# Capítulo

# 1

# O Enfoque de Linha de Produto para Desenvolvimento de Software

Itana Maria de Souza Gimenes e Guilherme Horta Travassos

***Abstract***

*This chapter presents the basic concepts of software product line that include the three essential activities: core asset development, product development and management. It also presents a survey of current approaches for software product lines. A case study that involves the development of a product line for satellite software is presented in order to illustrate the basic concepts.*

***Resumo***

*Este capítulo apresenta os conceitos básicos de linha de produto para o desenvolvimento de software, incluindo as três atividades essências para desenvolvimento de uma linha de produtos, a saber: desenvolvimento do núcleo de artefatos, desenvolvimento do produto e gerenciamento da linha de produto. Apresenta também uma revisão de abordagens existentes para linha de produto. Um estudo de caso sobre o desenvolvimento de uma linha de produto de software para controle de satélites é apresentado para ilustrar os conceitos básicos.*

## 1.1. Introdução

Há muito vem se formando uma consciência em engenharia de software de que para obtenção de produtos com alta qualidade e que sejam economicamente viáveis torna-se necessário um conjunto sistemático de processos, técnicas e ferramentas. A reutilização está entre as técnicas mais relevantes desse conjunto. Parte-se do princípio que reutilizando partes bem especificadas, desenvolvidas e testadas pode-se construir software em menor tempo e com maior confiabilidade. Muitas técnicas que favorecem a reutilização têm sido propostas ao

longo dos últimos anos. Entre estas técnicas estão: engenharia de domínio, *frameworks*, padrões, arquitetura de software e desenvolvimento baseado em componentes. Porém, o que falta nesse contexto é uma maneira sistemática e previsível de realizar a reutilização. Assim, o enfoque de linha de produto de software surge como uma proposta de construção sistemática de software baseada em uma família de produtos.

Em setores como o da indústria aeroespacial, automotiva e componentes eletrônicos, as técnicas de linhas de produto já vem sendo explorado há muito tempo [TRA 02]. Por exemplo, a linha de produtos da Mercedes-Benz em Sindelfinger (Alemanha) produz milhares de variantes das classes C-, E- e S- (existem aproximadamente 8000 variantes de *cockpits* e 10.000 variantes de assentos para classe-E). É difícil encontrar dois carros iguais saindo da linha de produtos no mesmo dia. O tipo certo de motor ou de qualquer outro componente deve estar disponível em qualquer momento na linha de produtos. O processo começa quando pessoas diferentes solicitam carros diferentes. O atendimento às solicitações dos clientes requer um grande esforço organizacional e técnico, o que envolve muito conhecimento de configuração do produto. Embora software seja construído de maneira completamente diferente de automóveis, aviões, ou computadores, engenheiros de software podem utilizar estratégias similares de produção.

Entende-se por família de produtos de software um conjunto de produtos de software com características suficientemente similares para permitir a definição de uma infra-estrutura comum de estruturação dos itens que compõem os produtos e a parametrização das diferenças entre os produtos. Por exemplo, uma família de produtos para sistemas de gerenciamento de *workflow* [WFM 95] tem uma arquitetura de software comum e um conjunto de componentes que povoam essa arquitetura, tais como escalonadores de tarefas, gerenciadores de recursos, interpretadores e gerenciadores de arquiteturas de *workflow*. Cada um desses componentes pode variar de um produto para outro, por exemplo, um produto pode requerer determinados tipos de recursos (ex. humano, equipamentos e salas). Cada produto pode ter uma política específica de escalonamento de tarefas (ex. prioridade de tarefas e classes de tarefas).

Uma linha de produto de software envolve um conjunto de aplicações similares dentro de um domínio que podem ser desenvolvidas a partir de uma arquitetura genérica comum, a arquitetura da linha de produto, e um conjunto de componentes que povoam a arquitetura. Esta abordagem tem por objetivo identificar os aspectos comuns e as diferenças entre os artefatos de software ao longo do processo de desenvolvimento da linha de produto, de modo a explicitar os pontos de decisão em que a adaptação dos componentes para geração de produtos específicos pode ser realizada. Para tal, durante o processo de desenvolvimento deve-se identificar os pontos de variabilidade que são pontos em que as características dos produtos podem se diferenciar. Esses pontos de variabilidade aparecem inicialmente na definição dos requisitos da linha de produto e devem ser representados ao longo de todo o processo de desenvolvimento. Dessa forma, pode-se decidir por uma característica ou outra para um produto específico tanto em nível de projeto quanto em nível de implementação.

Ainda existe uma grande controvérsia sobre o enfoque de linha de produto, considerando técnicas anteriores com propósitos similares, principalmente: engenharia de domínio e *frameworks*. O que parece mais produtivo é considerar o enfoque de linha de produto e essas técnicas como complementares. O enfoque de linha de produto pode ser visto como uma forma de organização dessas técnicas que nasceu da necessidade evidente de empresas como a Celsius Tech [CLE 01] que se encontrou em uma situação de sucesso por ter uma alta demanda por seus produtos, mas que não conseguiria atender sua demanda se não fosse encontrada uma forma de gerenciar sua família de produtos. Poulin [POU 97] afirma que linha de produto e domínios representam o mesmo nível de abstração e que o termo linha de produto foi criado para facilitar a comunicação entre o pessoal técnico e os gerentes que pensam em termos de produtos.

*Frameworks* e padrões também são importantes recursos para o enfoque de linha de produto. Para se conceber uma linha de produto é necessário partir de conceitos que levem a generalização de aplicações e que permitam instanciações para aplicações específicas. Os *frameworks* fornecem esse tipo de recurso. Um *framework* pode ser descrito como uma arquitetura de software semidefinida que consiste de um conjunto de unidades individuais e de interconexões entre elas, de tal forma a criar uma infra-estrutura de apoio pré-fabricada para o desenvolvimento de aplicações de um ou mais domínios específicos [LEW 95]. Portanto, *frameworks* são utilizados como mecanismos de apoio à linha de produto. Da mesma forma, os padrões permitem que a experiência de desenvolvedores de software seja documentada e reutilizada, registrando-se soluções de projeto para um determinado problema ou contexto particular [GAM 95]. Portanto, o enfoque de linha de produtos encontra nos padrões indicações de como construir tanto a arquitetura quanto os componentes desta.

A abordagem de linha de produto de software é recente na comunidade de engenharia de software. O primeiro *workshop* sobre esta abordagem [BAS 97] foi promovido pelo SEI em dezembro de 1996. Os resultados desse *workshop* sintetizam fatores envolvidos na prática de linha produtos. Neste *workshop* também foram analisados assuntos relacionados como arquitetura de software, pessoal, organização, administração e modelos empresariais.

O objetivo deste capítulo é apresentar os conceitos básicos de linha de produto para o desenvolvimento de software realizando uma revisão das pesquisas desenvolvidas na área. A seção 1.2 apresenta os elementos de uma linha de produto de software. A seção 1.3 apresenta uma revisão das abordagens mais relevantes de linha de produtos existentes atualmente. A seção 1.4 apresenta um estudo de caso de linha de produto de software. Finalmente, as conclusões e tendências são discutidas na seção 1.5.

No contexto de linha de produto de software herdamos do inglês três termos de difícil tradução: *features*, *commonalities* e *variabilities*. Neste texto, não traduziremos o termo *feature* por entender que o correspondente no português, característica, não traduz o que tecnicamente o termo feature significa (ver seção 1.2). Os termos *commonalities* e *variabilities* são traduzidos como características comuns e variabilidades, respectivamente.

## 1.2. Conceitos Básicos: Elementos de uma Linha de Produto de Software

Segundo Clements e Northrop [CLE 01] as três atividades essenciais da construção de uma linha de produto são:

- desenvolvimento do núcleo de artefatos;
- desenvolvimento do produto, e
- gerenciamento da linha de produto.

Essas três atividades estão intrinsecamente relacionadas de tal forma que a alteração em uma delas implica em analisar o impacto nas demais.

É importante notar que o desenvolvimento do núcleo de artefatos baseado em uma família de produtos faz a reutilização de artefatos para a linha de produtos previsível. Clements e Northrop [CLE 01] reconhecem explicitamente que a atividade de desenvolvimento do núcleo de artefatos tem sido chamada de engenharia de domínio e que a atividade de desenvolvimento do produto tem sido chamada de engenharia de aplicação. Weiss [WEI 99] também incorpora engenharia de domínio e engenharia de aplicação em sua abordagem denominada FAST (Family-Oriented Abstraction, Specification and Translation). Weiss reconhece que família de produtos também é conhecida pelo termo domínio.

O processo de desenvolvimento de uma linha de produto de software pode ser visto como dois modelos de ciclo de vida, conforme mostra a Figura 1.2.1.

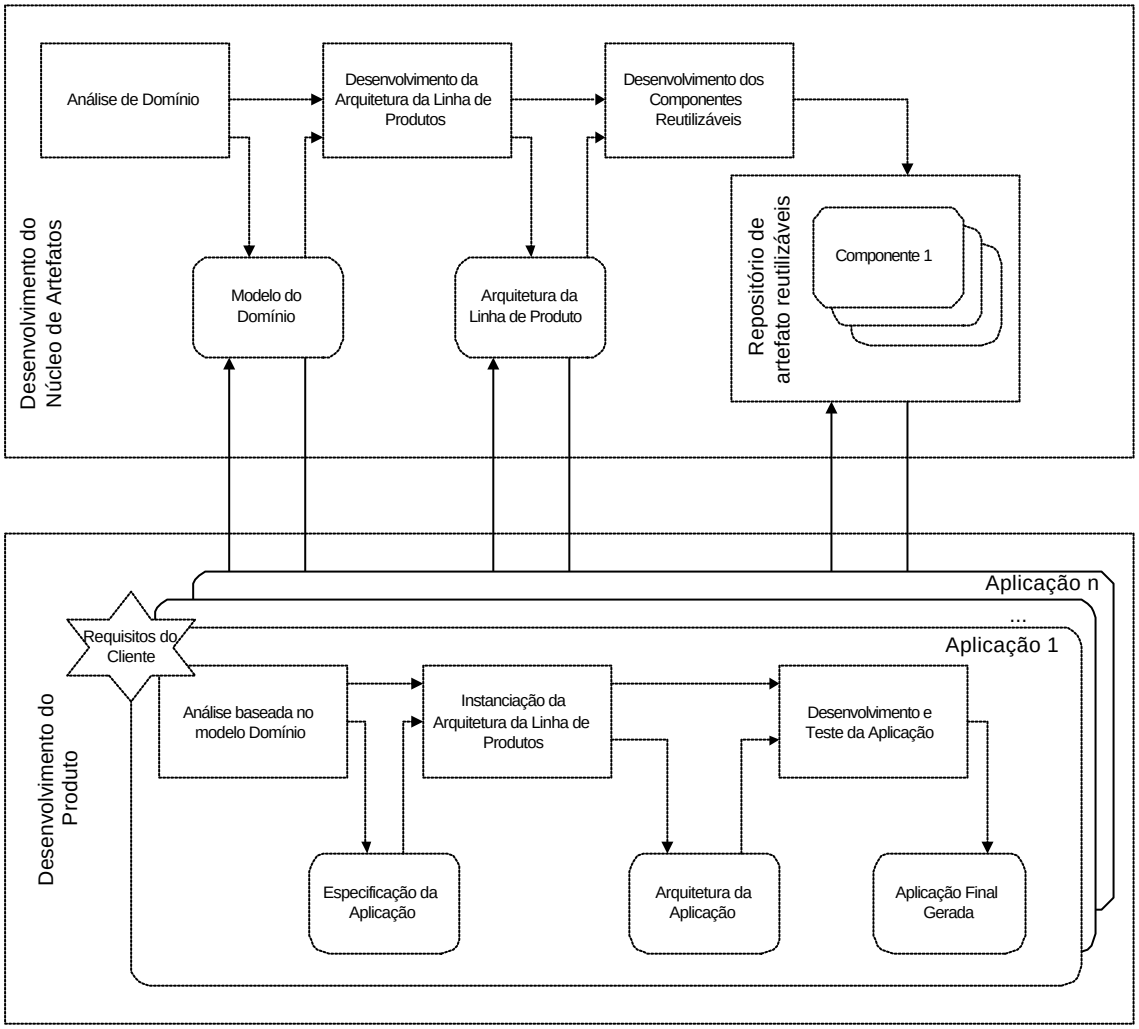


**Figura 1.2.1: O processo de desenvolvimento de uma linha de produto.**

O primeiro modelo de ciclo de vida representa o Desenvolvimento do Núcleo de Artefatos (Engenharia de Domínio), o qual engloba análise de domínio, desenvolvimento da arquitetura e desenvolvimento de componentes reutilizáveis. Este ciclo produz um modelo de domínio, uma arquitetura, um conjunto de componentes reutilizáveis e geradores de software para a linha de produtos. Um dos conceitos mais importantes no enfoque de linha de produto de software é o de varibilidade. Este conceito será explorado e definido na seção seguinte. O Desenvolvimento do Núcleo de Artefatos e o Desenvolvimento do Produto (Engenharia de Aplicação) serão descritos na seção 1.2.2 e 1.2.3 respectivamente.

