UNIVERSIDADE FEDERAL DA BAHIA
FACULDADE POLITÉCNICA / INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MECATRÔNICA

ANA PATRÍCIA FONTES MAGALHÃES MASCARENHAS

METODOLOGIA PARA DESENVOLVIMENTO DE PRODUTOS MECATRÔNICOS INTEGRANDO ENGENHARIA DE SOFTWARE E ENGENHARIA DE PRODUTOS

Salvador
2007

ANA PATRÍCIA FONTES MAGALHÃES MASCARENHAS

METODOLOGIA PARA DESENVOLVIMENTO DE PRODUTOS MECATRÔNICOS INTEGRANDO ENGENHARIA DE SOFTWARE E ENGENHARIA DE PRODUTOS

Dissertação apresentada ao Programa de Pós-graduação em Mecatrônica, Escola Politécnica / Instituto de Matemática, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Mecatrônica.

Orientadora: Profa. Dra. Aline Maria Andrade
Co-orientadora: Profa. Dra. Leila Maciel de A. e Silva
Co-orientador: Prof. Dr. Herman Lepikson

Salvador
2007

A
meus pais por me iniciarem no caminho do conhecimento.
Érico pelo incentivo e compreensão demonstrado desde os primeiros dias.
Felipe por ter me mostrado tantas coisas novas neste caminho.

AGRADECIMENTOS

Muitos são os agradecimentos, pois todos sempre me apoiaram nesta jornada.

A Aline, orientadora, pelo empenho, disposição e compreensão nesta jornada.

A Leila, co-orientadora pelo esforço e o interesse depositado neste trabalho.

A Herman, co-orientador pela ajuda nas áreas que eu tinha menos domínio.

A Rita Suzana, que me iniciou no caminho da pesquisa e me incentivou a seguir novos rumos.

A Antônio Carlos pela importante ajuda inicial e as idéias para o projeto.

A Lúcia, Socorro e Rebeca, secretárias sempre prestativas.

A Frederico, sempre disposto a ajudar, com a sua experiência em desenvolvimento.

Aos meus colegas pelas novas experiências que vivemos.

RESUMO

A Mecatrônica aborda uma categoria de produtos que pressupõe um conhecimento multidisciplinar nas áreas de Mecânica, Eletrônica e Computação. Este cenário de integração de conhecimentos requer uma atenção especial, pois considera diferentes técnicas de desenvolvimento e envolve pessoas de áreas distintas.

Esta dissertação especifica uma metodologia para o desenvolvimento de produtos mecatrônicos, a MdpM, que utiliza como base a linguagem Unified Modeling Language (UML) e o processo consolidado de desenvolvimento de software, o *Rational Unified Process* (RUP) integrado a técnicas de Engenharia Elétrica e Engenharia de Produtos apropriadas ao domínio das aplicações mecatrônicas.

A MdpM utiliza o potencial da equipe multidisciplinar para modelar uma solução adequada para produtos meatrônios, adiando ao máximo as definições tecnológicas a serem empregadas. Uma vez que o produto esteja bem conceituado, orienta quanto à identificação de soluções apropriadas para o modelo conceitual gerado.

A MdpM aborda também aspectos importantes para a Mecatrônica tais como a especificação de requisitos temporais e a aplicação de técnicas relacionados à confiabilidade do produto.

Palavras-chaves: Metodologia de Desenvolvimento de Produtos Mecatrônicos, RUP, UML, Engenharia de Software, Engenharia de Produtos.

ABSTRACT

Mechatronic approach presumes a multidisciplinary knowledge in Mechanic, Electrical and Computational areas. This integrated scenario requires special attention due to the use of different techniques and to the involvement of people from distinct areas.
This dissertation specifies a methodology for mechatronic products development, MdpM, based on Unified Modeling Language (UML) and the consolidated software development process Rational Unified Process (RUP), integrating Electrical Engineering and Product Engineering techniques appropriated to mechatronic applications domains.
MdpM uses multidisciplinary knowledge potential to model an adequate product solution, postponing technologies definitions to late stages. Only after perform a well conceptual design, the methodology conducts to the definition of an appropriated technical solution.
MdpM also considers important aspects on Mechatronic development as the specification of temporal requirements and the application of techniques related to product confiability.

Key words: Methodology for Mechatronic Products Development, RUP, UML, Software Engineering, Product Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMMI	Capability Maturity Model Integration
DCA	Aplicações de Controle Distribuído
DFD	Diagrama de Fluxo de Dados
FB	Function Blocks
FBDK	Function Block Development Kit
IEC	International Eletrotechnical Commission
OMG	Object Management Group
OO	Orientação a Objetos
PC	Pontos de Controle
RUP	Rational Unified Process
SPEM	Software Process Engineering Metamodel
UML	Unified Modeling Language
VEDA	Verification Enviroment for Distributed Applications

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento de produtos vem sofrendo muitas mudanças nos últimos anos, ditadas pelos avanços tecnológicos da Eletrônica e da Computação. Inicialmente, a fabricação de um produto tinha como foco principal a execução de uma atividade mecânica ou física. Mudanças em suas necessidades funcionais ocasionavam mudanças em seus componentes mecânicos o que, conseqüentemente, determinavam tempo e custos elevados de manutenção. Muitas vezes o *layout* de um produto era definido não pelos seus requisitos funcionais, mas pelas imposições mecânicas necessárias para a sua viabilidade [36].

A evolução da tecnologia tem influenciado fortemente as mais diversas atividades desempenhadas pelo homem, sejam elas no âmbito industrial, comercial ou no cotidiano das pessoas. Esta evolução possibilita a construção de produtos que integram hardware e software e permitem a criação tanto de aplicações críticas como, por exemplo, controle de tráfego aéreo e controle de uma planta industrial, quanto de aplicações comuns ao dia a dia dos indivíduos como, por exemplo, os aparelhos celulares, televisores, fornos de microondas, entre outros equipamentos.

Neste contexto inserem-se os produtos mecatrônicos. O termo Mecatrônica possui um domínio altamente multidisciplinar, onde as engenharias Mecânica, Elétrica e Computação colaboram entre si para o desenvolvimento de um produto comum [29]. A construção de produtos mecatrônicos faz parte de um trabalho de equipe que integra as diversas áreas envolvidas. Desta maneira, estes produtos não podem ser vistos como um objeto mecânico que agrega características eletrônicas para melhorar o seu desempenho, mas como um objeto que integra características de diversas áreas de conhecimento para prover soluções inovadoras.

Um sistema mecatrônico, como apresentado na Figura 1, possui uma estrutura básica composta por quatro elementos que interagem entre si: o sistema mecânico, sensores, atuadores e o controlador. Os sensores captam informações do sistema mecânico. Estas

informações são processadas pelo controlador e originam ações de controle que retornam ao ambiente, atuando e modificando o seu estado original [26]. Este alto grau de interação observado pode agregar novas características que devem ser abordadas durante o desenvolvimento de sistemas desta natureza.

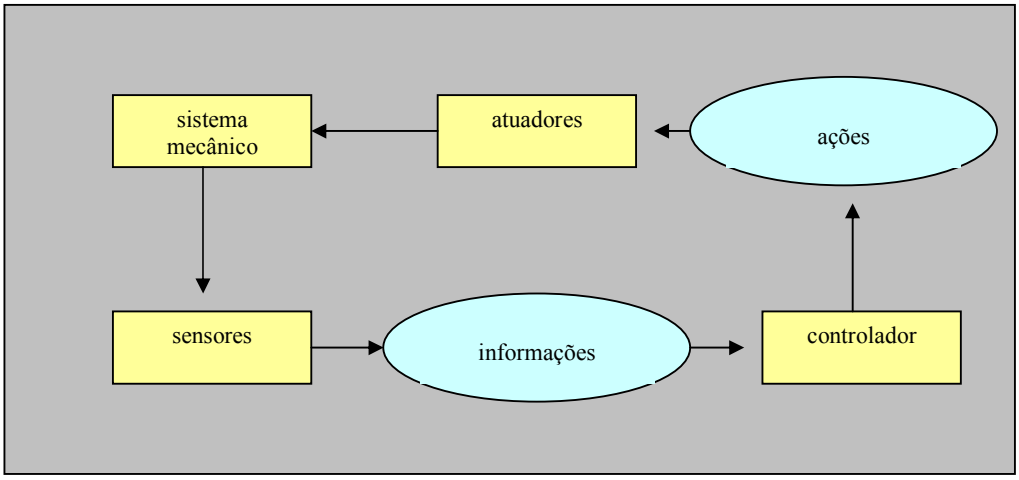


Figura 1 - Representação de um sistema mecatrônico (adaptado de [26])

Um produto mecatrônico muito freqüentemente é formado por um conjunto de partes, chamadas de componentes. Neste contexto, um ponto importante a ser tratado na modelagem destes produtos está relacionado ao nível de integração dos componentes, sejam de hardware ou de software. A divisão do produto em componentes traz benefícios tais como a facilidade de manutenção e de evolução. No entanto, também aumenta a complexidade, uma vez que se precisa especificar detalhadamente a comunicação entre estes componentes, pois problemas de funcionamento e comunicação podem comprometer o sistema como um todo. Além disso, características de detecção, tratamento e correção de erros devem ser definidas durante a especificação destes produtos.

Além das características técnicas particulares que envolvem este domínio de aplicações, outro ponto importante a ser considerado é o ambiente multidisciplinar da equipe de desenvolvimento. Este ambiente é composto por pessoas de diferentes formações que trabalham em conjunto em função de um objetivo comum. Cada uma destas pessoas enxerga o sistema sob uma perspectiva e tende a definir soluções voltadas à sua área de atuação. No entanto, o foco do desenvolvimento deve estar no aproveitamento deste potencial diversificado de conhecimentos para adoção da melhor solução integrada.

Assim, a necessidade de uma metodologia para o desenvolvimento de produtos mecatrônicos é visível. Por todos estes aspectos uma metodologia para esta categoria de

produtos deve considerar a multidisciplinaridade dos seus usuários no que se refere à adoção de uma linguagem de fácil compreensão e expressiva o suficiente para unificar as técnicas já consolidadas nas diversas áreas.

Nesta direção, algumas metodologias já estão sendo propostas. É consenso entre vários pesquisadores que a utilização das técnicas de Orientação a Objetos (OO) [28] é útil para o projeto de produtos mecatrônicos. Conceitos de reutilização, modularização, encapsulamento, herança, entre outros, presentes na OO, se adequam bem ao ambiente de integração de componentes e permitem uma melhor manutenção, portabilidade e evolução destes produtos. Também a utilização da *Unified Modeling Language* (UML) [5] está sendo adotada por oferecer modelos de fácil compreensão e padronização como pode ser observado em [3, 4, 31, 35, 19, 15].

No entanto, as metodologias propostas [3, 4, 31, 35, 19] estão voltadas para a modelagem do elemento controle do sistema. Dentre os trabalhos encontrados, apenas o [33] propõe uma metodologia que se aplique à modelagem de todo o produto e permita um nível de abstração que forneça independência de tecnologia.

Nesse contexto, este trabalho propõe uma metodologia, a Metodologia Unificada para o Desenvolvimento de Produtos Mecatrônicos (MdpM) que abrange várias fases do processo de desenvolvimento, desde as etapas iniciais de levantamento de necessidades até a construção de um modelo do produto. A MdpM enfatiza a aplicação de técnicas de validação que permeiam todo o processo de desenvolvimento com o objetivo de agregar certa confiabilidade ao produto. Esta confiabilidade é fortalecida com a aplicação da verificação formal a partes críticas do sistema. Além disso, a MdpM aborda o tratamento de falhas e a especificação de requisitos temporais, características importantes no desenvolvimento de produtos mecatrônicos.

A metodologia MdpM baseia-se no processo *Rational Unified Process* (RUP) [14], o qual aplica técnicas de análise orientada a objetos (OO), como também utiliza como linguagem de representação de modelos a UML. O uso da UML facilita a compreensão, pois possui modelos simples de representação do conhecimento, o que facilita também a comunicação dentro da equipe. A metodologia tem seu foco na modelagem do problema, abstraindo inicialmente a tecnologia de construção. Soluções técnicas de construção são definidas em estágios mais avançados dos trabalhos, quando um modelo conceitual do produto for concebido. Para auxiliar nestas decisões, a MdpM indica a utilização de técnicas que permitam selecionar a opção mais indicada para implementação. Assim como no RUP,

também faz parte da metodologia um modelo de gerência dos trabalhos que permite acompanhar o andamento das atividades e definir papéis e responsabilidades dentro da equipe de desenvolvimento. Esta gerência é uma atividade contínua que possibilita detectar e corrigir problemas que possam vir a significar riscos para o sucesso do projeto.

A metodologia considera também aspectos do ciclo de vida de projeto de produtos mecânicos e incorpora alguns artefatos da Engenharia de Produtos e da Engenharia Elétrica. Pode-se citar, por exemplo, a matriz da casa da qualidade, usada na análise dos requisitos e os *function blocks* (FB) usados para mapear diagramas UML em um modelo mais próximo da implementação.

Este trabalho está organizado como descrito a seguir. O capítulo 2 apresenta o desenvolvimento de produtos seja de software, hardware ou mecânico. Uma revisão de trabalhos relacionados e uma descrição de aspectos importantes para o desenvolvimento de produtos mecatrônicos são apresentados no capítulo 3. O capítulo 4 apresenta a metodologia MdpM, proposta neste trabalho, no qual é descrito o funcionamento, as características técnicas adotadas e a estrutura da metodologia. A descrição do processo de desenvolvimento de produtos mecatrônicos segundo a MdpM é apresentada no capítulo 5. O capítulo 6 descreve um experimento onde foi desenvolvido um produto utilizando a MdpM. No capítulo 7 são apresentadas as considerações finais e direções de trabalhos futuros. O apêndice A fornece uma descrição dos diagramas UML utilizados pela MdpM. O apêndice B apresenta um guia completo do processo de desenvolvimento segundo a MdpM, e o apêndice C detalha o experimento prático e apresenta todos os artefatos gerados durante o projeto.

2. DESENVOLVIMENTO DE PRODUTOS

Durante muitos anos a engenharia construiu seus produtos de maneira artesanal, sem utilizar técnicas que pudessem ajudar a projetar, acompanhar e avaliar o resultado dos trabalhos. No entanto, com o passar dos anos as exigências do mercado foram crescendo, novas necessidades foram surgindo e este processo de desenvolvimento começou a ser questionado, pois com freqüência eram acometidos pelo insucesso [25]. Diante deste panorama, técnicas foram sendo criadas que pudessem ajudar os desenvolvedores a construir produtos mais confiáveis, com maior qualidade e consequentemente a custos e prazos mais aceitáveis.

Atualmente, estas técnicas estão presentes em todos os ramos da Engenharia, seja na Engenharia Mecânica, na Engenharia de Computação ou Engenharia Elétrica, e já possuem certo grau de maturidade que possibilitam uma qualidade dos produtos gerados. No entanto, como já destacado anteriormente, o desenvolvimento de produtos mecatrônicos envolve o trabalho conjunto destas três áreas e, consequentemente, é importante analisar as técnicas já existentes em cada uma delas para identificar o que pode ser reaproveitado e adaptado na especificação de uma metodologia voltada para o domínio de aplicações mecatrônicas.

Desta maneira, as seções seguintes apresentam as técnicas atualmente desenvolvidas em cada uma das engenharias que foram conciliadas na especificação da metodologia descrita neste trabalho. A seção 2.1 descreve o processo de desenvolvimento de produtos segundo a Engenharia de Software. As seções 2.2 e 2.3 abordam o desenvolvimento de produto eletrônico e o desenvolvimento de produtos mecânicos, respectivamente.

2.1. DESENVOLVIMENTO DE PRODUTOS DE SOFTWARE

A Engenharia de Software engloba o estudo de métodos que visam à melhoria do processo de construção de sistemas computacionais. Ao longo do tempo, estes métodos vêm evoluindo para atender as necessidades impostas pelas novas tecnologias.

A análise estruturada moderna [38] pode ser considerada o primeiro método de desenvolvimento significativo na evolução da construção de software. Neste modelo, o desenvolvimento é realizado tendo como foco principal o mapeamento das funções do sistema. Com o passar do tempo, foram sendo observados alguns problemas nesta abordagem de pensamento tais como a redundância de informações e a dificuldade de evolução e manutenção dos sistemas, considerando a dinâmica das organizações. Surgiu então o modelo de entidades e relacionamentos, propondo o desenvolvimento voltado aos dados, deixando a especificação das funções para um segundo plano. [25]

A década de 80 foi marcada pelo surgimento do paradigma da orientação a objetos (OO) [28]. Esta nova abordagem abstrai o mundo real na forma de objetos, com características e comportamentos próprios. Esta maneira de mapear o mundo real torna mais simples a especificação dos sistemas, pois facilita a compreensão e consequentemente diminui a distância existente entre analistas e usuários durante o processo de especificação de um software.

Na década de 90, o desenvolvimento baseado em componentes se expandiu, com a necessidade do reuso na Engenharia de Software. Esta tecnologia permitiu o encapsulamento, a modularização e a reutilização, características presentes na OO e cada vez mais desejadas atualmente no desenvolvimento de sistemas de qualquer natureza [25].

A OO se fortaleceu ainda mais com a especificação de uma linguagem unificada de modelagem, *Unified Modeling Language* (UML), pois a partir desta os conceitos de OO começaram a ser mais amplamente utilizados no desenvolvimento de sistemas [28].

Neste contexto, metodologias de desenvolvimento de software foram propostas, dentre elas o *Rational Unified Process* (RUP). O objetivo do RUP é orientar os desenvolvedores para a construção de software de alta qualidade nos prazos e custos planejados [14]. Para isso, define um processo de construção do software, disciplinado com atividades e responsabilidades bem definidas que se baseia no paradigma OO e utiliza a UML como linguagem de modelagem.

O RUP utiliza o paradigma da OO, a tecnologia de componentes e a linguagem UML, além de ser um processo focado na garantia da qualidade dos produtos gerados. A facilidade de representação do conhecimento e de compreensão dos modelos, presentes na OO e na UML, são de grande importância para o ambiente mecatrônico, onde pessoas de diferentes formações precisam se comunicar e construir um produto único. Além disso,

oferece recursos que permitem mapear com eficiência as características necessárias para este domínio de aplicações. A tecnologia de componentes, onde o software é dividido em partes que podem ser reaproveitadas por outros software é uma característica importante a ser considerada na Mecatrônica, pois a decomposição de um produto em partes pode facilitar a compreensão, a construção, a montagem e a manutenção de produtos.

A especificação de processos de desenvolvimento de software fez surgir à necessidade de um metamodelo que permitisse representá-los. A metodologia proposta neste trabalho, assim como o RUP, está especificada utilizando o metamodelo *Software Process Engineering Metamodel* (SPEM).

As próximas seções estão assim estruturadas. Os conceitos básicos de OO e componentes são descritos na seção 2.1.1 e a seção 2.1.2 apresenta a linguagem UML. A seção 2.1.3 apresenta o processo RUP de desenvolvimento de software. A seção 2.1.4 descreve os principais elementos do metamodelo SPEM.

2.1.1. Paradigma de orientação a objetos (OO)

A tecnologia OO [28] procura construir sistemas de informação mapeando abstrações de objetos do mundo real em modelos computacionais. Neste contexto, utiliza vários tipos de modelos, cada um com a sua aplicação, sejam para identificação de requisitos, para representação estrutural ou comportamental de um sistema.

Na OO um sistema é formado por classes e objetos. As classes são estruturas que representam os modelos do mundo real e são utilizadas para criar objetos. A especificação de uma classe encapsula atributos (dados) e métodos (comportamento) além de definir claramente as interfaces de comunicação e o relacionamento existente entre as diversas classes do modelo. Assim, na visão da OO, um sistema é formado por um conjunto de objetos que são criados de acordo com as definições especificadas nas suas respectivas classes. Estes objetos possuem uma interface bem definida que especifica um conjunto de serviços para serem utilizados na comunicação com os outros objetos. Os objetos se comunicam por meio de mensagens enviadas aos métodos [23].

Na OO, uma classe é visualizada como uma cápsula com todas as definições necessárias para a criação dos objetos. Esta característica facilita a sua reutilização, pois o desenvolvedor não precisa conhecer todos os seus detalhes de especificação, apenas os serviços que disponibiliza na sua interface de comunicação.

A OO também implementa o conceito de herança, onde uma classe pode herdar definições de outras classes. Esta característica favorece a reutilização e facilita a evolução, pois na construção de um software novo, o desenvolvedor pode reaproveitar as definições das classes que já existem.

Dentro deste enfoque, pode-se dizer que a OO reúne características tais como reutilização, modularização, encapsulamento e herança que se adequam bem ao desenvolvimento de produtos modulares, ou seja, que podem ser vistos como um conjunto de partes que se integram formando um sistema. Uma vez que produtos mecatrônicos são construídos procurando integrar conhecimentos de áreas distintas, esta abordagem se torna bastante adequada. Dividir o produto em partes menores bem modularizadas e encapsuladas permite que estas sejam construídas mais facilmente, respeitando as interfaces definidas na especificação.

É comum confundir componentes com objetos. Componentes são unidades que provêem serviços específicos e disponibilizam uma interface para que estes serviços sejam requisitados. Na abordagem OO geralmente os componentes são formados por uma ou mais classes que agrupadas fornecem uma tarefa específica ao sistema. Isso acontece porque os objetos são consideradas unidades menores, que possuem uma interface, mas que não necessariamente respondem completamente por uma tarefa. Além disso, possui um estado (valores ao longo da sua execução) e uma identidade única dentro do sistema, ao contrário dos componentes que são identificados por cópias iguais.

Um componente geralmente é uma unidade independente que pode ser utilizada na montagem de um produto. Para isso, deve ser auto-suficiente, ter seus serviços encapsulados e suas interfaces bem definidas. Não pode ser utilizado em partes, quem o utilizar terá disponível todos os seus serviços. Desta maneira, quando um produto é construído, procura-se reutilizar os componentes que já existem prontos no mercado (que também já estão testados e funcionam), integrados aos componentes específicos que podem ser desenvolvidos para atender a novas demandas.

Estes conceitos são interessantes para o desenvolvimento de produtos mecatrônicos no que se refere à sua composição. Pode-se conceber o produto como um conjunto de componentes com objetivos e interface bem definidas. Estes componentes podem ser mecânicos, eletrônicos, de software ou híbridas (duas ou mais tecnologias) e poderão ser desenvolvidos, adquiridos ou reutilizados de outro produto.

2.1.2. A Linguagem UML

A *Unified Modeling Language* (UML) [5] é uma linguagem padrão para modelagem de sistemas que pode ser utilizada com a finalidade de visualização, especificação, construção e documentação. A UML se baseia na tecnologia de objetos e de componentes e tem se mostrado uma linguagem muito expressiva, capaz de representar diversas visões de um sistema. Por isso vem sendo utilizada na modelagem de problemas das mais diversas áreas de domínio, incluindo a Mecatrônica.

A UML é uma linguagem de modelagem. Como toda linguagem, possui um vocabulário e estabelece regras necessárias para a sua boa utilização. Estas regras e vocabulário concentram-se na conceituação e mapeamento do problema em modelos visuais de representação das diversas partes do sistema. A utilização da representação visual diminui a distância entre desenvolvedores e clientes, facilitando o entendimento. Esta é uma característica muito importante no desenvolvimento de um produto, pois identificar erros de concepção ainda durante as fases iniciais de levantamento diminui o tempo e o custo de desenvolvimento [19].

A UML possibilita a criação de uma série de modelos, cada um com finalidade própria. Para elaborar estes modelos três pontos devem ser observados [5]: a escolha do modelo adequado ao problema; o refinamento deste modelo; e o uso de vários modelos para representar aspectos diversos do sistema. Assim, para cada tipo de problema a ser especificado utilizam-se modelos que enfatizem as características necessárias. Estes modelos são refinados ao longo do processo de especificação até atingirem o nível de detalhamento necessário para o seu entendimento. A utilização de vários modelos é importante para se atingir um nível de maturidade suficiente no entendimento do problema, pois sistemas complexos necessitam de visões e abordagens diferentes e por isso devem ter modelos que mapeiem cada uma destas abstrações.

Os modelos da UML, estruturais, comportamentais ou de agrupamento, podem ser compostos por cinco elementos básicos: os blocos, que representam um elemento do modelo (uma classe, por exemplo); as notações, que oferecem explicações sobre os modelos; os relacionamentos, que estabelecem ligações de dependências, associações generalização e realização entre os elementos dos diversos modelos; as regras necessárias para gerar documentos bem formados tais como nomes e escopo; e os mecanismos para representação e extensão da linguagem.

Os modelos estruturais representam a parte estática do sistema, podendo ser conceitual ou físico. São exemplos, os modelos de classes e de componentes. Os modelos comportamentais modelam o comportamento dinâmico do sistema no tempo e espaço, como exemplo tem-se os diagramas de caso de uso, modelos de interação e a máquina de estados. Os modelos de agrupamento representam blocos de organização das partes do sistema como, por exemplo, o modelo de pacotes.

Basicamente os modelos UML são formados pela combinação dos blocos e relacionamentos descritos acima a partir dos mecanismos de representação e extensão e respeitando as regras de formação. Para cada item especificado nos modelos (ex. classe), é possível acrescentar uma especificação textual, seguindo regras específicas de sintaxe, que representem informações adicionais (ex. atributos, métodos, etc.) ao modelo e que permitem a definição de semânticas específicas.

A UML foi inicialmente projetada para o desenvolvimento de sistemas de informação. No entanto, com o passar do tempo, vem sendo utilizada também para outras áreas de domínio. Para suprir estes novos domínios, permite que extensões sejam criadas de maneira a definir a semântica necessária para cada domínio de aplicação. As extensões UML podem ser definidas a partir de estereótipos (*stereotypes*), valores rotulados (*tagged values*) e limitações (*constraints*). Por exemplo, pode-se criar uma extensão no modelo de componentes em um sistema mecatrônico com o uso de estereótipos que indiquem se o componente será de <<hardware>>, <<software>>, <<mecânico>> ou <<mecatrônico>>. Isso pode facilitar a compreensão do modelo de componentes do sistema.

A versão 2.0 da UML, utilizada neste trabalho, incorpora alguns modelos e nomenclaturas propostas em extensões anteriores da linguagem que são importantes para as aplicações mecatrônicas. Dentre elas está o tratamento de requisitos temporais. O apêndice A apresenta os principais conceitos e modelos UML utilizados neste trabalho.

2.1.3. Rational Unified Process (RUP)

O RUP [14] é um processo de Engenharia de Software que tem como principal objetivo garantir o desenvolvimento de sistemas com qualidade, respeitando os requisitos solicitados pelo cliente, em prazos e custos determinados.

Como base para a criação deste processo, algumas características foram adotadas visando a qualidade de software: o uso de iterações, onde o desenvolvimento é visto como um

processo de refinamentos sucessivos evoluídos em novas iterações até chegar a uma versão final; a gerência dos requisitos, para evitar que modificações de escopo ao longo do processo possam gerar descontrole dos prazos e custos inicialmente previstos; o desenvolvimento orientado a componentes, para facilitar a manutenção e a reutilização; a utilização de artefatos visuais, com a utilização da UML; o controle da qualidade; e o controle de mudanças.

O processo RUP está dividido em duas dimensões como apresentado na Figura 2. A dimensão de tempo, representada pelo eixo das abscissas da figura, apresenta o ciclo de vida do sistema dividido em quatro fases: concepção, elaboração, construção e transição. Cada uma destas fases está bem definida no tempo, embora permita várias iterações, e a cada nova iteração, a fase é refinada. Após várias iterações, chega-se a um nível de detalhamento suficiente para finalizar uma fase e iniciar a próxima.

O eixo das ordenadas representa as disciplinas (fluxos de trabalho) do processo RUP. Estas disciplinas agrupam atividades afins que são desenvolvidas ao longo do ciclo de vida do sistema. Pode-se observar, por exemplo, que na fase Concepção são desenvolvidas principalmente as disciplinas Modelagem de Negócio e Requisitos. No entanto, já se iniciam algumas atividades de Análise e Projeto, Implementação e Testes. Na fase Elaboração, a modelagem do negócio e os requisitos também são expressivos, embora a disciplina Análise e Projeto assuma o papel principal do processo. Observa-se também que a disciplina Gerenciamento do Projeto está presente em todas as fases.

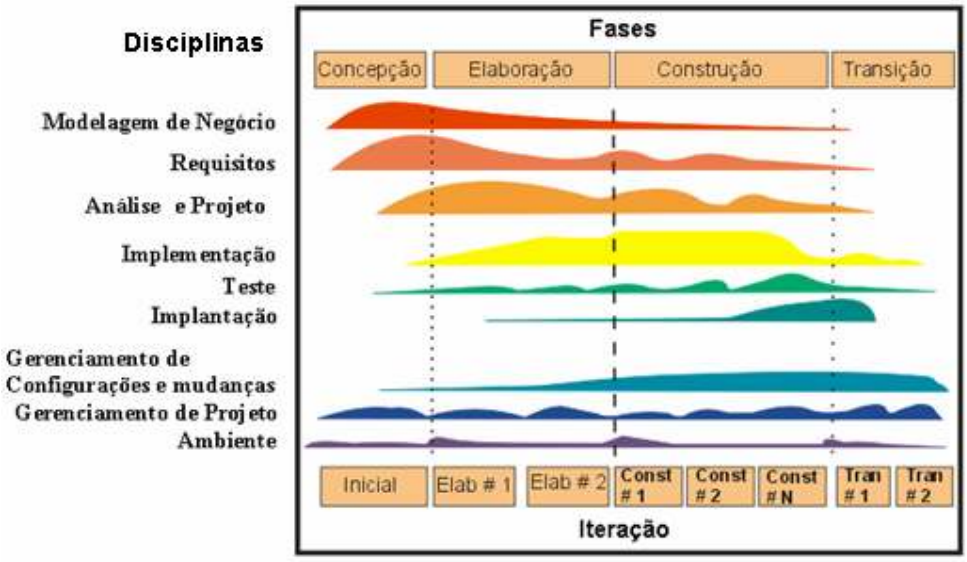


Figura 2 - Processo RUP [14]

A primeira fase do processo é chamada Concepção (*Inception Phase*) e seu objetivo é definir o escopo da versão que será construída a partir da especificação dos requisitos do sistema. Nesta fase são apresentados os requisitos funcionais e os critérios de aceitação do sistema. É importante também iniciar o levantamento dos possíveis riscos que possam comprometer o sucesso do projeto, avaliar uma possível arquitetura e sugerir um cronograma preliminar. Como produtos finais desta fase destacam-se os diagramas UML de caso de uso inicial e caso de uso do negócio, que em sistemas comerciais deve oferecer uma estimativa de retorno do investimento, cronograma e um protótipo preliminar.

A segunda fase, chamada Elaboração (*Elaboration Phase*) é responsável pela definição da arquitetura do sistema, pela avaliação dos riscos possíveis e pelo direcionamento do projeto para uma solução tecnológica possível. Para isso, os casos de uso são minuciosamente especificados. Também nesta fase é realizado o planejamento das atividades e dos recursos necessários e toda a infra-estrutura e o ambiente de desenvolvimento é elaborado. Outro item importante desta fase é a especificação da estrutura de componentes a ser utilizada, avaliando quais deles serão desenvolvidos, reutilizados ou adquiridos. Neste estágio de especificação, é possível definir os prazos e custos de todo o projeto. Como produtos finais desta fase, destacam-se o diagrama de casos de uso com todos os atores e casos de uso identificados e definidos, a especificação de arquitetura do software, o protótipo e o cronograma de todo o projeto.

A fase Construção (*Construction Phase*) é a terceira fase do processo e consiste basicamente no desenvolvimento do sistema. Esta é uma fase de construção de código, onde a ênfase principal é dada na gerência de recursos e controle das operações visando a otimização dos custos. Os componentes gerados nesta fase são testados e submetidos aos critérios de aceitação definidos anteriormente. No final desta fase o produto deve estar pronto para ser entregue ao usuário e deve conter basicamente o software construído e integrado às plataformas necessárias, o manual de usuários e uma descrição sobre esta versão.

A fase final do processo é chamada Transição (*Transition Phase*) e é responsável pela entrega do produto. Engloba a entrega, treinamento, suporte e manutenção. Esta fase inclui também os testes finais que avaliam se as expectativas do usuário foram realmente atendidas. São realizados testes comparativos com sistemas antigos, conversões de banco de dados e a divulgação do novo produto.

O RUP especifica um conceito de pontos de controle (PC) que permite a avaliação contínua do processo. Assim, em cada um destes pontos de controle, é avaliado o andamento

do projeto, considerando seus requisitos em relação à iteração e em relação a todo o sistema. Existem quatro pontos de controle que acontecem ao final de cada fase do projeto (Figura 3).

No ponto de controle I, após a fase Concepção, é realizada a primeira avaliação sobre o andamento do projeto. Deve ser verificada a coerência entre os requisitos e os casos de uso definidos, confrontado também com o protótipo gerado. Além disso, é importante avaliar os custos e prazos apresentados.

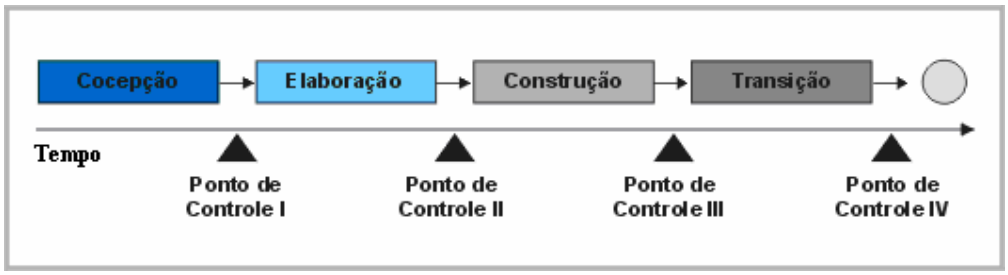


Figura 3 - Pontos de controle do RUP (Kruchten, 2000)

O ponto de controle II está logo após a fase Elaboração e é utilizado principalmente para avaliar se o protótipo apresentado está estável, ou seja, se foram resolvidos todos os pontos críticos que ofereciam riscos ao projeto. Além disso, é verificado se a especificação contém um nível de detalhe aceitável o suficiente para oferecer credibilidade ao cronograma apresentado.

Finalizada a fase de construção, tem-se o ponto de controle III . Neste ponto deve-se principalmente definir quando o sistema está pronto para ser entregue ao usuário. Para tanto deve ser avaliado se a nova versão está estável o suficiente para ser instalada e se os usuários estão aptos a utilizarem esta nova versão.

O último ponto de controle é realizado ao final da fase Transição. Neste ponto é decidido se os objetivos finais do projeto foram atendidos ou se uma nova versão deve ser iniciada.

Na prática, estas fases podem acontecer em paralelo entre versões diferentes, ou seja, durante a fase Transição da versão um pode-se iniciar a fase Elaboração da versão dois, desde que não haja dependência entre elas.

Desta forma pode-se dizer que no processo RUP, os riscos podem ser antecipados, as mudanças podem ser mais facilmente gerenciadas, há uma maior reutilização de código, a equipe de desenvolvimento se autodesenvolve juntamente com o sistema e consequentemente o software pode atingir um elevado nível de qualidade.

O RUP define também quatro elementos básicos que são utilizados em todo o processo: atores, atividades, artefatos e disciplinas (fluxos de trabalho).

A função de ator representa as responsabilidades de um ser externo que troca informação com o sistema. Pode ser um usuário, um grupo de usuários, outro sistema, uma máquina, dentre outras. Neste contexto, um usuário pode representar atores diferentes para atividades diferentes do sistema. As atividades representam as tarefas realizadas em cada disciplina. Elas são invocadas através do envio de informações pelos atores e processam estas informações para produzirem um resultado. Os artefatos representam tanto os dados enviados pelos atores às atividades quanto os resultados fornecidos pelas atividades após o seu processamento. O fluxo de trabalho (*workflow*) representa uma seqüência de atividades que produzem um resultado.

O RUP é um dos processos de desenvolvimento de sistemas mais utilizados atualmente porque além de utilizar a OO e a UML, mapeia o sistema de maneira gradual em iterações e permite um dinamismo entre as atividades (Figura 2). Pode-se observar que não se limita a definir fases, o conceito de disciplinas organiza as atividades afins e mapeia a influência destas em todas as fases do desenvolvimento, por exemplo, uma mesma disciplina pode ser desenvolvida em várias fases do processo. Esta dinâmica é muito importante nas aplicações mecatrônicas, pois torna flexível o processo uma vez que possibilita mapear vários aspectos do sistema em paralelo. Além disso, simplifica o entendimento, pois o uso de iterações permite que o projeto seja gradualmente detalhado. Também a política de gerência e o conceito de pontos de controle adotados pelo RUP facilitam o controle das atividades em uma equipe tão diversificada como a encontrada no ambiente da Mecatrônica.

Por ser um processo bem definido, o RUP permite também a adequação a modelos de qualidade. De acordo com [11] o RUP assegura cerca de 97% das práticas recomendadas pelo modelo *Capability Maturity Model Integration* (CMMI) nível dois. O CMMI [11], desenvolvido pelo *Software Engineering Institute*, é um modelo composto de práticas que visam ajudar as organizações a serem mais competitivas melhorando a sua eficiência, custo e satisfação dos clientes a partir da otimização de seus processos. Trata-se de um modelo e não de um método de trabalho, por isso as empresas o utilizam para adequar seus próprios processos, ou suas metodologias de trabalho. Assim como outros tipos de produtos, o desenvolvimento de produtos mecatrônicos também busca cada vez mais melhorar a sua qualidade e, portanto, a adoção de um modelo com este objetivo é importante.

2.1.4. *Software Process Engineering Metamodel* (SPEM)

O *Software Process Engineering Metamodel* (SPEM) é um metamodelo para definição de processos de desenvolvimento de software, especificado pela *Object Management Group*, baseado na abordagem OO na linguagem UML.

