html

[4] Mark W. Maier and Eberhardt Rechtin, The Art of Systems Architecting (second edition), CRC Press, 2000, p 8.

[5] Grande parte destas seções foram extraídas de Murray Cantor, "Leading Modern System Development ", publicado em The Rational Edge, janeiro de 2003, http://www.therationaledge.com/content/jan_03/f_modernSystemsDevelopment_mc.jsp

[6] Thomas P. Hughes, Rescuing Prometheus. Pantheon Books, 1998.

[7] David Longstreet, "Software Productivity Since 1970." http://www.ifpug.com/Articles/history.htm, 2002.

[8] Ibid.

[9] Paul Carlock and Robert Fenton, "System of System (SoS) Enterprise Systems Engineering for Information-Intensive Organizations," Systems Engineering, Vol. 4, No. 4, 2001.

[10] O diagrama de contexto do RUP SE é semanticamente equivalente ao Diagrama de Contexto Elaborado primeiramente descrito por Sanford Friedenthal Howard Lykins e Abe Meilich. Consulte http://www.omg.org/cgi-bin/doc?syseng/2001-09-05

[12] Esta distinção entre casos de uso e serviços é um recurso do OOSEM, loc cit.

[13] Para obter uma discussão do uso da semântica do classificador para subsistemas em UML, consulte o artigo de Fredrick Ferm, "The Why, What, and How of a Subsystem," The Rational Edge, June 2003. http://www.therationaledge.com/content/jun_03/t_subsystem_ff.jsp

---

Para obter mais informações sobre os produtos ou serviços discutidos neste artigo, clique aqui e siga as instruções fornecidas.

Obrigado!