## 1.2.1 Variabilidade

Para formar uma linha de produto é necessário representar as variações que podem ocorrer em cada artefato que faz parte desta. Produtos em uma linha de produto existem simultaneamente e podem se diferenciar em termos de comportamento (conjunto de ações), atributos de qualidade, plataforma, configuração física, fatores de escala, entre muitos outros [CLE 01].

Variações são diferenças tangíveis entre produtos que podem ser reveladas e distribuídas entre os artefatos da linha de produto, sejam eles a arquitetura, os componentes, as interfaces entre componentes ou as conexões entre componentes. As variações podem ser reveladas em qualquer fase do ciclo de desenvolvimento de uma linha de produtos, a começar da análise de requisitos. A variação mais conhecida em construção de sistemas é a existência de várias implementações para uma mesma especificação. Por exemplo, para uma especificação "classificar itens" que diz que itens entram desordenados e saem ordenados, pode-se implementar vários algoritmos como *Bublesort*, *Insertion sort* ou *Quicksort*. A expressão de uma variação pode ser obtida pela introdução de parâmetros instanciáveis em tempo de construção associados a componentes, subsistemas ou coleção de subsistemas a partir dos quais um produto pode ser configurado atribuindo-se um conjunto de valores a esses parâmetros [CLE 01]. Um tipo de variação simples de ser representado é a escolha de componentes diferentes para uma mesma arquitetura. Assim, produtos podem ter maior ou menor capacidade, ou características diferentes dependendo do tipo de componente escolhido para a arquitetura.

Em orientação a objetos variações podem ser representadas, em nível de projeto, por meio de especializações de determinadas classes. Por exemplo, suponha uma classe ferramenta que de acordo com o produto desejado possa ser interna ou externa. Isto pode ser representado por meio da especialização da classe ferramenta.

A integração dos componentes de uma arquitetura de linha de produtos tem também um papel fundamental. A instanciação de todas as variações possíveis deve ser consistente e ter um valor semântico que faça sentido para o produto desejado.

Existem vários mecanismos para tratamento de variabilidade. Por tratamento de variabilidade entende-se desde o momento de reconhecimento de um ponto de variação até o mapeamento do ponto para uma instância de variação.

Um conceito importante na representação de pontos de variação é o de *feature*. Este conceito tem origem na engenharia de domínio [KAN 90]. Uma feature é uma característica de um produto que usuários e clientes consideram importante na descrição e distinção de membros de uma família de produtos [GRI 00]. Um feature pode ser um requisito específico, uma seleção entre os requisitos opcionais e alternativos; ou pode estar relacionada a certas características do produto como funcionalidade, usabilidade e desempenho, ou pode estar relacionado às características de implementação como o tamanho, a plataforma de execução ou compatibilidade com certos padrões.

Um modelo de features consiste da representação dos tipos de features que podem ocorrer em uma família de produtos e seus interrelacionamentos. Um modelo de features é, geralmente, representado por um grafo em que os nós contém as features e seus atributos. Este grafo contém decorações para indicar features opcionais, obrigatórias e composição de features.

Apesar dos modelos de features apresentar alguma relação com o modelo de casos de uso, Griss et al. [GRI 98] apontam algumas diferenças entre esses modelos, que incluem:

- o modelo de casos de uso é orientado para o usuário e o modelo de features é orientado para o reutilizador;

- o modelo de casos de uso descreve os requisitos do usuário em termos de funcionalidades do sistema. O modelo de features organiza o resultado da análise dos aspectos comuns e variáveis, preparando uma base para reutilização;

- o modelo de casos de uso deve cobrir todos os requisitos de um sistema individual do domínio, enquanto que o modelo de features deve incluir apenas aquelas características que o analista do domínio considera importante pois este resume apenas os itens essenciais relativos aos objetivos do domínio;

- a notação utilizada para representar features é diferente daquele utilizada nos modelos de casos de uso pois nem todas as features podem ser automaticamente relacionadas com casos de uso;

- nem todas essas features aparecem nos casos de uso apesar de existir um grupo de features básicas que o usuário vê como capacidades do sistema. Algumas features surgem: no detalhamento da implementação; nas opções configuração do sistema, ou por sugestão de especialistas do domínio. Tais features podem aparecer somente nas fases de projeto e implementação.

Segundo a abordagem FeatuRSEB [GRI 98] a construção do modelo de features é um processo concorrente que tem como entrada sistemas exemplo, requisitos, especialistas do domínio e modelos anteriores do domínio. Essas entradas são alimentadas em um processo concorrente e contínuo de definição do contexto, modelagem de features e modelagem de casos de uso. O processo de construção do modelo de features pode ser resumido como segue:

1. mescle os modelos de casos de uso individuais para formar o modelo de caso de uso do domínio, mostrando os pontos de variação;

2. crie um modelo de features inicial com as features funcionais derivadas do modelo de casos de uso do domínio;

3. crie o modelo de objetos de análise, aumentando o modelo de features com features de arquitetura. Essas features estão relacionadas com a estrutura e configuração do sistema ao invés de funções específicas;

4. crie o modelo de projeto, aumentando o modelo de features com as features de implementação.

Jacobson et al. [JAC 97] discute features relacionadas com casos de uso. Uma feature é definida como um caso de uso ou parte deste. Neste contexto é sugerida a utilização do esteriótipo <<extend>> para representar aspectos de variação em casos de uso. O caso de uso estendido recebe uma marca, representada por um círculo preenchido para indicar um ponto de variação. Gurp et. al [GUR 01] propõem algumas extensões ao mecanismo de variabilidade proposto por Griss et al [GRI 98], considerando três categorias: entidade variante, entidade opcional e múltiplas entidades coexistentes.

Dessa forma, no projeto de uma linha de produto deve existir alguma forma de representar os aspectos que diferenciam os produtos membros da linha, como abordagem de features. Esses aspectos serão considerados na próxima fase, desenvolvimento do produto, para instanciar o produto específico adequado.

### 1.2.2. Desenvolvimento do Núcleo de Artefatos

No desenvolvimento do núcleo de artefatos três aspectos devem ser considerados: a definição do contexto da linha de produto, o núcleo de artefatos e o plano de produção. Um artefato é um item reutilizável de software utilizado como bloco de construção de uma linha de produto. Pode ser uma arquitetura de software, um componente, um framework, ou alguma documentação relativa à arquitetura de software ou a algum componente.

**Arquitetura da linha de produto**

Dentre os artefatos da linha de produto, a arquitetura tem um papel importante. Ela representa a infra-estrutura central da linha de produto. A arquitetura de um sistema de software engloba a definição de suas estruturas gerais, descrevendo os elementos que compõem os sistemas e as interações entre estes [BAS 98]. Além de descrever esses elementos e suas interações, uma arquitetura de software apóia também questões importantes de projeto, tais como: a organização do sistema como composição de componentes, as estruturas de controle globais, os protocolos de comunicação, a composição dos elementos do projeto e a designação da funcionalidade dos componentes do projeto.

Uma arquitetura representa um investimento significativo para uma organização. Em função disto, é natural que as organizações desejem maximizar este investimento por meio da reutilização da arquitetura em vários sistemas. Quando uma organização está produzindo vários sistemas similares, reutilizando a mesma arquitetura e os componentes associados a esta, pode reduzir seus custos de produção e o tempo necessário para disponibilizar um produto no mercado.

A arquitetura genérica ou framework de arquitetura da linha de produto representa um meta-projeto das aplicações que constituem a família de produtos. Alguns requisitos importantes da arquitetura de linha de produtos são [CLE 01]:

- ser estável, persistindo durante a vida da linha de produtos, sofrendo poucas alterações;
- permitir integração flexível de novas funcionalidades durante o ciclo de vida da arquitetura;
- permitir a representação explícita de variações de modo a maximizar a reutilização;
- fornecer mecanismos para implementar variações;
- permitir as variações inerentes da família de aplicação, e
- permitir as variações impostas pelo mercado.

O projeto da arquitetura de software tem sido objeto de muitas técnicas recentes, que devem ser consideradas no projeto de uma arquitetura de linha de produtos, em particular dos métodos de desenvolvimento baseado em componentes [Che 00] [D'So 99] [Kru 00] [BAS 98]. Outros trabalhos importantes são: a padronização de arquiteturas de software com UML da OMG [OMG 02] e as investigações dedicadas a articular o relacionamento entre especificação requisitos com o projeto da arquitetura de software [CAS 01].

**Componentes da linha de produto**

Um componente pode ser definido como uma unidade de software independente, que encapsula dentro de si seu projeto e implementação, e oferece interfaces bem definidas para o meio externo. Componentes têm pontos de interconexão chamados de Interfaces que concentram um conjunto de serviços relacionados. As interfaces são chamadas de Interfaces Fornecidas (*Provided Interfaces*) ou Interfaces Requisitadas (*Required Interfaces*). A interface fornecida define as operações que o componente oferece a outros componentes. A interface requisitada define as operações que o componente requisitará de outros componentes. A interface requisitada de um componente conecta-se à interface fornecida pelo componente que oferece os serviços correspondentes. Um componente é entendido por seus usuários por meio de sua especificação. A especificação de um componente deve estabelecer suas interfaces, argumentos, pré e pós-condições de cada serviço e invariantes. A conexão de componentes é realizada por protocolos de comunicação que vão desde simples chamadas de procedimentos até infra-estruturas de comunicação como DCOM e CORBA.

### 1.2.3. Desenvolvimento do Produto

O desenvolvimento do produto, também conhecido como engenharia de aplicação, em geral, inclui os seguintes artefatos:

- o modelo do domínio, base para identificar os requisitos do cliente;
- um framework de arquitetura de linha de produtos, base para especificar uma arquitetura para um membro da família;
- um conjunto de componentes reutilizáveis a partir do qual um subconjunto de componentes será integrado à arquitetura para gerar um produto.