De acordo com a especificação do SPEM, uma metodologia tem a seguinte estrutura: possui um ciclo de vida que consiste em uma seqüência de fases, definidas no tempo e formadas por pré-condições de entrada, critérios, objetivos e saídas; possui iterações, trabalhos que são realizados e que tem um limite de tempo para serem finalizados; possui fluxos de trabalho (disciplinas) onde cada um deles reúne um conjunto de atividades afins; gera artefatos; define papéis, com responsabilidades atribuídas as pessoas que desenvolverão o software.

A Figura 4 demonstra uma maneira de representação de uma disciplina. Todos os elementos da disciplina são agrupados em um pacote. No topo do pacote, por exemplo, são identificados os papéis responsáveis pela execução da disciplina.

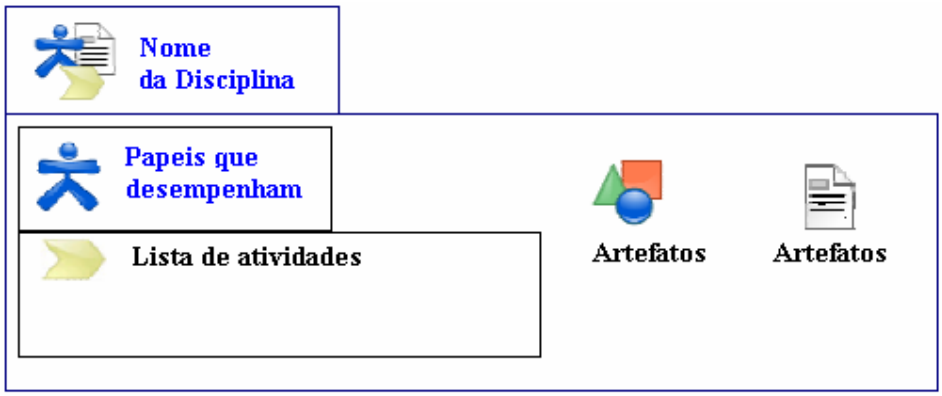


Figura 4 - Modelo de representação de uma disciplina no SPEM

Logo abaixo dos papéis é apresentado um quadro com todas as atividades que compõem a disciplina e do lado esquerdo os artefatos gerados, identificados por ícones e com uma descrição de acordo com a sua finalidade (Tabela 1).

O SPEM define uma série de símbolos que possuem um significado específico e que são utilizados na especificação dos processos, conforme apresentado na Tabela 1.

Tabela 1 - Símbolos utilizados pelo SPEM

Nome	Descrição	Símbolo
Pacotes de Processos	Agrupa elementos do processo.	
Fase	Etapa de uma metodologia.	
Artefato	Produto qualquer gerado ou utilizado por uma atividade.	
Documento	Documento gerado por uma atividade. Também considerado como artefato.	
Modelo UML	Representação de um modelo UML qualquer. Também considerado como artefato.	
Fluxo de trabalho (Disciplina)	Definição de um conjunto de atividades com objetivo especifico.	
Atividade	Item de uma disciplina.	
Papel	Papel desempenhado por uma pessoa que utiliza a metodologia.	

Para o SPEM um processo de desenvolvimento de software é formado por entidades abstratas, chamadas Pacotes de Processos, que colaboram entre si. O objetivo principal quando se define um processo de desenvolvimento é definir como serão estes pacotes e onde eles serão aplicados dentro do ciclo de vida do processo. Por exemplo, pode-se definir um pacote que represente uma disciplina. Este pacote será formado por uma série de elementos: as atividades que compõe a disciplina; e os artefatos utilizados ou gerados por ela.

2.2. DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

O desenvolvimento de produtos eletrônicos está em constante evolução. Como parte desta evolução é visível a crescente utilização de soluções híbridas, que envolvam hardware e software. Esta tendência tem sido motivada pela redução de custos característicos desta solução, pois quanto mais implementações forem feitas em software, mais barato o custo do produto.

O paradigma do *co-design*[18] trata de soluções desta natureza, ou seja, de produtos que podem ter uma implementação híbrida. Esta realidade pode ser fortemente observada também nos produtos mecatrônicos, onde a integração das diversas áreas envolvidas tende a este tipo de solução.

Outras técnicas permitem mapear requisitos para facilitar o processo de especificação, a exemplo do diagrama de *function blocks* (FB) [35], bastante referenciados na especificação de produtos [3, 4, 12]. No domínio da Mecatrônica os FBs também podem ser bastante úteis, pois oferecem uma linguagem de simples entendimento para os engenheiros.

As seções seguintes 2.2.1 e 2.2.2 descrevem, respectivamente, em linhas gerais a tecnologia de *co-design* e os diagramas de *function blocks*.

2.2.1. Co-design

O paradigma de c*o-design* trata especificamente dos sistemas que possuem partes eletrônicas, desenvolvidas em hardware, e partes desenvolvidas em software. Para isso, define um modelo de desenvolvimento, com regras e algoritmos, cujo objetivo é encontrar uma solução aceitável de implementação, seja ela em software ou em hardware. Esta solução está principalmente relacionada aos requisitos definidos durante a fase de análise do produto, e a questões relativas a desempenho e custo. Esta abordagem pode ser enriquecida com a possibilidade de geração automática dos modelos gerados na especificação em uma linguagem que permita distinguir as definições de hardware e software. Por exemplo, a linguagem SystemC [15] pode ser utilizada para esta finalidade, pois esta linguagem permite especificações de hardware e software em um alto nível de abstração além de geração automática de código.

Uma metodologia típica em co-design possui cinco passos (Figura 5): o detalhamento dos requisitos, funcionais e de desempenho; o particionamento destes requisitos em hardware e software; a definição da interface entre os elementos de hardware e software; a síntese de hardware; e a síntese de software [18].

Inicialmente é feito o levantamento de requisitos do sistema e identificadas às necessidades de desempenho. De posse destes dados, algoritmos são aplicados para chegar a melhor solução de particionamento em hardware-software que atenda aos requisitos. Uma vez particionado são definidas as interfaces e finalmente sintetizadas ambas as partes para que seja montado o produto final.

Como se pode observar, a etapa de particionamento é uma das mais importantes no *co-design*, pois consiste exatamente na definição de quais elementos são implementados em hardware e quais são implementados em software. O desafio do particionamento é ajudar

na definição de uma solução de implementação que atenda aos requisitos definidos para o produto com o melhor custo e em um desempenho aceitável.

O mapeamento de soluções que podem ser aplicadas em um problema desta natureza gera um número grande de combinações possíveis. Estas combinações representam possibilidades de implementação e podem ser limitadas a depender da heurística de convergência que for utilizada no algoritmo de particionamento.

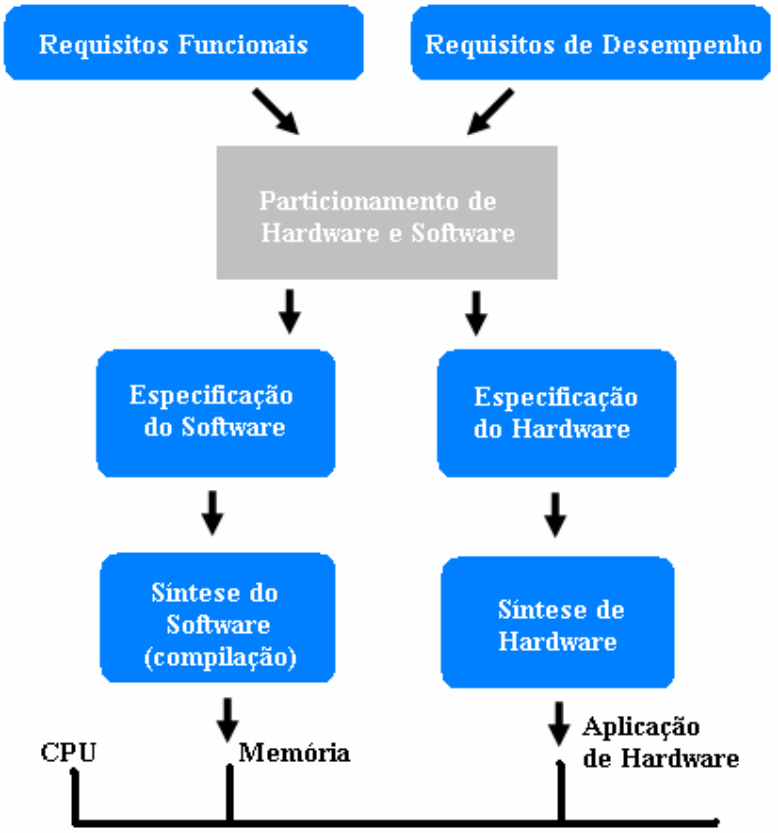


Figura 5 - Metodologia de *Co-design* (adaptado de [34])

Em geral, existem quatro formas de implementação possíveis de serem aplicadas pelo *co-design*: implementações apenas em software; apenas em hardware; em software com suporte de hardware; e em hardware com suporte de software, como é o caso dos dispositivos programáveis.

Já existem várias técnicas de particionamento que sugerem algoritmos para gerar as alternativas de solução, tais como os algoritmos genéticos [22], algoritmos de pesquisa [18], algoritmos baseado em programação linear [1], dentre outros.

O desenvolvimento de produtos mecatrônicos também requer em um determinado estágio o particionamento de hardware e software. O paradigma de *co-design* se adapta bem a esta necessidade, pois além de fornecer técnicas para avaliar a melhor solução de implementação, também se preocupa com a definição das interfaces entre os elementos particionados.

2.2.2. *Function Block* (FB)

O International Electrotechnical Commission (IEC) é um órgão responsável pela publicação e padronização de tecnologias elétricas e eletrônicas. Ele é responsável pela padronização do conceito de *Function Block* (FB) amplamente usado hoje no desenvolvimento de sistemas de controle. O FB pode ser definido como um mecanismo de abstração que permite encapsular algoritmos industriais de forma simples para serem utilizados por engenheiros não especialistas em desenvolvimento de sistemas [35].

A interface no FB, como pode ser observada na Figura 6, é composta por uma parte responsável pela entrada e saída de eventos e uma segunda parte que recebe e envia conexões de dados. Um modelo completo de representação de um sistema é composto por um conjunto de FB conectados a partir de suas entradas e saídas. Os eventos ativam o controle do FB, chamado *Execution Control Chart* (ECC), formado por especificações de como os eventos devem ser processados e quando os algoritmos internos do FB devem ser executados. A segunda parte do FB armazena os algoritmos internos que serão executados pelo ECC e os dados necessários.

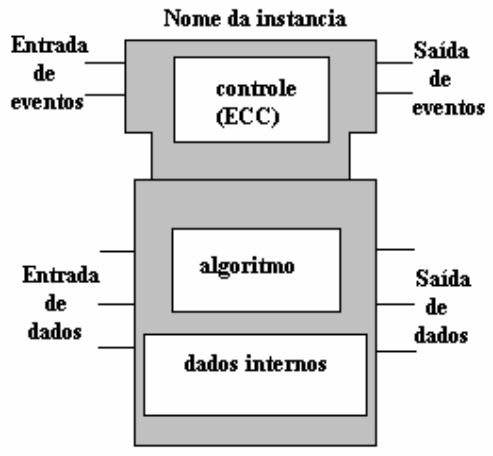


Figura 6 – *Function Block* (adaptado de [3])

Existem trabalhos, a exemplo do descrito em [30], que propõem agregar mais facilidades à implementação dos componentes definidos na UML através da migração de modelos UML para diagramas FB. De posse de um diagrama FB, o engenheiro pode continuar trabalhando com os procedimentos já estabelecidos na Engenharia Elétrica.

A utilização de FB vem sendo adotada na modelagem de produtos mecatrônicos porque além de provê uma maior facilidade de entendimento para os engenheiros, permite decompor um componente, representando o controle, o software e os dados separadamente.

2.3. DESENVOLVIMENTO DE PRODUTOS MECÂNICOS

Assim como a Engenharia de Software, a Engenharia de Produtos também vem ao longo do tempo aperfeiçoando técnicas que podem auxiliar o desenvolvimento de produtos de maneira a construí-los com maior qualidade e a um custo cada vez mais acessível ao consumidor.

A Engenharia de Produtos apresenta um modelo geral de desenvolvimento de produto composto por quatro etapas: definição do produto; projeto do produto; produção do produto; lançamento e acompanhamento do produto. Dentre estas etapas, o projeto do produto é o objeto de estudo desta dissertação. Ele é responsável pelas atividades que vão desde à especificação de projeto e detalhamento de suas operações até a geração da documentação necessária para a sua produção[9].

Para a especificação do projeto do produto, uma abordagem amplamente utilizada hoje é a sistemática, onde o projeto é visto como uma evolução sistemática de modelos. Assim, parte-se de modelos abstratos e simples e após sucessivos refinamentos tem-se uma visão mais concreta e detalhada do produto. Nesta abordagem, alguns modelos foram sugeridos para auxiliar o desenvolvimento, dentre eles, o modelo de fases merece destaque, pois reúne as características principais dos modelos precursores da engenharia. Ele divide o projeto em quatro etapas, conforme ilustrado na Figura 7.

O produto de cada etapa é um modelo refinado que servirá de base para o início da etapa seguinte, até atingir um nível de especificação que possa ser enviado para produção.

A primeira etapa do modelo chama-se projeto informacional e tem como foco o entendimento do problema, ou seja, o levantamento de todas as informações necessárias à especificação do produto. Como resultado tem-se as especificações do projeto, composta por

uma lista de objetivos que possibilita a definição de funções, propriedades e restrições do produto.

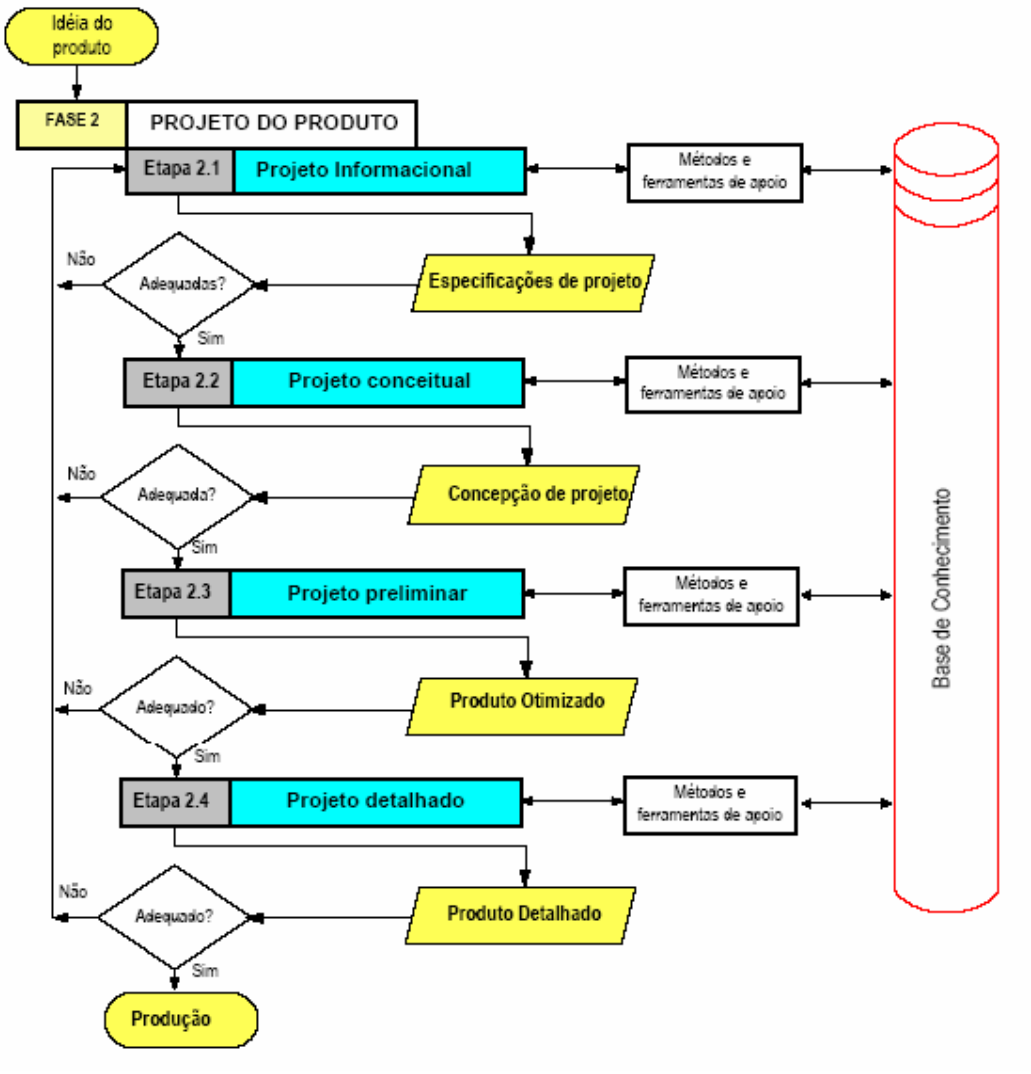


Figura 7 - Modelo de fases para o projeto de produto [9]

A segunda etapa, chamada de projeto conceitual, objetiva gerar uma concepção para o produto que atenda às especificações detalhadas na etapa anterior respeitando suas restrições. Como resultado é gerada uma solução, uma concepção para o produto que será construído.

A terceira etapa do modelo, projeto preliminar é responsável pelo desenvolvimento do projeto a partir da sua concepção. São desenvolvidos principalmente *layout* e formas para as concepções selecionadas, tendo como resultado um produto otimizado.

A quarta etapa do modelo, projeto detalhado, é responsável pelo detalhamento do projeto preliminar em um nível que possa ser enviado à produção [9].

Segundo [9], é fato hoje que as decisões tomadas durante todas as fases de especificação de um produto têm uma influência significativa na manufatura, qualidade, e custo para sua produção. Por este motivo, metodologias estão sendo especificadas para, além de especificar o produto, planejar a produção, montagem, *marketing*, entre outros, e abordar todos os aspectos do gerenciamento do ciclo de vida do produto. Dentre estas metodologias está a engenharia simultânea, que procura tratar não só a parte técnica de especificação, mas também da integração da equipe de especificação com outras equipes envolvidas no projeto, tais como a de produção. Além do gerenciamento e da troca de conhecimento entre os diversos setores, permite um paralelismo das atividades, o que facilita a antecipação de problemas. Estas características influenciam diretamente no tempo de desenvolvimento, custo e otimização de soluções (Figura 8).

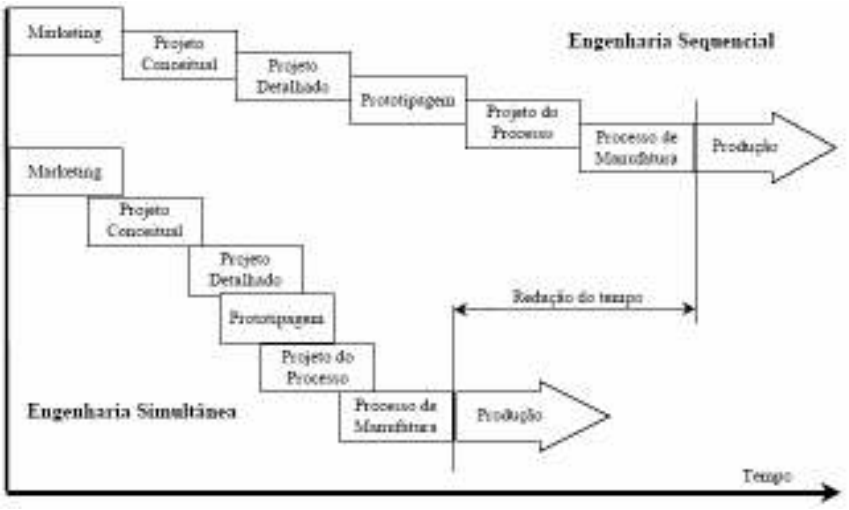


Figura 8 - Engenharia Seqüencial e Engenharia Simultânea [7]

Na Figura 8, são visíveis as diferenças entre o processo seqüencial e o simultâneo, principalmente no gerenciamento das atividades, pois no processo simultâneo equipes multifuncionais trabalham em conjunto nas diversas fases do projeto. Esta prática foi mais amplamente difundida com o apoio de ferramentas computacionais no projeto.

Observa-se então, que o desenvolvimento simultâneo pode ser útil ao desenvolvimento de produtos mecatrônicos por sua abordagem multidisciplinar e pela flexibilidade demonstrada nas suas atividades.

2.3.1. Ferramentas que atendem ao desenvolvimento de produtos

Para atender às metodologias utilizadas no desenvolvimento de produtos, algumas ferramentas de modelagem foram desenvolvidas e vem sendo utilizadas com sucesso. São exemplos delas a matriz da casa da qualidade e a matriz morfológica. Elas ajudam, respectivamente, na análise dos requisitos e na definição de soluções para o produto e parecem ser adequadas para equipes multidisciplinares como a Mecatrônica.

a) Matriz da casa da qualidade

A matriz da casa da qualidade é um método para análise e classificação de requisitos de um sistema. Para isso, estabelece relações entre as necessidades do cliente e os requisitos do projeto [26].

Entende-se por necessidades do cliente, o conjunto de solicitações do cliente para o produto. Estas necessidades são categorizadas e traduzidas em requisitos mensuráveis, chamados de requisitos de projeto. Por exemplo, a necessidade informada pelo cliente como Baixo aquecimento, pode ser traduzida no requisito de projeto Aquecimento externo, para ser mensurada em valores medidos em ^{o}C.

A casa da qualidade oferece uma visão global sobre o sistema e como os requisitos se relacionam entre si. Sua construção geralmente envolve os integrantes de todas as áreas participantes do projeto.

Pode-se observar na Figura 9 que a casa é formada principalmente pelas necessidades do cliente (linhas) e pelos requisitos do projeto (colunas). O cruzamento destes dados fornece uma avaliação quantitativa que varia segundo a escala apresentada na Figura 9: cinco para forte relacionamento; três para médio relacionamento; e um para fraco relacionamento. Para cada necessidade do cliente é estabelecida também uma importância (coluna ao lado das necessidades do cliente). Esta importância é similar a uma nota que varia de um a cinco de acordo com o valor que o cliente deposita àquela necessidade. Pode-se realizar também uma pesquisa de mercado e avaliar como se comportam as necessidades definidas pelo cliente em produtos concorrentes. Na Figura 9, as colunas da direita apresentam a pontuação das necessidades do cliente para quatro produtos existentes no mercado: A, B C e D. De posse destes dados, é possível gerar uma análise de mercado do produto, mostrando-se os pontos fracos e fortes do produto sob a ótica do próprio consumidor. A última linha da matriz apresenta a análise realizada para um forno de microondas indicando

uma escala de prioridades para os requisitos mapeados (importância dos requisitos para o projeto do produto).Existem algumas ferramentas computacionais, a exemplo de [8], que disponibilizam um ambiente para construção da matriz da casa da qualidade.

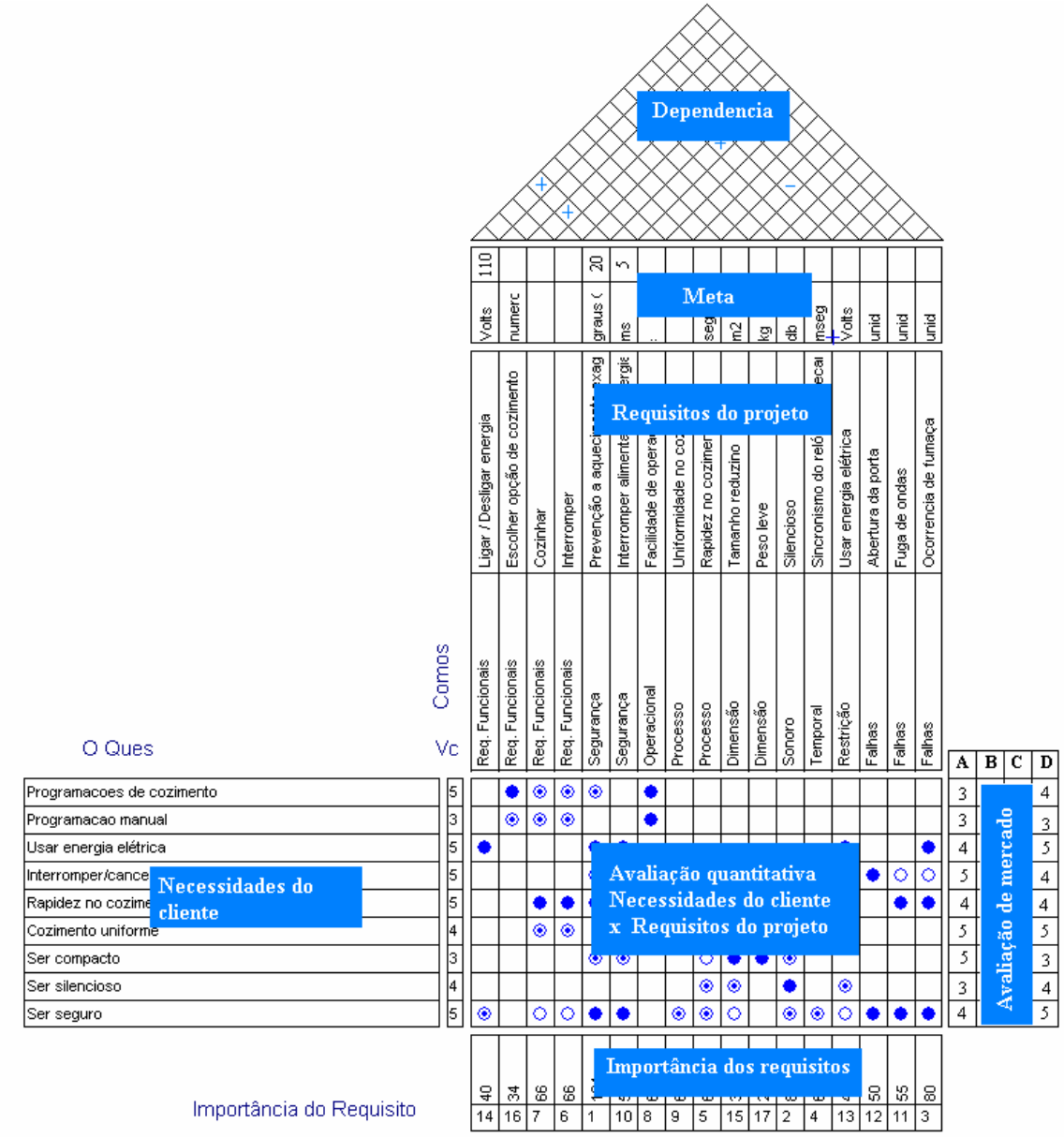


Figura 9 - Matriz da casa da qualidade de um forno de microondas

O telhado da casa tem uma grande importância nas definições futuras do projeto. É a partir dele que são estabelecidos os graus de dependência existentes entre os requisitos. Assim, é possível identificar afinidades e conflitos [26]. Por exemplo, o requisito Opções pré programadas de cozimento possui um conflito com o requisito Facilidade de utilização, pois quanto maior o número de programações mais complexo pode se tornar a utilização do

produto. Este conflito está refletido no telhado pelo símbolo – na posição que relaciona os dois requisitos.

Portanto, a matriz da casa da qualidade é considerada uma ferramenta poderosa, pois permite que análises diversificadas do projeto sejam submetidas a um grupo de pessoas com conhecimentos e enfoques diferentes. No ambiente da Mecatrônica ela se aplica com eficiência pelas possibilidades que pode oferecer na troca de conhecimentos e análise do produto numa equipe multidisciplinar.

b) Matriz morfológica

A matriz morfológica tem como objetivo ajudar o desenvolvedor a encontrar soluções para o produto. Para isso utiliza uma matriz onde realiza combinações de elementos e parâmetros oferecendo diversas possibilidades de construção para um mesmo item [7].

Tabela 2 - Matriz morfológica de um forno de microondas

Componentes	**Princípios de soluções**		
Força	Filtro de linha.	Estabilizador.	Tomada simples.
Controlador	Micro controlador; Sensores mecânicos.	Micro controlador; Sensor inteligente.	
Atuador	Emissor fixo no topo da carcaça.	Emissor fixo em algum ponto com refletores.	
Entrada / Saída	Visor digital, 1 botão para cada comando e um teclado numérico; Botões de execução e sirene.	Visor digital, 1 botão de opção e teclado numérico; Botões de execução e sirene.	Botão para aumentar e diminuir a temperatura; Botões de execução e sirene.
Prato	Prato redondo de vidro e mecanismo para girar prato.	Prato quadrado de vidro.	Prato redondo de plástico.
Porta	Porta de metal isolante com vidro isolante, sensores mecânicos acoplados e um micro controlador; Duplicação de sensores e de micro controlador para evitar falha.	Porta de metal isolante com vidro isolante e sensores inteligentes; Duplicação de sensores para evitar falha.	Campo magnético isolante com sensores inteligentes; Duplicação de sensores para evitar falha.

A Tabela 2 apresenta um exemplo de matriz morfológica para um forno de microondas. Para construir a matriz listam-se na primeira coluna todos os componentes que deverão formar o produto. Para cada um dos componentes, representado por cada linha da matriz, buscam-se os princípios de solução possíveis e completa-se a linha. Por exemplo, para o componente Prato da Tabela 2, é possível especificar três soluções: prato redondo de vidro; prato quadrado de vidro; e prato redondo de plástico.

A partir da matriz morfológica é possível montar concepções de projeto para o produto. Uma concepção é montada a partir da combinação de um princípio de solução para cada componente listado. Pode-se gerar várias concepções para serem avaliadas pela equipe do projeto.

Tabela 3 - Concepções de projeto para um forno de micro ondas

Componentes	**Concepções**	
	1	**2**
Força	Estabilizador.	Filtro de linha.
Controlador	Micro controlador e Sensores mecânicos.	Micro controlador e Sensores mecânicos.
Atuador	Emissor fixo no topo da carcaça.	Emissor fixo em algum ponto com refletores para atingir de maneira uniforme todo o forno.
Entrada / Saída	Visor digital, um botão para cada comando e um teclado numérico; Botões de execução e sirene.	Visor digital, um botão para cada comando e um teclado numérico; Botões de execução e sirene.
Prato	Prato redondo de vidro e mecanismo para girar prato.	Prato redondo de plástico.
Porta	Porta de metal isolante com vidro isolante, sensores mecânicos acoplados e um micro controlador; Duplicação de sensores e de micro controlador para evitar falha.	Campo magnético isolante com sensores inteligentes; Duplicação de sensores para evitar falha.

A Tabela 3 apresenta duas concepções de projeto para um forno de microondas. Nota-se que em cada uma das concepções, para cada componente, foi selecionado um princípio de solução.

A escolha de uma concepção para ser utilizada no produto em construção envolve a análise de aspectos importantes relativas à tecnologia, custos, montagem, fabricação, entre outros. Existem algumas técnicas desenvolvidas para nortear os projetistas nestas decisões a exemplo da Teoria de Solução Inventiva de Problemas (TRIZ) [26].

A matriz morfológica pode ser importante no desenvolvimento de produtos mecatrônicos porque permite mapear e combinar as diversas soluções de implementação para um produto. Além disso, apresenta uma visão geral da concepção do produto para a equipe com uma linguagem simples e de fácil entendimento.

3. DESENVOLVIMENTO DE PRODUTOS MECATRÔNICOS

A busca por procedimentos que possam ser aplicados ao projeto e desenvolvimento de produtos mecatrônicos é um campo amplo de pesquisa, uma vez que a indústria necessita de uma metodologia adequada para estas aplicações [3].

O ambiente de desenvolvimento de produtos mecatrônicos é caracterizado pela utilização de uma diversidade de métodos já consolidados utilizados pela Mecânica, Elétrica e Computação. Esta característica dificulta o processo de construção de produtos, pois cria uma resistência dos projetistas em abdicar de suas técnicas em prol de um processo unificado.

O desenvolvimento dos produtos mecatrônicos também aborda aspectos importantes nem sempre presentes em outras categorias de produtos. Por exemplo, os produtos mecatrônicos geralmente interagem com o ambiente onde estão inseridos e esta interação requer especificações adicionais tais como a sincronização produto-ambiente, que demanda restrições temporais. Um outro aspecto a ser considerado é a utilização de produtos mecatrônicos em situações de risco, o que requer uma atenção especial para aspectos de confiabilidade aos produtos.

Neste cenário, muitos trabalhos já estão sendo desenvolvidos e algumas metodologias estão sendo propostas e testadas [3, 4, 31, 35, 19]. A seção 3.1 apresenta uma visão geral sobre produtos mecatrônicos descrevendo algumas características importantes relacionadas ao seu desenvolvimento. As seções seguintes apresentam metodologias já propostas para este domínio de aplicação, destacando como elas abordam aspectos importantes relacionados ao desenvolvimento de produtos mecatrônicos. A seção 3.7 faz uma análise comparativa das metodologias apresentadas e efetua uma breve comparação com a metodologia proposta nesta dissertação.

3.1. UMA VISÃO GERAL SOBRE PRODUTOS MECATRÔNICOS

Os produtos mecatrônicos geralmente interagem com o ambiente por intermédio de sensores e atuadores e para que as interações funcionem de maneira satisfatória, é necessário haver um sincronismo entre o ambiente controlado e o produto [16]. Por este motivo, seu desenvolvimento contem características especiais que devem ser tratadas durante todo o processo de especificação.

Os produtos mecatrônicos em geral são sistemas de tempo real e, portanto, é necessário que a troca de informações entre o ambiente e o produto aconteça durante um tempo esperado para que as ações de atuação sejam satisfatórias e atinjam o seu objetivo. Por exemplo, em um forno de microondas, uma vez detectado que existe uma incidência de ondas acima do programado (a partir de um sensor), esta incidência deve ser regularizada (a partir de um atuador) em um tempo específico para que o alimento não sofra uma alteração indesejada. A ocorrência de um estímulo no ambiente que está sendo constantemente monitorado dispara uma ação no produto. No exemplo do forno é o aumento da intensidade das microondas que ativa o processamento, e o resultado é uma atuação que visa diminuir esta intensidade. Assim, quando um evento é identificado, é executado um conjunto de tarefas com características específicas, tais como, o intervalo máximo de tempo permitido para a execução da mesma (*deadline*), a sua periodicidade de ativação, e o seu tempo máximo de execução, para que seja possível projetar satisfatoriamente a dinâmica de funcionamento do produto.

Assim como em outras áreas, o desenvolvimento de produtos mecatrônicos demanda a utilização de técnicas que permitam inserir um maior nível de confiança aos resultados que serão gerados pelo produto. Uma técnica que pode ser aplicada com sucesso é a validação [25]. Ela envolve mecanismos para minimizar erros de especificação e de concepção a partir da realização de testes, simulações ou construção de protótipos. Por exemplo, é possível submeter a especificação de um produto a ferramentas computacionais de simulação para testá-lo antes mesmo de ele ser fisicamente construído. Desta forma, pode-se simular o funcionamento e analisar seu comportamento em situações excepcionais como, por exemplo, na presença de falhas. Esta simulação é interessante principalmente por ser puramente virtual.

No entanto, muitos produtos mecatrônicos são considerados de risco ou críticos. Estes produtos demandam um maior grau de confiabilidade, pois seu mau funcionamento

pode acarretar implicações graves. Um exemplo destes são os monitores de pacientes em hospitais. Nestes casos, erros de especificação podem acarretar problemas de funcionamento que comprometam seriamente a saúde do paciente. Neste contexto, uma técnica aplicada com sucesso é o método formal. Esta abordagem utiliza modelos matemáticos para especificar o produto onde são definidas propriedades que devem ser submetidas ao modelo de maneira que possa ser verificada a sua correção.

Mesmo que sejam utilizadas técnicas de validação e verificação para tentar garantir o perfeito funcionamento do produto e que sejam tratados todos os requisitos temporais, falhas imprevisíveis podem acontecer. Por exemplo, no forno de microondas, mesmo que seja especificado, testado e simulado que na detecção do aumento exagerado da intensidade das microondas estas devem ser reguladas por atuação direta do controlador, o mau funcionamento de um sensor pode comprometer o resultado final. Por isso, é importante especificar também para esta categoria de produtos um modelo de falhas que possa assegurar o seu bom funcionamento e oferecer maior confiança. Para se construir um modelo de falhas parte-se de hipóteses, as mais precisas possíveis, de falhas que podem acontecer com o produto. Estas falhas podem ser desde falhas físicas, em algum componente mecânico, por exemplo, até falhas de especificação e falhas humanas, de operação [16]. O importante é que elas sejam consideradas na especificação do produto para que possam ser detectadas e tratadas quando necessário.

3.2. METODOLOGIA DE BONFÉ

Em [3, 4] Bonfé propõe uma metodologia para sistemas mecatrônicos na qual se define a estrutura do sistema formada por um conjunto de subprocessos que possuem na sua maioria partes mecânicas, sensores, atuadores e controle. A metodologia está baseada na OO e nos modelos da UML e destaca a possibilidade do desenvolvedor especificar a sintaxe e a semântica mais adequada para o domínio das aplicações mecatrônicas a partir do uso de estereótipos, estendendo os padrões de classes, objetos, entre outros. Utiliza também normas da IEC 61499 [21, 6] para facilitar o mapeamento dos modelos UML em linguagens de controle. A metodologia possui uma característica peculiar ao introduzir o uso de diagramas de fluxo de dados (DFD), modelo da Análise Estruturada [38], em seu processo de desenvolvimento. O autor considera que a utilização de DFDs torna mais clara a definição de fluxo de eventos e sincronização.

O processo de análise e desenvolvimento descrito em [3, 4] se baseia em passos oriundos da análise orientada a objetos, da análise estruturada e alguns métodos de desenvolvimento de produtos. Os passos são os seguintes: definição dos requisitos funcionais do sistema, com a utilização dos diagramas de casos de uso; identificação dos objetos, a partir da análise, tanto dos requisitos funcionais quanto dos requisitos de Mecânica e Elétrica, na construção do diagrama de classes; identificação das interfaces entre os objetos definidos, construindo diagramas de fluxo de dados; especificação do comportamento do controlador, a partir de refinamentos sucessivos dos requisitos funcionais, gerando diagramas de *statecharts* [5] para as classes; verificação formal, para provar que a especificação realizada está correta em relação aos requisitos especificados; desenvolvimento detalhado da especificação, com a construção de *function blocks* (FB); e implementação do sistema, com a utilização de uma linguagem de programação.