De acordo com o ciclo de vida mostrado na Figura 1.2.1, a primeira atividade é a análise baseada no modelo do domínio. Nesta atividade, o modelo de domínio é utilizado para elicitar informações sobre a aplicação específica a ser desenvolvida e definir uma lista de características para a aplicação. Assim, o modelo do domínio representa as necessidades do cliente em termos da arquitetura de linha de produto e dos componentes disponíveis no repositório. Associado ao modelo de domínio tem-se o modelo de decisão que define os principais aspectos a serem considerados no desenvolvimento do produto. Este modelo permite ao desenvolvedor da aplicação, por exemplo, identificar os ajustes necessários nos valores iniciais das características descritas pelos componentes tendo em vista as restrições arquiteturais impostas pelas variabilidades. As necessidades não cobertas pelo modelo do domínio são denominadas novos requisitos. Os novos requisitos podem implicar na necessidade de estender a arquitetura da linha de produto e adquirir novos componentes. Para os requisitos não satisfeitos pelo modelo existente, deve ser investigado o custo de desenvolver o software por encomenda, modificar a arquitetura ou os componentes existentes. A lista de características da aplicação recuperada do modelo de domínio mais aquelas definidas nos novos requisitos são integradas para formar a especificação do produto. Informações sobre os requisitos que não foram antecipados no modelo de domínio devem ser realimentadas para estender o modelo de domínio e/ou o repositório de componentes existentes.

A segunda atividade é a instanciação da arquitetura do produto. O *framework* de arquitetura é percorrido de cima para baixo, tomando-se decisões recursivamente até que a arquitetura genérica seja transformada na arquitetura específica do produto. O processo de construção da arquitetura é iterativo. A instância inicial provavelmente contém pendências como pontos de variação e decisões ainda não resolvidos. Essas pendências, geralmente, ocorrem devido aos requisitos específicos da aplicação. A implementação desses requisitos pode provocar modificações no *framework* de arquitetura da linha de produto ou apenas afetar a instância da arquitetura da aplicação. A decisão depende da perspectiva de utilização do requisito por outros clientes.

A terceira atividade refere-se ao povoamento da arquitetura com os componentes adequados. As técnicas utilizadas neste processo são similares àquelas utilizadas para o desenvolvimento baseado em componentes. Como nas atividades anteriores, atenção especial deve ser dada a implementação dos requisitos específicos da aplicação. Os componentes podem ter várias origens, tais como:

- o repositório da linha de produtos pode conter o componente desejado;
- um novo componente é implementado, encomendado ou adquirido de terceiros, pois os requisitos do usuário não foram satisfeitos pelos componentes existentes no repositório;
- o requisito do cliente é implementado na própria aplicação, e
- um componente de prateleira (COTS) é adquirido;
- um componente é obtido de aplicações legadas.

Os testes da aplicação são feitos também de forma similar às técnicas de desenvolvimento baseado em componentes que incluem: teste de unidade e verificação de componentes, teste de integração e verificação geral das funcionalidades da aplicação.

### 1.2.4. Gerenciamento do produto

Grande parte do sucesso de uma linha de produto depende do compromisso da organização em construí-la e mantê-la. Assim, é importante que a organização estabeleça um plano de adoção da linha de produtos que descreva o estado desejado e as estratégias para atingir este estado [CLE 01]. Para tal é necessário prover a equipe responsável pela linha de produto com recursos e garantir que as atividades da linha de produtos sejam coordenadas e supervisionadas. Deve haver um sincronismo entre o grupo que desenvolve os artefatos e o grupo que gera os produtos. Deve também ser garantido o apoio ao desenvolvimento e à evolução dos artefatos da linha.

## 1.3. Abordagens para Desenvolvimento de Linha de Produto de Software

### 1.3.1. Introdução

A literatura técnica apresenta algumas abordagens e estratégias que suportam a tecnologia de linha de produto de software. Algumas delas apresentam uma solução abrangente para as questões relacionadas à representação de aspectos ligados à engenharia do domínio e aplicação bem como aos conceitos inerentes a evolução, gerenciamento, análise da relação custo e benefício, tomadas de decisão orientada ao mercado e avaliação de riscos. Outras fornecem métodos concentrados em alguns dos aspectos inerentes a tecnologia de linha de

produto, tais como a definição da família de produtos, construção da infra-estrutura arquitetural básica ou mesmo a implementação de componentes reutilizáveis. Neste caso, elas estão usualmente integradas a algum outro método específico de engenharia de software como, por exemplo, análise de *features*, desenvolvimento baseado em componentes ou mesmo a utilização da notação UML ao longo de um processo de desenvolvimento de software. Algumas destas abordagens são comentadas a seguir. As referências podem ser utilizadas pelos leitores para enriquecer os comentários encontrados nesta seção.

### 1.3.2. A iniciativa PLP (Product Line Practice) do SEI/CMU

Um exemplo de estratégia que pode ser utilizada para implementação de linha de produto é a estrutura para prática de linha de produto de software (PLP) desenvolvida pelo *Software Engineering Institute*. Neste contexto, o desenvolvimento de uma linha de produto envolve atividades relacionadas ao desenvolvimento de artefatos centrais e desenvolvimento e gerenciamento do produto utilizando estes artefatos centrais. Para executar estas atividades torna-se necessária a definição de áreas de trabalho que representam conjuntos menores de atividades, porém mais gerenciáveis. Cada área de trabalho (*Practice Area*) possui um plano de trabalho e métricas associadas que permitem acompanhar sua execução e avaliar o sucesso dos trabalhos realizados. Usualmente estas áreas de trabalho produzem artefatos concretos que levam a criação, de alguma forma, aos artefatos centrais que serão utilizados na linha de produto. É uma descrição de abordagem similar a do CMM. Basicamente, a estratégia PLP define três áreas de trabalho:

- Engenharia de Software: necessária para aplicar a tecnologia apropriada à criação e evolução dos artefatos ou componentes centrais e do produto.

- Gerenciamento técnico: relacionada à criação e evolução dos artefatos ou componentes centrais.

- Gerenciamento organizacional: utilizada para o gerenciamento dos esforços relacionados à linha de produto como um todo.

A versão corrente da iniciativa PLP define todas as atividades para as áreas de trabalho e vem evoluindo continuamente através das experiências obtidas de diferentes estudos de caso realizados com a colaboração extensa existente entre o SEI e as organizações que estudam a utilização de linha de produto, além de informações recebidas da comunidade de software. Uma descrição mais detalhada desta estratégia pode ser obtida no endereço http://www.sei.cmu.edu/plp/framework.html#framework_toc.

### 1.3.3. A Abordagem Synthesis

Synthesis foi elaborado pelo Software Productivity Consortium [http://www.software.org/pub/products/productlines.asp] e descreve uma metodologia

abrangente para a construção de sistemas de software representando instâncias de uma família de sistemas que possuem descrições similares. Suas características principais são:

- formalização de domínios como famílias de sistemas que compartilham várias características em comum, e que também variam em formas e configurações bem definidas;

- redução da construção de Sistemas à resolução de requisitos com decisões de engenharia representando as características de variações existentes no domínio;

- reutilização de artefatos de software através de adaptação mecânica dos componentes para satisfazer aos requisitos e as decisões de engenharia, e

- análise baseada na descrição do modelo do sistema de forma a auxiliar a compreensão das implicações de decisões de construção para com o sistema, permitindo assim avaliar as possíveis alternativas de construção.

Quatro princípios básicos norteiam a aplicação de Synthesis:

- família de programas: especificando que desenvolvedores deveriam construir programas de software não como artefatos únicos, mas como instâncias de uma família de programas similares;

- processo iterativo: considerando o desenvolvimento de produtos de software como uma repetição de atividades de produção e utilização;

- especificações: descrevendo as propriedades verificáveis a serem consideradas para um produto membro ou um conjunto de produtos membros a serem produzidos, e

- reutilização baseada em abstrações: fornecendo um meio para representar uma família de produtos como um conjunto de produtos membro adaptáveis e também para derivar instâncias de cada produto membro e produzir um sistema produto em particular.

Essencialmente Synthesis descreve os aspectos principais do desenvolvimento de uma linha de produto de software, comuns a todas as metodologias relacionadas a este tópico. Um detalhamento sobre a metodologia pode ser obtido em [SPC 93].

### 1.3.4. A Abordagem FAST

A análise de características comuns é um aspecto marcante da utilização de **FAST** (Family-oriented Abstraction, Specification and Translation) [ARD 97]. Esta análise serve para

identificar o contexto, descrever o domínio, fornecer um conjunto padrão de termos técnicos chave, identificar características comuns e variabilidades, quantificar as variabilidades através do fornecimento de parâmetros de variação, identificar e registrar várias informações úteis que aparecem durante a análise.

A análise de características comuns é realizada por meio de uma série de reuniões com especialistas de domínio, facilitadas por um moderador. A equipe de análise produz o documento em grupo durante a reunião, decidindo sobre seu conteúdo por consenso e guiada pelo moderador. Tipicamente, cada especialista é especializado em um ou mais aspectos do domínio, enquanto o moderador deve possuir experiência no processo de análise de características comuns e possuir habilidade para reconhecer quando as definições estão precisas, claras e bem formadas, identificar características comuns, variabilidades, parâmetros de variação e questões úteis que deverão compor o documento.

A análise de características comuns é organizada em cinco estágios: preparação, planejamento, análise, quantificação e revisão externa. A Figura 1.3.1 apresenta a estrutura do processo de análise de características comuns de FAST.

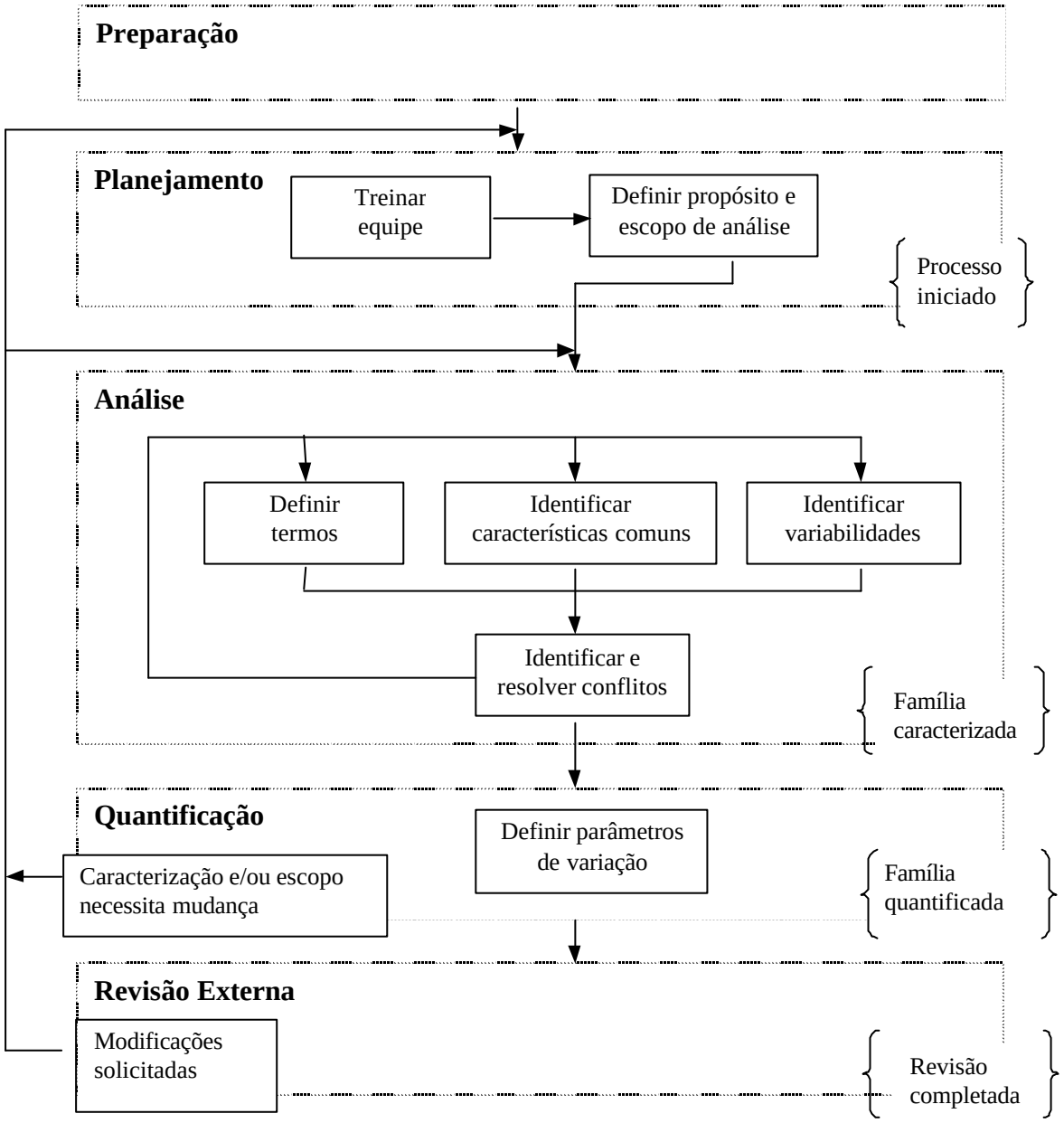


**Figura 1.3.1 – Estrutura do Processo de Análise de Características comuns de FAST [TRA 02].**

No estágio Preparação, o moderador assegura que todos os recursos necessários estão disponíveis para as seções seguintes. O moderador prepara um modelo de documento para registrar a definição e descrição dos termos, características comuns, variabilidades e outras informações de saída do processo de análise de características comuns. Além do esqueleto do documento, algumas definições iniciais de alguns termos e conceitos, incluindo características comuns e variabilidades, são preparadas e incluídas no documento permitindo iniciar a discussão.

O estágio de planejamento começa com uma breve revisão dos conceitos do processo de análise de características comuns. Então, a equipe revisa os documentos iniciais, os aprova, e determina as fronteiras do domínio. Durante a execução dos estágios seguintes do processo de análise de características comuns estas informações são usualmente revisadas visando definir a família de produto e tornar mais explícitas suas interfaces com outros domínios.

Durante o estágio de análise a equipe gera os termos técnicos e suas definições, as características comuns e variabilidades, identificando, ainda, as questões relacionadas ao domínio. Existe uma relação estreita entre a definição de termos e a descoberta de características comuns. Normalmente, os termos são identificados através da utilização da relação "*é-um*". As características comuns normalmente são identificadas sob a forma "*tem atributo*". Além dos atributos comuns, alguns produtos membro usualmente possuem um conjunto de atributos especiais ou opcionais representando as variabilidades. Existem dois tipos de variabilidade: funções diferentes executadas por produtos membro e possíveis modificações de alguns produtos membro que devem ser realizadas no futuro. Questões, normalmente, representam termos, características comuns ou variabilidades com informação insuficiente para sua perfeita identificação. Para cada questão um número de alternativas é estabelecido. Então, a questão é associada a um dos membros da equipe para investigação adicional. O estágio termina quando nenhum novo termo, característica comum ou variabilidade possa ser descoberto e a descrição das características já identificadas não sofrem modificações durante as reuniões.

O estágio de quantificação consiste na geração dos parâmetros de variação. Um parâmetro de variação é uma visão quantificada da variabilidade especificando a faixa de valores e o momento em que deve ser utilizada para tomada de decisão. Para cada variabilidade a equipe define o tipo de decisão que ela representa, quantifica a faixa de valores, especifica o tempo de ligação, e fornece um valor *default*, se existir algum, para a tomada de decisão de projeto.

A revisão externa tem por objetivo primariamente descobrir hipóteses e definições incorretas ou omitidas, e inconsistências na análise de características comuns. Além disso, ela revela a qualidade externa documento, já que diferentes especialistas de domínio ou usuários não envolvidos no processo de análise estarão utilizando o documento e avaliando sua clareza, legibilidade e compreensão.

O processo de análise de características comuns do FAST começou como um projeto experimental na Lucent em 1992 e ainda se encontra em desenvolvimento. Algumas questões relacionadas à alocação de recursos, atributos que caracterizam um bom moderador, definição de trabalhos para os membros da equipe que não foram discutidos aqui podem ser encontrados na literatura [WEI 99].

### 1.3.5. A Abordagem PULSE

**PuLSE** (Product line Software Engineering) [BAY 99] é uma metodologia de construção e utilização de linhas de produto. A estrutura geral de PuLSE inclui fases de desenvolvimento (iniciação, construção de infra-estrutura, utilização da infra-estrutura e evolução e gerenciamento), componentes técnicos (PuLSE-Eco utilizado para identificação de escopo para um membro da linha de produto; PuLSE-CDA contém análise de domínio e elaboração do modelo de decisão que será utilizado para a criação da arquitetura da linha de produto; PuLSE-DSSA suporta a definição de arquitetura específica de domínio o que implica no desenvolvmiento incremental da estrutura de linha de produto guiado por cenários; PuLSE-I executa a instanciação do produto na fase que o utiliza, e PuLSE-EM que lida com o gerenciamento de configuração e questões relacionadas a evolução do produto), e componentes de suporte (Pontos de entrada de projeto que customizam PuLSE para tipos maiores de projeto; Escala de maturidade que fornece caminhos para integração e evolução; e, Questões organizacionais que fornecem diretrizes para configuração e manutenção da estrutura de organização). A metodologia PuLSE serve como base para vários de projetos industriais e científicos, como por exemplo, KobrA descrito neste seção.

PuLSE-Eco (Economic Scoping) é uma forma de identificação e representação do contexto econômico de uma linha de produto que compreende as seguintes atividades:

- determinação antecipada de produtos;
- mapeamento e avaliação das características dos produtos membro;
- determinação das melhores características e melhores candidatos para uma linha de produto, e
- caracterização dos candidatos escolhidos e fornecimento da análise de benefício para cada candidato.

O resultado apresenta o contexto econômico da linha de produto que consiste de uma lista de características significantes para o domínio em questão e um mapa do produto. Este mapa é uma tabela que fornece uma classificação de produtos membro existentes, futuros e potencias de acordo com as caracerisitcas que eles possuem.

PuLSE-CDA (Customizable Domain Analysis) inicialmente refina o contexto econômico para especificar os limites da linha de produto. Para tal, ele utiliza uma técnica de modelagem chamada *storyboard*. Os *storyboards* capturam tipos de seqüência de ações relevantes para o domínio. Estes tipos variam para diferentes domínios. Exemplos são diagramas de *workflow* e diagramas de seqüência de mensagens. Esta técnica de modelagem presume a elicitação da informação sobre sistemas simples em *storyboards* não formatados e a consequente consolidação deste conhecimento em *storyboards* genéricos. Dessa forma, o

componente PuLSE-CDA captura as variabilidades. Para derivar a especificação de produtos membro de *storyboards* genéricos, um modelo de decisão é criado contendo um conjunto estruturado de decisões. Cada decisão corresponde à variabilidade no *storyboard*, juntamente com um conjunto de possíveis soluções.

PuLSE-DSSA implica numa especificação incremental da arquitetura guiada por cenários. PuLSE-DSSA categoriza cenários como genéricos (descrevendo requisitos funcionais) ou relacionadas à propriedade (descrevendo aspectos de qualidade independente do domínio). Os cenários genéricos são derivados do *storyboard* genérico que são representações de modelo de domínio em PuLSE e juntamente com o modelo de decisão compõem o resultado final da fase PuLSE-CDA. O desenvolvimento da arquitetura começa com um conjunto inicial de cenários genéricos que é utilizado para criar a estrutura de arquitetura inicial. Em seguida, um conjunto de cenários genéricos é escolhido de acordo com sua importância para a arquitetura e aplicados à estrutura de arquitetura inicial. Dessa forma, é possível o refinamento da estrutura de arquitetura inicial e sua extensão para que suporte todos os cenários genéricos, evoluindo assim para seu estágio final.

A aplicação dos cenários genéricos pode resultar em mais de uma alternativa de arquitetura. Neste caso, os cenários relacionados às propriedades e adicionados aos cenários genéricos correntemente utilizados, são aplicados e classificados visando avaliar a aderência do candidato à estrutura arquitetural. O resultado desta classificação é um candidato a fazer parte da estrutura arquitetural que deverá será estendido.

O modelo de decisão produzido durante a fase PuLSE-CDA é também melhorado no contexto do processo PuLSE-DSSA. Algumas decisões específicas de implementação são colecionadas enquanto a estrutura arquitetural está evoluindo. Estas decisões terão que ser resolvidas durante a instanciação da estrutura arquitetural com o propósito de criação de uma arquitetura para o produto membro. Capturando estas decisões ao longo do processo com suas possíveis resoluções formam um modelo de configuração que estende o modelo de decisão.

### 1.3.6. A Abordagem Bosch

Bosch [BOS 00] considera uma linha de produtos em três dimensões que estão organizadas em três conjuntos de elementos, a saber: (1) arquitetura, componente e sistema; (2) negócios, organização, processos e tecnologia; e (3) desenvolvimento, aplicação e evolução.

A primeira dimensão divide o domínio da linha de produto no que diz respeito aos artefatos iniciais que são parte do desenvolvimento baseado em reutilização. Esta abordagem é adotada por [JAC 97]. Os artefatos principais são resumidos a seguir.

- **Arquitetura:** o primeiro artefato de uma linha de produto é a arquitetura de software. O objetivo principal é projetar uma arquitetura que cubra todos os requisitos dos

produtos da linha de produto e inclua características que possam ser compartilhadas entre esses produtos.

- **Componente:** o segundo conjunto de artefatos de uma linha de produto contém os componentes identificados para arquitetura. A arquitetura de linha de produto identifica os componentes e a variabilidade requerida para esses. Segundo Bosch [BOS 00], componentes tendem a ser entidades relativamente grandes, chegando a ter mais de cem mil linhas código sendo mais relacionados à *frameworks* orientados a objetos do que a classes nos sistemas tradicionais.

- **Sistema:** o conjunto final de artefatos compreende os sistemas construídos e baseados na arquitetura de linha de produto e seus componentes. A construção do sistema requer uma adaptação da arquitetura de linha de produto para adequar a arquitetura no sistema. Este processo pode requerer a adição ou remoção de componentes na arquitetura, bem como a adição ou remoção de relacionamentos entre componentes da arquitetura. Pode ainda ser necessário o desenvolvimento de extensões específicas do sistema para os componentes restantes, a configuração dos componentes do sistema e o desenvolvimento de código específico para esses componentes.

A segunda dimensão, de acordo com que se pode ver a linha de produto está relacionada com as diferentes visões de uma organização. Bosch [BOS 00] relata que os *workshops* do SEI utilizam estas visões para descrever questões sobre linha de produto. Estas questões são discutidas abaixo com mais detalhes.

- **Negócio:** todas as atividades de uma organização devem, de alguma forma, ser justificadas em termos de benefícios de negócio. Com respeito às linhas de produto, um importante aspecto é que geralmente se requer um investimento considerável para se converter de uma abordagem orientada a um produto, para uma abordagem de desenvolvimento orientada à reutilização e baseada em linha de produto [BOS 00]. O projeto da arquitetura de linha de produto e o desenvolvimento dos componentes geralmente são caros, e o mais importante, normalmente atrasa o tempo inicial de comercialização dos produtos.

- **Organização:** a abordagem tradicional de linha de produto sugere o uso de um domínio que cria um departamento responsável pelo desenvolvimento e evolução da linha de produto e dos componentes reutilizáveis. Abordagens alternativas não empregam necessariamente um domínio que cria unidades, mas geralmente têm efeitos em toda organização. Por exemplo, componentes reutilizáveis são desenvolvidos por projetos de engenharia de domínio e pessoas que se envolveram neste processo. Assim, a mudança da abordagem tradicional de desenvolvimento para o desenvolvimento de software baseado em reutilização também cria efeitos organizacionais, que necessitam ser tratados.