3.3. METODOLOGIA CORFU

A metodologia CORFU, apresentada em [31, 35], foi especificada para o desenvolvimento de projetos mecatrônicos, com ênfase na construção de Aplicações de Controle Distribuídos (DCA). Utiliza a UML para definir o processo de levantamento de requisitos e análise e utiliza também os conceitos de FB para a fase de desenvolvimento, de acordo com as regras definidas pelo padrão IEC 61499. O processo de desenvolvimento CORFU define uma série de procedimentos que englobam as fases de levantamento de requisitos, análise, desenvolvimento, testes e implementação. O levantamento de requisitos é realizado a partir da construção do diagrama de casos de uso, que especifica as interações do sistema com o mundo externo. Na fase de análise dois passos são realizados: captura do comportamento, a partir da construção de diagramas de interações (seqüência ou colaboração); e captura da visão estática do sistema a partir da identificação dos objetos e da construção do diagrama de classes correspondente. Após a análise, inicia-se o processo de desenvolvimento a partir do mecanismo de tradução dos modelos UML para FBs. As informações contidas em cada uma das classes são mapeadas em um FB. Desta forma, após o processo de tradução, o diagrama de classes se transforma no diagrama de FBs. Finalizando a metodologia, tem-se as etapas de avaliação e verificação.

A metodologia CORFU é um *framework* de desenvolvimento que contém ferramentas de suporte e oferece integração com produtos já consolidados no mercado, como, por exemplo, o ROSE [14], ferramenta para desenvolvimento de projetos orientados a objetos.

O *framework* dispõe de ferramentas, tais como o *Function Block Development Kit* (FBDK) [35] e o *Verification Enviroment for Distributed Applications* (VEDA) [35], desenvolvidas para auxiliar o processo de construção e tradução dos modelos UML em FB.

3.4. METODOLOGIA *MODEL INTEGRATED MECHATRONICS*

Esta metodologia, especificada em [33], é uma evolução da CORFU citada no item anterior. Nela, propõe-se uma arquitetura de desenvolvimento de sistemas mecatrônicos orientada a modelos que abrange desde as etapas iniciais de especificação do produto até a sua construção. Como ponto central da arquitetura proposta é colocado o foco no reuso de componentes, o que gera uma significante redução no tempo de desenvolvimento.

A arquitetura atende a três características básicas: reuso de componentes; agilidade no desenvolvimento e flexibilidade. Estas características permitem adaptações no desenvolvimento frente a mudanças no ambiente. Além disso, utiliza uma linguagem de modelagem criada exclusivamente para o domínio de engenharia simultânea e para componentes mecatrônicos. Para isso usa uma arquitetura orientada a modelos, originada dos avanços da orientação a objetos, e utiliza o IEC61499 *funtion block*, da Engenharia Elétrica.

A arquitetura MIM pode ser visualizada em duas dimensões: a dimensão de integração e a de evolução.

A integração é composta de quatro camadas. A camada mais externa, chamada de mecatrônica, é responsável pela especificação conceitual do produto e utiliza diagramas UML, como, por exemplo, casos de uso e de classes. A segunda camada, chamada de aplicação, é responsável pelas definições de controle do sistema. A terceira camada, chamada de recursos, detalha a infra-estrutura, a comunicação entre os softwares / hardware, entre outras atividades. A camada mais baixa é a mecânica, onde são projetados componentes do mundo real.

A dimensão de evolução intercepta verticalmente a outra dimensão dividindo as camadas em três segmentos distintos: análise, projeto e implementação. Definições de interface devem ser desenvolvidas para as quatro camadas de forma que para um componente mecatrônico ser interconectado ele precisa ter as interfaces mecânica, eletrônica e de software compatíveis.

A arquitetura da metodologia é enriquecida com a construção de uma ferramenta computacional, chamada Archimedes[33], que permite conduzir todo o processo de

desenvolvimento do sistema. Esta ferramenta engloba um guia de uso da metodologia, um *framework* que provê a tecnologia necessária para usar a metodologia, um ambiente de desenvolvimento para o componente mecatrônico e um sistema de suporte à engenharia com ferramentas de suporte ao desenvolvimento, validação, configuração, entre outras atividades necessárias.

A abordagem adotada pela MIM utiliza a engenharia concorrente para modelar componentes mecatrônicos, composto de partes mecânicas, eletrônicas e de software. A arquitetura abrange a conceituação dos componentes que compõem o sistema, o mapeamento de seus respectivos relacionamentos e os padrões de construção que podem ser utilizados.

3.5. METODOLOGIA DE MROZEK

Em [19] Mrozek apresenta um estudo sobre a utilização de diagramas UML para o desenvolvimento de sistemas mecatrônicos. Uma vez gerada a especificação e feita a análise, utilizam-se ferramentas de projeto auxiliado por computador, produção auxiliada por computador e engenharia auxiliada por computador para detalhar o projeto. Ferramentas de simulação são sugeridas para analisar o comportamento do produto antes da sua construção física. O processo de desenvolvimento está especificado de acordo com os seguintes passos: construção de casos de uso com visões preliminares sobre o problema; descrição dos cenários mapeados pelos casos de uso; identificação dos objetos através da construção de um diagrama de classes preliminar; construção de diagramas de seqüência e colaboração que mapeia as ações e interações entre objetos; construção de diagramas de *statecharts;* construção de diagrama de atividades para visualização de atividades paralelas. A metodologia prevê o mapeamento de requisitos não funcionais, a partir de uma linguagem descritiva de restrições, a *Object Constraint Language*, que utiliza notações e pseudo-códigos para representar estes requisitos nos modelos da UML.

A metodologia destaca também um modelo de gerenciamento para o processo de desenvolvimento de sistemas mecatrônicos, considerando o ambiente multidisciplinar existente. O processo de construção é dividido em quatro etapas, que englobam os passos anteriormente descritos: especificação de requisitos; análise conceitual; detalhamento, prototipagem, teste e implementação. Para cada uma destas etapas são destacadas as decisões e responsabilidades que devem ser definidas para o bom andamento do projeto.

3.6. METODOLOGIA DE ISERMANN

Esta metodologia, apresentada em [12], destaca a importância da engenharia simultânea no desenvolvimento de produtos mecatrônicos, visto que a integração das áreas deve estar presente desde o início das especificações. Classifica a integração dos sistemas mecatrônicos de duas formas: a integração de componentes e a integração de processamento de informação.

A integração de componentes, ou integração de hardware, consiste na integração dos sensores, atuadores e da parte mecânica. Neste caso, integrações com microcomputadores podem estar encapsuladas formando sensores ou atuadores inteligentes.

A integração de processamento de informação, ou integração de software, possui foco principal nas aplicações de controle, abordando tanto o tratamento dos sinais de controle recebidos do ambiente quanto a supervisão do sistema, tolerância a falhas, entre outros aspectos.

Nesta metodologia, o desenvolvimento está organizado em camadas responsáveis por níveis de controle. Para cada nível são especificadas funcionalidades, bases de conhecimento e interface entre os mecanismos.

3.7. PROCESSO IPPROCESS

Componentes embarcados podem ser úteis ao projeto de produtos mecatrônicos. O desenvolvimento destes componentes geralmente aborda uma tecnologia denominada *System on Chip* (*SoC*) que permite colocar o sistema construído em um determinado *chip*. A construção de SoC tem sido realizada a partir de técnicas que enfatizam a reutilização de componentes chamados *Intellectual Property Core* (*IP-core*).

O trabalho especificado em [15], intitulado IPPROCESS, utiliza a Engenharia de Software e se baseia no RUP e no *Extreme Programming* [2] para especificar um processo aplicado ao desenvolvimento de *IP-core*. O objetivo do IPPROCESS é utilizar uma metodologia de desenvolvimento que permita descobrir erros de concepção ainda na fase de projeto, antes que o sistema seja embarcado em um chip.

O IPPROCESS é composto de fases e disciplinas. As fases possuem uma seqüência no tempo e são importantes para nortear o processo de especificação. O IPPROCESS compreende quatro fases: concepção, responsável pelo entendimento dos requisitos do *IP-core* a ser desenvolvido a partir das necessidades do cliente; arquitetura, que

utiliza os requisitos mapeados na fase anterior; projeto RTL, responsável pela construção do mecanismo de verificação funcional dos componentes anteriormente projetados; e protótipo físico, responsável pela implementação física do *IP-core* projetado. As disciplinas agrupam atividades relacionadas e são desenvolvidas ao longo das fases que compõem o processo. O IPPROCESS define cinco disciplinas: requisitos, responsável pelo entendimento, estimativas de custo e tempo e definição de *inteface* externa ao *IP-core*; análise e projeto, responsável pela definição da arquitetura do *IP-core*; implementação RTL, responsável pela implementação do código RTL de cada componente, testes e integração; verificação funcional, responsável por avaliações de qualidade e validações necessárias; e prototipação, que agrupa as atividades necessárias para implementar fisicamente o *IP-core*.

A partir desta estrutura de fases e disciplinas o IPPROCESS gera como resultado o *IP-core* fisicamente implementado e testado, construindo ao longo do processo artefatos importantes para a documentação da especificação. Além disso, destaca a gerencia das atividades dentro da equipe a partir da definição de papeis e responsabilidades para cada um dos integrantes. Propõe validações em todo o processo, buscando antecipar possíveis problemas de especificação.

3.8. CONSIDERAÇÕES SOBRE OS TRABALHOS RELACIONADOS

Nos trabalhos apresentados sobre construção de produtos mecatrônicos pode-se observar a preocupação em aproveitar os métodos já utilizados pelas áreas específicas, adaptando-os a este domínio de aplicação. Desta forma, a maioria deles procura unificar os processos já consolidados da Engenharia de Software aos processos de desenvolvimento da Engenharia de Produtos e aos padrões da Engenharia Elétrica tal como o IEC 61499-1 [6].

As metodologias apresentadas em [3, 4, 35, 31] utilizam os diagramas FB para diminuir a distância entre a especificação UML gerada, que está em um nível alto de abstração, e a linguagem utilizada pelos engenheiros de controle. No entanto, como diferencial, a metodologia [35, 31] define um conjunto de regras para realizar uma tradução automática dos modelos UML em diagramas de FB, com o objetivo de facilitar o trabalho e reduzir a possibilidade de erros. Estas regras contemplam a extração de informações dos diagramas de interação e do diagrama de classes. O processo descrito em [15] destaca-se por utilizar processos já consolidados da Engenharia de Software, a exemplo do RUP.

Um ponto de destaque apresentado pela metodologia [3, 4], em relação aos demais trabalhos analisados, é a preocupação com a verificação formal dos objetos do modelo

mecatrônico. A metodologia apresenta um mecanismo de tradução de modelos em um formato que pode ser entendido por uma ferramenta de verificação formal automática.

Considerações sobre a especificação e tratamento dos requisitos não funcionais, tais como os temporais foram abordados apenas em [19] com a utilização da *Object Constraint Language*, não sendo mencionados pelas demais metodologias, embora este tipo de tratamento possa ser mapeado com a ajuda dos diagramas de *statecharts* utilizados em [3].

Nas metodologias descritas, apenas a metodologia [12], que utiliza a engenharia simultânea, faz referência a formas de tratamento de falhas em sistemas mecatrônicos, tais como detecção, prevenção, tolerância, entre outras.

Um ponto importante é que todas estas metodologias enfatizam a modelagem do elemento Controle do produto. Apenas a metodologia [33] se preocupa com um modelo que conceitua todo o produto e, portanto, permite aproveitar o potencial da equipe multidisciplinar também na modelagem dos demais componentes.

A tabela 4 apresenta um resumo das metodologias descritas nesta seção, analisando aspectos importantes para o desenvolvimento de produtos mecatrônicos, tais como o método utilizado como base para especificação, os padrões adotados, entre outros.

Tabela 4 – Características de alguns trabalhos relacionados

Aspecto observado	**Metodologia de Bonfe**	**Metodologia CORFU**	**Metodologia de MROZEK**
Métodos Processos utilizados	Orientação a objetos Análise estruturada Function blocks	Orientação a objetos Function blocks	Orientação a objetos
Padrões	UML IEC	UML IEC	UML
Validação	Não	Testes	Simulação
Verificação formal	SIM	Sugere	Não
Tratamento de falhas	NÃO	NÃO	NÃO
Aspectos temporais	NÃO	NÃO	SIM
Foco	Controle	Controle	Controle

Como pode ser observado, as três metodologias comparadas acima utilizam método de análise orientada a objetos como principal técnica para especificação do produto e a linguagem UML como padrão de modelagem. Além destas, pode-se observar que características importantes para sistemas mecatrônicos nem sempre são abordadas por estas metodologias.

4. METODOLOGIA UNIFICADA PARA DESENVOLVIMENTO DE PRODUTOS MECATRÔNICOS - MdpM

O ciclo de vida de um produto é composto basicamente de três etapas conforme ilustrado na Figura 10. Na primeira etapa, chamada de pré-desenvolvimento, é realizado um estudo de viabilidade para o desenvolvimento do produto, entre outras tarefas. A segunda etapa consiste no desenvolvimento do produto e engloba o projeto do produto, o projeto do processo de fabricação, a preparação e o lançamento do produto. A última etapa corresponde ao pós-desenvolvimento que acontece depois que o produto já está pronto e disponibilizado ao cliente e consiste no seu acompanhamento e posterior descarte [27].

Figura 10 - Ciclo de vida de um produto

A MdpM é uma metodologia de desenvolvimento de produtos especificada para atender às necessidades de construção de produtos mecatrônicos. Considerando o ciclo de vida apresentado na Figura 10, está localizada na etapa de desenvolvimento do produto, mais especificamente no projeto do produto. A metodologia está organizada em fases que abrangem desde o levantamento dos requisitos até a construção de um modelo do produto pronto para ser enviado à linha de produção.

Um produto mecatrônico é aquele que reúne características de diversas áreas tecnológicas diferentes tais como a Elétrica, a Mecânica e a Computação. Esta integração

permite construções mais robustas que aproveitem os recursos de todas estas áreas. No entanto, envolve um desenvolvimento mais complexo que precisa de um tratamento diferenciado em relação aos métodos e técnicas utilizadas pelo desenvolvimento de produtos em uma área específica.

A complexidade do desenvolvimento de um produto mecatrônico pode ser visualizada já na composição da equipe de trabalho, que é multidisciplinar. A diversidade de perfis característica desta equipe permite que sejam criadas soluções inovadoras e criativas oriundas da união dos conhecimentos. No entanto, vem acompanhada de problemas tais como a comunicação entre os membros e a escolha das técnicas de desenvolvimento apropriadas para esta categoria de produto.

Em uma equipe multidisciplinar, a tarefa de entendimento das necessidades do cliente para mapeamento dos requisitos do projeto, por exemplo, deve ser realizada com o apoio de uma linguagem de modelagem única e simples para que possa ser facilmente compreendida por todos. Esta linguagem além de simples, precisa ser expressiva o suficiente para fornecer subsídios que permitam representar adequadamente os elementos de um produto mecatrônico.

O ambiente multidisciplinar também requer atenção especial na diversidade de técnicas e métodos disponíveis para os desenvolvedores, oriundos das diversas áreas envolvidas. Este acervo é importante, pois se tratam de técnicas e métodos já conhecidos e fundamentados nas suas áreas de domínio. No entanto, é importante que estes sejam analisados e adaptados para a Mecatrônica antes de serem utilizados, para que se tenha uma uniformidade dentro da equipe.

A metodologia MdpM é centrada no trabalho realizado por uma equipe multidisciplinar e define que esta deverá atuar conjuntamente desde o início das especificações até a construção do modelo do produto para ser enviado à linha de produção. O trabalho da equipe é utilizar as técnicas descritas na metodologia para construir um modelo puramente conceitual do produto. O objetivo é extrair o máximo de contribuição de cada integrante para entender o produto e modelar uma solução que seja independente de tecnologia. Com este propósito, definições de implementação são adiadas para estágios mais avançados de especificação, onde a compreensão do produto já esteja suficientemente amadurecida.

Produtos mecatrônicos requerem o mapeamento de características nem sempre comuns aos produtos convencionais. Estas características são necessárias devido à freqüente interação destes produtos com o ambiente e a conseqüente criticidade que os envolve. Por exemplo, em um sistema de navegação, um robô capta imagens do ambiente onde está inserido e as utiliza para traçar um caminho de locomoção neste ambiente. Para este exemplo, mapear características temporais é indispensável para o bom funcionamento do robô, pois ele precisa receber as imagens, processá-las e se locomover em um tempo suficiente para não causar acidentes. A MdpM também indica, ao longo do seu processo de desenvolvimento, os pontos onde devem ser observadas estas característias especiais.

O tratamento para garantir confiabilidade está presente na metodologia MdpM através de procedimentos de validações e verificações, além de orientações para a necessidade da detecção e tratamento de falhas.

As seções 4.1 e 4.2 descrevem, respectivamente, quais as técnicas utilizadas para compor a metodologia MdpM e qual a sua estrutura.

4.1. CARACTERÍSTICAS TÉCNICAS

A MdpM utiliza como base o processo *Rational Unified Process* (RUP) [14] de desenvolvimento de software. O RUP foi escolhido por apresentar um processo bem definido que atende a todos os estágios do desenvolvimento, desde a especificação de requisitos até a entrega do sistema. Além disso, possui algumas características que se adaptam muito bem às necessidades da Mecatrônica, apresentadas a seguir.

O RUP utiliza a tecnologia OO e de componentes e usa a linguagem UML para construção de seus modelos. Permite a estruturação do produto em partes que agregam funcionalidades, facilidade de montagem e manutenção, reutilização, entre outros aspectos. O processo de desenvolvimento é subdividido em interações, o que permite que a equipe de desenvolvimento possa gradualmente se aprofundar nos detalhes do produto. Assim, sempre que necessário, uma nova interação é iniciada para refinar a especificação desenvolvida.

O RUP implementa uma política de controle de qualidade baseada em pontos de checagem ao longo do processo que permite acompanhar as atividades desenvolvidas e corrigir possíveis erros de definições que venham a comprometer o projeto. Esta característica é essencial no ambiente multidisciplinar que envolve pessoas de culturas diferentes e com visões específicas sobre as necessidades do produto que será construído.

A MdpM utiliza a versão 2.0 da UML pois esta já incorpora recursos necessários à especificação de requisitos temporais, que serão abordados ao longo do processo.

A MdpM integra ao RUP técnicas oriundas do desenvolvimento de produtos mecânicos e de hardware. Estas técnicas foram inseridas à metodologia no formato original em que elas são aplicadas na sua área ou adaptadas a um modelo UML existente. Por exemplo, a MdpM utiliza o diagrama de casos de uso para levantamento de requisitos do produto, atividade presente no desenvolvimento de software e no desenvolvimento de produtos. No entanto, com o objetivo de oferecer uma visão mais ampla de análise destes requisitos, foi inserida na metodologia a utilização da matriz da casa da qualidade, ferramenta já utilizada com sucesso no ambiente de desenvolvimento de produtos mecânicos. As diversas análises possíveis de serem realizadas nesta matriz ajudam a equipe multidisciplinar na definição do escopo e na tomada de decisões importantes para o projeto. O RUP possui um artefato, a matriz de rastreabilidade, que permite rastrear um elemento do projeto com elementos correlatos, possibilitando um melhor gerenciamento de mudanças, impactos destas mudanças no projeto, entre outros aspectos. Nesta metodologia preferiu-se adotar a matriz da qualidade, pois além de fornecer análises entre os elementos existente, permite que as definições do produto em construção sejam comparadas com similares, gerando *benchmark* entre produtos. A matriz da qualidade também é uma ferramenta largamente utilizada na Engenharia de Produtos.

A MdpM sugere o uso de métodos que possam facilitar ainda mais a compreensão dos modelos pelos engenheiros participantes do projeto, a exemplo do diagrama de function blocks (FB). Estes diagramas podem ser automaticamente gerados por ferramentas automatizadas.

Assim como o RUP, a MdpM está especificada no metamodelo *Software Process Engeneering Metamodel* (SPEM).

4.2. ESTRUTURA BÁSICA

A estrutura básica da MdpM está representada na Figura 11. O processo tem início a partir de demandas do mundo real expressas sob a forma de necessidades do cliente, custos, normas, restrições, entre outras. Estas informações servem de subsídios para o processo de desenvolvimento, composto por um conjunto de passos descritos ao longo da metodologia. Cada um destes passos pode gerar artefatos que servirão como documentação a ser anexada ao produto. Ao final do processo é gerado um *modelo do produto* para ser

entregue à linha de produção. Para a MdpM um modelo do produto é prototipado a partir da integração de componentes especificados durante o projeto do produto.

O processo descrito na metodologia tem início com o entendimento preliminar das necessidades do produto que será construído a partir da fase Concepção, conforme apresentado na Figura 11. A partir das necessidades do cliente a equipe deve conceituar o produto, definir requisitos, funcionais e não funcionais, identificar os fatores de risco, entre outros aspectos importantes para o entendimento pleno do produto que será construído. Este trabalho de especificação de requisitos é gradual e depende integralmente das informações fornecidas pelo mundo real. Também durante o entendimento podem ser realizadas análises comparativas com produtos similares existentes no mercado, analisados aspectos tais como a relação entre os requisitos e os conflitos que possam existir entre eles. Todo este trabalho tem como resultado a definição do *escopo* do produto que será efetivamente construído, ou seja, a identificação, dentre os requisitos levantados, dos que efetivamente vão ser projetados para o produto.

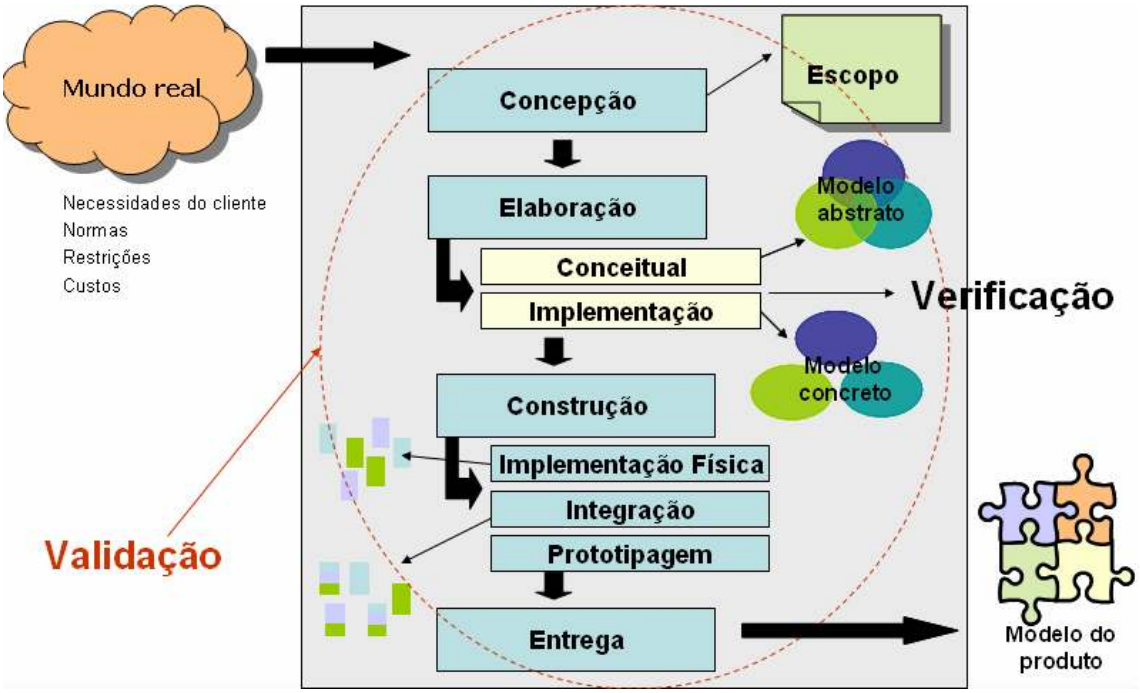


Figura 11 - Estrutura da MdpM

Os requisitos que compõem o escopo do produto são a base para iniciar a fase Elaboração. Esta fase compreende a especificação conceitual, responsável por definições estruturais e comportamentais do produto, e a especificação da implementação, que contempla as definições de soluções técnicas para a construção do produto.

A especificação conceitual se inicia com o detalhamento dos requisitos e a definição da arquitetura do produto, a partir da modelagem das classes que o compõem. Em seguida, é modelado o comportamento do produto a partir da identificação de aspectos dinâmicos. Desta forma, especifica-se como será o funcionamento do produto dentro do modelo estrutural projetado. Este comportamento é definido a partir dos diagramas de interação e do diagrama de estados. Os modelos estruturais e comportamentais representam o alicerce para a construção do artefato *modelo abstrato* do produto, representado na UML pelo diagrama de componentes, e na Figura 11 por círculos que se cruzam. Esta representação indica que os componentes ainda estão definidos em um nível de abstração conceitual, sem qualquer definição de como serão implementados, ou seja, se cada círculo representasse uma área específica e os componentes estivessem inseridos nestes círculos, seria possível observar a existência de componentes mecânicos, eletrônicos, computacionais e híbridos, que pertencem a mais de uma destas áreas. A MdpM busca adiar ao máximo as decisões de implementação, com o objetivo de aproveitar o potencial da equipe multidisciplinar na concepção de um modelo integrado para o produto. Por este motivo, pode-se notar que até este momento decisões de implementação ainda não foram tomadas. É importante destacar, no entanto, que verificações formais de propriedades do produto já podem ser realizadas sob o modelo comportamental definido.

Uma vez concebido o modelo abstrato, com as definições conceituais necessárias, inicia-se a especificação de implementação, que é responsável pela migração deste modelo abstrato para um *modelo concreto.* Este modelo define a solução tecnológica adequada para à construção de cada componente, ou seja, indica qual será a área responsável pela construção de cada um deles. Este é um dos pontos chaves do processo de especificação e, portanto, a MdpM define técnicas importantes para auxiliar a tomada de decisões pela equipe multidisciplinar. O processo de definição da tecnologia é realizado com o auxílio da matriz morfológica, método que permite mapear os princípios de solução possíveis para cada um dos componentes do modelo abstrato. É possível que sejam identificadas soluções que permitam construir componentes totalmente mecânicos, totalmente eletrônicos, totalmente computacionais ou híbridos. Na MdpM, um componente híbrido deve ser particionado em subcomponentes de forma que se chegue a uma granularidade onde cada subcomponente seja desenvolvido por apenas uma área de domínio. Para isso, primeiramente é definido, para cada componente ou subcomponente do modelo abstrato, se ele será comprado, reaproveitado ou desenvolvido. O segundo passo é identificar, para os que serão desenvolvidos, quais as

possíveis soluções de construção. Cada componente ou subcomponente poderá ter várias soluções possíveis e todas elas devem ser representadas na matriz morfológica. Para soluções híbridas, podem ser utilizados métodos de *co-design* que permitam definir a melhor solução de particionamento para o desenvolvimento. Com o auxílio da matriz morfológica, é possível selecionar também concepções de projeto que serão efetivamente construídas. Como resultado da identificação de soluções é gerado o *modelo concreto*. A Figura 11 representa este modelo também utilizando círculos, conforme o modelo abstrato apresentado anteriormente, no entanto os círculos não mais se sobrepõem, indicando que os componentes ou subcomponentes já foram completamente dissociados e agora pertencem a uma área específica.

Embora a metodologia utilize uma linguagem visual que seja de fácil entendimento, é possível também neste momento fazer um mapeamento dos modelos gerados em UML para *function blocks* visando facilitar a implementação por parte dos engenheiros. A MdpM adota a UML 2.0, que já incorpora conceitos tais como porta e interface, que permitem dissociar aspectos de comunicação de aspectos computacionais facilitando o mapeamento dos FBs. Alem disso, o uso da UML permite efetuar o mapeamento automático, a partir das regras e ferramentas apresentadas em [35].

A fase de construção é onde efetivamente os componentes ou subcomponentes definidos serão implementados fisicamente por cada uma das áreas, integrados e o produto prototipado.

A Implementação física pode envolver um novo processo, com o uso de uma metodologia arbitrária adotada pela área responsável pelo desenvolvimento. A MdpM não interfere nesta construção, apenas orienta que, para evitar problemas na etapa de integração, as definições das interfaces devem ser respeitadas. Neste cenário, eventualmente, alguma construção pode necessitar de novas definições ou redefinições da equipe multidisciplinar, demandando uma revisão das especificações genéricas do produto. Ao final da construção tem-se um conjunto de componentes ou subcomponentes prontos para formar o produto. A Figura 11 representa cada componente ou subcomponente com uma só cor, indicando que eles pertencem a uma única área.

A Integração é responsável pela montagem dos componentes ou híbridos. Cada uma de suas partes, subcomponentes, foi implementada por uma área. Portanto o componente precisa ser montado e testado antes de iniciar a construção do produto. Na Figura 11, após a integração, existem componentes que possuem mais de uma cor, indicando que são híbridos.

O *protótipo* do produto é formado pela composição dos seus componentes. Ele será submetido a validações e resultará no modelo do produto a será enviado à produção.

Pode-se notar que a metodologia prevê uma etapa de verificação formal que pode ser aplicada em algumas partes do processo a partir dos modelos UML. Por exemplo, existem abordagens que migram os modelos UML para verificadores formais possibilitando analisar se os modelos estão de acordo com os requisitos identificados no início do processo [3]. A verificação é importante principalmente para os sistemas críticos e deve ser utilizada sempre que a equipe multidisciplinar julgar necessário.

A validação do produto é um ponto de destaque na metodologia e pode ser observada ao longo de todo o processo, pois visa permitir uma maior qualidade ao produto que será construído.

A validação pode ser aplicada ainda em estágios iniciais da especificação, com a definição de critérios de aceitação que serão avaliados no final do processo. No decorrer de todo o processo, outras atividades de validação são realizadas, pois identificar problemas ainda em um nível de especificação evita construções erradas, auxiliando na gerência de prazos e custos. Desta forma são definidos, por exemplo, testes que serão submetidos aos componentes e ao protótipo do produto. Além disso, a MdpM define que, antes de iniciar a implementação física de cada componente definido no modelo concreto, deve-se avaliar a validade deste modelo. Esta avaliação é efetuada com o auxílio de ferramentas computacionais de simulação, a exemplo do MATLAB/SIMULINK [10]. Para efetuar a simulação, o modelo concreto é mapeado em um modelo de simulação, conforme a ferramenta computacional escolhida. Cada componente, por exemplo, pode ser representado por uma caixa preta que possui uma interface e executa um serviço específico. Em muitos casos é necessário programar a caixa preta para simular a execução de seus serviços. Estas caixas são conectadas de maneira que possam se comunicar, a partir das interfaces, e a simulação é realizada submetendo-se os casos de testes especificados para o produto. Deve-se considerar nesta simulação os requisitos funcionais e os não funcionais de maneira que seja possível identificar se o modelo projetado atende a suas especificações iniciais. É possível, por exemplo, injetar falhas em um componente para analisar o comportamento do produto nestes casos. A simulação proposta é um processo puramente virtual que visa encontrar, antes da construção física do componente, possíveis problemas de especificação.

A validação está presente também na construção dos subcomponentes que deverão compor um componente híbrido. Nestes casos, recomenda-se que seja efetuada uma

integração contínua destes subcomponentes, ou seja, sempre que os subcomponentes forem sendo construídos, eles são integrados e validados a partir dos testes especificados no início do processo. Esta integração contínua permite identificar precocemente erros de especificação, de construção ou de integração.

Uma vez construídos todos os componentes, é montado o protótipo do produto que também é validado de acordo com os casos de teste desenvolvidos na especificação.

Assim, ao final do processo, tem-se o protótipo testado do produto que é chamado pela MdpM de *modelo do produto*, pronto para ser enviado à linha de produção. Toda a documentação gerada durante a especificação realizada pela equipe multidisciplinar e a documentação gerada pelas equipes específicas durante a construção dos componentes é anexada ao modelo do produto. É importante enfatizar que a especificação do processo de produção e montagem do produto não faz parte do escopo desta metodologia.

5. PROCESSO DE DESENVOLVIMENTO MdpM

O processo de desenvolvimento especificado a seguir e apresentado na Figura 12 descreve a metodologia proposta, MdpM.

Para a metodologia MdpM foi especificada a seguinte estrutura:

- possui um ciclo de vida que consiste em uma seqüência de fases, definidas no tempo;
- cada fase pode conter uma ou mais iterações;
- possui um conjunto de disciplinas, onde cada uma delas reúne atividades afins que são desenvolvidas ao longo das fases do ciclo de vida.

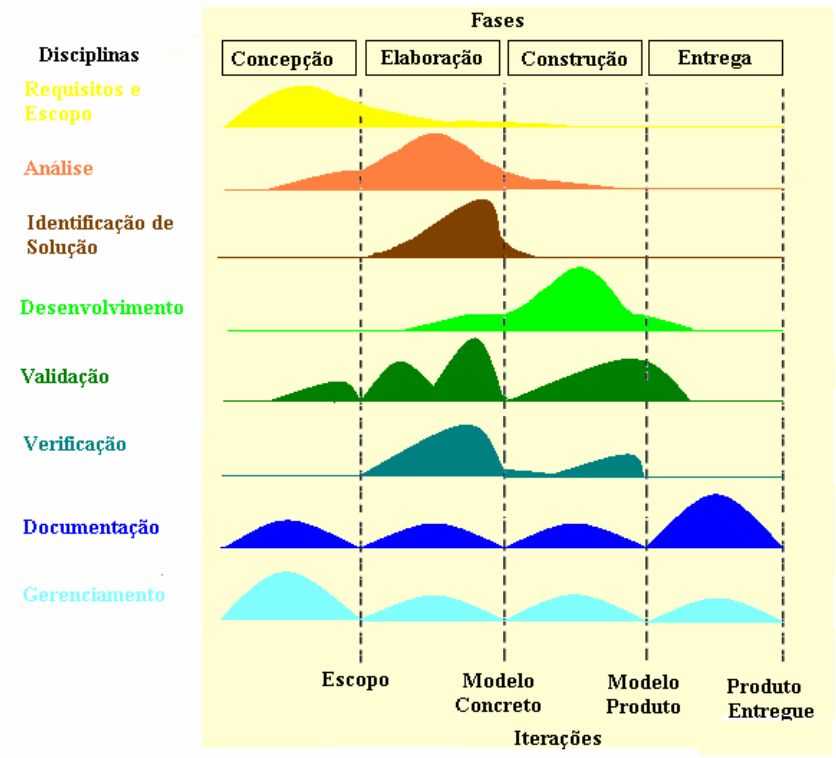


Figura 12 - Processo de desenvolvimento MdpM

Conforme ilustrado na Figura 12, o processo está organizado em duas dimensões. A primeira dimensão (eixo das abscissas) representa a estrutura dinâmica da metodologia e está dividida em quatro fases, Concepção, Elaboração, Construção e Entrega, onde cada uma destas fases pode ser subdividida em iterações. Assim, uma fase pode conter uma ou mais iterações a depender de características tais como o tamanho do produto que será construído, a dificuldade de entendimento ou o nível de risco que este ofereça. As iterações servem como refinamentos da especificação, até que as atividades de cada fase estejam completas.

Ao final de cada fase é importante verificar se os objetivos foram atingidos, antes de iniciar a fase seguinte, conforme especificado no controle de qualidade do RUP. Assim, a metodologia proposta define quatro pontos de controle (PC), denominados PC1 (Escopo), PC2 (Modelo Concreto), PC3 (Modelo do Produto) e PC4 (Produto Entregue) realizados ao final de cada uma das fases com critérios pré-estabelecidos para avaliação das atividades realizadas.

A segunda dimensão (eixo das ordenadas) é representada pela estrutura estática da metodologia, composta por oito disciplinas que agrupam atividades afins do processo de desenvolvimento do produto: Requisitos e Escopo; Análise; Identificação de solução; Desenvolvimento; Validação; Verificação; Documentação; e Gerência.

Observando a Figura 12 pode-se notar que a disciplina Análise tem mais expressividade durante a fase Elaboração, embora ela possa ser iniciada durante a Concepção e se estender até a Construção. Isso acontece porque embora o foco principal da Concepção seja o levantamento de requisitos, durante este levantamento alguns detalhes podem ser observados que levem a equipe a iniciar um trabalho preliminar de análise. Da mesma forma, na fase Construção pode haver a necessidade de se redefinir algum detalhe que somente neste momento foi visualizado.

A especificação da MdpM segue a estrutura básica utilizada pelo RUP para organização do processo em duas dimensões, conforme ilustrado na Figura 12. Embora esteja dividida em quatro fases e utilize os conceitos de disciplina, cada uma das fases e disciplinas foi totalmente redefinida para atender ao processo proposto.

Assim como no RUP, a MdpM também se preocupa com a definição dos papéis e responsabilidades de cada integrante da equipe participante do projeto. Assim, antes de iniciar um processo de especificação de um produto deve ser definida a equipe de trabalho.

As fases, papéis e disciplinas que compõem a MdpM serão detalhadas a seguir, nas seções 5.1 , 5.2 e 5.3 respectivamente.

5.1. FASES

Como destacado anteriormente, o ciclo de vida da MdpM está dividido em quatro fases denominadas Concepção, Elaboração, Construção e Entrega. Estas fases acontecem ao longo do tempo em seqüência (Figura 13). Isso significa que uma fase posterior só é iniciada quando a fase que a antecede estiver completamente finalizada e seu ponto de controle correspondente tiver sido analisado e aprovado por toda a equipe. Por exemplo, ao final da fase Elaboração espera-se que um *modelo concreto* do sistema tenha sido gerado, então, a fase Desenvolvimento só pode ser iniciada se todas as definições que envolvem o ponto de controle PC2 estiverem finalizadas.