- **Processos:** a terceira questão diz respeito aos processos orientados a reutilização de uma organização. O fato de se ter disponível uma arquitetura de linha de produto e um conjunto de componentes implica em efeitos consideráveis no processo associado a um produto ou ao desenvolvimento de um sistema. Existem também, processos específicos de linha de produto relacionados ao projeto da arquitetura de linha de produto e ao desenvolvimento dos componentes reutilizáveis.

- **Tecnologia:** a questão final de uma organização orientada a reutilização está relacionada com a tecnologia utilizada para apoiar o desenvolvimento de software baseado em reutilização. Bosch [BOS 00] emprega a noção de tecnologia em sentido amplo, incluindo os artefatos da organização, como os componentes reutilizáveis, por exemplo. A tecnologia ou técnicas utilizadas no desenvolvimento de componentes como também na reutilização destes componentes também estão incluídas. Exemplo disso são os *frameworks* orientados a objetos e os padrões de projeto.

A terceira dimensão pela qual se pode descrever a linha de produtos de software está baseado no ciclo de vida de cada um dos artefatos da organização e envolve desenvolvimento, aplicação e evolução. Em [BOS 00], o autor enfatiza esta dimensão como sendo preliminar porque representa as fases iniciais para que uma organização adote a abordagem baseada em linha de produto.

Como resultado dos processos de desenvolvimento, aplicação e evolução, vários artefatos são produzidos. A Figura 1.3.2 apresenta uma visão geral desses artefatos.

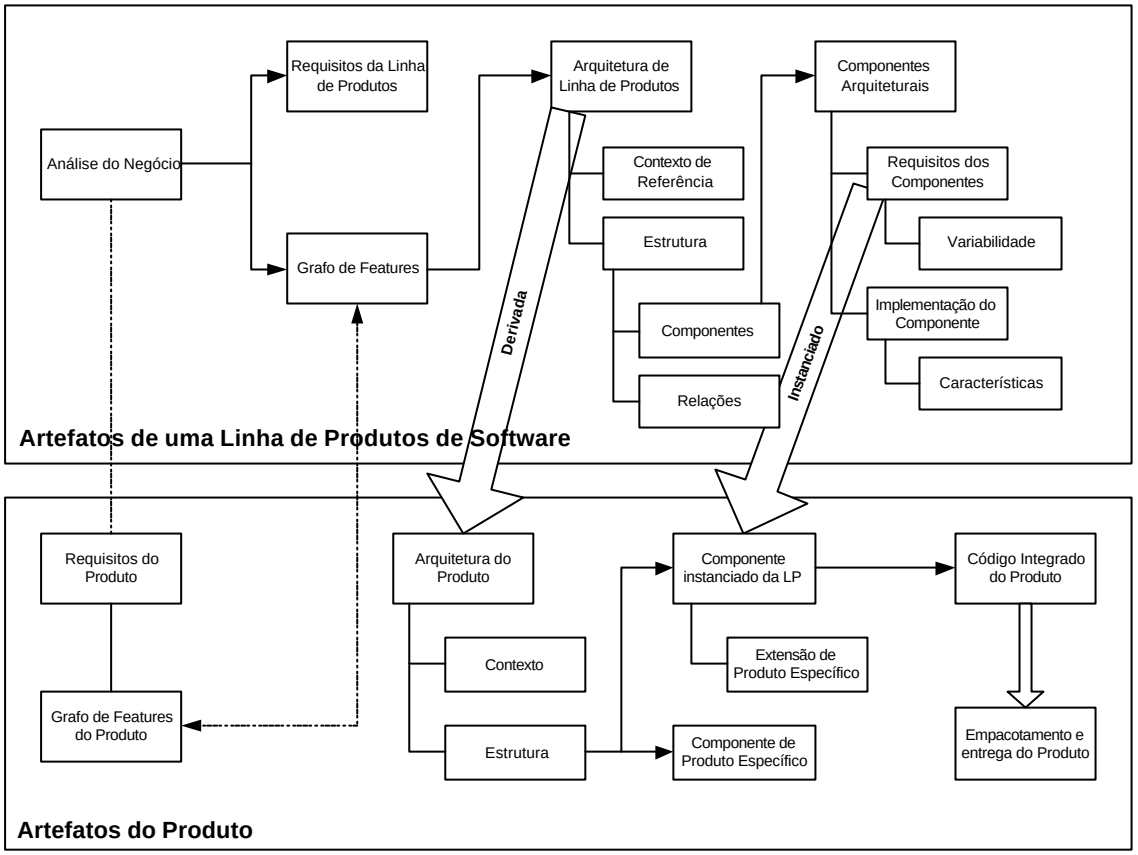


**Figura 1.3.2: Artefatos de uma Linha de Produto de Software[BOS 00].**

Os artefatos são divididos em duas categorias principais: os artefatos que são compartilhados pelos membros da linha de produto e os artefatos de produtos específicos. Os artefatos compartilhados entre produtos e linha de produto, são conseqüência da análise do negócio e dos requisitos da linha de produto. Além disso, os artefatos devem incluir a arquitetura de linha de produto, um ou mais contextos de referência e a estrutura da arquitetura de software em termos dos componentes e suas relações.

A arquitetura de linha de produto define a associação dos componentes arquiteturais com os requisitos desses componentes e suas possíveis implementações. Cada implementação de componente possui características associadas, descrevendo semelhanças com produtos particulares.

Os artefatos de produtos específicos consistem dos requisitos desses produtos e de um grafo de features do produto. Além disso, define-se a arquitetura para um produto derivado da arquitetura de linha de produto, incluindo a descrição do contexto e a estrutura.

De acordo com a arquitetura, as implementações de componentes instanciados da linha de produtos de software podem ser configuradas por extensões de produtos específicos. Por fim, o código integrado, o empacotamento e a liberação do produto finalizam os artefatos de um produto específico.

### 1.3.7. Outras Abordagens e Aplicações

**KobrA** [ATK 00] é um método que integra os conceitos de linha de produto e desenvolvimento baseado em componentes. Os princípios de construção e utilização de linhas de produto são baseados na metodologia PuLSE. KobrA considera o desenvolvimento e manutenção de *Framework* o que inclui todas as alternativas possíveis de realização da arquitetura de produtos membro. O conceito central do método é um componente KobrA (*Komponent*) descrito em diferentes níveis de abstração e organizados em forma de uma árvore. Cada *Komponent* é descrito em dois níveis de abstração – uma especificação definindo as propriedades externas visíveis de um *Komponent* e comportamentos, e uma realização descrevendo como um *Komponent* preenche suas responsabilidades quando relacionados a outros *Komponents* de mais baixo nível. A descrição da especificação e realização de um *Komponent* é realizada através da utilização de diagramas UML.

A especificação de um *Komponent* é derivada da estrutura arquitetural da linha de produto e fornece a definição da interface do *Komponent* utilizando quatro modelos básicos: estrutural, comportamental, funcional e decisão. Estes modelos representam respectivamente o estado do *Komponent*, sua reação a estímulos externos, seu impacto sobre entidades externas e a natureza de trocas no comportamento do *Komponent* dependendo do ambiente.

A realização do *Komponent* é também representada por quatro modelos principais: interação, estrutural, atividade e decisão. Estes modelos são relacionados aos modelos de especificação, mas ao contrário deles, representam como um *Komponent* realiza sua funcionalidade ao invés de mostrar o que é a funcionalidade. Se a interface especificada de um Komponent está de acordo com a interface fornecida por um componente pré-existente, como por exemplo, COTS ou algum sistema legado reestruturado, ele poderia ser utilizado normalmente com algum ajuste.

O método KobrA está sendo correntemente avaliado num contexto industrial através do desenvolvimento de um estudo de caso para o domínio de Planejamento de Recursos Organizacionais.

**GenVoca** [BAT 98] é uma metodologia baseada nos conceitos de máquina virtual, camadas de componentes (representando uma implementação da máquina virtual) e arquitetura-*realm* (um conjunto de componentes de mesmo tipo). GenVoca considera também os mecanismos de integração e operação para esses conceitos os quais incluem equações de tipos, regras de projeto, gramáticas e geradores. A utilização da metodologia GenVoca tem mostrado algumas vantagens como por exemplo, melhor desempenho da aplicação produzida, aumento de produtividade e redução de erros de desenvolvimento em comparação com técnicas de desenvolvimento convencionais. Entretanto, existem ainda algumas barreiras técnicas e não-técnicas para as quais os autores da metodologia intencionam resolver futuramente.

A tecnologia de linha de produto vem sendo aplicada com sucesso em diferenetes projetos na indústria. Por exemplo, [CZA 99] aplicaram a tecnologia de linha de montagem para a criação

de alguns sistemas em escala real no domínio de bibliotecas para tratamento de matrizes, ou mesmo o programa STARS (Software Technology for Adaptable, Reliable Systems) vem sendo desenvolvido de acordo com a tecnologia de linha de produto. Projetos reacionados a utilização da abordagem STARS podem ser encontrados no endereço http://www.asset.com/stars/home.html#progslide.

## 1.4 Estudo de Caso: A Experiência de Construção de Uma Linha de Produto de Software para Direcionamento, Navegação e Controle de Satélites

### 1.4.1. Introdução

A experiência que descrevemos a seguir representa uma iniciativa de implantação de uma linha de produto para o domínio de software para navegação, direcionamento e controle de satélites. Este projeto foi desenvolvido durante os anos de 1998-2000 no *Flight Software Branch* (FSB) do *Goddard Space Flight Center* (GSFC), um dos centros de pesquisa da NASA especializado no desenvolvimento de software controle e monitoração de espaçonaves, incluindo dentre estas, por exemplo, todo o controle do telescópio *Hubble*. Este projeto contou com a participação do NASA/SEL - *Software Engineering Laboratory (http://sel.gsfc.nasa.gov/)*, uma iniciativa conjunta entre a Universidade de Maryland e a NASA. Por questões contratuais e confidencialidade algumas informações não podem ser diretamente descritas. O texto apresenta observações pessoais realizadas por um dos autores e também participante do projeto. Dessa forma, recomendamos ao leitor buscar informações adicionais nos diferentes endereços e artigos referenciados nesta seção e disponíveis na literatura técnica.

### 1.4.2. Motivação

O FSB foi formado como resultado de uma reestruturação do GSFC. Os profissionais e especialistas oriundos de quatro diferentes organizações apesar de serem experientes, traziam consigo as especificidades das organizações de origem. Dessa forma, as práticas correntes e observadas para desenvolvimento de software no FSB apresentavam problemas tais como: o desenvolvimento de produtos com qualidade, mas alto custo; a redução do tempo de desenvolvimento, mas sem qualquer melhoria do processo de desenvolvimento de software utilizado e conseqüente aumento dos riscos; o conhecimento não estava institucionalizado, fazendo com que informações importantes sobre o problema ficassem dispersas nas mentes dos especialistas; o crescimento da complexidade do produto não estava sendo controlado e a tecnologia vinha sendo aplicada de forma não uniforme.

O domínio de aplicação básico do FSB relaciona-se com o desenvolvimento de sistemas (software e hardware em alguns casos) para o direcionamento, navegação e controle de satélites. Basicamente, o software é produzido para ser utilizado como software embarcado, executando em tempo real, mas com velocidade de processamento (20 Hz) razoavelmente baixa visando garantir a confiabilidade dos sistemas. Cada lançamento de um satélite

(espaçonave) é denominado uma missão e representa uma aplicação. Isto implica que para cada satélite a ser lançado, independente de já ter sido construído anteriormente algum sistema similar, um novo software é produzido. Apesar de cada missão apresentar diferenças, seja em relação à órbita poder ser geoestacionária ou não, ou mesmo aquelas relacionadas às dimensões e forma da espaçonave e sensores, algumas características comuns podem ser identificadas, das quais destacam-se:

- a determinação da posição e atitude (comportamento) da espaçonave a partir dos dados fornecidos pelos sensores de bordo;

- a identificação de órbitas em relação a Terra (por exemplo, não se considerava missões para Marte), e

- ao fato de possuir um sistema de dados de vôo instalado como parte do sistema operacional nativo da espaçonave.

Uma descrição mais detalhada do domínio pode ser encontrada em [McC 00]. Este domínio era bem definido e se encontrava em situação razoavelmente estável, representando um domínio maduro com algumas aplicações já finalizadas e utilizadas, possuindo algum código legado que deveria ser aproveitado e, o mais importante talvez, a existência de famílias de espaçonaves com aplicações semelhantes que poderiam ser utilizadas como ponto de apoio para o desenvolvimento. Além disto, experiências anteriores (1996) levaram à construção de uma biblioteca de módulos reutilizáveis, escrita em Ada, para direcionamento, navegação e controle (DNC) demonstrando a possibilidade de explorar reutilização de forma abrangente. Esta biblioteca, que oferecia funcionalidades específicas de direcionamento, navegação e controle para os dispositivos em terra, estava disponível e já tinha sido possível observar ganhos em tempo e esforço de projeto a partir de sua utilização na construção de algumas aplicações.

Entretanto, o uso da biblioteca DNC não teve continuidade pelos usuários que relataram algumas questões que dificultavam seu uso relacionado à engenharia da aplicação: documentação de difícil leitura ou em alguns casos inexistente; dificuldade de configuração (neste caso, uma das hipóteses era relacionada à documentação inadequada), a não familiaridade dos usuários com o paradigma de desenvolvimento utilizado (no caso orientação a objetos), e uma questão política envolvendo diferentes perfis de usuários (matemáticos vs. pessoal de software); bem como aquelas relacionadas a problemas nos produtos finais gerados a partir da utilização da biblioteca, principalmente os relacionados com a interface com o usuário. Em parte, esses problemas foram provocados devido aos esforços para construção da biblioteca DNC terem se concentrado mais na engenharia do domínio, ficando as questões relacionadas à engenharia da aplicação para um momento seguinte. Além disto, observou-se também uma substituição gradual do uso da biblioteca DNC pelo MATLAB, que passou a ser utilizado como linguagem de desenvolvimento e programação do ambiente produzindo modelos e códigos que demonstraram a viabilidade de reutilização de missão para

missão. Este cenário, juntamente com a existência de conhecimento do FSB em relação ao problema e a arquitetura utilizada para sistemas de controle de vôo, a experiência do NASA/SEL com análise de processos de software e as lições aprendidas no desenvolvimento e utilização da biblioteca DNC, facilitaram a decisão de se procurar implantar uma linha de produto para software de satélite.

### 1.4.3. Objetivos do Projeto

Para se chegar aos objetivos do projeto foi realizada uma avaliação das expectativas de cada participante e instituição. Basicamente, dois grupos com perspectivas diferentes foram identificados, um representando o ponto de vista dos patrocinadores do projeto, no caso NASA/GSFC e o outro os cientistas e pesquisadores do NASA/SEL.

Neste sentido, os patrocinadores esperavam que o projeto pudesse:

- utilizar o estado da arte em abordagem para reutilização de artefatos;
- reduzir o custo e o esforço do desenvolvimento, com conseqüente redução do tempo de ciclo de vida, e
- documentar a abordagem utilizada de forma a transformar o conhecimento tácito em explícito e desta forma disseminá-lo pela organização aplicando esta tecnologia a outros domínios.

Por se tratar de uma oportunidade impar de implantação de uma tecnologia recente, permitindo o acompanhamento, observação e aprendizado em processos para linha de produto de software, alguns objetivos científicos foram identificados. Dessa maneira, o grupo de pesquisadores do SEL buscava:

- executar um estudo de caso em engenharia de domínio e em um domínio de aplicação peculiar;
- avaliar empiricamente a relação entre custo, esforço, cronograma, retorno de investimento e usabilidade, e
- executar experimentos e estudos que permitissem avaliar:
  - a relação entre desembolso (*payoff*) e o retorno do investimento,
  - a variação relacionada ao custo, cronograma e incidência de defeitos, e,

  - o desembolso, mas utilizando uma perspectiva negativa: quantas aplicações no domínio poderiam não ser desenvolvidas a partir da engenharia do domínio;

- Avaliar e identificar ferramentas para aprimorar a reutilização e,

- Como evitar falhas no software para direcionamento, navegação e controle de satélites, aprimorando:

  - documentação,

  - composição,

  - treinamento.

Em linhas gerais, identificou-se como objetivos fundamentais:

- Prover a infra-estrutura para melhoria contínua com impacto mínimo sob o esforço de desenvolvimento corrente;

- Reduzir o tempo de desenvolvimento sem sacrificar a qualidade;

- Permitir que equipes de desenvolvimento menores pudessem construir estes sistemas, ou então que equipes de mesmo tamanho pudessem construir sistemas maiores, e

- Fornecer capacidades para prototipação rápida incluindo modelos de alta fidelidade e habilidade de exportar/importar código de ferramentas de análise, como por exemplo, o MATLAB [http://www.mathworks.com/products/matlab/].

### 1.4.4. O Plano Inicial

Para a implantação desta abordagem foi identificada a necessidade de planejar atividades que contemplassem diferentes aspectos do desenvolvimento. Por ser uma experiência com um processo de software que apresentava um diferencial sobre os anteriores, em alguns momentos não foi possível realizar um planejamento detalhado destas atividades por falta de conhecimento explícito e experiência anterior, fazendo com que o caráter exploratório e experimental direcionasse a execução da maioria delas. Mesmo assim, um plano básico foi elaborado visando apoiar as etapas que deveriam ser, a princípio, tratadas, conforme descritas a seguir.

**Avaliação e adaptação dos métodos existentes**

Aqui, partiu-se para a avaliação das abordagens de linha de produto disponíveis na literatura técnica. Desde o início, sabíamos que não seria fácil implantar tal metodologia de trabalho e por isto uma avaliação dos métodos existentes visando principalmente identificar o esforço necessário para adaptar esta metodologia de engenharia de domínio ao contexto do projeto foi realizada. A seleção foi, de certa forma, influenciada pelo conhecimento anterior adquirido pelo grupo em cursos e consultorias realizadas nas abordagens Synthesis e FAST.

**Acompanhamento e medição informal do domínio**

Definiu-se que, neste contexto, o acompanhamento e medição do domínio seria realizado a partir da construção e avaliação de protótipos. Do ponto de vista da engenharia de domínio, conceitos seriam explorados e definidos a partir da reutilização de sistemas legados ou através da reconstrução. Para a engenharia de aplicações utilizando estes conceitos de domínio definiu-se por reconstruir uma aplicação previamente existente, produzindo três versões diferenciadas pelo acréscimo de funcionalidade. A partir deste resultado, uma nova aplicação deveria ser construída explorando a abordagem.

**Desenvolvimento e aplicação da abordagem de linha de produto no desenvolvimento de software**

A partir da avaliação do modelo proposto, implantar o enfoque de linha de produto, incluindo seu processo, para desenvolver os componentes de software e produtos para o software de direcionamento, navegação e controle.

O enfoque principal de acordo com a motivação do projeto foi explorar inicialmente engenharia de domínio, pois se fazia necessário a formalização do conhecimento dos conceitos de domínio que se encontravam dispersos pelos diferentes especialistas da equipe. Conforme discutido anteriormente, as abordagens Synthesis e FAST influenciaram a estrutura de trabalho, fornecendo os enfoques para análise do domínio, avaliação e identificação das características comuns e variabilidades da arquitetura e para a construção do glossário As atividades de engenharia de domínio objetivam o desenvolvimento de componentes de domínio (em grande parte módulos de código, mas também requisitos, modelos de análise e projeto) que irão apoiar o desenvolvimento das aplicações no domínio de software do FSB.

O processo de desenvolvimento foi influenciado pelo modelo proposto em [RUM 92], tendo em vista iniciativas de outros projetos de linha de produto e análise de domínio [GOS 95] de outros projetos e visando aproveitar o conhecimento da equipe nesta abordagem. Desta maneira, as abordagens para análise, projeto e implementação e também a parte relacionada ao projeto do sistema refletem em grande parte as características originais propostas pelo método OMT. Para a representação gráfica dos modelos foi adotado UML [RUM 98] já que alguns autores [COH 98] [KEE 99] utilizando orientação a objetos para análise de domínio

sugeriam sua utilização e devido a esta representar uma notação amplamente utilizada pela indústria e de fácil compreensão e assimilação por parte dos usuários. As atividades de engenharia da aplicação objetivam o desenvolvimento de uma aplicação a partir da reutilização de componentes de domínio previamente desenvolvidos. Se observarmos ao longo do tempo, podemos notar que a engenharia de domínio é uma atividade contínua, enquanto o desenvolvimento de aplicações acontece em períodos definidos. As atividades de engenharia de domínio devem começar muito antes que a primeira atividade para engenharia da aplicação. Uma representação da organização das atividades de engenharia de domínio e aplicação para este projeto pode ser vista na Figura 1.4.1.