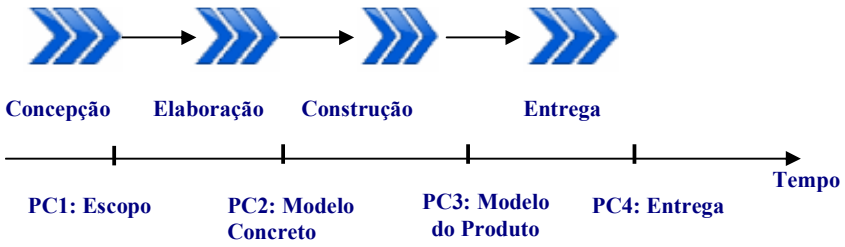


Figura 13 - Fases da MdpM

É válido mencionar que a utilização de pontos de controle não caracteriza a metodologia em um formato do tipo cascata. Os pontos de controle foram inseridos para facilitar o gerenciamento das atividades indicando itens que devem ser desenvolvidos antes de se iniciar uma nova fase. No entanto, o mecanismo definido pelas disciplinas que podem estar presentes em várias fases do ciclo de vida fornece a flexibilidade necessária ao processo.

5.1.1. Concepção

Concepção é a fase inicial de um projeto. O objetivo principal desta fase é o entendimento do produto a ser construído com o mapeamento dos requisitos e a definição do respectivo escopo. Também faz parte desta fase a avaliação dos possíveis riscos que possam ameaçar o sucesso do projeto e a definição dos critérios de aceitação que são utilizados para avaliar o modelo do produto construído.

Esta fase está centrada em reuniões com o cliente. Nestas reuniões, o cliente informa quais as suas necessidades para que a partir delas sejam mapeados os requisitos do produto. Estas reuniões de levantamento de requisitos devem ser sempre multidisciplinares, ou seja, devem ser realizadas por uma equipe composta por representantes de cada uma das áreas participantes do projeto. Mais detalhes sobre a formação desta equipe serão descritos na seção 5.2.

Uma vez mapeados todos os requisitos, é definido o escopo do produto. Durante a fase Concepção são desenvolvidas as atividades relativas à disciplina de Requisitos e Escopo, Análise, Validação, Gerência e Documentação.

5.1.2. Elaboração

A fase Elaboração tem como objetivo detalhar os requisitos que compõem o escopo do produto que será construído e definir a arquitetura do sistema. Esta fase também é realizada em reuniões que devem ser multidisciplinares. A partir dos requisitos identificados e detalhados, são mapeados aspectos específicos do produto tais como a estrutura estática e comportamental, o relacionamento entre os diversos objetos destas estruturas, a comunicação entre eles, dentre outros aspectos.

As decisões de implementação devem ser adiadas ao máximo na especificação do projeto. Estas decisões estão diretamente ligadas à arquitetura do sistema e devem ser definidas no final desta fase do projeto. O resultado desta fase é um modelo, chamado de *modelo concreto* do sistema, já simulado, com sua solução de implementação definida e particionado nos componentes de software, eletrônicos e mecânicos que serão construídos.

Durante a fase Elaboração são desenvolvidas as atividades relativas às disciplinas de Requisitos e Escopo, Análise, Identificação de Solução, Desenvolvimento, Validação, Verificação, Documentação e Gerência.

5.1.3. Construção

A fase Construção tem como entrada o modelo concreto gerado no Desenvolvimento e seu objetivo é construir os componentes projetados, visando montar um protótipo do produto que será exaustivamente testado antes que o projeto seja enviado para a linha de produção.

O produto final desta fase é um *modelo do produto* já testado e pronto para ser fabricado. Durante a fase Construção são desenvolvidas as atividades relativas às disciplinas de Requisitos e Escopo, Análise, Validação, Verificação, Identificação de Solução, Desenvolvimento, Documentação e Gerência.

5.1.4. Entrega

A Entrega é a fase responsável pela liberação do modelo do produto para fabricação. Toda a documentação complementar relativa ao processo de fabricação deve ser gerada para ser entregue junto com o projeto piloto. Embora a metodologia não aborde o planejamento do processo de produção, cada área deve fornecer a documentação necessária para a fabricação dos seus elementos. Durante a fase Entrega são desenvolvidas as atividades relativas às disciplinas Desenvolvimento, Validação, Documentação e Gerenciamento. Especificamente a Documentação tem um papel estratégico nesta fase, pois é neste momento que todos os artefatos gerados ao longo do processo são reunidos para compor a documentação final do produto. Esta documentação serve de base para outras atividades que não fazem parte desta metodologia, tais como a especificação do processo de produção.

5.2. PAPÉIS

O gerenciamento do processo de desenvolvimento é um ponto importante, pois envolve controlar as atividades de um grupo de pessoas que trabalham juntas colaborando para um resultado único. Isso se torna mais necessário quando considerado o ambiente multidisciplinar da Mecatrônica, onde se tem um grupo de pessoas oriundas de formações diferentes e que tendem a enxergar o produto sob diferentes aspectos. Além disso, estão acostumadas a trabalhar com técnicas específicas de suas áreas e, portanto, podem ser levadas a querer adotar uma solução que lhes seja mais familiar.

É importante que cada integrante possa contribuir positivamente com os seus conhecimentos, mas esta contribuição envolve pensar no produto como um elemento único que será construído e que deve ser projetado aproveitando o que melhor existe em cada área. Este é o ganho real de se ter uma equipe multidisciplinar. No entanto, com tantos pensamentos diversificados, é muito importante que a metodologia conduza os trabalhos de maneira que consiga nortear as atividades para que elas sejam realmente desenvolvidas em conjunto, em prol de um objetivo comum.

Para atender a esta necessidade de gerenciamento, a MdpM utiliza o conceito de papéis. Cada participante da equipe de desenvolvimento possui um papel bem definido com responsabilidades a cumprir. Assim, o primeiro passo quando o projeto se inicia é a criação de uma equipe responsável pelo trabalho, onde é definindo de forma clara quem são os integrantes e qual o papel desempenhado por cada um deles dentro do processo.

Como pode ser observado na Tabela 5, a metodologia MdpM possui vários papéis.

Tabela 5 - Papéis da MdpM

Papel	Responsabilidade
 Cliente	O cliente representa a conexão entre os desenvolvedores e o mundo real. Suas responsabilidades são: definir as necessidades e características que o produto deve conter; identificar a importância de cada uma destas necessidades; definir o escopo do produto; e definir critérios de aceitação. Desta forma, tem papel fundamental nas definições conceituais do produto.
 Gerente	O gerente do projeto é o principal responsável pelo projeto no perfil técnico. Ele terá a responsabilidade de decidir as prioridades, avaliar riscos, acompanhar e coordenar o andamento das atividades junto aos líderes. O gerente deve estar presente nas reuniões multidisciplinares. Assim, deve ser uma pessoa de conhecimento geral sobre todas as áreas participantes.
 Líder	O projeto deve conter um líder para cada área de conhecimento envolvida. O líder deve participar ativamente de todo o processo, estando presente em todas as reuniões (multidiciplinares e da sua área), pois ele é o responsável direto pelas decisões de projeto relativas à sua área de conhecimento.
 Especialista de Domínio	O especialista de domínio é representado por técnicos de áreas específicas que detém o conhecimento necessário para especificar o produto. Podem ser recrutados pelo seu líder sempre que necessário para definir particularidades da sua área.
 Desenvolvedor	Os responsáveis pelo desenvolvimento dos componentes nas respectivas áreas são os desenvolvedores. Eles receberão uma especificação detalhada e deverão construir o componente de acordo com esta especificação.
 Arquiteto	O projeto deve conter um arquiteto, pessoa que possua um conhecimento multidisciplinar e que será o mediador das áreas específicas no desenvolvimento da arquitetura do produto.
 Testador	A metodologia prevê dois tipos diferentes de testes: os testes de componentes, e os testes do produto. Os testadores são as pessoas responsáveis por executar estes testes de acordo com os casos de teste definidos. Pode haver testadores técnicos e testadores clientes.
 Engenheriro de Verificação	O engenheiro de verificação é responsável pelas atividades necessárias para realizar a verificação formal das partes críticas do produto.

Recomenda-se que a equipe formada para desenvolver o projeto de um produto mecatrônico deve ser composta por um representante do cliente, um gerente de projeto, vários líderes, um para cada área, e pode requisitar a participação de outros papéis quando necessário, a exemplo do especialista de domínio. Também deve contar em alguns momentos com testadores e desenvolvedores.

Devem ser realizadas reuniões no decorrer do desenvolvimento, que devem ser registradas em atas:

- reuniões multidisciplinares, com a participação do cliente, do gerente, arquiteto, de todos os líderes, um para cada área, e se necessário de outros papéis com perfis técnicos;
- reuniões específicas, com a participação do líder de uma área e seus especialistas, desenvolvedores e testadores, se necessário.

A Tabela 5 apresenta os papéis definidos pela MdpM e suas respectivas responsabilidades dentro da metodologia.

Dentre estes papéis apenas os representantes de cliente, gerente e líderes participam integralmente do processo de desenvolvimento, os demais são requisitados à medida que for necessária a sua presença na equipe.

5.3. DISCIPLINAS

Assim como na metodologia RUP, cada uma das fases definidas na MdpM contém disciplinas. Uma disciplina envolve um conjunto de atividades afins (Figura 14) que podem ou não ser desenvolvidas, a depender da necessidade do projeto. O produto gerado por estas atividades são os artefatos. Artefatos representam aspectos importantes do sistema como, por exemplo, os requisitos, a estrutura estática das classes que o compõe, o comportamento, as normas e restrições a serem seguidas. Cada atividade pode ser desempenhada por um ou mais papéis definidos pela metodologia.

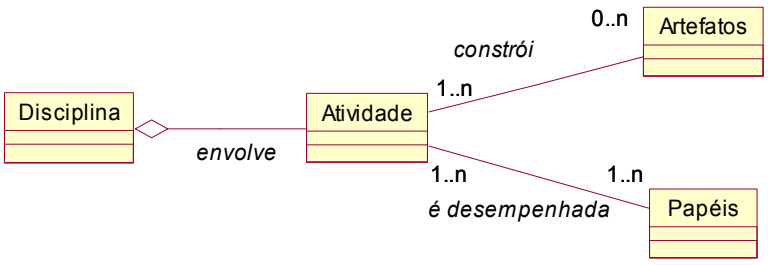


Figura 14 - Estrutura das disciplinas de acordo com a MdpM

Para o domínio das aplicações mecatrônicas propõe-se o uso de oito disciplinas, conforme apresentado no diagrama de pacotes da Figura 15.

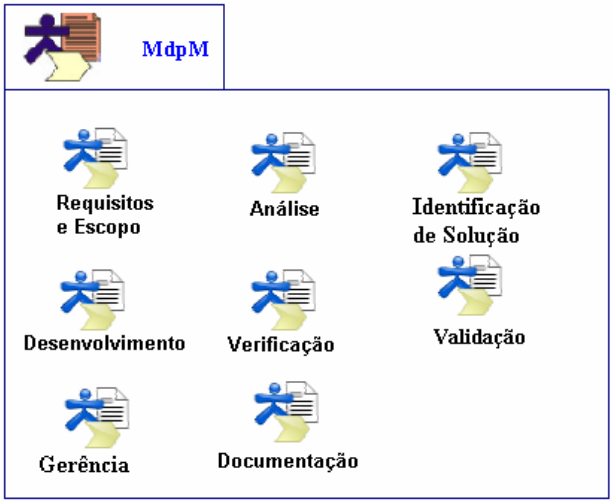


Figura 15 - Diagrama de pacotes com Disciplinas da MdpM

As disciplinas identificadas na MdpM estão descritas nas seções a seguir.

5.3.1. Requisitos e Escopo

O objetivo da disciplina Requisitos e Escopo é identificar as necessidades do cliente para o produto a ser construído e a partir destas necessidades mapear os requisitos e definir o escopo do produto.

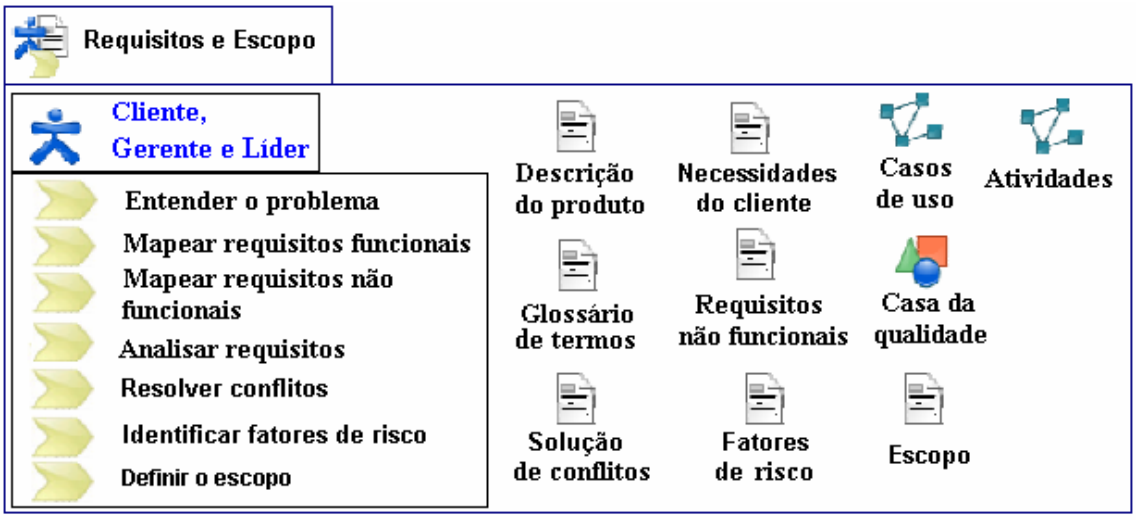


Figura 16 - Diagrama de pacotes com a disciplina Requisitos e Escopo

O diagrama de pacotes ilustrado na Figura 16 agrupa os elementos que compõem a disciplina Requisitos e Escopo. Pode-se observar que a disciplina é desenvolvida pelo gerente do projeto e pelos líderes das áreas específicas, sempre com a ajuda do representante do cliente. Isso acontece porque as atividades desempenhadas são, na maioria, realizadas em reuniões de entendimento do produto. Estas reuniões representam o subsídio necessário para que a equipe possa identificar os principais requisitos para definir o escopo.

De acordo com a Figura 16, nota-se que são realizadas sete atividades ao longo da disciplina: Entender o problema; Mapear requisitos funcionais; Mapear requisitos não funcionais; Analisar requisitos; Resolver conflitos; Identificar fatores de risco; e Definir o escopo. Artefatos também são gerados como resultado destas atividades e estão apresentados na figura como ícones, tais como, a descrição do produto, o diagrama de casos de uso, o diagrama de atividades, entre outros.

A Figura 17 apresenta o fluxo das atividades desenvolvidas na disciplina Requisitos e Escopo. O trabalho se inicia com o entendimento do problema onde é gerado um documento com a descrição do produto e um documento com a lista de necessidades apontadas pelo cliente. Em seguida é realizado o mapeamento dos requisitos funcionais e não funcionais. Estes requisitos são então validados pelo cliente. Se houver alguma inconsistência, o processo se repete até que o entendimento esteja de acordo com o solicitado. A próxima atividade então consiste em analisar estes requisitos. A análise envolve a construção da matriz da casa da qualidade, onde são observados aspectos tais como a definição do nível de importância destes requisitos para o cliente, a comparação do produto com outros similares no mercado e a detecção de requisitos conflitantes. Neste caso, propostas de solução para estes conflitos são elaboradas.

A identificação de fatores que possam oferecer risco ao projeto também é uma atividade importante no processo, pois com isso é possível detectar problemas tais como a necessidade de ter pessoas com conhecimentos especiais na equipe, necessidades de treinamento, de aquisição de equipamentos, entre outros aspectos e definir estratégias para resolvê-los. A última atividade desenvolvida nesta disciplina é a definição do escopo do produto que será construído.

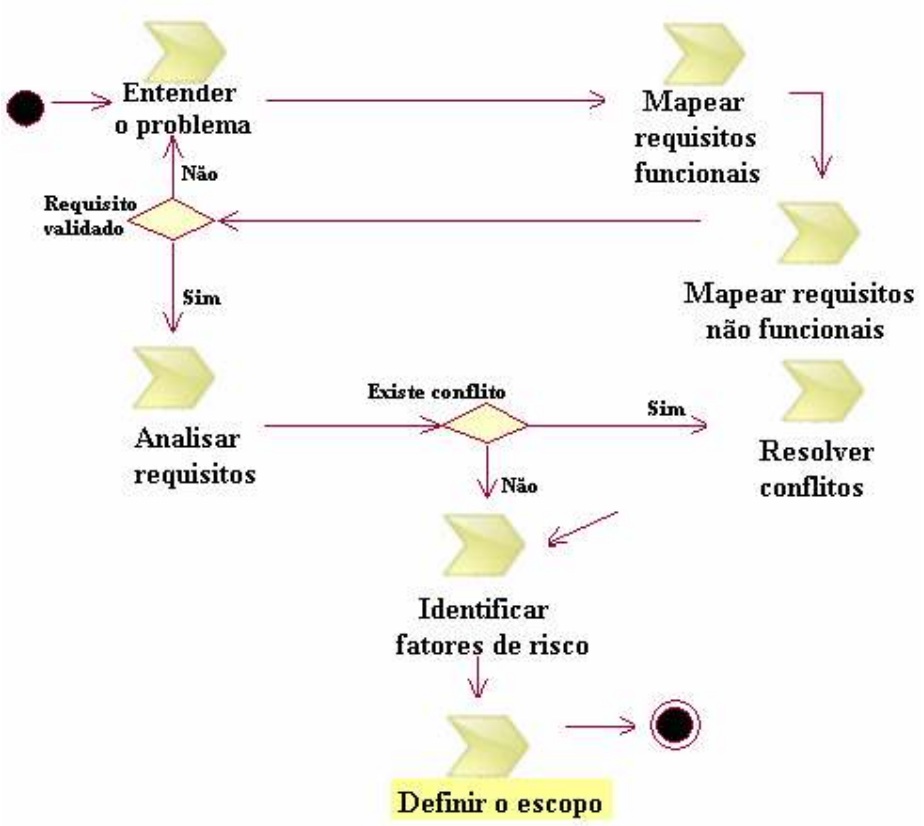


Figura 17 - Diagrama de atividades da disciplina Requisitos e Escopo

Para cada uma das atividades ilustradas na Figura 17, é definido um conjunto de passos que devem ser executados para atingir o objetivo da atividade. Assim, a metodologia propõe que se utilizem, para cada atividade, uma tabela que ilustra o objetivo da atividade, os passos que devem ser seguidos para atingir este objetivo, os artefatos por ela gerados, a principal fase onde a disciplina é desenvolvida e quais os papéis que desempenham esta atividade. Como exemplo, a Tabela 6 apresenta a atividade Mapear requisitos funcionais.

Entende-se por requisitos funcionais os requisitos do sistema que estão diretamente ligados ao seu funcionamento. Entende-se por requisitos não funcionais aqueles requisitos que são necessários para o bom funcionamento do sistema, mas que não dizem respeito à sua função objetivo, eles contribuem para que esta função seja executada de maneira satisfatória. Por exemplo, em um forno de microondas, pode-se citar como requisito funcional o cozimento uniforme do alimento e como requisito não funcional um limite de temperatura aceitável para aquecimento da carcaça do forno.

Para efeito de padronização, nesta metodologia, os requisitos temporais e os relativos à prevenção de falhas são tratados como requisitos não funcionais. Desta maneira, se

algum requisito funcional necessitar de um tratamento de falhas ou alguma especificação temporal, estas são adicionadas em um novo requisito na lista de requisitos não funcionais. No exemplo do forno de microondas citado anteriormente, pode-se mapear para o requisito funcional desligar forno, um requisito não funcional que indique as condições de tempo necessárias para que o forno interrompa suas operações antes que haja alguma alteração nas condições do alimento. Além destes, o desenvolvimento de sistemas mecatrônicos pode requerer o mapeamento de requisitos adicionais para atendimento a normas e padronizações, tais como as normas de uso e segurança. Estes requisitos também devem ser adicionados à lista de requisitos não funcionais.

Tabela 6 - Exemplo de uma atividade da disciplina Requisitos e Escopo

Atividade: Mapear requisitos funcionais
Objetivo: Identificar as necessidades do cliente e mapeá-las em requisitos funcionais.
Passos: • Realizar reuniões com o cliente; • Identificar as necessidades do cliente; • Mapear os requisitos funcionais; • Descrever cenários a partir da construção de diagramas de casos de uso com as visões necessárias do produto; • Elaborar diagramas de atividades para detalhar o funcionamento dos casos de uso mais complexos dentro do cenário; • Criar um glossário com os principais termos associados ao projeto; • Validar artefatos com o cliente.
Artefatos: Documento de necessidades do cliente, casos de uso, diagrama de atividades e glossário de termos.
Fase: Concepção.
Papéis: Cliente e líderes técnicos das áreas específicas. Eventualmente pode participar o gerente.
Observação: Caso de uso que expressem um requisito que deve ser tolerado a falhas, devem ser tratados como uma exceção e ilustrados como uma extensão do caso de uso normal, usando o estereótipo <<extends>>.

Conforme apresentado na Tabela 6 e ilustrado na Figura 18, a atividade Mapear requisitos funcionais que se inicia com o cliente informando suas necessidades. A partir destas necessidades desencadeiam-se as demais tarefas de identificação de requisitos, construção e detalhamento dos cenários e elaboração do glossário de termos necessário para esclarecer e nivelar o conhecimento dentro da equipe. Cada um destes passos pode gerar artefatos importantes como resultado dos trabalhos. Por exemplo, o mapeamento de requisitos gera como resultado o diagrama de casos de uso com os possíveis cenários de utilização. Podem-se detalhar estes requisitos construindo-se diagramas de atividades em UML que ilustrem passo a passo o funcionamento de um determinado requisito. Este diagrama permite também ilustrar atividades paralelas já em estágios iniciais de especificação. Pode-se gerar também um glossário de termos importantes que ajudem à equipe no entendimento do

problema, pois além de definir conceitos, padroniza e nivela o conhecimento da equipe. Note que estes três passos são executados em paralelo e se complementam. À medida que as iterações vão acontecendo, eles ganham níveis de abstração mais detalhados e a equipe adquire maior conhecimento sobre o problema.

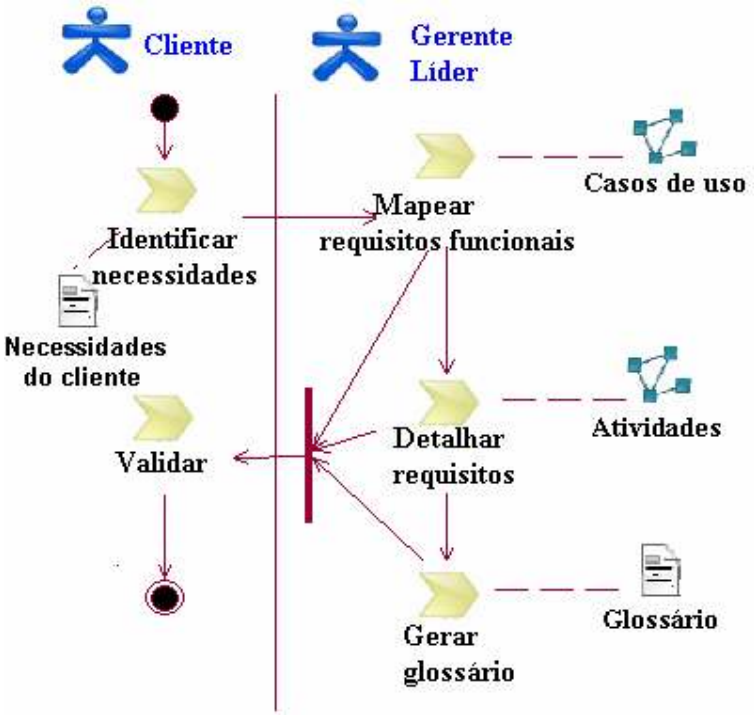


Figura 18 - Diagrama das atividades com os passos para mapear requisitos funcionais

As tabelas com as definições dos passos para as demais atividades que compõem a disciplina Requisitos e Escopo estão descritas no Apêndice B.

5.3.2. Análise

A disciplina Análise se baseia no escopo do produto definido na fase anterior. Seu objetivo é detalhar este escopo e iniciar a definição da arquitetura do produto. Para isso, cada um dos requisitos que compõem este escopo, seja ele funcional ou não funcional, é utilizado para construir a estrutura estática e definir o comportamento do produto.

O diagrama de pacotes ilustrado na Figura 19 agrupa os elementos que compõem a disciplina Análise. Pode-se observar que a disciplina é desenvolvida pelos líderes de cada área envolvida no projeto e tem em alguns momentos a participação do cliente. Isso acontece porque o líder detém um conhecimento sobre a área que lhe permite, além de participar das definições do projeto, recrutar especialistas de domínio em assuntos específicos. O cliente é

importante no esclarecimento das dúvidas que esta equipe possa ter. Além destes a participação do arquiteto é importante na definição integrada da arquitetura do produto.

A disciplina Análise é composta por seis atividades, listadas no diagrama de pacotes: Detalhar requisitos funcionais; Detalhar requisitos não funcionais; Montar o modelo estático; Montar a estrutura dinâmica; Modelar estados; e Construir o modelo abstrato. Os artefatos gerados por estas atividades também fazem parte do pacote e estão ilustrados na forma de ícones, a exemplo da descrição dos requisitos e do diagrama de classes.

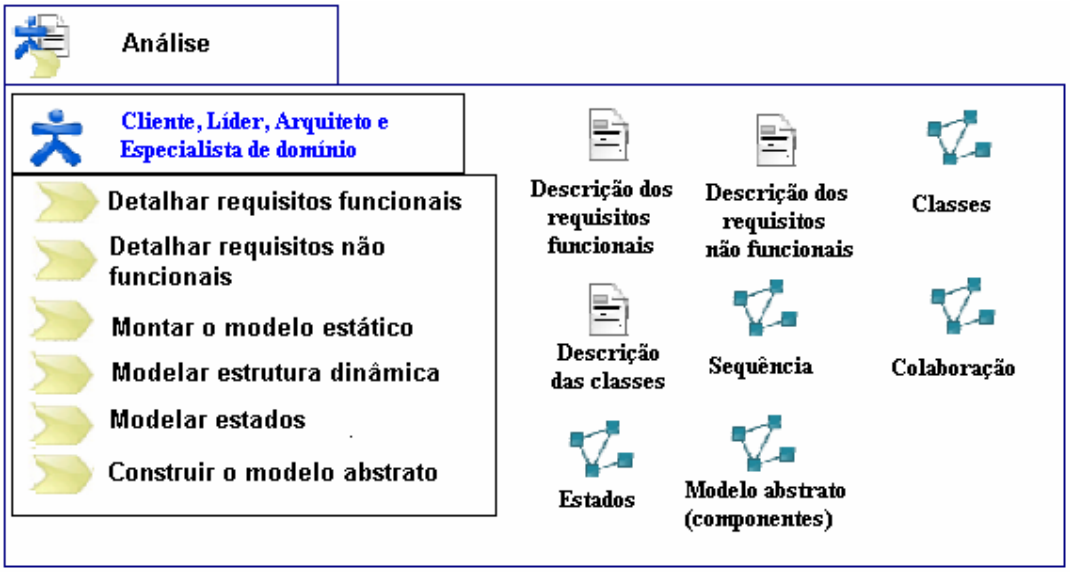


Figura 19 - Diagrama de pacotes da disciplina Análise

A Figura 20 apresenta o fluxo das atividades desenvolvidas na Análise. A atividade inicial consiste no detalhamento dos requisitos tanto funcionais quanto não funcionais. Este detalhamento é importante para especificar como cada um dos requisitos deve se comportar quando selecionado. Além disso, permite aprofundar ainda mais o nível de abstração da especificação. Após o entendimento completo dos requisitos, as próximas atividades são as de modelagem da estrutura estática e da dinâmica, para que possa ser gerado um modelo abstrato do produto. Este modelo contemplará todos os componentes que farão parte do produto e como será a comunicação entre eles, mas não inclui a definição da tecnologia que será utilizada para implementá-los.

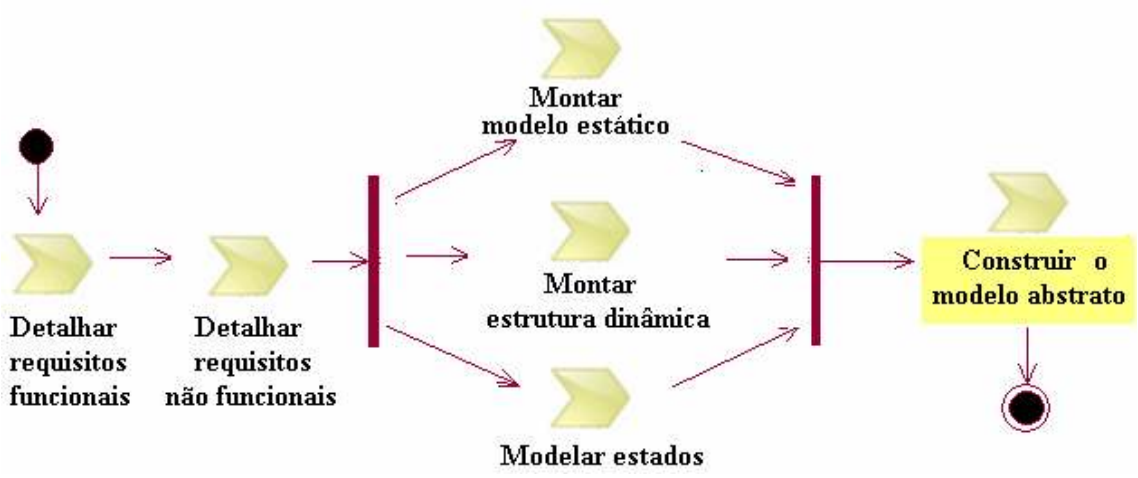


Figura 20 - Diagrama de atividades da disciplina Análise

Similarmente à disciplina anterior, a Tabela 7 ilustra os passos realizados para executar a atividade Detalhar requisitos funcionais, bem como o seu objetivo, artefatos gerados, papéis responsáveis pela sua execução, a principal fase onde ela é desenvolvida e observações pertinentes.

Tabela 7 – Exemplo de uma atividade da disciplina Análise

Atividade: Detalhar requisitos funcionais
Objetivo: Produzir uma descrição detalhada de cada requisito funcional que compõe o escopo do produto.

Passos:
• Realizar reuniões com o cliente;
• Revisar os casos de uso especificados durante a disciplina de Requisitos e Escopo;
• Elaborar documento de Descrição dos requisitos funcionais de acordo com o detalhamento fornecido pelo cliente. Este documento deverá conter para cada caso de uso definido as seguintes características:
  ○ quem interage com o caso de uso (atores);
  ○ quais as pré-condições para que o caso de uso seja executado;
  ○ como será o processamento normal executado pelo caso de uso;
  ○ exceções ao processamento normal e;
  ○ pós-condições de processamento.
• Validar a Descrição dos requisitos com o cliente.

Artefatos: Documento de descrição dos requisitos funcionais.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas e cliente. Pode ser solicitada a presença de especialistas de domínio.
Observação: Vide modelo do documento de descrição dos requisitos funcionais no experimento prático.

A Figura 21 apresenta o fluxo dos trabalhos executados na atividade Detalhamento de requisitos funcionais. Nota-se que o fluxo está dividido em duas áreas indicando que a atividade tem parte técnica, desenvolvida pelos líderes e pelos especialistas, e parte relacionada a definições, que são desempenhadas pelo cliente.

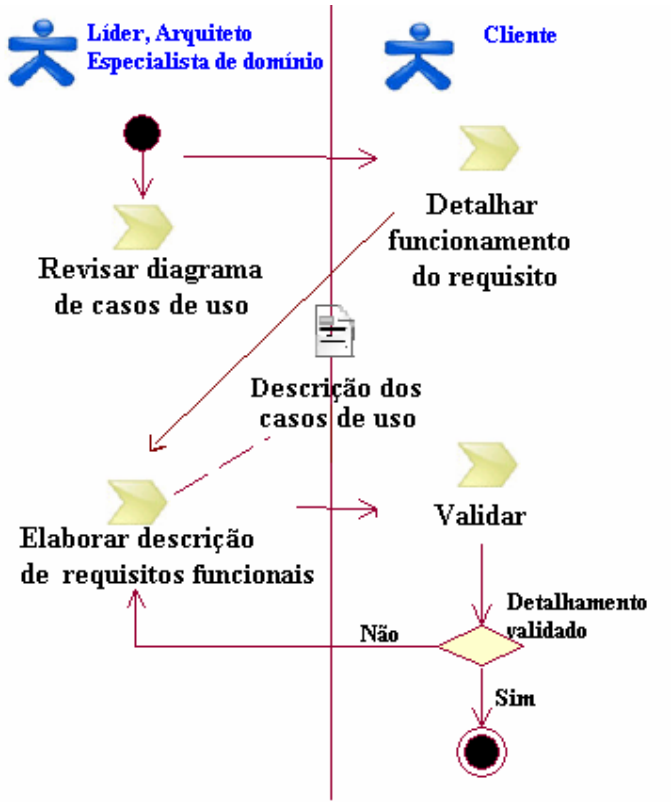


Figura 21 - Fluxo da atividade Detalhar Requisitos Funcionais

Como se pode observar inicialmente os líderes, arquiteto e especialistas fazem uma revisão nos casos de uso previamente definidos e solicitam que o cliente detalhe o funcionamento do requisito correspondente. É a partir deste detalhamento que é elaborado o documento de descrição dos casos de uso com o entendimento dos líderes sobre o que foi detalhado pelo cliente. O detalhamento dos casos de uso engloba a descrição do processamento, a identificação dos atores, pré-condições, pós-condições e o tratamento das exceções que podem ocorrer ao curso normal de processamento. Estas exceções terão que ser tratadas para não significarem falhas no produto. O experimento prático ilustrado no capítulo 6 apresenta um modelo de documento para o artefato descrição dos casos de uso. Finalmente, este documento é submetido à validação do cliente e pode ser reescrito caso tenha erros de interpretação. A participação conjunta dos líderes tem o objetivo de aproveitar a experiência e o conhecimento de uma equipe tão diversificada e tornar a especificação o mais independente possível da tecnologia.

As demais atividades da disciplina de análise estão descritas no Apêndice B.

5.3.3. Identificação de Solução

A disciplina Identificação de solução faz parte da definição da arquitetura do produto. Seu objetivo é definir como serão desenvolvidos os componentes ou subcomponentes do modelo abstrato, ou seja, qual será a área responsável pela sua construção: Eletrônica, Mecânica ou Computação.

Esta disciplina deve ser realizada em reuniões multidisciplinares entre líderes de cada área envolvida e o arquiteto, pois é importante a participação de todos nas definições dos caminhos de implementação a serem seguidos.

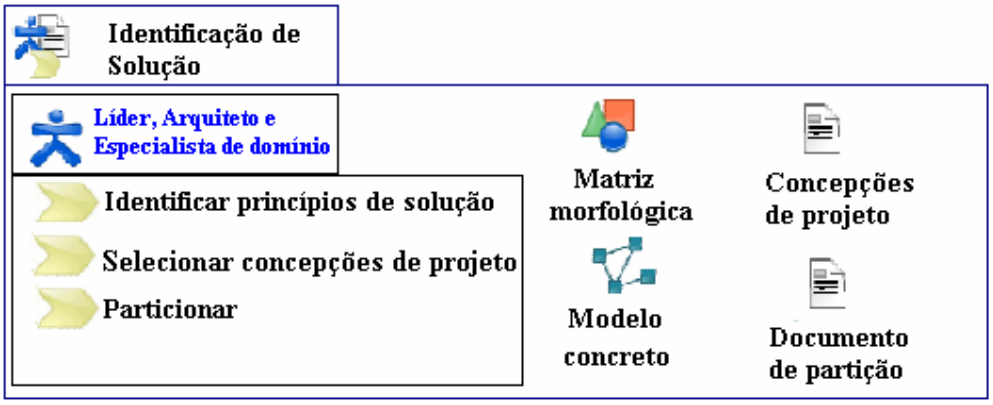


Figura 22 - Diagrama de pacotes da disciplina Identificação de Solução

O diagrama de pacotes ilustrado na Figura 22 agrupa os elementos que compõem a disciplina. São sugeridas três atividades a serem desenvolvidas: a identificação dos princípios de solução; a seleção de concepções de projeto; e o particionamento. Estas atividades geram artefatos, representados na forma de ícone no diagrama de pacotes, tais como a matriz morfológica e o modelo concreto.

O fluxo das atividades desenvolvidas é ilustrado no diagrama de atividades da Figura 23. Observa-se que as atividades Definir testes e Simular não fazem parte desta disciplina, elas estão inseridas na disciplina Validação, no entanto, foram representadas no diagrama por ter uma relação direta com a atividade Particionar.

Assim, a atividade inicial consiste em identificar os princípios de solução possíveis para cada componente do modelo abstrato. Esta atividade gera uma matriz morfológica onde são representados para cada componente os vários princípios de solução possíveis. Com base nestes princípios, são geradas algumas concepções de projeto que continuam no processo de desenvolvimento do produto. Estas concepções representam

soluções tecnológicas projetadas para cada componente do modelo abstrato do produto. No entanto, existem casos em que são necessárias análises adicionais para definir qual a melhor solução de implementação. Nestes casos usa-se a atividade Particionar. Nela é escolhida, para os componentes que podem ser desenvolvidos tanto em hardware quanto em software, qual a melhor solução de implementação a ser adotada. Em caso de componentes híbridos, a partir deste ponto, eles serão visualizados através de subcomponentes pertencentes a uma única área.

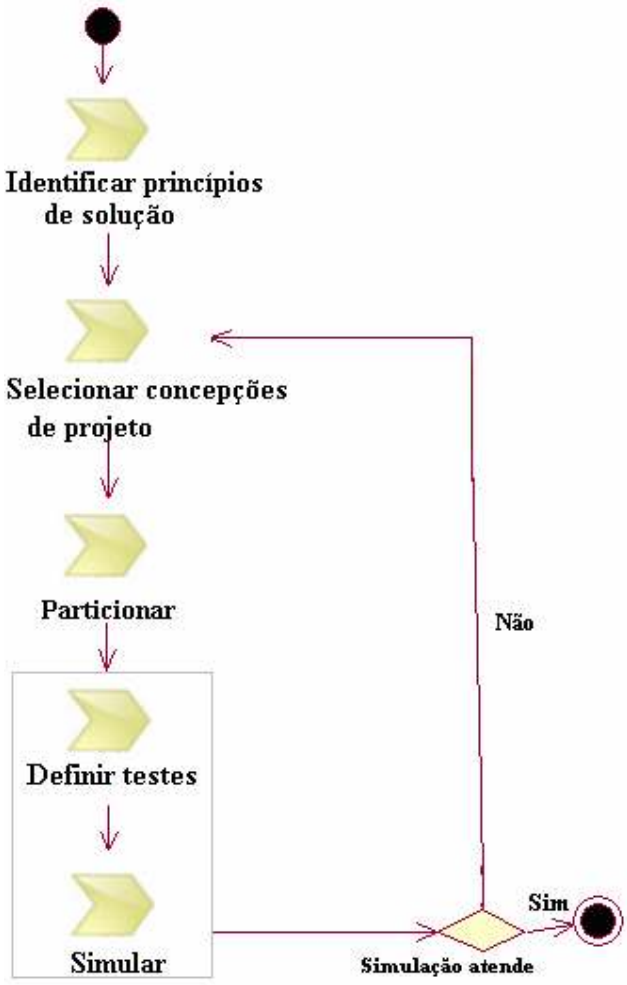


Figura 23 - Diagrama de atividades da disciplina Identificação de solução

A atividade Particionar pode ser auxiliada por algoritmos de *co-design*. Estes algoritmos analisam a melhor solução de implementação sob o aspecto de custo e eficiência. É importante colocar que a existência de requisitos temporais para o produto deve ser considerada nestes algoritmos, pois se sabe que implementações em hardware são mais

eficientes que em software e, portanto, estes requisitos podem interferir diretamente na solução adotada.

Recomenda-se que seja utilizado um estereótipo para identificar a área tecnologica responsável pelo componente. Assim, os estereótipos <<mecatrônica>>, <<mecânica>>, <<elétrica>> e <<software>> devem ser utilizados. Após a geração dos subcomponentes, um componente <<mecatrônica>> se transforma em subcomponentes com outros tipos de estereótipos.

De maneira análoga às atividades anteriores, a atividade identificar princípios de solução pode ser sumarizada na Tabela 8.

Tabela 8 – Exemplo de uma atividade da disciplina Identificação de Solução

Atividade: Identificar princípios de solução
Objetivo: Definir as alternativas de soluções para cada componente que será desenvolvido.

Passos:
• Para os componentes que serão desenvolvidos:
  o identificar os princípios de solução para cada componente;
  o montar a matriz morfológica.

Artefatos: Matriz morfológica.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas, arquiteto e especialistas.
Observação: O modelo do documento com as concepções de projeto encontra-se no experimento prático no capítulo 6.

5.3.4. Desenvolvimento

O objetivo da disciplina Desenvolvimento é construir fisicamente um protótipo do produto especificado.

Um produto é formado de componentes ou subcomponentes, mecânicos, eletrônicos ou de software, de forma que componentes e subcomponentes são sempre desenvolvidos em uma única área. Este desenvolvimento pode se basear em qualquer técnica escolhida pela equipe da área responsável, desde que respeite as interfaces definidas na especificação.

O diagrama de pacotes ilustrado na Figura 24 agrupa os elementos que compõem a disciplina Desenvolvimento. Como podem ser observadas as atividades são realizadas por especialistas de domínio, desenvolvedores e testadores, sob a supervisão dos líderes de cada área específica.

O Desenvolvimento contempla a execução de três atividades, conforme apresentado na Figura 24: construir; integrar; e prototipar. Estas atividades podem gerar artefatos tais como os componentes e subcomponentes implementados e o protótipo do produto pronto para ser testado.

O processo se inicia com a construção de cada componente ou subcomponente pela área responsável, a partir das especificações recebidas. A ocorrência de problemas que impeçam ou dificultem os desenvolvedores de seguir estas especificações prévias deve necessariamente demandar uma reunião multidisciplinar para redefinição da especificação.

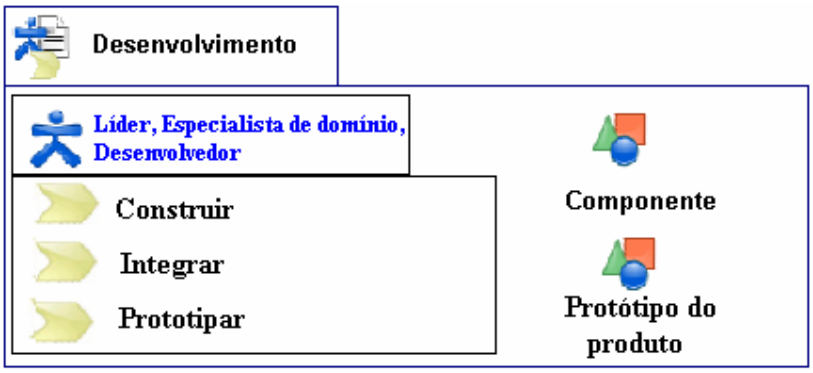


Figura 24 - Diagrama de pacotes da disciplina Desenvolvimento

A segunda atividade consiste da integração dos subcomponentes formando um componente híbrido. Este deverá ser submetido aos testes recebidos junto à sua especificação.

A terceira atividade consiste na montagem do protótipo do produto a partir dos componentes prontos. Seguindo o padrão da metodologia, a Tabela 9 descreve as características da atividade Integração.

Tabela 9 – Exemplo de uma atividade da disciplina Desenvolvimento

Atividade: Integrar
Objetivo: Montagem do componente.
Passos: • Sintetizar software / hardware; • Realizar reuniões interdisciplinares para: ○ Montar componente híbrido a partir dos subcomponentes sintetizados.
Artefatos: Componente.
Fase: Construção.
Papéis: Desenvolvedor, sob a supervisão do líder.

Como ilustrado na Tabela 9, a integração é formada por três passos. Primeiro o componente é sintetizado, depois os componentes híbridos são montados a partir da

composição de seus subcomponentes e finalmente o componente pronto é submetido aos testes especificados na disciplina de validação.

A MdpM orienta que seja utilizada a técnica de integração contínua de componentes. Nesta técnica, não é necessário aguardar que todos os componentes estejam prontos para iniciar a integração, pois ela é realizada à medida que os componentes vão sendo produzidos. Esta maneira de conduzir a integração é importante, principalmente para um ambiente de tecnologias diversificadas, pois permite que os componentes sejam testados logo que estiverem prontos, antecipando os problemas e diminuindo a possibilidade de erros. Assim, observa-se certo grau de paralelismo entre a integração e o processo de montagem do protótipo do produto que permite diminuir a possibilidade de atrasos no projeto. Esta integração contínua envolve também uma periodicidade de encontros agendados entre os testadores e desenvolvedores das áreas específicas onde eles poderão montar o componente híbrido e realizar os testes necessários.

5.3.5. Validação

O objetivo da disciplina Validação é validar a especificação para garantir certo grau de confiabilidade ao produto. Para isso, algumas atividades são realizadas ao longo do processo e englobam a definição de critérios de aceitação, que possam ser utilizados na entrega do produto, a definição de testes, a simulação do modelo concreto, a realização dos testes dos componentes produzidos e, finalmente, a realização de testes no protótipo do produto. Todas estas atividades visam encontrar possíveis falhas no produto e avaliar se o este atende às expectativas do cliente.

O diagrama de pacotes ilustrado na Figura 25 agrupa os elementos que compõem esta disciplina. Observa-se que ela é realizada pelos líderes das áreas específicas e pelos testadores, podendo ter a participação do gerente e do cliente. A disciplina compreende seis atividades: a definição dos critérios de aceitação; a definição dos testes que serão realizados nos componentes e subcomponentes; a definição dos testes que serão aplicados no protótipo do produto; a simulação do modelo concreto; e a realização propriamente dita dos testes técnicos; e a realização dos testes de uso. Os artefatos produzidos por estas atividades constam dos documentos com os critérios de aceitação, dos casos de testes, do modelo simulado do produto e do modelo do produto.

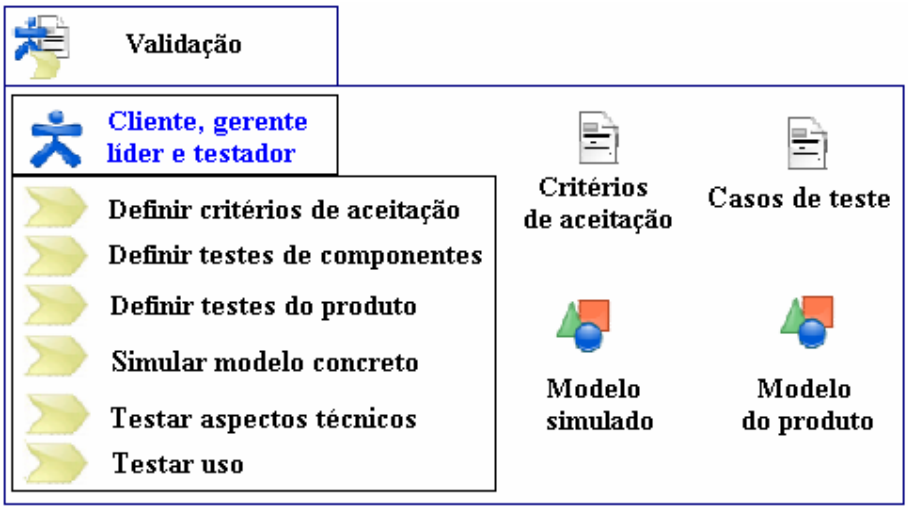


Figura 25 - Diagrama de pacotes da disciplina Validação

Testes são importantes para validar se o produto construído atende aos requisitos especificados no início do projeto e para dar maior confiabilidade aos resultados por ele produzidos. Considerando o nível de complexidade dos produtos mecatrônicos onde o produto é uma composição de componentes que podem ser oriundos de diversas áreas, se torna indispensável testar os componentes e a integração destes componentes. Desta forma, os testes técnicos validam o produto sob a visão técnica, ou seja, a integração dos componentes, os requisitos funcionais e também os não funcionais. Por isso, são realizados pelos testadores sob a supervisão dos líderes. Os testes de uso destacam a visão do cliente sobre o produto e devem ser realizados no protótipo do produto, tanto pelos testadores quanto pelos próprios usuários finais, tendo como resultado o modelo do produto. Pode-se, também submeter o modelo do produto, antes de ser enviado à linha de produção, a uma amostra de clientes em potencial e avaliar os resultados.

Além de testes também é importante simular o produto antes mesmo da sua construção física, ainda em um nível de especificação. Esta simulação permite submeter o modelo especificado aos requisitos funcionais e não funcionais para analisar se os resultados estão de acordo com o esperado. Com isso, é possível identificar problemas antes do produto ser realmente construído, economizando tempo e reduzindo custo.

A atividade Simular modelo concreto da disciplina Validação é descrita na Tabela 10.

Tabela 10 – Exemplo de uma atividade da disciplina Validação

Atividade: Simular modelo concreto
Objetivo: Realizar uma simulação do modelo concreto (diagrama de componentes) para visualizar o funcionamento dos componentes. Validar as suas interfaces internas e externas para detectar possíveis problemas de definição.
Passos: • Definir a ferramenta de simulação a ser usada; • Desenvolver o modelo de simulação; • Criar o modelo concreto no simulador; • Efetuar a simulação, aplicando os casos de teste; • Avaliar os resultados.
Artefatos: Modelo simulado.
Fase: Elaboração.
Papéis: Líderes das áreas específicas e especialistas.

A simulação não utiliza nenhuma forma de prototipação. O modelo de simulação é montado como caixas que possuem funções (comportamento) e interfaces de comunicação com as outras caixas. Atividades tais como a prototipação e o *layout* de formas são um detalhamento de cada área específica de acordo com o método que por ela for utilizado para construir o componente.

Além de avaliar a definição das interfaces, a simulação é importante também para verificar se os requisitos não funcionais tais como tolerância à falhas, requisitos temporais ou de desempenho estão mapeados corretamente, diminuindo os riscos do projeto. Assim, é importante que este modelo permita simular além dos requisitos funcionais, os requisitos temporais e que seja possível injetar defeitos no modelo para visualizar o comportamento especificado para o tratamento das falhas.

5.3.6. Verificação

Métodos formais devem ser utilizados para verificar se a especificação gerada para um produto está de acordo com as necessidades definidas pelo cliente. No caso de produtos mecatrônicos, estes métodos são indicados principalmente devido ao alto grau de confiabilidade exigido.

A disciplina de Verificação da MdpM orienta que seja utilizada a especificação formal para realizar verificação das partes do produto que requeiram maior confiabilidade. Recomenda-se que a verificação seja realizada utilizando a técnica de verificação de modelos. Para isso pode-se, por exemplo, utilizar as definições de [3] para, a partir dos modelos UML construídos no processo de especificação, gerar um modelo formal do sistema que será utilizado na verificação. A recomendação do uso de verificação de modelos pela MdpM se

baseia nesta possibilidade de geração automática do modelo e na existência de ferramentas computacionais [3] que tornam o processo de verificação viável na prática.

O diagrama de pacotes ilustrado na Figura 26 agrupa a estrutura da disciplina. Como pode ser observada, esta disciplina é realizada pelos líderes das áreas específicas e por especialistas de domínio que neste caso são os especialistas em verificação. São desenvolvidas duas atividades: identificação do comportamento do produto; e verificação do modelo propriamente dita. Estas atividades geram como artefatos um modelo de autômatos, propriedades e um documento com o resultado da verificação, representados na figura na forma de ícones.

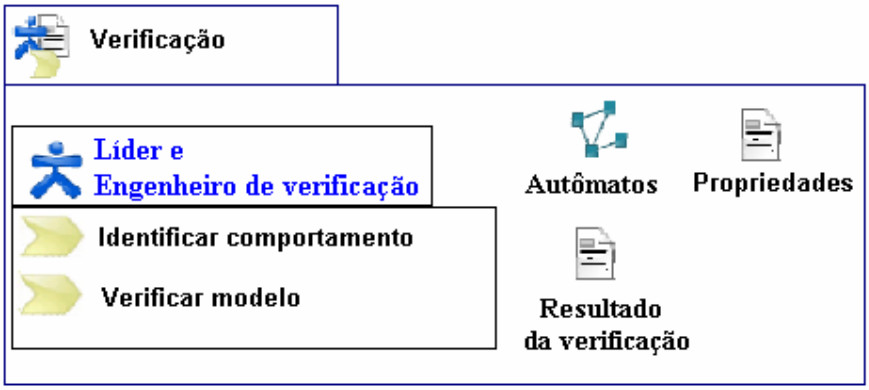


Figura 26 - Diagrama de pacotes da disciplina Verificação

Similarmente às atividades anteriores, a atividade Identificar comportamento é descrita na Tabela 11.

Tabela 11- Exemplo de uma atividade da disciplina de Verificação

Atividade: Identificar comportamento
Objetivo: Construir o modelo formal para as partes do produto que precisam de uma verificação formal e definir propriedades que possam ser submetidas a este modelo formal.
Passos: • Selecionar um verificador de modelos; • Mapear os diagramas de estados em autômatos; • Especificar propriedades em uma linguagem apropriada.
Artefatos: Autômatos e documento de propriedades.
Fase: Construção.
Papéis: Engenheiro de verificação com o apoio dos líderes das áreas envolvidas.

Observa-se que o primeiro passo consiste em mapear o comportamento do produto em autômatos. Para isso, utilizam-se os diagramas de estados gerados durante a Análise. A geração destes autômatos pode, por exemplo, ser realizada automaticamente a

partir de regras definidas em [3]. É importante selecionar qual o verificador e a linguagem que será utilizada para que possam ser definas as propriedades de verificação da correção do modelo em relação aos requisitos identificados no início da especificação.

A segunda atividade da verificação consiste em submeter o modelo gerado a um verificador de modelos juntamente com as propriedades definidas. Este verificador fará as checagens necessárias e apresentará um resultado indicando se o modelo atende ou não às especificações iniciais do projeto.

5.3.7. Documentação

A disciplina Documentação está presente em todas as fases do desenvolvimento e é responsável pela geração da documentação final do produto. Compreende os artefatos existentes nas diversas disciplinas e os documentos gerados na implementação dos componentes específicos de cada área.

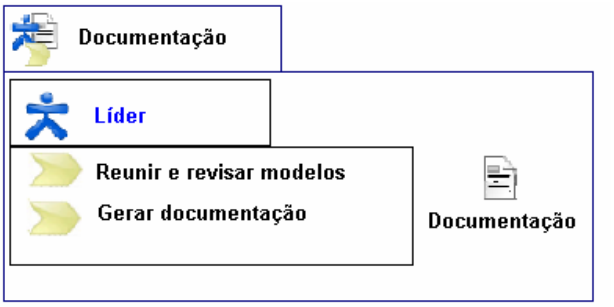


Figura 27 - Diagrama de pacotes da disciplina Documentação

O diagrama de pacotes ilustrado na Figura 27 agrupa os principais elementos da disciplina. Como se pode observar, esta disciplina é de responsabilidade dos líderes e nela são desenvolvidas duas atividades: reunião e revisão dos modelos; e geração da documentação que deverá acompanhar o modelo do produto.

Na fase Entrega do produto, esta disciplina se torna mais significativa, pois tem como objetivo reunir e revisar todos os documentos gerados ao longo do processo de desenvolvimento. Deve-se construir um documento que será enviado à linha de produção para fabricação do produto, bem como às demais áreas para geração dos demais documentos a serem criados para uso, manutenção, descarte, propaganda, entre outras atividades necessárias.

5.3.8. Gerência

Para atingir sucesso no desenvolvimento de um trabalho é importante a utilização de uma metodologia de desenvolvimento que permita guiar a equipe na execução das etapas do trabalho. No entanto, apenas a utilização de técnicas especializadas não garante que o processo seguirá da forma desejada. Associada a estas técnicas é preciso também definir algumas regras que permitam controlar as etapas, medindo sua evolução no decorrer do tempo, para prevenir problemas que venham a comprometer o objetivo final.

O objetivo da disciplina Gerência é acompanhar todo o processo de construção do produto para garantir o bom andamento das demais disciplinas e suas atividades.

O diagrama de pacotes ilustrado na Figura 28 agrupa os principais elementos desta disciplina. Observa-se que a disciplina é executada pelo gerente do projeto, pelos líderes das áreas específicas e pelo cliente, pois o gerente é o responsável direto pelo projeto, os líderes são os responsáveis pelas definições relativas à sua respectiva área de conhecimento e o cliente é o responsável pelas definições de necessidades.

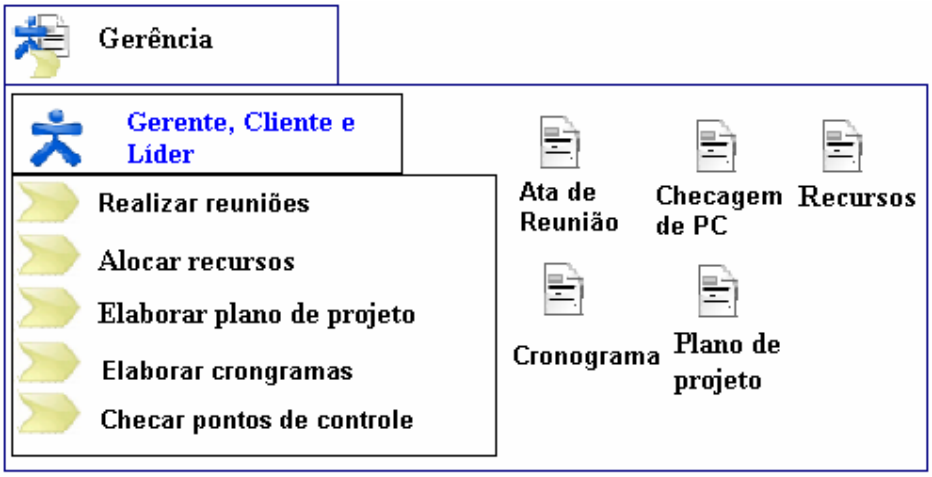


Figura 28 - Diagrama de pacotes da disciplina Gerência

A gerência é realizada em todo o processo de desenvolvimento do produto a partir de quatro atividades: a realização de reuniões, que acontecem no decorrer do processo; a alocação de recursos, que pode acontecer durante todo o projeto; a elaboração de um plano para o projeto, que deverá ser seguido pela equipe; a elaboração de cronogramas, que servirão de guia para saber se o plano de projeto está sendo executado conforme planejado; e a checagem de pontos de controle. Como resultado destas atividades gera-se artefatos,

representados como ícones na Figura 28, tais como atas de reunião, cronogramas, checagem de pontos de controle.

Os passos para execução da atividade Checar pontos de controle são executados à medida que as fases da metodologia vão sendo completadas, ou seja, ao final de cada fase existe um ponto de controle a ser checado. Como ilustrado na Tabela 12, estes pontos tem o objetivo de orientar a equipe de maneira que se tenha certeza que a fase pode ser finalizada e que uma nova fase pode ser iniciada. Para isso, define perguntas que checam se pontos importantes do processo já foram finalizados.

Tabela 12 – Exemplo de uma atividade da disciplina Gerência

Atividade: Checar pontos de controle
Objetivo: Avaliar o bom andamento do projeto de acordo com características pré-definidas.

Passos:
• PC1: acontece sempre ao final da fase Concepção e tem os seguintes passos:
  o verificar a validade dos requisitos. Se não existem outros requisitos para outros usuários;
  o verificar a consistência dos requisitos . Se os conflitos foram resolvidos;
  o verificar a viabilidade dos requisitos. Se é possível implementar o requisito com a tecnologia atual;
  o analisar a facilidade de avaliação de um requisito. Se é possível o cliente avaliar de maneira simples se o produto está atendendo aos requisitos;
  o Avaliar se o escopo definido para a versão é viável ao projeto sob vários aspectos diferentes, procurando atender aos seguintes questionamentos:
  ▪ as funcionalidades selecionadas agregam realmente o valor desejado ao produto?
  ▪ o cronograma inicial está de acordo com as expectativas?
• PC2: acontece após a fase Elaboração. Considerando que todo o levantamento de requisitos foi finalizado e uma arquitetura foi definida, é necessário então observar se existe algum fator que possa impactar o sucesso do projeto, seja ele físico, técnico ou pessoal.
Este ponto de controle deverá responder as seguintes perguntas:
  o os requisitos estão completamente entendidos e mapeados nos modelos do sistema?
  o todos os componentes definidos no modelo abstrato tem sua estratégia de implementação claramente definida?
  o os testes foram simulados e todos os problemas contornados?
  o existe concordância entre os líderes e gerentes quanto às soluções adotadas?
• PC3: realizado após a fase Construção. Considerando que neste ponto o modelo do produto se encontra pronto, devem-se analisar as condições de desempenho, confiabilidade e tolerância a falhas.
• PC4: aplicado quando o produto já está pronto para ser entregue à produção. Deve responder as seguintes perguntas:
  o a documentação está atualizada e de acordo com o produto construído?
  o é necessário iniciar as atividades de construção de outra versão do produto?
  o os critérios de aceitação foram atendidos?

Artefatos: Documento de checagem de pontos de controle (PC).
Fase: Ao final de cada fase.
Papéis: Cliente, gerentes e líderes das áreas.

5.4. FERRAMENTAS DE APOIO AO DESENVOVIMENTO

Ao longo do desenvolvimento de produtos mecatrônicos vários artefatos são gerados que necessitam de ferramentas especiais para serem desenhados. A tabela 13 apresenta uma sugestão de ferramentas que podem ser utilizadas pela equipe na construção destes artefatos.

Tabela 13 – Ferramentas aplicadas na construção dos artefatos da MdpM

Ferramenta	Artefatos
Rational ROSE	Diagramas UML
SISCOI	Matriz da casa da qualidade
MATLAB/SIMULINK	Simulações
CORFU	Geração de FB

5.5. ESFORÇO DE ESPECIFICAÇÃO DA METODOLOGIA

A MdpM especifica um conjunto de artefatos importantes para o domínio de aplicações mecatrônicas. O esforço, em termos quantitativos, para a especificação destes artefatos é apresentado na Tabela 14. Pode-se observar que foram definidas quatro fases para compor o processo de desenvolvimento de um produto mecatrônico. Ao longo destas fases, oito disciplinas agrupam 32 atividades que são executadas pelos nove papéis participantes do projeto. Estas atividades constroem e utilizam 36 artefatos que serão a base da documentação do produto.

Tabela 14 – Resumo das especificações da MdpM

Conceito	Quantidade de especificações
Fases	4
Disciplinas	8
Papéis	7
Atividades	32
Artefatos	36

6. EXPERIMENTO PRÁTICO

Este capítulo apresenta a especificação de um produto mecatrônico seguindo as orientações descritas na MdpM, tendo como principal objetivo a validação do fluxo de atividades descritas na metodologia.

Devido à indisponibilidade de pessoal, o experimento foi realizado por uma pessoa da área de Computação e validado pelas demais áreas. Por este motivo, a especificação possui um nível alto de abstração.

A especificação descrita neste capítulo baseia-se nas orientações descritas em [37] para construção de um robô. As seções que se seguem estão organizadas por fase da metodologia e destacam os produtos gerados pelas disciplinas executadas em cada uma das fases.

A documentação completa do experimento encontra-se disponível no Apêndice C.

6.1. CONCEPÇÃO

A fase Concepção se inicia com uma descrição do produto que será construído. Para este experimento, o produto mecatrônico selecionado é um robô.

O robô será utilizado para exploração de ambientes que não permitam a presença humana. Como principal objetivo deve se locomover neste ambiente capturando imagens e enviando-as para um ambiente externo, representado na Figura 29 como o *controle remoto*. Por exemplo, o robô pode ser utilizado para capturar imagens de um lugar afetado por um gás tóxico qualquer que não possa ser respirado pelo ser humano. Desta forma, ele deverá se locomover pelo ambiente, capturar as imagens e enviá-las para um lugar seguro onde possam ser analisadas.

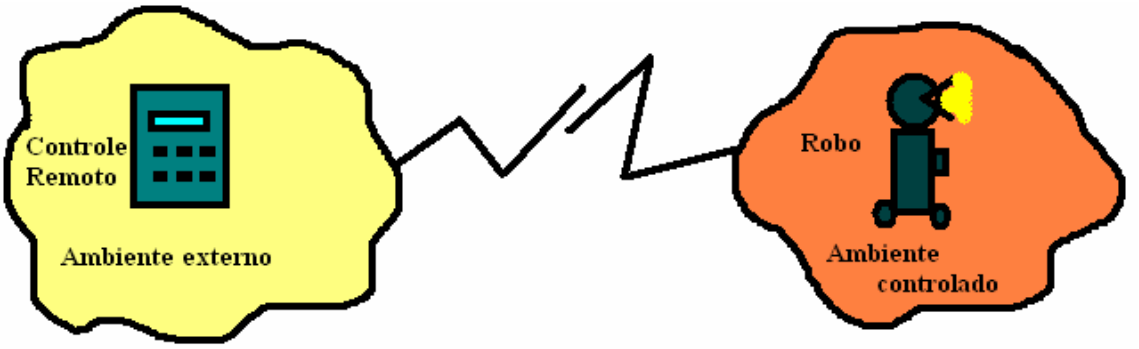


Figura 29 – Visão geral do sistema

O robô é controlado remotamente pelo controle remoto (CR). Este controle pode ser realizado de duas formas: manual, por uma pessoa, ou automático, através de um sistema inteligente. Quando for controlado manualmente, uma pessoa será responsável por analisar as imagens transmitidas e guiar o robô pelo ambiente. Quando for controlado automaticamente, as imagens transmitidas servirão de base para um sistema de navegação que indica a locomoção do robô. Desta forma, a partir das imagens capturadas o sistema de navegação identifica obstáculos e define o caminho a ser percorrido. O robô fica em operação pelo controle automático até que este seja desativado.

O controle manual também permite efetuar ajustes na câmera para melhorar a qualidade das imagens capturadas tais como foco e distância (*zoom*).

A interface do controle remoto deve ser amigável, simples e flexível, pois se espera que este robô tenha um leque grande de aplicabilidade. O robô deve se locomover nas quatro posições (norte, sul, leste e oeste) com uma velocidade constante. Além disso, deve ser extensível a utilização de outros dispositivos, além do inicialmente projetado para filmagem, com poucas modificações.

Uma vez definida a descrição inicial do produto, as seguintes necessidades do cliente são destacadas, que devem ser consideradas na construção do robô:

- gerar imagens de um ambiente onde o homem não possa estar presente e enviá-las para um local remoto;
- permitir ajuste da câmera na captura da imagem;
- locomoção nas quatro direções: norte, sul, leste e oeste;
- ser controlado manualmente via controle remoto;
- ser controlado automaticamente via controle remoto;

- usar uma tecnologia que permita variedade de controladores remotos;
- usar uma tecnologia que permita adicionar novos dispositivos para serem controlados além do dispositivo de filmagem;
- ter uma interface de comunicação simples;
- ter um custo acessível.

A partir destas necessidades iniciais são definidos os requisitos funcionais, representados pelos casos de uso ilustrados na Figura 30.

Como se pode observar, o diagrama de casos de uso apresenta dois atores, que representam as duas formas de controle identificadas para o robô: o controle manual representado pelo ator *controlador manual* e o controle automático representado pelo ator *controlador automático*.

O controlador manual possui algumas funcionalidades exclusivas tais como ligar e desligar o robô, ativar e desativar o controle remoto e controlar dispositivos.

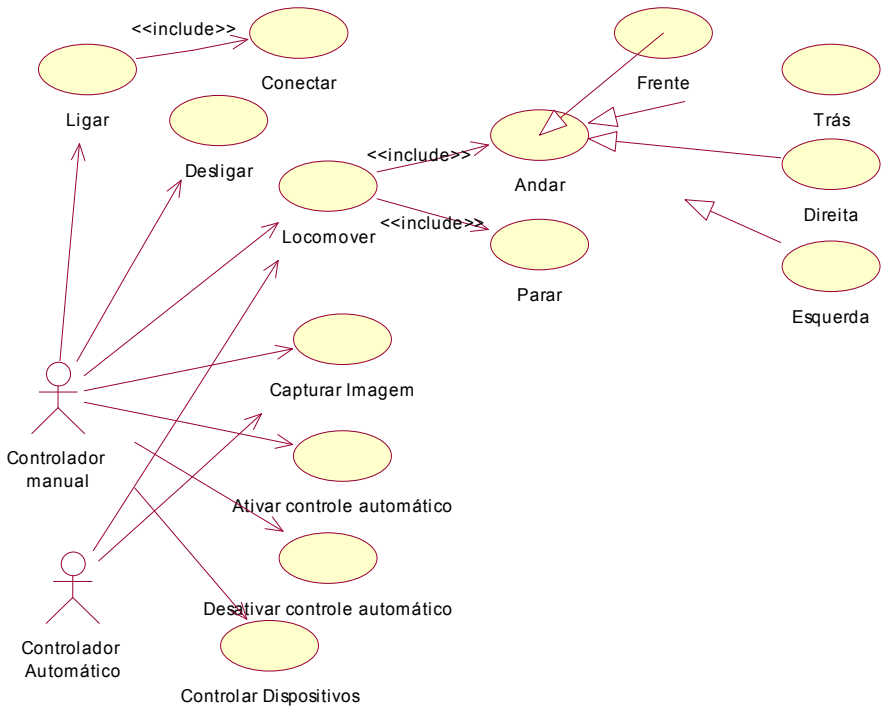


Figura 30 - Diagrama de casos de uso com as funcionalidades do robô

O robô é ligado a partir do caso de uso Ligar. Esta opção é responsável por estabelecer uma conexão entre o controle remoto e o robô, carregando configurações iniciais no robô e deixando-o pronto para receber novos comandos. Caso o controlador manual deseje ativar o controle automático ele deverá utilizar o caso de uso Ativar controle automático. A partir deste momento, o robô passa a ser controlado pelo sistema automático de navegação até que o caso de uso Desativar controle automático seja selecionado. Enquanto o robô estiver em operação é possível utilizar o caso de uso Locomover para andar (nas quatro posições) ou parar o robô. Também é possível, para o controlador manual, controlar dispositivos. O dispositivo previsto até o momento deve suprir apenas necessidades de filmagem, portanto é possível, por exemplo, ajustar o *zoom*. Sempre que o robô estiver se locomovendo, imagens estarão sendo capturadas e enviadas para o controle remoto. Finalmente, quando o controlador manual desejar, poderá desligar o robô a partir da opção Desligar, que encerra a conexão salvando a configuração atual para ser recarregada da próxima vez que o robô for ligado.

É importante destacar que a precisão na locomoção e captura de imagens do robô depende da velocidade com que as solicitações do controle remoto são atendidas. No caso do controle automático depende também da velocidade de processamento e análise da imagem para guiar a locomoção.

Considerando que o robô estará sendo controlado à distância, é importante que este tenha um mecanismo de monitoria da conexão com o controle remoto. Em caso de perda de conexão, um estado de segurança deve ser projetado para preservar a integridade do robô até que a conexão seja restabelecida.

Para atender as necessidades descritas os seguintes requisitos não funcionais são considerados, distribuídos por categoria.

- Extensibilidade:
    - usar um sistema de comunicação que permita adicionar novos dispositivos com facilidade;
    - projetar um robô com previsão de expansão de dispositivos.
- Usabilidade:
    - interface de controle amigável;
    - a imagem capturada deve ser nítida o suficiente para ser entendida pelo controlador manual ou pelo sistema de navegação.

- Falha:
    - qualquer situação de falha deve parar a locomoção do robô, inclusive a perda de comunicação entre a central e o robô, até que a falha seja recuperada.
- Tempo:
    - o deslocamento deve ocorrer em um intervalo de tempo *Td* específico para que o robô seja preciso em seus movimentos;
    - o envio da imagem deve ocorrer em um intervalo de tempo *Ti* específico para que estas imagens sejam válidas;
    - o tempo de processamento do sistema de navegação deve ser suficiente para que a atuação no ambiente tenha o resultado esperado.
    - a parada do robô em caso de falhas ou em caso de situações anormais (obstáculos, por exemplo) deve ocorrer em um tempo *Tp* aceitável.
- Custo:
    - optar por tecnologias mais accessível, tanto no robô quanto no controle remoto;
- Gerais:
    - peso *P* específico
    - comprimento *C* específico
    - largura *L* especifica
    - altura *A* específica
    - Consumo de energia

A partir dos requisitos, funcionais e não funcionais, é montada a matriz da casa da qualidade, definindo-se a importância de cada um destes requisitos para o cliente, identificando as relações existentes entre eles e comparando-os com os produtos similares de mercado, se for necessário (Figura 31).

Observa-se que são analisados os relacionamentos existentes entre as necessidades do cliente, representado pelas linhas da figura, e os requisitos mapeados para o produto, representados pelas colunas. O cliente atribui a cada necessidade um grau de importância como, por exemplo, a necessidade Envio remoto de imagens, que tem grau de

importância cinco. A partir destas definições gera-se o grau de importância de cada requisito para o projeto, representado na última linha da Figura 31 como uma escala de um a doze.

No telhado da casa também é possível observar que alguns conflitos foram identificados. Por exemplo, para atender ao requisito Novos dispositivos o produto terá que ser mais flexível e isso poderá deixar a interface menos amigável. Além disso, esta característica pode significar um aumento no custo do produto, pois os componentes que serão projetados devem atender a requisitos de expansão.

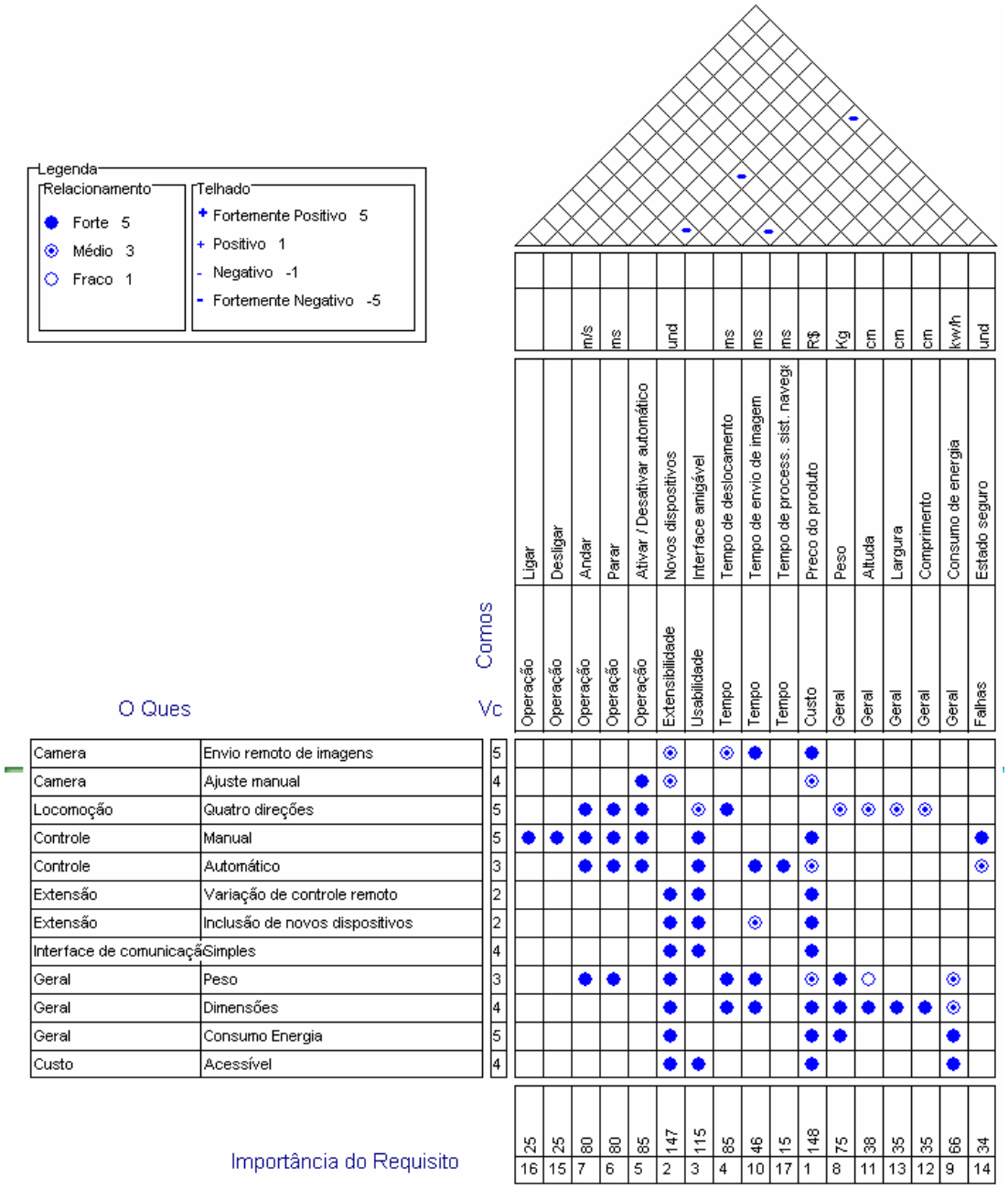


Figura 31 - Casa da qualidade para o robô

Para finalizar a fase Concepção, alguns fatores são considerados como de risco e, portanto, foram destacados para serem analisados de maneira a não comprometer o sucesso do projeto, são eles:

- tempo de execução das tarefas desde a solicitação de locomoção pelo controle remoto até o processamento das imagens enviadas pelo robô;
- tecnologia utilizada para conexão entre o controle remoto e o robô, que deve ser segura o suficiente para que não haja perdas constantes de comunicação.