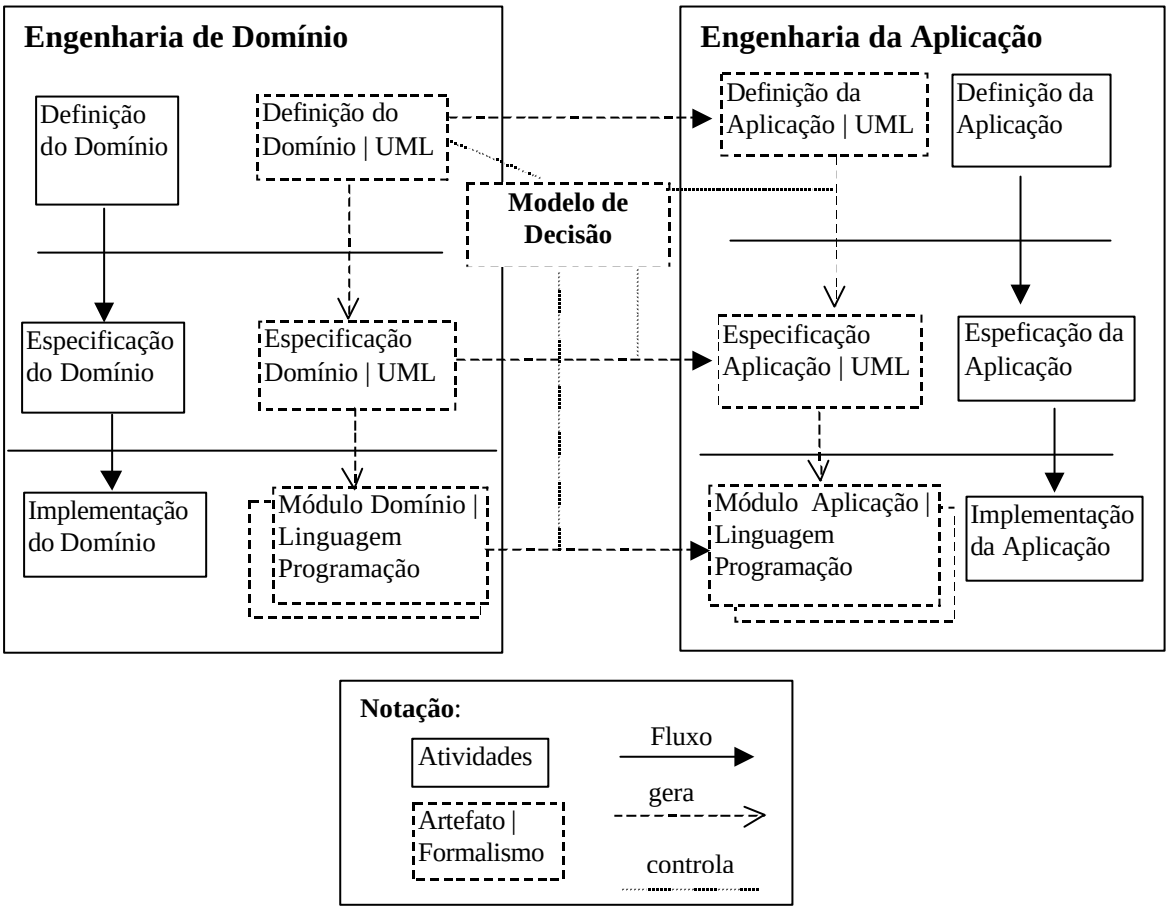


**Figura 1.4.1: A estrutura de desenvolvimento da Linha de Produto**

Ao decidirmos utilizar os modelos e notações específicos para desenvolvimento convencional de aplicações em uma estrutura que contempla a definição abrangente de conceitos e modelos de domínio nos deparamos com alguns problemas de representação, principalmente nos pontos de aderência dos instrumentos convencionais de desenvolvimento (i.e. UML) com os modelos necessários para o desenvolvimento de uma linha de produto. A aplicação destes instrumentos não pode ser realizada diretamente e algumas extensões necessitaram ser definidas. Para representar as variabilidades foi proposta uma extensão ao diagrama de classes UML. Alguns destes questionamentos já haviam sido discutidos por [KEE 99] quando propuseram a utilização de padrões de projeto para representar os discriminantes do modelo, entretanto, para o caso específico do domínio de software do FSB, observamos que a

utilização de padrões não seria adequada para capturar as características dos conceitos e por isto regras e mecanismos adicionais precisaram ser formalizados de forma a permitir expressar corretamente as cardinalidades e capturar as diferenças diretamente nos diagramas, principalmente aquelas que afetam as hierarquias de classe, sem impor características de projeto a modelos de mais alto nível de abstração. Neste caso, explorou-se a utilização de estereótipos para marcar as classes que apresentavam diferenças em relação à arquitetura geral da linha de produto. Uma ferramenta CASE (Figura 1.4.2) foi especialmente adaptada na Universidade de Maryland para explorar estas novas representações relacionadas aos modelos de domínio [MOR 00].

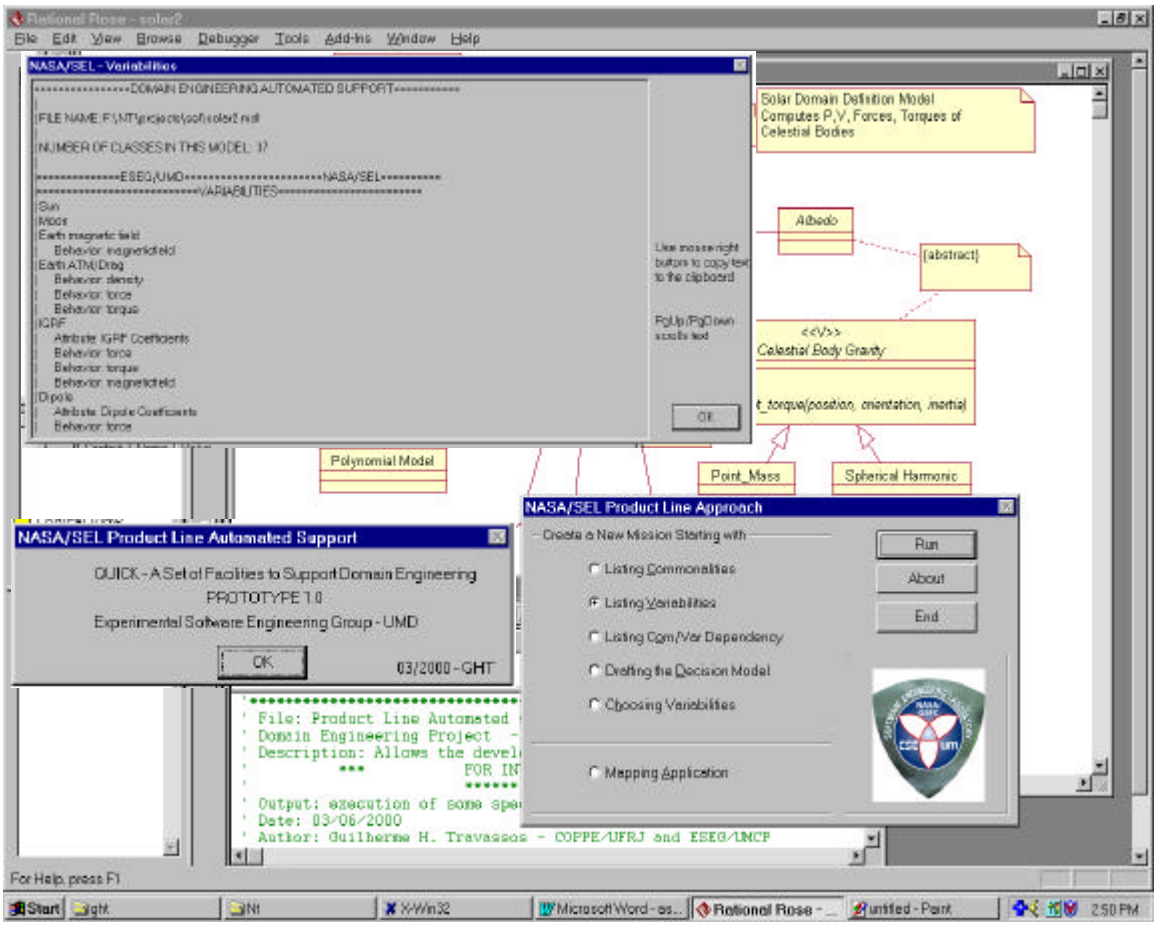


Figura 1.4.2 – Adaptação de ferramenta CASE para tratar os modelos de domínio.

Estas modificações nos permitiram verificar a existência de mecanismos que possibilitam a automação da construção do modelo de decisão, elemento fundamental de uma linha de produto, e a capacidade de transformação semi-automática dos modelos de análise para projeto e implementação, explorando este modelo de decisão. De forma geral e devido a estas características de mapeamento, o caminho inverso também se torna possível, melhorando a questão da rastreabilidade, embora ainda não resolvendo completamente o problema. A Figura 1.4.3 apresenta um exemplo de um diagrama inicial de parte do modelo

de domínio representando os elementos utilizados para determinação da órbita da espaçonave. Diagramas deste tipo apóiam o modelo de decisão, estando as diferenças arquiteturais marcadas com o estereótipo <<V>>. A descrição semântica desta representação pode ser encontrada em [MOR 00].

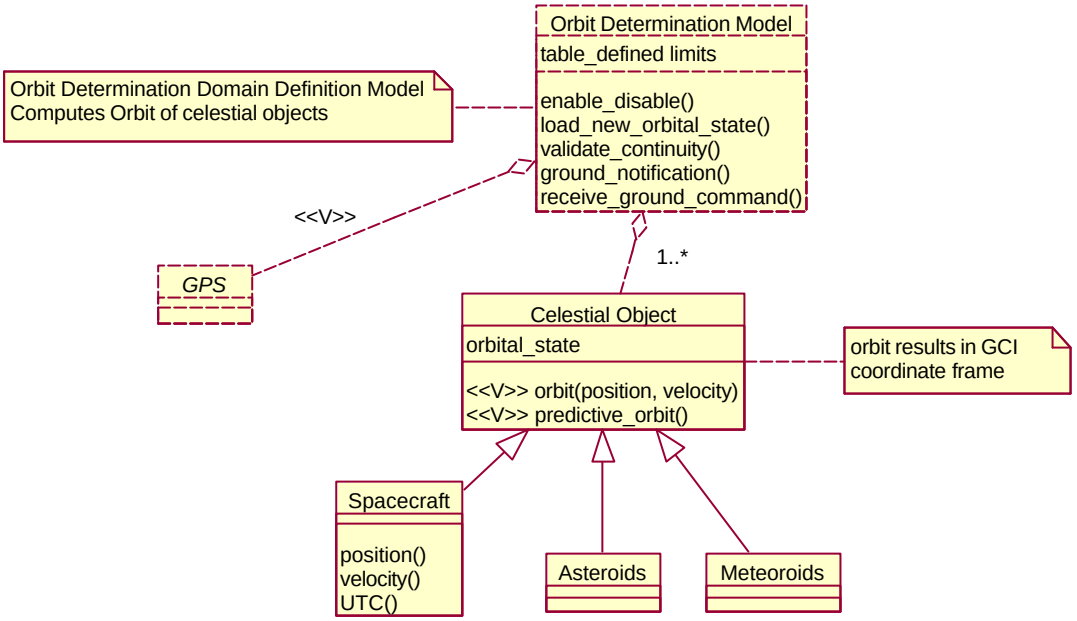


**Figura 1.4.3 – Modelo inicial de parte do domínio representando características comuns e variabilidades**

### 1.4.5. Comentários sobre o Projeto

Após a aplicação do processo descrito anteriormente, foi possível observar algumas características que podem ser utilizadas como pontos para reflexão sobre a adoção de estratégias de linha de produto para o desenvolvimento de software em domínios que possuem características de estabilidade e repetição entre aplicações.

A abordagem de prototipação para avaliação dos trabalhos de desenvolvimento se mostrou adequada. Os protótipos, representados por pequenas aplicações que capturavam as funcionalidades essenciais para direcionamento, navegação e controle de espaçonaves, permitiram avaliar a arquitetura básica do software para monitoração de vôo e também os diferentes conjuntos de atividades de desenvolvimento inerentes a linha de produto. Por não possuírem complexidade exagerada, permitiu aos engenheiros de software acompanhar as atividades de forma mais direta, fornecendo a flexibilidade necessária para se modificar, medir e avaliar estas modificações durante as diferentes versões da aplicação. Três protótipos foram previstos para 2000. Destes, dois foram concluídos com sucesso, sendo o primeiro implementado para tratar modelos orbitais simples e o segundo, acrescentando mecanismos que permitem investigar o projeto das interfaces de hardware com os sensores da espaçonave. Versões subseqüentes começaram a incluir as características necessárias a capturar a arquitetura básica definida para a linha de produto. Por outro lado, os protótipos não ajudaram a realizar uma clara distinção entre o que seriam produtos de domínio e

aplicação. Os produtos são normalmente orientados ao desenvolvedor da aplicação, enquanto os usuários incluem analistas de domínio, testadores e desenvolvedores de ferramentas. Foi possível, por exemplo, identificar o que estava implícito no código fonte, mas, para ser útil, o modelo de decisão e a engenharia de aplicação precisam estar explicitamente descritos e permitir rastreamento.

Do ponto de vista dos métodos utilizados, as técnicas e instrumentos foram aplicados durante o desenvolvimento dos protótipos. Para complementar a avaliação a experiência da organização foi explorada como verificador real. Verificamos que as modificações nos métodos originais deveriam ser mantidas no menor nível possível, mas assegurando que os produtos reutilizáveis fossem reutilizados pelos usuários corretos. O foco principal se deu na análise do domínio em face da necessidade de formalização dos conceitos utilizados para o desenvolvimento das aplicações. Embora Synthesis e FAST tenham contribuído com a abordagem para análise de características comuns e variabillidades foi preciso adicionar notação para representar diferenças em modelos UML, construir matrizes para relacionar missões com capacidades funcionais e missões com clientes. A procura por características básicas de outros métodos, como, por exemplo, PulSE, também foi realizada. Neste sentido, entendemos que processos e produtos são variáveis que podem ser manipuladas de maneira a fazer a engenharia da aplicação o mais fácil possível para todos os usuários e tornar mais fácil a automação das técnicas de construção da aplicação.