Com as informações levantadas até o momento definiu-se que o escopo do projeto do robô deve conter todos os requisitos funcionais (representados pelos casos de uso) e não funcionais.

Como instrumento de avaliação do modelo do produto quando este já estiver pronto, são identificados também os seguintes critérios de aceitação:

- atender aos requisitos funcionais e não funcionais especificados;
- obter resultado positivo em 100% dos testes realizados;
- ter sido submetido a uma simulação real, piloto, com resultados positivos.

Ao longo da fase Concepção alguns artefatos foram gerados e validados. Esta validação faz parte da checagem de pontos de controle que acontece sempre ao final de cada fase e está ilustrada no Documento de validação apresentado na Tabela 15.

Tabela 15 - Documento de validação da primeira fase de especificação do robô

Documento de Validação	
Data de início do projeto: 27/10/06	
Equipe Responsável	
Identificação	Nome
Gerente do cliente	Especificações de [37]
Gerente Técnico	Ana Patrícia
Líder Computação	Ana Patrícia
Líder Mecânica	Hermam
Concepção	
Data da validação	Artefatos
30/10/06	Casos de uso
02/11/06	Lista de requisitos não funcionais
04/11/06	Casa da qualidade e Soluções de conflitos
05/11/06	Fatores de risco
08/11/06	Documento de escopo

6.2. ELABORAÇÃO

A fase Elaboração se inicia com o detalhamento dos requisitos do produto. Assim, para cada requisito funcional (caso de uso) é elaborada uma tabela que apresenta informações tais como: qual a pré-condição para que o caso de uso funcione; qual o ator responsável por iniciar o caso de uso; o objetivo do caso de uso; seu funcionamento normal detalhado; como deve funcionar em casos excepcionais; qual a pós-condição; e observações pertinentes.

Tabela 16 - Descrição do caso de uso Ligar

Caso de uso	**Ligar**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô desligado.		
Ator:	Controlador manual.		
Objetivo:	Ligar o robô. Estabelecer uma conexão entre a central e o robô.		
Caso normal			
Ator		Robô	
Selecionar opção Ligar.		Conecta com o controle remoto; Inicializa configuração; Liga o dispositivo de filmagem; Aguarda novos comandos.	
Exceção 1: Central não consegue se conectar			
Mensagem de conexão não efetuada.			
Pós – condições	Robô aguardando comando.		

A Tabela 16 descreve o caso de uso Ligar. Este caso de uso é ativado pelo controlador manual quando o robô está no estado de desligado. Para isso é selecionada no controle remoto a opção Ligar. Sua função é estabelecer uma conexão com o robô para iniciar as atividades. Quando a conexão é efetuada com sucesso, o robô recebe a solicitação do controle remoto e restaura a última configuração utilizada. O dispositivo de filmagem também é iniciado e o robô fica aguardando os próximos comandos. Caso aconteça algum problema que impeça a comunicação inicial do controle remoto com o robô, uma mensagem é apresentada ao controle remoto e nada mais acontece.

Além dos casos de uso, os requisitos não funcionais também são detalhados conforme tabela a seguir. Observa-se que para cada um dos requisitos uma descrição detalhada é elaborada. Características temporais relacionadas ao requisito devem ser detalhadas, como apresentado na Tabela 17.

Tabela 17 - Descrição dos requisitos não funcionais

Requisitos não Funcionais		
Data:	27/10/06	
Categoria	**Requisito**	**Descrição**
Extensibilidade	Usar um sistema de comunicação que permita adicionar novos dispositivos com facilidade.	A comunicação entre o controlador manual e o robô será remota. Ela será implementada a partir de um protocolo de comunicação que permita trocar comandos, dados e imagens. Este protocolo deve ser flexível o possível para permitir que novos dispositivos acoplados ao robô sejam controlados também remotamente.
Usabilidade	Interface de controle amigável.	A interface do controlador remoto deve ser simples, fácil de ser utilizada.
Tempo	O deslocamento deve ocorrer em um intervalo de tempo específico para que o robô seja preciso em seus movimentos.	A precisão no deslocamento do robô está diretamente relacionada ao intervalo de tempo existente entre a solicitação do comando de locomoção e a sua execução pelo robô. Assim, foram definidas as seguintes características: *deadline* 50ms*; t*empo de execução *40ms; e p*eriodicidade aperiódica.
	O envio da imagem deve ocorrer em um intervalo de tempo específico para que estas imagens sejam válidas.	A captura da imagem pela câmera e o envio desta para a central deve acontecer em um tempo suficiente para que estas imagens possam ser analisadas pelo controlador. Assim, foram definidas as seguintes características: *deadline 30ms;* tempo de execução *25ms;* periodicidade periódica.
	O tempo de processamento do sistema de navegação deve ser aceitável para que possa atuar no ambiente.	O tempo de processamento da imagem pelo sistema de navegação quando o robô estiver sendo operado em modo automático deve atender as seguintes restrições: *deadline 20ms;* tempo de execução *15ms;* e periodicidade periódica.
	A parada do robô em caso de falhas ou em caso de situações anormais (obstáculos, por exemplo) deve ocorrer em um tempo *T* aceitável.	O tempo de interrupção dos motores que locomovem o robô deve ser *50ms*.
Custo	Optar por uma tecnologia barata, tanto no robô quanto no controle remoto.	A tecnologia deve ser atender às expectativas de custo do projeto.
Falha	Qualquer situação de falha deve parar a locomoção do robô, inclusive a perda de conexão, até que a falha seja recuperada.	Caso ocorra algum problema de comunicação entre o controlador e o robô, este deverá ter suas atividades de locomoção suspensas. Isso deve ser feito em no máximo *50ms*.
Gerais	Peso, altura, largura e comprimento.	O robô deve ter um peso máximo de 15kg, uma altura máxima de 80cm, uma largura máxima de 50cm e um comprimento máximo de 50cm.
	Consumo de energia	O robô deve consumir um valor máximo X de energia.

A partir das informações colhidas, é então montado o diagrama estático com a estrutura preliminar do robô. Este diagrama ilustra as classes já identificadas para o produto, com seus atributos e métodos.

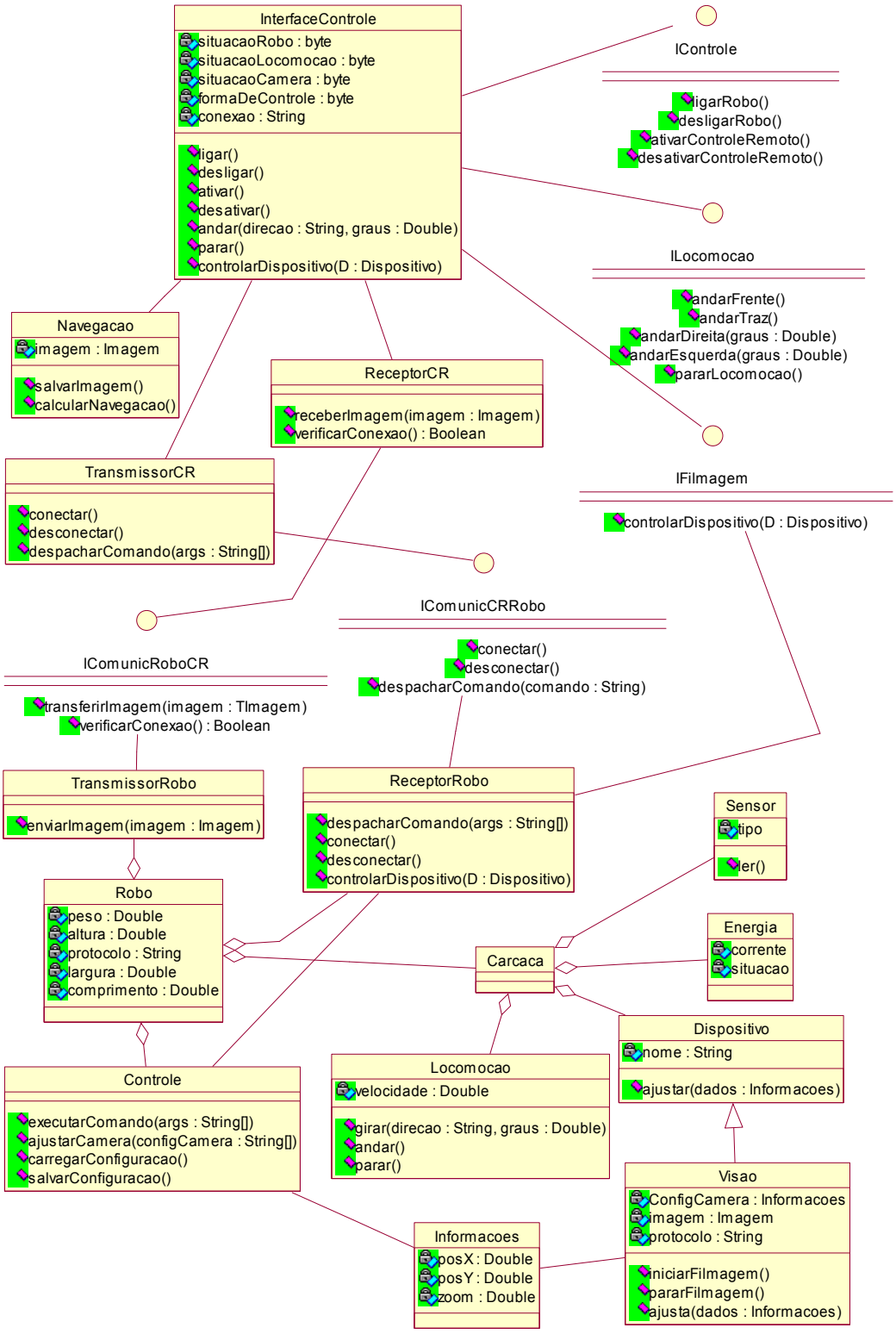


Figura 32 - Modelo estático do robô

Pode-se observar a partir da Figura 32 que a estrutura do robô está dividida em duas partes que se comunicam através de um conjunto de interfaces. A primeira parte, formada pelas classes InterfaceControle, Navegação, TransmissorCR e ReceptorCR referem-se ao controle remoto (CR). A segunda parte formada pelas demais classes, refere-se ao robô propriamente dito. Nota-se então que o robô possui características gerais tais como peso e tamanho, e é composto por um conjunto de classes responsáveis pela comunicação com o CR (TransmissorRobo e ReceptorRobo), pelo controle local do robô (Controle) e pela sua estrutura física (Carcaça). É importante observar que a forma de modelar a visão do robô a partir de uma classe chamada Dispositivos já demonstra uma preocupação com a inclusão futura de novos dispositivos.

Para cada uma das classes do diagrama ilustrado na Figura 32, é elaborada uma tabela com uma descrição que detalha seu funcionamento e define cada atributo e método identificado. Também as interfaces implementadas pela classe são listadas no final da tabela.

A Tabela 18, por exemplo, mostra a descrição da classe InterfaceControle. Pode-se observar a existência de cinco atributos e sete métodos. O atributo situacaoRobo, por exemplo, indica a situação atual do robô, podendo assumir o valor um para ligado ou dois para desligado.

Tabela 18 - Descrição da classe InterfaceControle

Classe: InterfaceControle.			
Data da elaboração:	10/11/2006	Última alteração	26/12/2006
Descrição: Representa a classe principal do controle remoto do robô. É responsável por todo o controle seja ele manual ou automático. Relaciona-se diretamente com três outras classes, a ReceptorCR, a TransmissorCR e Navegacao.			
Atributos		**Descrição**	
SituacaoRobo: *byte*		Indica a situação atual do robô, se ligado (1) ou desligado (2).	
situacaoLocomocao: *byte*		Índica se o robô está parado (1) ou em movimento (2).	
situacaoCamera: *byte*		Índica se a câmera esta ligada (1) ou desligada (2).	
formaDeControle: *byte*		Indica se o robô está sendo operado manualmente (1) ou automaticamente (2).	
conexão: *String*		Comando de conexão com o robô.	
Métodos		**Descrição**	
ligar()		Liga o robô.	
desligar()		Desliga o robô.	
ativar()		Ativa o controle remoto.	
desativar()		Desativa o controle remoto.	
andar (direção: *String*, graus: *double*)		Locomove o robô na direção indicada.	
parar()		Para o robô.	
controlarDispositivo(d: dispositivo)		Configura o dispositivo passado como parâmetro.	
Interfaces implementadas:			
IControle, IFilmagem e ILocomocao.			

A partir do diagrama de classes modela-se a estrutura dinâmica do robô. Esta estrutura representa a maneira como os objetos das classes interagem entre si trocando informações. A Figura 33, por exemplo, apresenta o diagrama de seqüência necessário para ligar o robô.

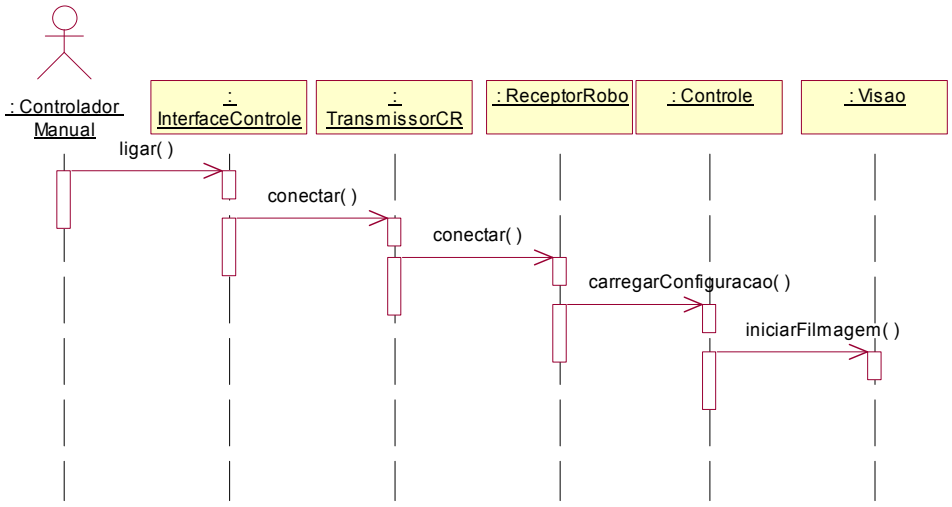


Figura 33 - Diagrama de seqüência para ligar o robô

Pode-se observar que a ação se inicia a partir do ator controlador manual, solicitando à interface de controle que o robô seja ligado. A partir desta solicitação é então estabelecida uma conexão entre o transmissor de dados do CR e o receptor de dados do robô. A primeira atividade executada no robô quando ele é ligado é carregar as configurações que existiam quando ele foi desligado pela última vez e depois iniciar a filmagem enquanto aguarda novas instruções.

Análogamente, a Figura 34 apresenta o comportamento do robô quando a filmagem está sendo efetuada. Pode-se notar que o controlador do robô controla todo o processo. Primeiramente ele envia uma mensagem à classe Visão para que a filmagem seja iniciada. A partir deste momento, imagens são capturadas, tratadas, a partir da extração de características, e envidadas para o CR a partir da comunicação já estabelecida entre o transmissor do robô e o receptor do CR. Quando a imagem é recebida pelo CR ela é armazenada. Nota-se que todo este processo acontece de maneira periódica a cada intervalo de tempo *Ti*.

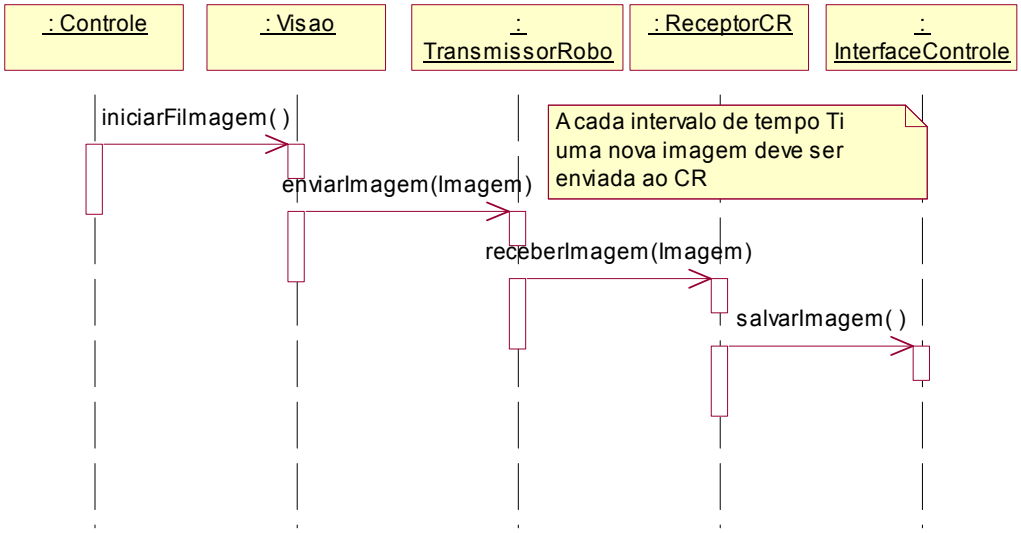


Figura 34 - Diagrama de seqüência da filmagem

A modelagem comportamental do produto envolve também a definição dos possíveis estados que este pode assumir durante sua utilização.

Observa-se na Figura 35 o diagrama de estados do controle remoto do robô. O CR sempre é iniciado em modo Manual, podendo ser colocado ou tirado do Automático pelo controlador manual.

No modo manual, inicialmente o robô está desligado. Quando é solicitado que ele seja ligado, o CR fica no estado de Conectando até que a conexão com o robô seja estabelecida e ele passe para o estado Ligado. Caso não seja possível efetuar a conexão, volta para o estado Desligado.

Quando o robô está ligado, o CR inicialmente fica no estado de Aguardando comando. Assim que o usuário controlador informa um comando qualquer, este comando é enviado para o robô (Enviando comando) e o estado retorna a Aguardando comando. Caso esteja sendo executada uma filmagem, o CR pode também receber do robô uma imagem, que será armazenada (Salvando imagem), depois retornar ao estado de Aguardando comando.

Quando o robô está sendo operado em modo automático, o CR fica sempre aguardando uma imagem. Uma vez que a imagem é recebida, ela é salva (Salvando imagem) e enviada para que seja efetuado o cálculo da navegação do robô (Calculando navegação). Depois que o cálculo é efetuado, um comando de navegação é enviado ao robô (Enviando comando) e o estado retorna ao estado Aguardando imagem para repetir todo o processo.

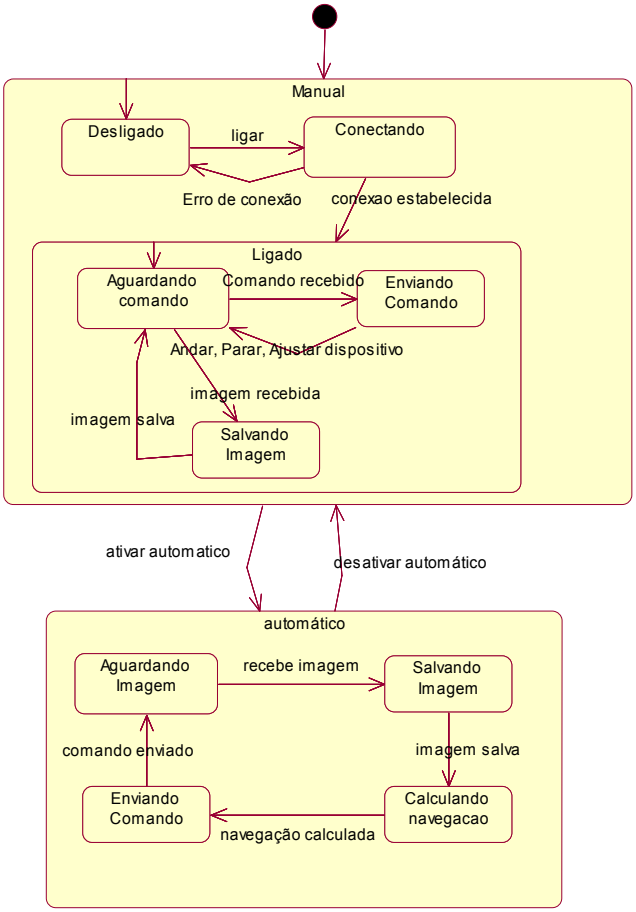


Figura 35 - Diagrama de estados do controle remoto

A partir dos modelos estruturais e comportamentais definidos é desenvolvido o diagrama de componentes representado na MdpM pelo modelo abstrato do produto. A estratégia de definição dos componentes baseia-se no agrupamento das classes. Desta maneira, um componente pode atender por completo a uma necessidade específica. O modelo, ilustrado na Figura 36, apresenta os componentes definidos para o controle remoto e para o robô. Nota-se que o componente Comunicação agrupa as classes TransmissorCR e ReceptorCR, provendo desta maneira todas os procedimentos necessários para efetuar a comunicação com o robô. De maneira análoga o componente ComunicacaoRobo agrupa as classes TransmissorRobo e ReceptorRobo, provendo esta funcionalidade para o robô.

Basicamente este projeto possui dois produtos, representados na figura como componentes compostos por subcomponentes, que são o controle remoto e o robô propriamente dito.

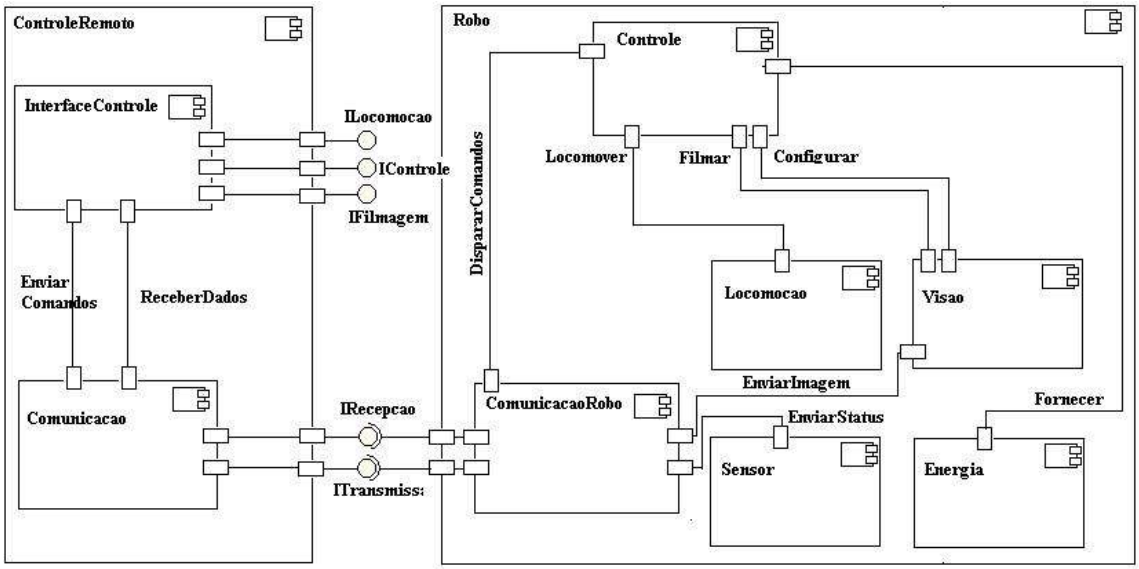


Figura 36 - Modelo abstrato do produto

O primeiro componente, observado do lado esquerdo da figura, é o ControleRemoto. Ele compreende os subcomponentes InterfaceControle e Comunicação e possui cinco portas de comunicação com o meio externo. As três primeiras portas implementam as interfaces ILocomoção, IControle e IFilmagem, respectivamente, e permitem que um agente externo qualquer tenha acesso ao componente, pois representam os serviços disponíveis para serem utilizados pelo controlador manual na operação do robô. As duas outras portas implementam as interfaces IRecepcao e ITransmissao, e fornecem acesso direto ao componente Comunicação. O componente InterfaceControle se comunica com o subcomponente Comunicação através de duas portas. A primeira delas, EnviarComandos é responsável pelo envio dos comandos e dados que devem ser transmitidos para o robô. A segunda, ReceberDados, é responsável pelo recebimento dos dados enviados pelo robô para o controle remoto.

O segundo grande componente é o Robô. Ele fornece duas portas com as interfaces requeridas para que haja a comunicação com o componente ControleRemoto. Desta forma, os subcomponentes Comunicação e ComunicacaoRobo se comunicam entre si a partir das interfaces definidas para eles, transmitindo dados e comandos. Cada comando recebido pelo subcomponente ComunicacaoRobo é enviado ao controle do robô a partir da porta

DispararComandos para que seja traduzido e executado nos demais componentes que formam o robô. Existe uma comunicação direta entre o subcomponente Visão e o subcomponente ComunicacaoRobo que permite enviar as imagens capturadas diretamente para o componente ControleRemoto. Existe também uma comunicação direta entre o subcomponente Sensor e o subcomponente ComunicacaoRobo que permite enviar um status com a situação atual do robô.

Pode-se dizer que o projeto do robô já atingiu um nível conceitual bastante detalhado neste momento. A partir de então, é iniciado o processo de definição da melhor solução tecnológica para implementação de cada um dos componentes. A identificação da solução começa com a análise do modelo abstrato para construção da matriz morfológica ilustrada na Tabela 19. Observa-se que para cada um dos componentes identificados no modelo sugerem-se alguns tipos de implementação, nas diversas áreas envolvidas.

Tabela 19 - Matriz morfológica do robô

Componente	**Opções de Implementação**							
	Software		**Mecânica**	**Eletrônica**		**Mecatrônica**		
InterfaceControle	Software para PC	Software para PalmTop						
Comunicação	Internet			Radio	Rede s/ fio			
ComunicacaoRobo	Internet			Radio	Rede s/ fio			
Controle				Micro controlador		PalmTop		
Visão			Sensor			Câmera		
Locomoção			Receptor IR Micro controlador Motor Carcaça					
Sensor			Sensor de presença					
Energia				Estabilizador	Fonte			

A partir da matriz morfológica apresentada acima, escolhe-se uma concepção de projeto para o robô, combinando as soluções sugeridas. A concepção escolhida para o projeto do robô é apresentada na Tabela 20.

Tabela 20 - Concepção de projeto para o robô

Componente	**Implementação**	**Concepção escolhida**
InterfaceControle	Construir	Sistema em um PC
Comunicação	Adquirir	Rede sem fio
ComunicacaoRobo	Adquirir	Rede sem fio (PalmTop)
Controle	Adquirir	PalmTop
Locomoção	Construir	Receptor IR, micro controlador, motor e carcaça
Visão	Adquirir	Câmera filmadora
Sensor	Adquirir	Sensor de presença
Energia	Adquirir	Estabilizador

A partir das decisões de implementação, pode-se observar o modelo concreto conforme definido na Figura 37.

Nota-se uma simplificação do diagrama após a escolha da concepção de projeto que será produzido. Por exemplo, o modelo concreto unifica os componentes Controle e ComunicaçãoRobo com a utilização do PalmTop, que já provê recursos para atender a estas duas necessidades. Esta decisão não demandou revisão do diagrama de classes, pois o subcomponente PalmTop continua implementando as mesmas interfaces previamente definidas.

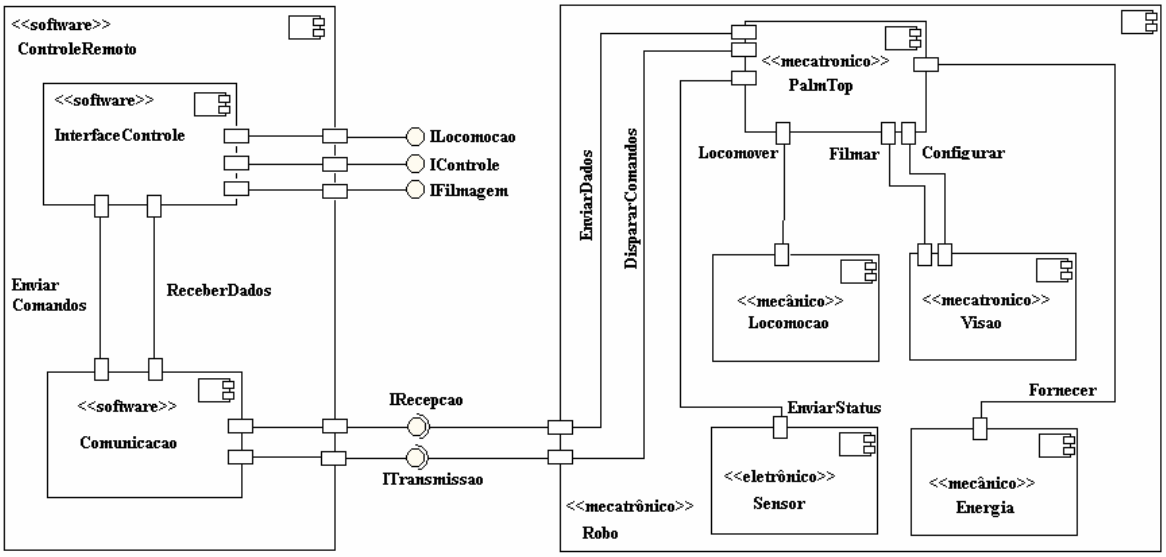


Figura 37 - Modelo concreto do produto

Com base nos componentes do modelo concreto, são definidos os testes que deverão ser realizados quando estes estiverem implementados.

Para cada um dos componentes são especificadas as situações que devem ser testadas. A Tabela 21 apresenta os testes do componente comunicação. Nota-se que é identificado o papel responsável pelo teste, o que será testado e qual o resultado que se espera obter quando o teste for realizado. O primeiro teste, por exemplo, é realizado pelo testador técnico e analisa se o robô esta respondendo ao comando de ligar. Para que o funcionamento esteja correto, o resultado esperado é que o robô esteja conectado ao CR após o processamento do comando.

Tabela 21 – Testes para o componente comunicação

Componente:	**Comunicação**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber como entrada o comando ligar com uma conexão válida.	Conectar com o robô.
	Receber como entrada o comando ligar com uma conexão inválida.	Mensagem que não consegue se conectar.
	Estando conectado, receber como entrada um comando válido qualquer.	Enviar o comando para o robô.
	Sem estar conectado, receber como entrada um comando válido qualquer.	Mensagem informando que não está conectado ao robô.
	Receber uma imagem do robô.	Chamar o componente Interface para armazenar a imagem.
	Receber solicitação para verificar conexão.	Enviar mensagem de conexão.

Além dos testes individuais dos componentes, também são projetados os testes do produto como um todo, conforme ilustrado na Tabela 22.

Tabela 22 - Teste do produto

Teste do produto		
Quem realiza	O que testar	Resultado esperado
Testador Cliente / Testador Técnico	Ligar o robô desligado.	Conexão estabelecida e configurações iniciais carregadas; Mensagem de robô ligado aguardando comando.
	Ligar o robô ligado.	Nada deve acontecer, pois o robô já está em operação.
	Andar com o robô ligado.	Iniciar a locomoção no sentido selecionado; Receber imagens no tempo de 1 milisegundo.
	Andar com o robô desligado.	Nada acontece
	Parar com o robô em locomoção.	Robô parado em no máximo 1 milisegundo.
	Ajustar dispositivo com o robô ligado. Informar dados de parâmetro para o dispositivo.	Dispositivo ajustado corretamente.
	Ajustar dispositivo com o robô ligado; Informar dados errados ou em branco para os parâmetros.	Nada muda no dispositivo; Mensagem indicando que os parâmetros estão errados.
	Ligar automático com o robô ligado.	Automático ativado; Sistema de navegação em operação.
	Ativar automático com o robô desligado.	Nada acontece.
	Desativar automático com o robô no automático.	Automático desativado; Robô parado aguardando comando.
	Desativar automático com o robô no manual.	Nada acontece.
	Colocar obstáculo perto do robô com o robô em movimento.	Robô parado.

Durante as atividades da segunda fase, validações foram realizadas conforme apresentado na Tabela 23.

Tabela 23 - Documento de validação da fase Elaboração

Documento de Validação	
Data de início do projeto:	27/10/06
Equipe Responsável	
Identificação	Nome
Gerente do cliente	Especificações de [37]
Gerente Técnico	Ana Patrícia
Líder Computação	Ana Patrícia
Líder Mecânica	Hermam
Concepção	
Data da validação	Artefatos
30/10/06	Casos de uso
02/11/06	Lista de requisitos não funcionais
04/11/06	Casa da qualidade Soluções de conflitos
05/11/06	Fatores de risco
08/11/06	Documento de escopo
Elaboração	
Data da validação	Artefatos
10/11/06	Descrição dos casos de uso
15/11/06	Modelagem estrutural (Diagrama de classes)
20/11/06	Modelagem comportamental (Diagrama de estados)
25/11/06	Modelo abstrato (diagrama de componentes)
26/12/06	Modelo concreto (diagrama de componentes particionado)
26/12/06	Casos de teste

6.3. DESENVOLVIMENTO

O desenvolvimento consiste na construção, por cada uma das áreas específicas, dos seus respectivos componentes. Após a construção, estes componentes são integrados, prototipados e testados para gerar o modelo do produto que será enviado à linha de produção.

Esta fase não foi desenvolvida neste experimento prático por falta de recursos financeiros e de tempo.

6.4. ENTREGA

Nesta fase os modelos construídos durante a especificação do robô são reunidos e a documentação do produto é gerada.

7. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho define uma metodologia unificada de construção de produtos mecatrônicos, a MdpM. Esta metodologia contribui e inova em vários aspectos, conforme apresentados na seção 7.1. Além disso, destacam-se na seção 7.2 alguns pontos importantes que não foram abordados e que merecem um estudo mais aprofundado.

7.1. CONCLUSÕES

A metodologia MdpM, proposta neste trabalho, utiliza como base o RUP, integrando a este técnicas de Engenharia Mecânica e Engenharia Elétrica consideradas relevantes para o desenvolvimento de produtos mecatrônicos [17].

O RUP é um processo que se mostra adequado para guiar as atividades de desenvolvimento num ambiente mecatrônico principalmente porque trabalha com o conceito de componentes, usa uma linguagem simples, a UML, é focado em iterações, enfatiza a gerência de requisitos e fornece uma gerência de pessoal bastante eficiente. Além disso, pode-se observar nas fases do RUP certa relação com o modelo seqüencial de desenvolvimento proposto pela Engenharia Mecânica, e no dinamismo proporcionado pelo uso de iterações e pelas disciplinas a semelhança com os conceitos da engenharia simultânea, apropriados à Mecatrônica. Em se tratando de modelos de qualidade, o processo RUP pode ser adaptado a níveis aceitáveis de maturidade, pois consegue atingir quase 100% das práticas definidas para o CMMI [11] nível dois. Isso significa que tem uma boa condução de seus processos e se preocupa com a qualidade dos produtos gerados.

As técnicas de Engenharia Mecânica foram selecionadas porque podem agregar maior valor ao processo MdpM. A matriz da casa da qualidade, por exemplo, facilita a visualização dos requisitos, apresentando-os sob vários aspectos, o que é de suma importância no ambiente multidisciplinar. Esta visualização permite analisar os requisitos, identificar conflitos, representar a importância destes requisitos para o cliente, entre outros. A matriz

morfológica ajuda na definição de como cada componente será efetivamente desenvolvido, através do mapeamento de soluções relevantes e da seleção de concepções de projeto que realmente seguirão no processo de desenvolvimento.

A MdpM prevê também a utilização opcional dos diagramas *funcition blocks*, da Engenharia Elétrica, para mapear os modelos UML em diagramas mais próximos da implementação, na disciplina de análise.

A metodologia proposta neste trabalho tem uma contribuição importante para o desenvolvimento de produtos mecatrônicos primeiramente porque prevê as várias fases do processo, desde o início da especificação de requisitos até a construção física de um modelo do produto. Aproveita o potencial da equipe multidisciplinar na definição dos diversos componentes, não se limitando apenas ao componente de controle. Além disso, a adoção da UML como linguagem de modelagem é um facilitador de comunicação e compreensão entre os envolvidos que permite um detalhado mapeamento do produto e oferece recursos importantes para a modelagem dos diversos aspectos que envolvem a Mecatrônica.

Outra contribuição importante é o foco nas definições interdisciplinares. Ao tratar o produto como um conjunto de componentes que se comunicam a partir de interfaces bem definidas, a MdpM abrange as definições relacionadas a este modelo de componentes e permite certo grau de liberdade para as áreas responsáveis pela sua implementação. Estas áreas ficam livres para efetuarem suas construções de acordo com as especificações definidas conjuntamente. Além disso, uma vez definido o modelo conceitual, a MdpM adota técnicas que apóiam e facilitam o processo de identificação de soluções de implementação, muitas vezes uma atividade complexa de ser resolvida.

Nos produtos mecatrônicos é marcante a interação destes com o ambiente, seja captando informações ou atuando sobre este. Desta interação surgem requisitos particulares que devem e são tratadas pela MdpM, a exemplo dos requisitos temporais. Com a utilização da UML 2.0 a MdpM abrange a especificação destes requisitos permitindo que eles sejam mapeados e representados nos modelos necessários.

Confiabilidade é um ponto importante no desenvolvimento de produtos mecatrônicos que também é abordado pela MdpM sob a forma de validações. Estas validações estão presentes em todas as fases do desenvolvimento do produto, pois a MdpM possui uma disciplina dedicada completamente a elaboração de planos e aplicação de testes em várias etapas do processo, além de simulações e definição de critérios de aceitação do

produto. Para produtos críticos, a MdpM também orienta quanto à utilização de verificações dos modelos desenvolvidos. Estas verificações são importantes para aumentar o grau de confiabilidade do produto construído. A UML também é útil neste ponto, pois os modelos podem ser migrados para verificadores automáticos, facilitando o processo de verificação. Além destes aspectos, o tratamento da confiabilidade pode ser observada também na condução para o tratamento de falhas durante o desenvolvimento do produto. A MdpM orienta o tratamento de falhas com a especificação destas como requisitos não funcionais. Estes requisitos são depois detalhados e servem de guia para decisões de projeto, tais como a duplicação de componentes.

Além destes pontos, um problema comum existente hoje no desenvolvimento de produtos, seja qual for a área, são as constantes mudanças que ocorrem em suas especificações e podem comprometer o processo de desenvolvimento gerando inconsistências, mudanças de prazos e aumentando o custo. Assim como no RUP, a MdpM trata este problema com o gerenciamento de requisitos. São definidos pontos de controle, distribuídos ao longo de todo o processo, que permitem conduzir as atividades de maneira organizada. As mudanças são avaliadas antes de serem efetivamente adotadas e os cronogramas são atualizados. Tudo isso com o acompanhamento e aprovação direta do cliente.

Não só os processos e técnicas, mas também as pessoas possuem um lugar de destaque na metodologia. A MdpM possui um conceito de papel que define as responsabilidades de cada pessoa envolvida no desenvolvimento. Esta característica permite uma melhor organização da equipe, pois cada um sabe seus limites e deveres, e consequentemente uma melhor condução dos trabalhos pode ser implementada.

Desta forma, pode-se dizer que a MdpM contribui para o desenvolvimento de produtos mecatrônicos principalmente porque integra técnicas importantes das diversas áreas e as utiliza mapeadas em uma linguagem simples e expressiva. Esta integração se torna importante na condução e padronização das atividades, mas principalmente na geração de artefatos que formam uma documentação completa e consistente para o produto. Também se destaca por abordar o desenvolvimento aspectos bastante relevantes para a Mecatrônica, tais como, o tratamento do tempo e de falhas. Inova por utilizar como base um processo já consolidado na Engenharia de Software, o RUP, integrado a técnicas de desenvolvimento de produtos. Sinaliza a necessidade de adequar seus processos a um modelo de qualidade, como o CMMI contemplado parcialmente pelo RUP. Por todos estes aspectos permite a construção

de produtos com menor possibilidade de erros, principalmente por se preocupar com validações ao longo de todo o processo, e conduz a uma maior confiabilidade a partir de testes, simulações e da verificação formal dos modelos. Além destes aspectos técnicos, agrega valor ao abordar a gerência de mudanças e de pessoal, possibilitando um melhor controle de todo o processo.

A Tabela 24 apresenta um quadro comparativo da MdpM com alguns dos trabalhos relacionados enfatizando alguns aspectos considerados importantes para o desenvovimento de produtos mecatrônicos.

Tabela 24 – Aspectos relavantes da MdpM comparados a trabalhos relacionados

Aspecto observado	**Metodologia de Bonfe**	**Metodologia CORFU**	**Metodologia de MROZEK**	**Metodologia MdpM**
Métodos / Processos utilizados	Orientação a objetos Análise estruturada Function blocks	Orientação a objetos Function blocks	Orientação a objetos	Processo RUP Orientação a objetos Casa da qualidade Matriz morfológica Function blocks
Padrões	UML IEC	UML IEC	UML	UML
Validação	Não	Testes	Simulação	Testes Simulação
Verificação formal	SIM	Sugere	Não	SIM
Tratamento de falhas	NÃO	NÃO	NÃO	SIM
Aspectos temporais	NÃO	NÃO	SIM	SIM
Foco	Controle	Controle	Controle	Produto mecatrônico

Como pode ser observado, a MdpM diferencia-se das demais metodologias apresentadas principalmente por três aspectos básicos: a utilização do RUP como base para a sua especificação, que já adota os conceitos de orientação a objetos; a incorporação de técnicas de Engenharia de Produtos julgadas pertinente para os produtos mecatrônicos, tais como a casa da qualidade e a matriz morfológica; e o foco no desenvolvimento do produto mecatrônico. Além destes aspectos, preocupa-se com o aumento da confiabilidade do produto gerado com a adoção de métodos de validação e verificação e orienta quanto ao tratamento de falhas e o tratamento de requisitos temporais, característicos dos produtos que possuem interação com o ambiente.

7.2. TRABALHOS FUTUROS

A MdpM aborda o processo de especificação e desenvolvimento de produtos mecatrônicos, tendo como objetivo final a construção de um modelo do produto para ser enviado à linha de produção. No entanto, não inclui qualquer especificação ou planejamento da linha de produção. Assim, como uma extensão à MdpM seria interessante a construção de uma metodologia complementar que permitisse o planejamento do processo de fabricação do produto.

Ainda em relação à produção, um ponto de destaque no desenvolvimento de produtos atualmente são as técnicas de montagem do produto. Geralmente, quando um produto é projetado, além de atender aos requisitos especificados, é ideal que ele seja fácil de ser montado após a sua produção. Neste contexto, um trabalho importante seria também a avaliação das técnicas atuais de montagem para posterior sugestão de integração com a MdpM.

Um aspecto importante no desenvolvimento de produtos de qualquer natureza é o custo. É certo que a utilização de uma metodologia é o primeiro passo na otimização das atividades e prevenção de problemas oriundos de falta de controle. A MdpM é um processo detalhado que aborda várias fases do projeto, tais como validação, e oferece práticas que tendem a diminuir a incidência de problemas no final do projeto. No entanto, nenhum estudo aprofundado sobre técnicas de otimização de custos foi desenvolvido. Assim, uma boa contribuição de projetos futuros seria a realização de uma análise das técnicas existentes para verificar onde elas poderiam ser inseridas na MdpM para construir produtos cada vez mais competitivos. Uma das técnicas que poderiam ser abordadas seria a utilização de métricas que pudessem ajudar os desenvolvedores a medir o esforço necessário para construção de um produto. Isso permitiria melhores definições de recursos, pessoal, prazos e consequentemente custos.

A MdpM possui uma disciplina de validação que objetiva agregar confiabilidade ao processo de desenvolvimento aumentando a qualidade do produto gerado. Dentre as atividades desta disciplina destaca-se a simulação do modelo concreto. Existem várias formas de simulação, com finalidades específicas, tais como o teste de escala, a verificação de comportamento na presença de falhas, entre outros. Considerando a importância da simulação no processo de desenvolvimento de produtos, uma atividade importante é a analise de trabalhos que orientem quanto a realização de simulações para possam enriquecer o processo de validação da MdpM.

O desenvolvimento de produtos mecatrônicos envolve definições de hardware e software que devem colaborar para um objetivo específico. A UML permite extensões que se adaptam a este tipo de mapeamento de forma satisfatória. No entanto, esforços estão sendo realizados entre o grupo responsável pela especificação da UML, a OMG, e o *International Council on System Engineering,* em parceria com empresas, para especificação de uma linguagem que seja voltada a especificações de componentes que integrem software e hardware, a SysML[15]. Assim, um trabalho importante seria o estudo desta linguagem para possivelmente adotá-la como linguagem de modelagem da MdpM.

Os processos atualmente estão cada vez mais sendo submetidos a modelos de qualidade que permitam aprimorá-los. Segundo [11], o RUP, atinge quase 100% das necessidades mapeadas para alcançar o nível dois do CMMI. A MdpM pode se beneficiar desta característica pois utiliza como base o RUP. No entanto, além de não utilizar o RUP em sua totalidade também integra técnicas que não fazem parte deste processo base. Portanto, um trabalho interessante seria avaliar qual o nível de maturidade, em relação ao CMMI, que pode ser atingido por um produto que utilize a MdpM como metodologia de desenvolvimento. Além disso, analisar qual o nível aceitável de qualidade para um produto mecatrônico, sugerindo adequações à MdpM para se ajustar à realidade destes produtos.

Finalmente, a MdpM foi submetida a um experimento prático, a construção de um robô. Este experimento visou avaliar os processos descritos na metodologia, sua consistência, seqüência de atividades, entre outros aspectos. Ele foi realizado não por uma equipe multidisciplinar, mas por um projetista de uma das áreas, seguindo as especificações de um robô já pronto [37]. Um trabalho importante para ser futuramente realizado é a utilização desta metodologia em um projeto real, onde esta possa ser aplicada por uma equipe realmente multidisciplinar e os processos possam ser testados na íntegra.

REFERÊNCIAS

[1] ARATÓ, P. *et al*. *Hardware-Software partitioning in embedded system design*. *IEEE International Symposium on Intelligent Signal Processing (WISP)*, Budapeste, Hungria, p. 197-202, set. 2003.

[2] BECK, K. *Extreme Programming Explained.*Addison-Wesley, 2001, Indianápolis.

[3] BONFE, M. *Design and Verification of Mechatronic Object-Oriented Models for Industrial Control Systems*. *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Lisboa, Portugal, p. 253-270, set 2003.

[4] BONFE, M.; DONATI, C.; FANTUZZI, C.. *An Application of Software Design Methods to Manufacturing Systems Supervision and Control*. IEEE *Conference of Control Applications (CCA)*, Glasgow, Scotland, p. 850-855, set 2002.

[5] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: guia do usuário*. 2.ed. Rio de Janeiro: Elsevier, 2005.

[6] CAI, X.; VYATKIN, V. *Design and Implementation of a Prototype Control System According to IEC 61499*. IEEE, 2003.

[7] COLENAM, D.; HAYES F.; BEAR S. *Introducing objectcharts or how to use statecharts in object – oriented design*. *IEEE Transactions on Software Engineering*. 1.ed. NJ, USA: Piscatawa, Janeiro 1992. p. 9-18 (v. 18).

[8] FERREIRA, C. V.; OGLIARI A.; FORCELLINI E. *SISCOI – Software de Apoio 'a Definição das Especificações de Projeto de Componentes Injetados*, 3. CBGDP Congresso Brasileiro de Gestão de Desenvolvimento de Produtos, 2001, Florianópolis.

[9] FORCELLINI, F. *Projeto Conceitual*. Universidade Federal de Santa Catarina, Centro Tecnológico. Núcleo de Desenvolvimento Integrado de Produtos, Março 2003.

[10] GRANDA, J. *The role of bond graph modeling and simulation in mechatronics systems*: *An integrated software tool: CAMP-G, MATLAB-SIMULINK*. Mechatronics, California: Elsevier, v. 12, p. 1271-1295, 2002.

[11] GRUNDMANN, M. *A CMMI maturity level 2 assessment of RUP. Developer works, USA*. Disponível em http://www-28.ibm.com/developerworks/rational/library/dec05/grundmann. Acessado em 25/09/2006

[12] ISERMANN, R. *Modeling and design methodology for mechatronic systems*. *IEEE/SME Transactions on mechatronics*, v.1, n.1, Março 1996.

[13] KROLL, P.; KRUCHTEN, P. *The Rational Unified Process Made Easy. A Practitioner's Guide to the RUP*. Addison-Wesley, Boston, 2004.

[14] KRUCHTEN, P. *The Rational Unified Process: an introdution*. Massachusetts: Addison Wesley, 2000.

[15] LIMA M.. *IPProcess: Um processo para desenvolvimento de Ip-Cores com prototipação FPGA*, Dissertação de Mestrado, Universidade Federal de Pernanbuco, Recife, 2005.

[16] MACEDO, R., *et al*. *Tratando a previsibilidade em sistemas de tempo-real distribuídos: especificação, linguagens, middleware e mecanismos básicos*. In: ____. *Minicursos do SBRC2004*. Gramado, UFRGS, 2004. Cap X, p.

[17] MAGALHAES, A., *et al*. *Uma metodologia para o desenvolvimento de produtos*

mecatrônicos integrando engenharia de software e engenharia de produtos. XXVI Encontro nacional de engenharia de produção ENEGEP, Fortaleza, Outubro, 2006.

[18] MITRA, R. *Hardware-software partitioning: a case for constraint satisfaction*. *IEEE Intelligent systems*, 2000.

[19] MROZEK, Z. *Design of the mechatronic systems with help of UML diagrams. 3-rd Workshop on robot motion and control.* Bukowy Dworek, Poland, p. 243-248, novembro 2002.

[20] NEGRI, V. *Sistemas Automáticos: Conceitos, Modelos e Projeto*. Universidade Federal de Santa Catarina, Centro Tecnológico. Departamento de Engenharia Mecânica. Laboratório de Sistemas Hidráulicos e Pneumáticos. Florianópolis, Março 1997.

[21] SCHNAKENBOURG C.; FAURE M.; LESAGE J. *Towards IEC 61499 function blocks diagrams verification*. *IEEE International conference on systems, man and cybernetics*, v. 3, 6 pp, outubro, 2002.

[22] SAHA, D.; MITRA, R.; BASU, A. *Hardware software partitioning using genetic algorithm. IEEE 10 th International conference on VLSI Design*, p. 155-160, 1996.

[23] SANTOS, R. *Introdução à programação orientada a objetos usando Java*. São Paulo: Campus, 2003.

[24] SELIC, B. *On the semantic foundations of standard UML 2.0. SFM-RT 2004 Lecture Notes in Computer science LNCS 3185*, Berlin, p. 181-199, 2004.

[25] SOMMERVILLE, I. *Engenharia de software*. 6. ed. São Paulo: Pearson Education-Addison Wesley, 2004.

[26] ROSÁRIO, J. *Princípios de mecatrônica*. São Paulo: Prentice Hall, 2005.

[27] ROSENFELD, H. *et al*. *Gestão de desenvolvimento de produtos*: uma referencia para melhoria da processo. 1. ed. São Paulo: Saraiva, 2006.

[28] RUMBAUGH, J. *et. al*. *Modelagem e projetos baseados em objetos*. Rio de Janeiro: Campus, 1994.

[29] TOMATIS, N. *et al*. A *complex mechatronic system: from design to application*. *International Conference on Advanced Intelligent Mechatronics Proceedings IEEE / ASME*, Italia, p. 278-283, jul. 2001.

[30] THRAMBOULIDIS, K. *Using UML in control and automation: a model driven approach. IEEE International Conference on Industrial Informatied INDIN'04*. Alemanha, 2004.

[31] THRAMBOULIDIS, K. *Development of Distributed Industrial Control Applications: The CORFU Framework. IEEE International Workshop on Factory Communication Systems*. Sweden, p. 1-8, ago. 2002.

[32] THRAMBOULIDIS, K.; TRANORIS, C. *Developing a CASE tool for distributed control applications. The International journal of Advanced Manufacturing Tehnology*, v. 24, n. 1-2, p. 1-12, jul. 2004. Disponivel em http://seg.ee.upatras.gr/thrambo/dev/Papers/IJAMT-04paper.pdf. Acesso em 15 mar 2006.

[33] THRAMBOULIDIS, K. *Model-Integrated mechatronics – toward a new paradigm in the development of manufacturing systems. IEEE Transactions on Industrial Informatics*. v.1, n.1, p. 54 – 61, fev. 2005.

[34] THOMAS, D.; ADAMS, J.; SCHMIT H. *A model and methodology for Hardware-*

Software Codesign. *IEEE Design & Test of Computers*, Los Alamitos, CA, USA p. 6-15, set.1993.

[35] TRANORIS, C.; THRAMBOULIDIS, K. *Integrating UML and the function block concept for the development of distributed control applications*. *Emerging Technologies and Factory Automation*, Lisboa, v. 2, p. 87-94, set. 2003.

[36] WIESE, P. *In the Beginning. Manufacturing Engeneer*, Agosto 1995.

[37] WILLIAMS, D. *PDA robotics: using your personal digital assistant to control your robot*. USA: McGraw-Hill, 2003.

[38] YOURDON, E. *Análise Estruturada Moderna*. Ed. Campus, 1990.

[39]ZHANG M.; FISHER, W.; WEBB, P. *Functional Model Based Object-Oriented Development Framework for Mechatronic Systems. IEEE International Conference on Robotics & Automation*, Taiwan, p. 2153-2158, set. 2003.

GLOSSÁRIO

- Artefato: elemento qualquer consumido, produzido ou modificado por um processo da metodologia.
- Componente: unidade com função específica. Pode ser composta por um ou mais objetos. Um componente pode ser puramente mecânico, eletrônico, computacional ou pode ser híbrido, mecatrônico.
- Encapsulamento: capacidade de ocultar informações dentro de uma estrutura, classe, por exemplo, permitindo que estas sejam cessadas de forma controlada.
- Fatores de Risco: fatores que possam influenciar no sucesso do projeto. Ex. conhecimento, tecnologia, mercado.
- Função Objetivo: função principal do sistema.
- Interface: conjunto de operações que representam o serviço disponibilizado por uma classe ou por um componente. Independe da implementação.
- Metamodelo: linguagem de modelagem utilizada para especificar metodologias.
- Metodologia de Engenharia de Software: processo para a produção organizada de software, com utilização de uma coleção de técnicas predefinidas e convenções notacionais [25].
- Modelo do produto: produto desenvolvido e testado, pronto para ser enviado à linha de produção.
- Necessidade do cliente: solicitações efetuadas pelo cliente para serem implementadas no produto.
- Protótipo do produto: produto desenvolvido conforme especificação, já montado, mas não testado.
- Requisitos: Conjunto de itens que devem ser implementados no produto. São mapeados a partir das necessidades do cliente.
- Requisitos Funcionais: requisitos que representam as funcionalidades que o sistema deverá provê. Oriundos do levantamento de Necessidades do cliente.

- Requisitos não funcionais: requisitos necessários para o bom funcionamento do sistema, mas que não dizem respeito à função objetivo do sistema. Características temporais e de tolerância a falhas estão sendo consideradas como requisitos não funcionais.
- Sistema: produto final gerado pela metodologia MdpM.
- Subcomponente: parte desenvolvida em uma única tecnologia. Um componente híbrido é dividido em subcomponentes para que possa ser ser desenvolvido em uma única tecnologia.

APÊNDICE A - Diagramas UML

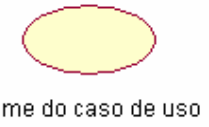
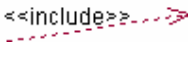
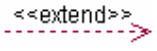
A metodologia proposta neste trabalho utiliza alguns dos diagramas especificados na UML 2.0. Neste apêndice serão apresentados estes diagramas a exemplo do diagrama de casos de uso, o diagrama de classes e o diagrama de componentes.

Os diagramas UML são classificados em estruturais e comportamentais. Os diagramas comportamentais são utilizados para modelar aspectos dinâmicos do produto. Dentre eles, destacam-se o diagrama de casos de uso, o diagrama de atividade e os diagramas de interação. Os diagramas estruturais são utilizados para modelar aspectos estáticos do produto. Dentre eles destacam-se o diagrama de classes e o diagrama de componentes.

a) Diagrama de Casos de Uso

Este diagrama tem como principal objetivo identificar o comportamento desejado para o sistema sem detalhar como este comportamento deverá funcionar. Os casos de uso representam as interações do mundo externo com o sistema e podem representar também casos excepcionais ou variações ao comportamento normal de um caso de uso. São bastante utilizados nas especificações iniciais, pois servem de ferramenta para o entendimento do problema e são de fácil compreensão e representação.

Tabela 25 - Elementos do diagrama de casos de uso

Elemento	Finalidade	Representação
Ator	Papéis que um agente externo desempenha no sistema.	
Caso de uso	Comportamento do sistema.	
Interação	Interação do ator com o caso de uso.	
Relacionamento de generalização	Indica que um caso de uso herda o comportamento do caso de uso pai.	
Relacionamento de inclusão	Indica que um caso de uso incorpora o comportamento de outro.	
Relacionamento estendido	Especifica extensões de comportamento de um caso de uso.	

A MdpM sugere que o diagrama de casos de uso seja fortalecido com o uso de notas sempre que for necessário representar características importantes dos requisitos que não possam ser expressas no diagrama. Como exemplo pode-se citar características temporais de uma tarefa executada pelo produto, tais como o tempo de execução e o *deadline*.

A Tabela 25 apresenta os principais elementos que compõem o diagrama de casos de uso. Cada elemento tem uma finalidade específica e uma representação gráfica. Por exemplo, o elemento Ator cuja finalidade é indicar a existência de um agente externo que desempenha um papel junto ao produto, tem como representação gráfica no diagrama a figura de um boneco.

A Figura 38 apresenta um exemplo de diagrama de casos de uso para um forno de microondas. É fácil observar que no caso de uso Usar forno, o ator Usuário pode utilizar o forno de três maneiras diferentes: selecionando a opção Cozinhar, Descongelar ou Aquecer.

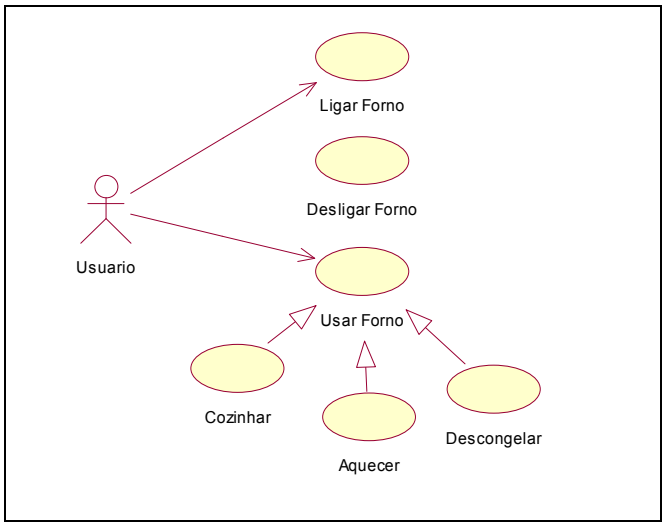


Figura 38 - Diagrama de casos de uso de um forno de microondas

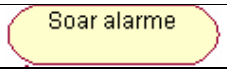
b) Diagrama de Atividade

O diagrama de atividade é utilizado para representar o fluxo de controle entre as atividades desempenhadas por um sistema. Neste diagrama é possível modelar uma seqüência de atividades, representar ramificações de uma atividade ou modelar atividades concorrentes. Esta última característica é muito importante para as aplicações mecatrônicas, onde atividades concorrentes são freqüentes devido à característica reativa destes produtos.

A Tabela 26 ilustra os principais elementos em um diagrama de atividades, sua finalidade e representação visual. Por exemplo, o elemento Atividade, tem como finalidade

indicar execuções realizadas no diagrama e é representada por um retângulo de cantos arredondados com o nome da atividade ao centro.

Tabela 26 - Elementos do diagrama de atividades

Elemento	Finalidade	Representação
Início	Representa o início de um fluxo.	
Término	Representa o término de um fluxo.	
Atividade	Execuções realizadas no diagrama.	
Fluxos de controle	Passagem de uma atividade para outra.	
Ramificação	Caminhos alternativos para o fluxo de controle.	
Bifurcação e união	Representação de atividades concorrentes, paralelismo.	

A Figura 39 ilustra um exemplo de diagrama de atividades para a utilização de um forno de microondas. É representada uma coluna de atividades do usuário e uma de atividades do forno. O processo começa com o usuário depositando o alimento, fechando a porta do forno e solicitando que este inicie o cozimento. Durante todo o processamento existem atividades paralelas de monitoria do forno para efeitos de segurança. Quando o processamento do forno termina, é soado um alarme e o usuário pode então retirar o alimento finalizando o processo.

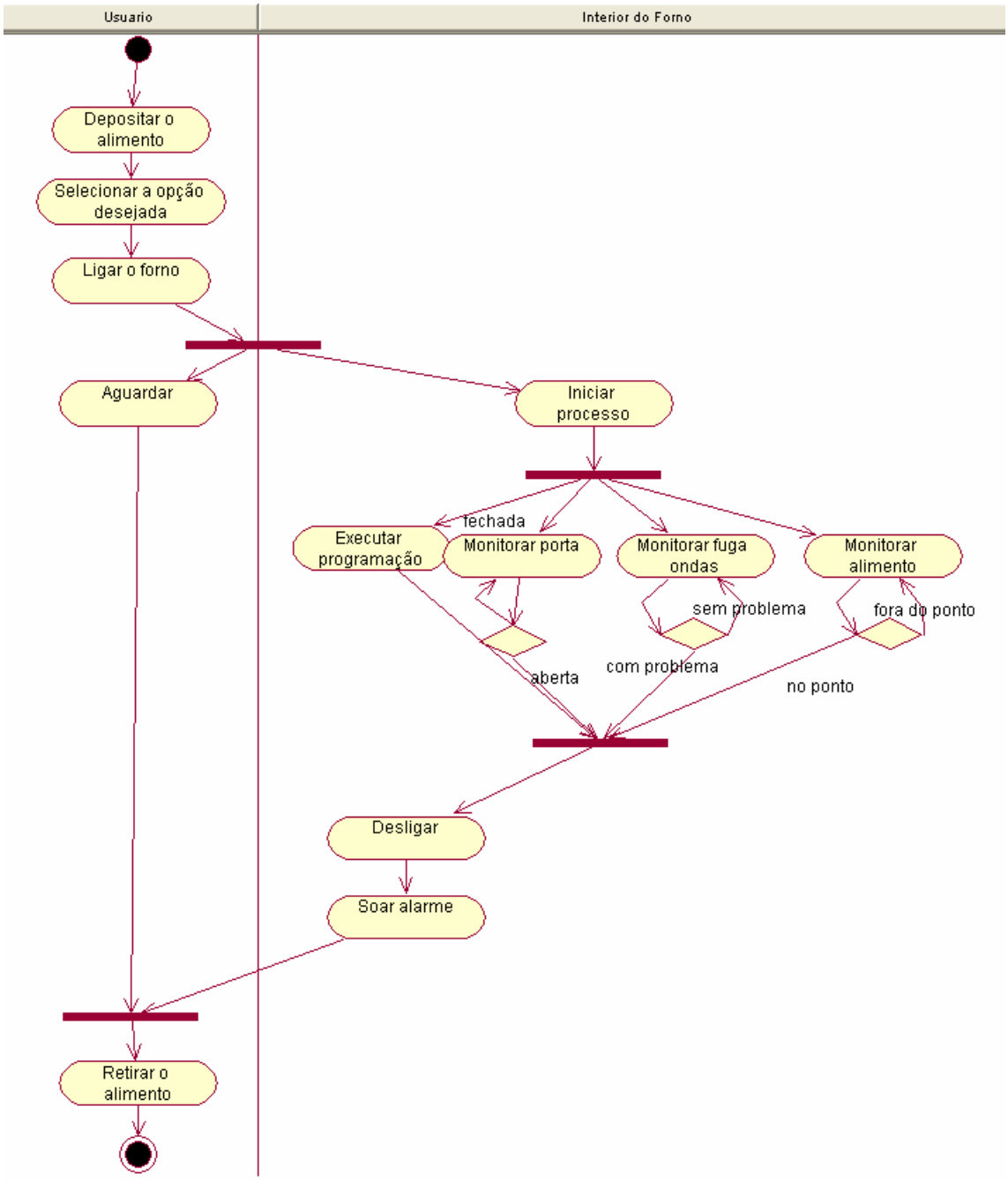


Figura 39 - Exemplo de diagrama de atividades de um forno de microondas

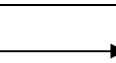
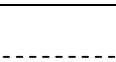
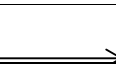
c) Diagramas de Seqüência

O diagrama de seqüência é um tipo de diagrama de interação que serve para ilustrar a interação entre os objetos de um sistema a partir da troca de mensagens entre eles.

O diagrama de seqüência é adequado para especificar processos mecatrônicos, pois permite visualizar a ordenação temporal das mensagens. Desta forma, este diagrama permite que sejam especificadas periodicidades e tempo de execução das mensagens.

A Tabela 27 ilustra os principais elementos de um diagrama de seqüência, a finalidade de cada um destes elementos e sua representação visual no diagrama.

Tabela 27 - Elementos do diagrama de seqüência

Elemento	Finalidade	Representação
Objetos	Objetos cuja interação será representada.	
Linhas da vida	Expressam o tempo de vida do objeto.	
Foco de controle	Ilustra o período de execução de um método pelo objeto.	
Mensagem Síncrona	Chamadas a métodos de um objeto de maneira síncrona.	
Retorno de uma mensagem síncrona		
Mensagem Assíncrona	Chamadas a métodos de um objeto de maneira assíncrona.	
Tags	Indicam execuções opcionais, condicionais, paralelas ou laços.	

Como pode ser observado, a Figura 40 apresenta um exemplo de diagrama de seqüência para a utilização de um forno de microondas. Neste exemplo é mapeada a seqüência de mensagens disparadas quando o forno inicia a execução de uma programação.

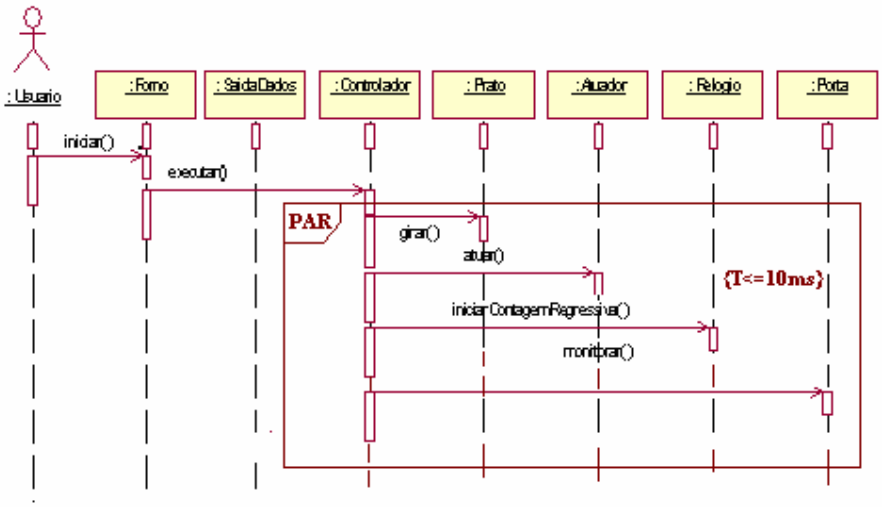


Figura 40 – Exemplo de diagrama de seqüência para o forno de microondas

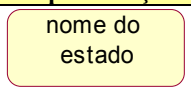
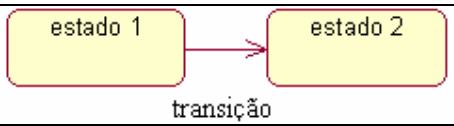
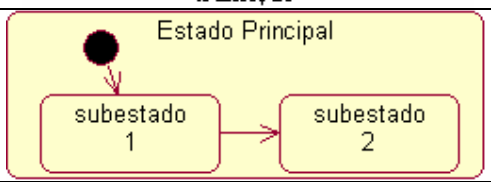
Nota-se que são executadas quatro mensagens em paralelo: uma mensagem para o atuador responsável pela emissão de microondas; uma mensagem para o prato indicando que ele deve começar a girar; uma mensagem para o relógio para que seja iniciada uma contagem regressiva de acordo com a programação; e uma mensagem para a porta (sensor) para que se inicie a monitoria. Estas quatro mensagens devem ter uma entrega de no máximo 10ms especificado com o símbolo {t<=10ms}.

d) Diagrama de Estados

O diagrama de estados permite mapear os estados possíveis para um determinado objeto do sistema e ilustra a transição entre estes estados na ocorrência de eventos. Pode ser utilizado para representar interações com o ambiente, mapear estados seguros, entre outros. Pode ser aplicado também para geração de autômatos na verificação formal de sistemas.

Análogamente, a Tabela 28 detalha os elementos do diagrama de estados.

Tabela 28 - Elementos do diagrama de estados

Nome	Finalidade	Representação
Estado	Situação do objeto em um dado momento.	
Estado inicial	Estado inicial do objeto.	
Estado final	Estado final do objeto.	
Evento	Ocorrência significativa no tempo e espaço que pode acarretar uma mudança de estado.	
Transição	Passagem de um estado para outro.	
Subestados	Estados aninhados. Pode ser independentes ou concorrentes.	
Esccolha	Define uma bifurcação onde uma transição pode seguir caminhos distintos.	

Como pode ser observado no exemplo ilustrado na Figura 41, um forno de microondas inicialmente possui dois estados principais, ligado e desligado, que indicam se ele está ou não sendo alimentado por uma força (energia). Quando o forno está ligado, ele pode assumir vários subestados: parado, em execução, e monitorando a temperatura. O estado em execução indica que o forno está em funcionamento. Neste caso, novos subestados precisam ser especificados, pois o forno terá que controlar atividades paralelas que poderão levar a estados diversos, tais como em execução e estados de monitoria, necessários para garantir a segurança do forno.

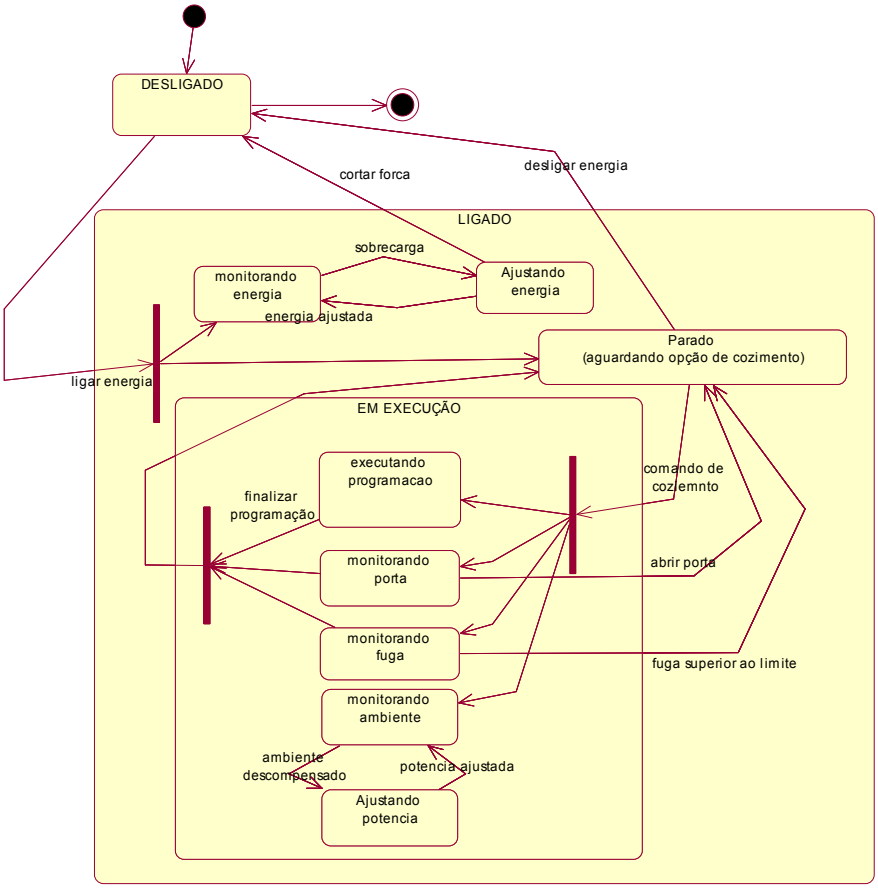


Figura 41 - Exemplo de diagrama de estados para um forno de microondas

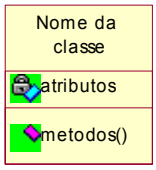
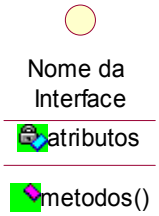
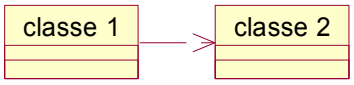
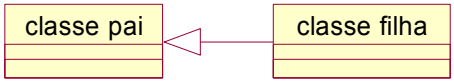
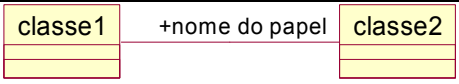
e) Diagrama de Classes

O diagrama de classes oferece uma visão estática e estrutural do sistema em um nível ainda conceitual. Nele é modelada a estrutura das classes que compõem o sistema e seus relacionamentos além das suas interfaces de acesso.

Como o diagrama de classes está em um nível conceitual, onde decisões de implementação ainda não foram tomadas, esta característica é particularmente interessante para a Mecatrônica, pois permite visualizar a estrutura do produto e mapear os requisitos nestas estruturas, sem que seja necessário definir como cada uma delas será implementada, ou seja, se serão mecânicas, eletrônicas ou computacionais. Permite também definir claramente as interfaces que cada classe terá para se comunicar com as demais.

A Tabela 29 ilustra os principais elementos de um diagrama de classes, sua finalidade e representação gráfica.

Tabela 29 - Elementos do diagrama de classes

Nome	Finalidade	Representação
Classe	Descrição dos objetos que possuem os mesmos atributos, operações e relações.	
Interface	Estereótipo que define os serviços que a classe provê ou requer.	
Relacionamento de dependência	Índica que uma classe usa outra classe.	
Relacionamento de generalização	Indica uma relação entre classes (pais) gerais e classes mais específicas (filhas).	
Relacionamento de associação	Indica uma relação estrutural de conexão entre objetos de classes diferentes.	
Papel	Define o comportamento de uma classe em relação à outra.	