Para tratar o domínio, não diferente da abordagem convencional de engenharia de domínio, verificamos a importância de limitar explicitamente o domínio a ser considerado, identificando os subdomínios básicos que deveriam ser utilizados, reduzindo assim a complexidade do problema. Isto facilitou o acompanhamento e avaliação do processo já que é possível lidar com um conjunto menor de conceitos de domínio. A representação destes subdomínios por diagramas de contexto identificando interfaces básicas, diagramas de dependências informais entre subdomínios e descrição de hipóteses a respeito das características comuns e das variabilidades [STA 00] facilitou a compreensão e ajudou principalmente aos engenheiros de software que não possuíam experiência anterior no domínio a compreender o problema e a avaliar o processo. Este processo de avaliação deve ser repetido toda vez que uma nova funcionalidade é acrescentada, incluindo avaliação de gastos. Entretanto a divisão do domínio em subdomínios não é trivial e implica na definição anterior de critérios que permitam executar esta atividade.

Algumas avaliações adicionais ainda se fazem necessárias. Completar a definição do processo e as definições é uma tarefa essencial, já que os usuários de uma linha de produto possuem perspectivas diferenciadas e não são compostos apenas por desenvolvedores de software. Os protótipos, da forma como estão, precisam ser aprimorados e inseridos no contexto de uma aplicação real. Alguns testes em simuladores foram realizados, mas é preciso reutilizá-los numa abordagem de linha de produto de forma a podermos avaliar se os objetivos definidos pelos patrocinadores foram alcançados. Do ponto de vista científico, apesar de vários resultados importantes, muito trabalho (interessante e importante) se inicia a partir deste

ponto. Os leitores interessados em detalhes adicionais são convidados a encontrar estas informações nas diferentes publicações referentes ao projeto.

## 1.5. Conclusões

Linha de produto de software é vista como uma área muito promissora pois oferece uma maneira sistemática, planejada e prática de reutilização de software. A literatura da área mostra vários exemplos de aplicação bem sucedida do enfoque de linha de produtos na indústria [CLE01, BOS 00, CZA 99]. Muito desse sucesso deve-se ao fato da indústria enxergar no enfoque de linha de produtos uma maneira de resolver problemas como *time-to-market*, redução de custo e tempo de produção, aumentando assim a sua produtividade.

A efetiva aplicação do enfoque de linha de produtos está atualmente favorecida por um lado pelo amadurecimento das técnicas de engenharia de software e por outra pela disponibilidade de algumas tecnologias de apoio. Com relação às técnicas de engenharia de software pode-se afirmar que a consciência da importância da arquitetura de software tem aumentado consideravelmente nos últimos anos. A arquitetura de software é o coração da linha de produtos de software. Técnicas que tornem a sua especificação precisa assim como permita a flexibilização necessária para produção de diferentes produtos de uma mesma família são fundamentais para o enfoque de linha de produtos. O amadurecimento das técnicas de desenvolvimento de software baseado em componentes também contribui muito com o enfoque de linha de produtos. Os componentes são os elementos que povoam a arquitetura da linha de produtos, assim o oferecimento de mecanismos para especificar e construir componentes de forma a permitir variações entre estes e que possam facilitar a geração de aplicações com características diferentes com menor esforço de desenvolvimento é também um ponto crucial para o enfoque de linha de produtos. Esses métodos consideram, em seu contexto, conceitos importantes como padrões e frameworks. Também é relevante considerar o amadurecimento da comunidade de engenharia de software no que diz respeito à importância de se ter um processo de software bem definido. Do ponto de vista de tecnologias de apoio, as técnicas de objetos distribuídos e a disponibilização de implementações de *middleware* como CORBA, DCOM e EJB facilitam a implementação dos conceitos de linha de produto.

Outra questão crucial para o sucesso do enfoque de linha de produto de software está nos aspectos gerenciais e culturais da organização. Este enfoque exige uma mudança de sistemática de trabalho para a organização, portanto é fundamental o comprometimento dos níveis gerenciais com a implantação da linha de produto. Essa questão é favorecida pela evidência que a indústria tem mostrado, em medidas palpáveis, o sucesso da implantação de linhas de produto.

## 1.6. Referências


[ARD 97] ARDIS, M., Weiss D., "Defining Families: The Commonality Analysis", Proceedings of the Nineteenth International Conference on Software Engineering, , May 1997, pp. 649-650.

[ATK 00] ATKINSON, C., Bayer, J., Muthig, D., "Component-Based Product Line Development: The KobrA Approach", 1st International Software Product Line Conference, Pittsburgh, August 2000.

[BAS 97] BASS, L., Clements, P., Northrop, L., Withley, J., Product Line Practice Workshop Report. Tecnhical Report (CMU/SEI-97-TR-003). Software Engineering Institute, Carnigie Mellon University, 1997. 36p.

[BAS 98] BASS, Len, Clements, P., Kazman, R., Software Architecture in Practice. Addison Wesley Longman, 1998. 452 p.

[BAY 99] BAYER, J., Flege, O., Knauber, P., Laqua, R., Schmid, K., Widen, T., DeBaud, J., PuLSE: A methodology to develop software product lines", Symposium on Software Reusability (SSR99), May 1999.

[BAT 98] Batory, D., "Product-Line Architectures", Smalltalk and Java Conference, Erfurt, Germany, October 1998.

[BOS 00] BOSCH, J., Design & Use of Software Architectures: adopting and evolving a product-line approach. Great Britain: Ed. Addison Wesley, 2000. 354 p.

[BOS 98] BOSCH, J., Product-Line Architectures in Industry: A Case Study. Disponível em <http://www.ipd.hk-r.se/bosch/>. Acesso em: 02 oct. 1999.

[CAS 01] CASTRO, J., STRAW 2001 – From Requirements to Architectures, ICSE 2001, Toronto, Canadá, 2001.

[COH 98] COHEN, S., Northrop, L. M., Object-Oriented Technology and Domain Analysis, Fifth International Conference of Software Reuse, June 1998.

[CZA 99] CZARNECKI, K., Eisenecker, U., "Components and Generative Programming", Proceedings of the 7th European Software Engineering Conference, held jointly with the 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE '99), Toulouse, France, September 1999. LNCS 1687, Berlin/Heidelberg/New York, 1999, pp. 2-19.

[CHE 01] CHEESMAN, J., Daniels J., UML Components, A Simple Process for Specifying Component-Based Software, Addison-Wesley, 2001.

[CLE 01] CLEMENTS, P., Northrop, L. Software Product Lines: Practices and Patterns, SEI Series in Software Engineering, Addison-Wesley. 563 p, 2001.

[D'SO 99] D'SOUZA, D., WILLS, A., Objects, Components and Frameworks with UML – The Catalysis Approach, Addison-Wesley Publishing Company, 1999.

[GAM 95] GAMMA, E., Helm, R., Johnson, R, Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[GOS 95] GOSSAIN, S.; Batory, D.; Gomaa, H.; Lubars, M.; Pidgeon, C. and Seidewitz, E. Objects and domain engineering (panel), Proceedings of the tenth annual conference on object-oriented programming systems, languages, and applications (OOPSLA95), October 1995.

[GRI 98] GRISS, M., Favaro, J., Alessandro, M. "*Integrating Feature Modeling with RSEB*", 5th International Conference on Software Reuse (ICSR-5), ACM/IEEE, Victoria, Canadá, Junho 1998.

[GRI 00] GRISS, M., "Implementing Product-Line Features with Component Reuse", 5th International Conference on Software Reuse, ACM/IEEE, Vienna, Austria, Junho 2000.

[GUR 01] GURP, J., Bosch, J., Svahnberg, M., On the Notion of Variability in Software Product Lines, Proceeedings of WICSA 2001, 28-31 August 2001, Amsterdam, THE NETHERLANDS.

[JAC 97] JACOBSON, I., Griss, M., Jonsson, P., "Software Reuse – Architecture Process and Organization for Business Success", New York: Ed. Addison-Wesley, 1997.

[JAZ 00] JAZAYERI, M., Ran, A., van der Linder, F. Software Architecture for Product Families – Principles and Practice. Addison-Wesley, 2000. 257 pg.

[KAN 90] KANG, K. et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study, (CMU/SEI-90TR-21, ADA 235785), Pittsburgh, PA:SEI CMU, 1990.

[KAN 98] KANG, K., Kim, S., Lee, J., Shin, E., Huh, M., FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5, 143-168(1998).

[KEE 99] KEEPENCE, B., Mannion, M. Using Patterns to Model Variability in Product Families, *IEEE Software*, July 1999, pp. 102-108.

[KRU 00] KRUCHTEN, P., *Rational Unified Process: An Introduction*, Addison-Wesley, 2000.

[LEW 95] LEWIS, T., et al., Object-Oriented Application Frameworks, Manning publications Co, 1995.

[McC 00] McCOMAS, D.; Leake, S.; Stark, M.; Morisio, M.; Travassos, G. H.; White, M. Addressing Variability in a Guidance, Navigation, and Control Flight Software Product Line. In: The Proceedings of the First Software Product Line Conference, SPLC1, 2000, Denver. 2000.

[MOR 00] MORISIO, M.; Travassos, G. H.; Stark, M. Extending UML to Support Domain Analysis. In: Proceedings of the IEEE International Conference on Automated Software Engineering - ASE'00, 2000, Grenoble. Los Alamos: IEEE Computer Press, 2000.

[OMG 02] Object Management Group, Model Driven Architecture, http://www.omg.org/mda/, último acesso Maio, 2002.

[POU 97] POULIN, J. "*Software Architectures, Product Lines, and DSSAs: Choosing the Appropriate Level of Abstraction*", WISR8, 1997.

[RUM 92] RUMBAUGH, J.; Blaha, M.; Premerlani, W.; Eddy, F. and Lorenson, W.(1992). Object Oriented Modeling and Design. Prentice-Hall. ISBN 0136298419

[RUM 98] RUMBAUGH, J., Jacobson, I., Booch, G. The Unified Modeling Language Reference Manual, Addison-Wesley, 1998.

[SOF 93] SOFTWARE PRODUCTIVITY CONSORTIUM, Reuse-Driven Software Processes Guidebook, SPC-92019-CMC version 02.00.03, November 1993.

[SPC 93] Software Productivity Consortium, Reuse-Driven Software Processes Guidebook, SPC-92019-CMC version 02.00.03 November 1993.

[STA 00] STARK, M., McComas D., Travassos, G., Morisio, M. Developing a Product Line Approach for Flight Software. In: 25TH ANNUAL SOFTWARE ENGINEERING WORKSHOP, 2000, Greenbelt. Proceedings of the 25th Software Engineering Workshop. Greenbelt: NASA/Goddard Space Flight Center, 2000. p.1-13.

[TRA 02] TRAVASSOS, G. H., Gurov, D., TABA Workstation: Towards a Product Line for the Developing of Software Engineering Environments. Publicações Técnicas do Projeto TABA, RT-21/02. Programa de Engenharia de Sistemas e Computação. COPPE/UFRJ.2002.

[WEI 99] WEISS, D., et al. Software Product-Line Engineering: a family-based software development process. [S.l]: Ed. Addison Wesley, 1999. 426 p.

[WFM 95] Workflow Management Coalition. “Workflow Reference Model”. Document number TC00-1003, January 19, 1995. 55 p.