A Figura 42 apresenta um exemplo de diagrama de classes. Pode-se observar três classes: o forno, a configuração e um controlador do forno. Cada classe define uma interface de comunicação com o mundo externo e com as demais classes. Por exemplo, a classe Forno implementa a interface IEForno que permite a um usuário selecionar uma opção pré-programada para execução do forno.

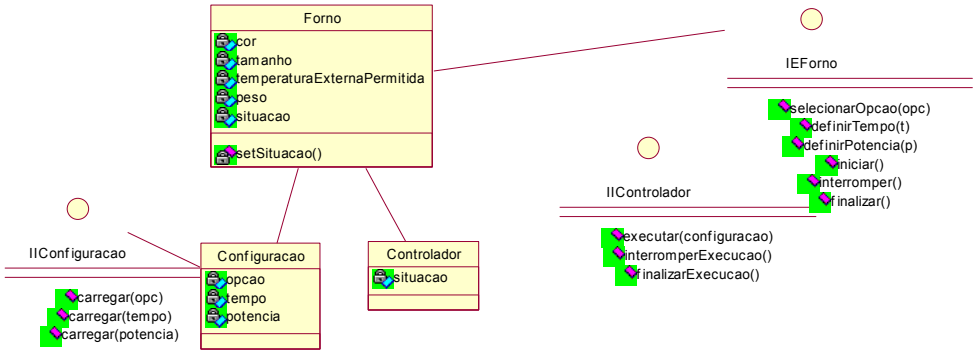


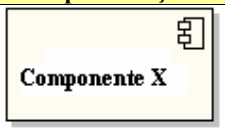
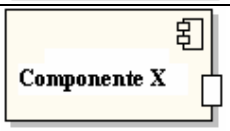
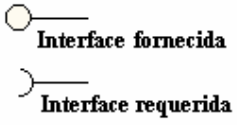
Figura 42 – Exemplo de um diagrama de classes de um forno de microondas

f) Diagrama de Componentes

O diagrama de componentes ilustra a organização de um produto como uma composição de componentes, portas, interfaces e relacionamentos. Estes componentes podem ser formados por uma ou mais classes de maneira que possam ser conectados para compor um produto. Um componente também pode ser composto de outros componentes menores, chamados de subcomponentes. Com isso tem-se uma maior flexibilidade para construção, reutilização e manutenção.

A Tabela 30 ilustra os principais elementos de um diagrama de componentes, sua finalidade e representação gráfica.

Tabela 30 - Elementos do diagrama de componentes

Elemento	Finalidade	Representação
Componente	Elemento responsável por um determinado serviço. Possui portas de comunicação com outros componentes, a partir de interfaces bem definidas.	
Porta	Janela (retângulo localizado na extremidade do componente) que aceita mensagens de acordo com a interface definida.	
Interface	Especifica os serviços de um componente. Existem dois tipos de interfaces: fornecida e requerida. A interface fornecida indica o serviço que o componente provê para outros componentes. A interface requerida indica os serviços que outro componente deve provê para se comunicar com um componente.	

A Figura 43 apresenta um exemplo de diagrama de componentes de um forno de microondas com cinco componentes. O componente Entrada_Saída é responsável pela interação com o usuário do forno, e de acordo com o que for solicitado, aciona o componente Controlador. O controlador provê uma interface chamada Ligar forno que pode ser utilizada de duas maneiras: através da porta Programação, se o usuário selecionar uma programação já existente no forno, ou através da porta Aleatório, se o usuário desejar informar uma programação aleatória. O controlador então inicia a atuação, chamando o componente Atuador que por sua vez atua sobre o componente Carcaça do forno, ligando os motores para emissão de microondas e giro do prato. Com o objetivo de aumentar a confiabilidade do produto, o controlador também inicia uma monitoria no forno através da leitura de sensores.

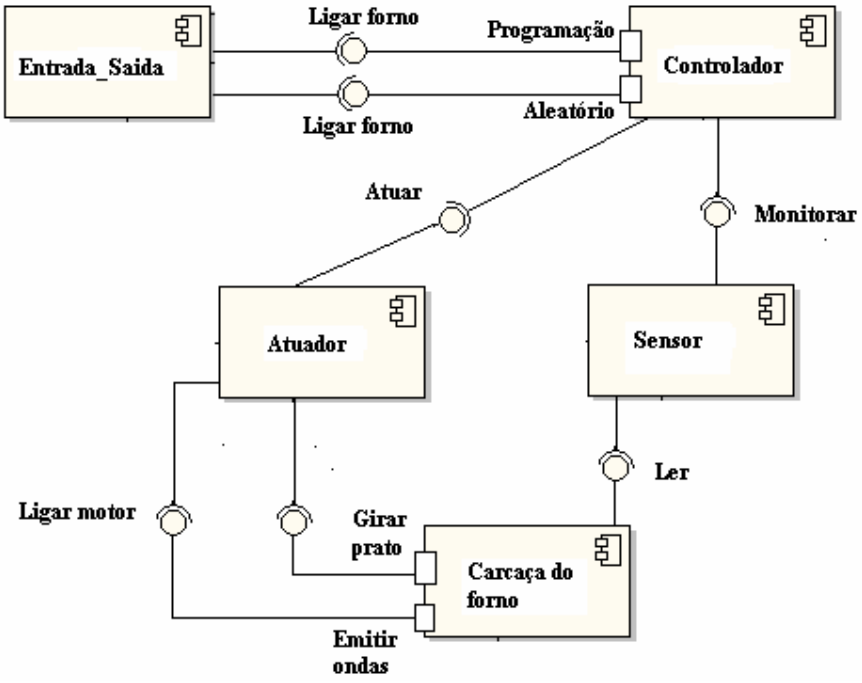


Figura 43 – Exemplo do diagrama de componentes de um forno de microondas

APÊNDICE B - Processo da MdpM

O objetivo deste apêndice é fornecer um guia ao leitor para utilização da metodologia, detalhando cada disciplina definida na MdpM.

No entanto, antes de iniciar o processo, a primeira tarefa a ser realizada é a definição das pessoas que deverão compor a equipe multidisciplinar. Para isso deve-se definir claramente quem exercerá os papéis de representante do cliente, gerente de projeto e líderes para cada uma das áreas envolvidas. Os demais papéis podem ser identificados ao longo do processo, quando forem necessários.

a) Disciplina Requisitos e Escopo

A disciplina Requisitos e Escopo possui várias atividades conforme apresentado no diagrama a seguir. Cada uma destas atividades possui um guia de utilização representado por uma tabela com os passos a serem seguidos para realizar a atividade de forma satisfatória.

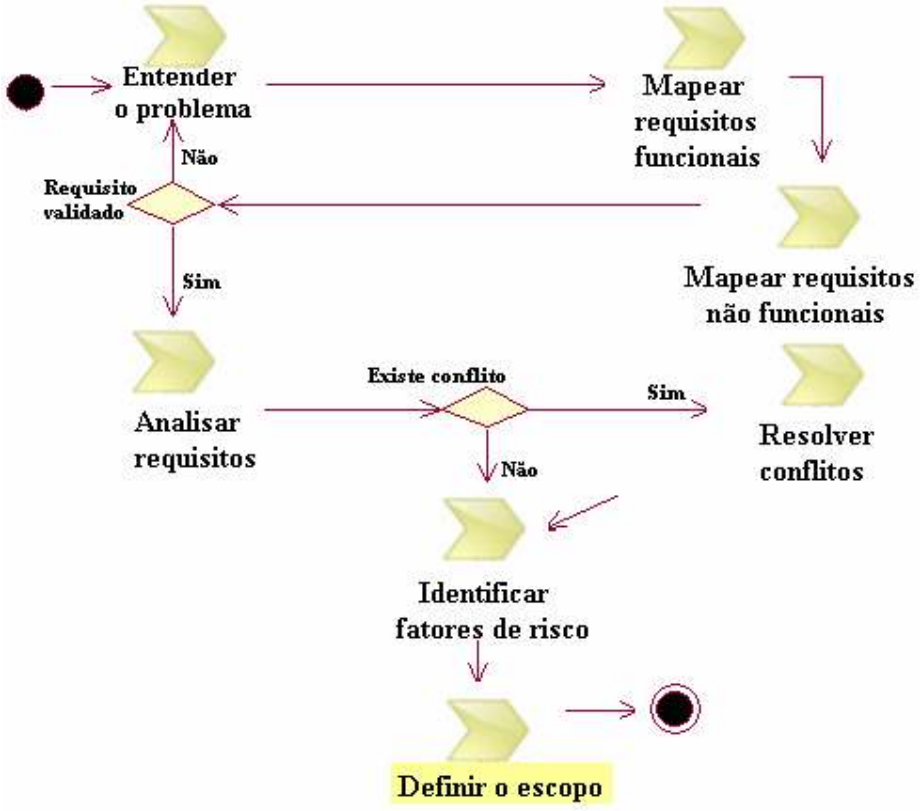


Figura 44 – Atividades da disciplina Requisitos e Escopo

Os trabalhos da disciplina Requisitos e Escopo se iniciam em reuniões com o cliente para entendimento do problema, como representado na Tabela 31. Estas reuniões vão gerar como resultado um documento contendo a descrição do produto.

Tabela 31 - Detalhamento da atividade Entender o problema

Atividade: Entender o problema
Objetivo: Descrever de maneira clara o que se entende sobre o produto.
Entrada: Necessidades do cliente, custos, normas, restrições.
Passos: • Realizar reuniões com o cliente; • Descrever a função principal do produto (função objetivo); • Validar o objetivo do produto com o cliente.
Artefatos: Documento de descrição do produto.
Fase: Concepção.
Papéis: Cliente, gerente e líderes técnicos das áreas específicas.

Após a atividade acima, o trabalho prossegue em reuniões de levantamento dos requisitos funcionais do produto (Tabela 32).

Tabela 32 - Detalhamento da atividade Mapear requisitos funcionais

Atividade: Mapear requisitos funcionais
Objetivo: Identificar as necessidades do cliente e mapeá-las em requisitos funcionais.
Entrada: Necessidades do cliente, custos, normas, restrições.
Passos: • Realizar reuniões com o cliente; • Identificar as necessidades do cliente; • Mapear os requisitos funcionais; • Descrever cenários a partir da construção de diagramas de casos de uso com as visões necessárias do produto; • Elaborar diagramas de atividades para detalhar o funcionamento dos casos de uso mais complexos dentro do cenário; • Criar um glossário com os principais termos associados ao projeto; • Validar artefatos com o cliente.
Artefatos: Documento de necessidades do cliente, casos de uso, diagrama de atividades e glossário de termos.
Fase: Concepção.
Papéis: Cliente e líderes técnicos das áreas específicas. Eventualmente pode participar o gerente.
Observação: Sempre que um caso de uso expresse um requisito que deve ser tolerado a falhas, tratar esta tolerância como uma exceção e ilustrá-la como uma extensão do caso de uso normal, usando o estereótipo <<extends>>.

O cliente será responsável por informar à equipe as suas necessidades, gerando o documento de necessidades. A partir destas necessidades, a equipe efetua o mapeamento dos requisitos funcionais em diagramas de casos de uso. Requisitos muito complexos têm seu funcionamento modelado em diagramas de atividades para que fique registrada toda a seqüência de atividades necessárias para o seu funcionamento. Sugere-se também a

construção de um glossário com termos importantes sobre o produto. Desta forma, vários artefatos são gerados, sempre com o foco no entendimento, e todos eles devem ser validados no final da atividade.

Na atividade Mapear os requisitos não funcionais (Tabela 33), os requisitos não funcionais são especificados para o produto e registrados no documento de requisitos não funcionais. Esta atividade é desempenhada essencialmente pelos líderes das áreas específicas, ficando o cliente responsável pela validação do documento gerado no final dos trabalhos.

Tabela 33 - Detalhamento da atividade Mapear requisitos não funcionais.

Atividade: Mapear os requisitos não funcionais
Objetivo: Identificação de requisitos que têm impacto sobre o projeto, mas que não dizem respeito à funcionalidade do produto, tais como: requisitos operacionais (desempenho, falhas, temporais, etc); restrições (geométricas, espaciais, fatores de risco para a construção do produto); normas (segurança, uso, qualidade), entre outros.
Entrada: Necessidades do cliente, custos, normas, restrições, requisitos funcionais.
Passos: • Realizar reuniões técnicas; • A partir dos documentos gerados até o momento, identificar os requisitos não funcionais; • Construir uma lista com os requisitos não funcionais classificando-os quanto a uma categoria: segurança, ergonomia, restrições, normas, tempo, falhas ou outra categoria importante para o domínio da aplicação que está sendo modelada; • Validar os requisitos com o cliente, caso necessário.
Artefatos: Documento de requisitos não funcionais.
Fase: Concepção.
Papéis: Líderes técnicos das áreas específicas. Cliente, para efetuar a validação.

Tabela 34 - Detalhamento da atividade Analisar requisitos

Atividade: Analisar requisitos
Objetivo: Efetuar um cruzamento entre as necessidades do cliente e os requisitos mapeados para o produto, para comparar o impacto existente entre eles e para comparar com os produtos similares já existentes no mercado.
Entrada: Necessidades do cliente, requisitos, outros produtos.
Passos: • Efetuar o cruzamento entre necessidades e requisitos. Neste cruzamento devem ser usados todos os requisitos funcionais e não funcionais; • Identificar a importância, em uma escala de 0 a 5, de cada necessidade do cliente; • Pesquisar produtos similares no mercado; • Efetuar comparação com os produtos similares existentes no mercado, de acordo com a importância definido pelo cliente; • Identificar metas para o produto a ser construído.
Artefatos: Casa da qualidade.
Fase: Concepção.
Papéis: Líderes técnicos das áreas específicas, o gerente e o cliente.

A análise dos requisitos mapeados, (Tabela 34) é a próxima atividade desta disciplina e consiste na análise de todos os requisitos, sejam eles funcionais ou não

funcionais. Esta análise é realizada com o apoio da matriz da casa da qualidade e permite obter uma visão mais ampla do produto, relacionando os requisitos para identificar conflitos, destacar prioridades e compará-los com produtos similares que existam no mercado. Por esta razão, esta é uma atividade que deve ser realizada por toda a equipe multidisciplinar.

A atividade seguinte consiste da análise dos conflitos (Tabela 35) identificados na matriz da casa da qualidade e objetiva antecipar problemas que possam vir a prejudicar o bom andamento do projeto. Desta forma, para cada conflito identificado deverá ser apontada uma possível alternativa de solução, no documento de soluções de conflitos. Soluções detalhadas podem ser adiadas, desde que a equipe visualize que é possível resolver o conflito. Caso contrário, o produto, com as especificações definidas, pode gerar uma especificação de projeto inviável.

Tabela 35 - Detalhamento da atividade Resolver conflitos

Atividade: Resolver conflitos
Objetivo: A partir da análise dos requisitos, identificar os possíveis conflitos existentes entre eles e apontar uma solução.
Entrada: Casa da qualidade.
Passos: • Identificar os conflitos a partir do telhado da casa da qualidade; • Apontar soluções para cada conflito encontrado; • Gerar documento com soluções.
Artefatos: Documento com Soluções de conflitos.
Fase: Concepção.
Papéis: Líderes técnicos das áreas específicas.

Tabela 36 - Detalhamento da atividade Identificar fatores de risco

Atividade: Identificar fatores de risco
Objetivo: Identificar os fatores que podem oferecer risco ao bom andamento do projeto.
Entrada: Casa da qualidade, normas, restrições.
Passos: • Realizar reuniões técnicas; • A partir dos documentos gerados até o momento, identificar os fatores de risco; • Construir uma lista com os fatores de risco.
Artefatos: Documento de fatores de risco.
Fase: Concepção.
Papéis: Líderes técnicos das áreas específicas. Eventualmente pode participar o cliente e o gerente.

Todo projeto pode envolver riscos. Estes riscos já foram minimizados com a identificação e solução dos conflitos. No entanto, outros fatores além dos conflitos podem significar riscos para o projeto (Tabela 36). Como exemplo, pode-se citar a necessidade de treinamento de pessoal em uma tecnologia específica para torná-las aptas a operar o produto

quando este estiver pronto. Estes riscos devem ser identificados e ações devem ser tomadas para resolvê-los desde o início do projeto.

A atividade final da disciplina Requisitos e Escopo é a definição do escopo do produto. Esta é uma tarefa que envolve toda a equipe e consiste em selecionar, dentre os requisitos identificados, qual será o escopo do produto que será efetivamente construído. O trabalho é realizado com base nos artefatos gerados até o momento e deve ser registrado em ata de reunião.

Tabela 37 - Detalhamento da atividade Definir o escopo

Atividade: Definir o escopo
Objetivo: Selecionar, dentre os requisitos mapeados, quais os que deverão fazer parte da versão do produto que será construído.
Entrada: Casa da qualidade.
Passos: • Realizar reunião com o usuário; • Selecionar os requisitos (funcionais e não funcionais) que farão parte da versão a ser construída do produto; • Elaborar um documento com os requisitos selecionados e aprovados na reunião.
Artefatos: Documento de escopo.
Fase: Concepção.
Papéis: Gerente, cliente e líderes de todas as áreas.

b) Disciplina de Análise

A disciplina Análise agrupa um conjunto de atividades responsáveis pelo detalhamento do produto e definição da estrutura estática e comportamental, conforme ilustrado na Figura 45

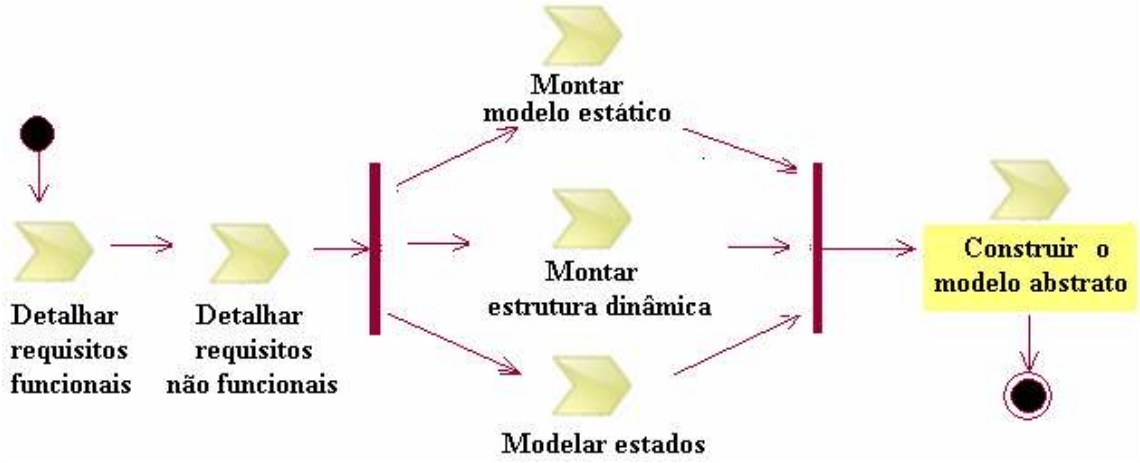


Figura 45 – Atividades da disciplina Análise

Cada uma das atividades tem a sua execução definida em uma tabela que serve de guia para a equipe de desenvolvimento. As tabelas da disciplina Análise são descritas a seguir.

A primeira atividade é o detalhamento dos requisitos que compõem o escopo do produto. Basicamente, os casos de uso identificados são descritos com a ajuda do cliente. A identificação de quem interage com o caso de uso, quais as condições que devem estar satisfeitas para que o caso de uso seja executado, como será a execução do caso de uso detalhadamente e a definição de exceções, são exemplos de informações que devem constar no documento de descrição dos requisitos funcionais, gerado nesta atividade.

Tabela 38 - Detalhamento da atividade Detalhar requisitos funcionais

Atividade: Detalhar requisitos funcionais
Objetivo: Produzir uma descrição detalhada de cada requisito funcional que compõe o escopo do produto.
Entrada: Casos de uso, diagrama de atividades.

Passos:
• Realizar reuniões com o cliente;
• Revisar os casos de uso especificados durante a disciplina de Requisitos e Escopo;
• Elaborar documento de Descrição dos requisitos funcionais de acordo com o detalhamento fornecido pelo cliente. Este documento deverá conter para cada caso de uso definido as seguintes características:
  o quem interage com o caso de uso (atores);
  o quais as pré-condições para que o caso de uso seja executado;
  o como será o processamento normal executado pelo caso de uso;
  o exceções ao processamento normal e;
  o pós-condições de processamento.
• Validar a Descrição dos requisitos com o cliente.

Artefatos: Documento de descrição dos requisitos funcionais.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas e cliente. Pode ser solicitada a presença de especialistas de domínio.
Observação: Vide modelo do documento de descrição dos requisitos funcionais no experimento prático.

Tabela 39 - Detalhamento da atividade Detalhar requisitos não funcionais

Atividade: Detalhar requisitos não funcionais
Objetivo: Produzir uma descrição detalhada de cada requisito não funcional associado ao sistema.
Entrada: Documento de requisitos não funcionais.
Passos: • Realizar reuniões técnicas; • Analisar a lista de requisitos não funcionais; • Elaborar o documento de descrição dos requisitos não funcionais, detalhando cada um dos requisitos e estabelecendo metas e / ou métricas que possam quantificar o requisito e facilitar a sua validação; • Para requisitos temporais especificar: *deadline*, tempo de execução e periodicidade; • Validar a descrição dos requisitos com a equipe.
Artefatos: Documento de descrição dos requisitos não funcionais.
Fase: Elaboração.
Papéis: Líderes das áreas específicas, especialistas de domínio e eventualmente o cliente.

Os requisitos não funcionais também são detalhados a partir da descrição do seu correto funcionamento e da definição de características específicas tais como as relativas aos requisitos temporais.

A Tabela 40 apresenta os passos para iniciar a construção da arquitetura do produto a partir do diagrama de classes. Geralmente as classes são mapeadas a partir dos requisitos funcionais identificados no início dos trabalhos. No entanto, é importante analisar também o documento de requisitos não funcionais para verificar se eles podem ser atendidos com a estrutura modelada. A partir desta análise, modificações na estrutura podem ser necessárias.

Tabela 40 - Detalhamento da atividade Montar o modelo estático

Atividade: Montar o modelo estático
Objetivo: Construir a arquitetura preliminar do produto.
Entrada: Documento de detalhamento dos requisitos funcionais e não funcionais.
Passos: • A partir dos documentos gerados, definir as classes do sistema; • Definir os atributos; • Identificar o comportamento (métodos) de cada classe; • Definir as interfaces da classe; • Identificar os relacionamentos existentes entre as classes; • Verificar se os requisitos podem ser atendidos com a estrutura modelada; • Elaborar a descrição das classes detalhando suas características.
Artefatos: Diagrama de classes, descrição das classes.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas, arquiteto e especialista de domínio.

Até este nível de modelagem, a análise está sendo realizada abstraindo-se detalhes de implementação, preocupando-se apenas com o conceito, o objetivo, o problema. Não tendo sido necessário definir o que será construído em Mecânica, Elétrica ou Computação. Por exemplo, a definição de um objeto que represente uma porta de um cofre pode conter atributos (dados), métodos (operações) e as interfaces abrir e fechar a porta. No entanto, esta porta pode ser mecânica, construída com algum tipo de metal e conter componentes eletrônicos e de software para validar senha. Pode também ser uma porta virtual. Abstraindo-se ainda mais, pode ser uma porta construída com um campo magnético e, portanto, sem nenhum componente mecânico. Desta maneira, o modelo estático enfatiza o conceito. Seja qual for à solução de implementação que venha a ser adotada no futuro, o conceito modelado é válido.

Assim como a estrutura estática, a estrutura dinâmica do produto também precisa ser detalhada. Ela será responsável por definir como os objetos das classes especificadas na estrutura estática se comportam, ou seja, como acontece a comunicação entre eles. Esta definição pode ser realizada sob várias visões onde cada uma delas vai representar a interação entre os objetos de um determinado conjunto de classes para executar um requisito específico. É recomendado que se modele uma visão para cada um dos casos de uso definidos ou pelo menos para os mais importantes.

Tabela 41 - Detalhamento da atividade Modelar estrutura dinâmica

Atividade: Modelar estrutura dinâmica
Objetivo: O objetivo principal desta atividade é modelar a comunicação entre objetos do produto, a partir do envio de mensagens. Para produtos com características temporais, esta atividade assume papel fundamental, pois permite modelar o comportamento das mensagens no tempo.
Entrada: Diagrama de classes, descrição dos requisitos funcionais e não funcionais.
Passos: • Identificar a seqüência de interação entre as classes; • Caso necessário, definir a resposta de cada interação (resposta das mensagens ao objeto); • Caso existam requisitos temporais mapeados, devem ser representadas nestes modelos as características temporais definidas na especificação destes requisitos para as tarefas envolvidas, tais como: periodicidade, *deadline* e tempo de execução.
Artefatos: Diagrama de seqüência / colaboração.
Fase: Elaboração.
Papéis: Líderes das áreas específicas, arquiteto e especialistas de domínio.

Tabela 42 - Detalhamento da atividade Modelar estados

Atividade: Modelar estados
Objetivo: Esta atividade faz parte da estrutura dinâmica do sistema e mapeia para cada classe os possíveis estados que ela pode assumir na ocorrência de mensagens do sistema.
Entrada: Documento de descrição dos requisitos funcionais e não funcionais.
Passos: • Identificar, para cada classe, os possíveis estados; • Identificar os eventos que podem resultar em mudança de estados; • Mapear os estados e eventos no diagrama de estados.
Artefatos: Diagrama de estados.
Fase: Elaboração.
Papéis: Líderes das áreas específicas, arquiteto e especialistas de domínio.

A modelagem do comportamento do objeto se aprofunda com a geração do diagrama de estados que o representa (Tabela 42). A criação deste modelo é importante também para a disciplina Verificação, pois é a partir dele que se inicia o processo de verificação formal dos requisitos do produto, conforme apresentado nas próximas sessões. Além disso, como o diagrama de estados representa todos os estados possíveis para o produto,

podem-se modelar aqui os estados seguros que poderão ser utilizados em casos de detecção de falhas.

A construção do *modelo abstrato* do produto (Tabela 43) é um passo significativo na definição de uma arquitetura que dará origem a arquitetura física do produto. Nesta atividade definem-se os componentes que deverão ser construídos e que juntos irão compor o modelo do produto. Estes componentes serão definidos a partir do diagrama de classes. Cada uma das classes pode formar um componente, ou um componente poderá ser formado pelo agrupamento de várias classes. A utilização de componentes é indicada para a mecatrônica porque permite a reutilização, padronização e conseqüente rápida construção a custos mais acessíveis.

Tabela 43 - Detalhamento da atividade Construir o modelo abstrato

Atividade: Construir o modelo abstrato
Objetivo: Definir um modelo de componentes que represente a arquitetura do produto. O modelo, denominado modelo abstrato, representa os componentes do produto como caixas que se comunicam trocando mensagens através de interfaces.
Entrada: Diagramas de classes, documento de detalhamentos dos requisitos funcionais e não funcionais.
Passos: • Compor os componentes do produto identificando as classes que poderão ser agrupadas, considerando características de reutilização, montagem, manutenção e funcionamento; • Definir os relacionamentos e interações, entre os componentes; • Definir a interface disponível e requerida para cada componente; • Identificar os componentes que deverão ter um tratamento especial quanto à presença de falhas, determinando a forma de detectar e tolerar estas falhas.
Artefatos: Modelo abstrato (diagrama de componentes).
Fase: Elaboração.
Papéis: Líderes das áreas específicas, arquiteto e especialistas de domínio.

A seleção das classes que deverão compor um componente é uma tarefa importante realizada pela equipe multidisciplinar. Em geral, quando se define um componente deve-se: agrupar classes afins que juntas executem de maneira completa uma funcionalidade; visar sempre o reaproveitamento do componente; considerar a facilidade de montagem e a facilidade de manutenção, pois isso pode significar tempo e custo tanto no processo de fabricação, quanto no processo de evolução do produto; especificar claramente a interface entre os componentes, pois este é um ponto crucial para o seu bom funcionamento. Estas características precisam ser ponderadas pela equipe, pois produtos com um grande número de componentes podem ser fáceis de montar, mas podem demandar uma interface complexa. Além disso, quanto maior o número de componente maior a possibilidade de ocorrência de erros, pois pode haver erro de comunicação, erro de definição da interface, entre outros.

Assim, a definição dos componentes é uma atividade importante da análise, pois é preciso encontrar um ponto de equilíbrio para ter um número eficiente de componentes.

É importante deixar claro que até o momento soluções de implementação ainda não foram definidas. Portanto, um componente poderá vir a ser totalmente mecânico, totalmente eletrônico, totalmente computacional ou híbrido, envolvendo mais de uma tecnologia na sua construção.

Outro passo importante desta atividade é a identificação de quais componentes precisam ser monitorados quanto à detecção de possíveis falhas e como estas serão tratadas de maneira que possam assumir um estado seguro para o produto, não comprometendo o resultado final. Um exemplo pode ser a duplicação espacial de elementos, que já pode ser indiada no modelo abstrato com a duplicação de algumas classes.

c) Disciplina Identificação de Solução

A disciplina Identificação de Solução ganha destaque pela responsabilidade em definir a solução tecnológica para cada componente do modelo abstrato. Para tanto, um conjunto de atividades foram especificadas, conforme ilustrado na Figura 46.

A primeira atividade na busca pela solução tecnológica adequada é a identificação dos princípios de solução possíveis para cada um dos componentes do modelo abstrato (Tabela 44). Entende-se por princípio de solução, uma forma possível de implementação para um componente. Por exemplo, para um forno de microondas, considerando um componente que represente a entrada de dados, princípios de solução possíveis poderiam ser: um teclado numérico com botões representando os números de zero a nove e um botão para ligar e desligar o forno; ou uma área de apresentação para números e opções de menu em uma tela *touch screen*. Podem ser mapeados quantos princípios de solução a equipe desejar para um mesmo componente.

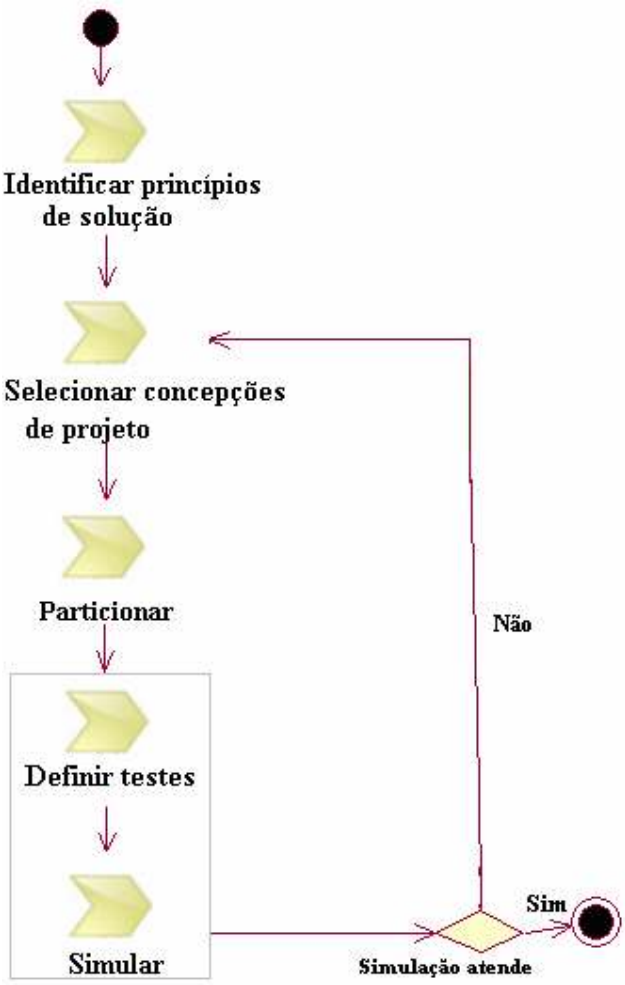


Figura 46 – Atividades da disciplina Identificação de Solução

Tabela 44 - Detalhamento da atividade Identificar princípios de solução

Atividade: Identificar princípios de solução
Objetivo: Definir as alternativas de soluções para cada componente que será desenvolvido.
Entrada: Modelo abstrato, documento de detalhamento dos requisitos funcionais e não funcionais.
Passos: • Para os componentes que serão desenvolvidos: o identificar os princípios de solução para cada componente; o montar a matriz morfológica.
Artefatos: Matriz morfológica.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas, arquiteto e especialistas.
Observação: O modelo do documento com as concepções de projeto encontra-se no experimento prático no capítulo 6.

A definição dos princípios pode incluir alternativas para serem avaliadas em algoritmos de *co-design* quando houver uma indicação de desenvolvimento em software ou hardware. Neste caso, requisitos temporais devem ser incluídos no algoritmo de particionamento dos componentes visto que implementações em hardware são mais rápidas que as implementações em software e, portanto, a decisão de particionamento pode interferir diretamente na solução a ser adotada.

A seleção de concepções de projeto é realizada a partir da combinação dos diversos princípios de soluções mapeados na matriz morfológica para os componentes do produto. A depender do porte do projeto, pode-se selecionar várias concepções para serem desenvolvidas como *protótipo* do produto.

Tabela 45 – Detalhamento da atividade Selecionar concepções de projeto

Atividade: Selecionar concepções de projeto
Objetivo: Identificar as combinações de concepções de projeto possíveis de serem construídas.
Entrada: Matriz morfológica.
Passos: • Analisar a matriz morfológica, destacando concepções possíveis de projeto; • Selecionar as concepções que seguirão o processo de construção; • Definir se o componente será adquirido, desenvolvido ou reaproveirado de outro produto.
Artefatos: Documento com as concepções de projeto.
Fase: Elaboração.
Papéis: Líderes técnicos das áreas específicas, arquiteto e especialistas.

Tabela 46 - Detalhamento da atividade Particionar

Atividade: Particionar
Objetivo: Escolher o melhor particionamento de hardware / software para o componente que será desenvolvido. Esta atividade é auxiliada pela matriz morfológica e pelos algoritmos de *co-design* e poderá subdividir um componente em subcomponentes.
Entrada: Concepções de projeto, documentos de detalhamento de requisitos funcionais e não funcionais.
Passos: • A partir das alternativas de concepção elaboradas na atividade anterior, aplicar algoritmos de *co-design*; • Analisar os resultados fornecidos pelos algoritmos; • Comparar as questões de segurança com os resultados gerados pelos algoritmos; • Definir solução para o componente que poderá ser totalmente mecânica, totalmente eletrônica, totalmente de software ou híbrida; • Para os componentes híbridos, dividi-los em subcomponentes totalmente eletrônico, totalmente mecânico ou totalmente de software, e definir as interfaces de comunicação entre estes.
Artefatos: Modelo concreto e documento de partição, que indica para cada componente a ser desenvolvido a decisão de particionamento adotada pela equipe.
Fase: Elaboração.
Papéis: Líderes das áreas específicas e arquiteto.

A atividade Particionar pode ser realizada em paralelo à atividade anterior ou pode ser aplicada às concepções de projeto escolhidas. É utilizada nos princípios de soluções que podem ser implementados tanto em hardware quanto em software. Consiste em utilizar os algoritmos de *co-design* para ajudar a definir qual o melhor particionamento de hardware/software possível, considerando as questões de custo e desempenho.

O *modelo concreto* consiste do modelo de componentes, onde um componente pode ter subcomponentes (elementos que serão construídos por uma única área). Soluções que envolvam componentes híbridos devem subdividir o componente de maneira que este seja formado por subcomponentes que possam ser construídos em uma única área. Esta subdivisão envolve a definição da interface de comunicação entre estes subcomponentes.

d) Disciplina de Desenvolvimento

A disciplina Desenvolvimento utiliza como instrumento de trabalho o modelo concreto já simulado e as especificações geradas até o momento. Desta forma, cada uma das áreas recebe os componentes ou subcomponentes que lhe foi designado para efetuar a construção. Esta construção poderá utilizar qualquer forma de desenvolvimento que a equipe técnica julgar apropriada, desde que atenda às especificações da equipe multidisciplinar.

Tabela 47 – Detalhamento da atividade Construir

Atividade: Construir
Objetivo: Construir o código fonte dos elementos de software. Construir um protótipo dos elementos mecânicos e eletrônicos.
Entrada: Modelo concreto, documento de partição.
Passos: • Programar os componentes de software, conforme especificação; • Construir os componentes mecânicos e eletrônicos, conforme especificação.
Artefatos: Componentes e subcomponentes.
Fase: Construção.
Papéis: Líder de cada área específica, especialistas de domínio e desenvolvedor.

A primeira atividade da disciplina de Desenvolvimento consiste na construção do componente. Para componentes de software, corresponde ao desenvolvimento do software. Para componentes eletrônicos, consiste no desenvolvimento do hardware específico. Para componentes mecânicos, o desenvolvimento do equipamento.

Tabela 48 – Detalhamento da atividade Integrar

Atividade: Integrar
Objetivo: Montagem do componente.
Entrada: Subcomponentes, documento de partição, modelo concreto.
Passos: • Sintetizar software / hardware*;* • Realizar reuniões interdisciplinares para: o Montar componente híbrido a partir dos subcomponentes sintetizados.
Artefatos: Componente.
Fase: Construção.
Papéis: Desenvolvedor, sob a supervisão do líder.

Uma vez construído um subcomponente, parte de um componente híbrido, este deve ser logo que possível integrado aos demais subcomponentes para compor o componente final. Quanto mais cedo ocorrer esta integração, mais cedo a possibilidade de se descobrir erros no componente e de corrigi-los sem grande impacto no prazo final do produto. A integração é um trabalho realizado pelos desenvolvedores das áreas envolvidas. Esta atividade está diretamente ligada à atividade de teste da disciplina Validação, pois o componente gerado deverá ser submetido aos testes especificados para ele.

A prototipação é a última atividade desta disciplina e consiste da montagem do protótipo do produto unindo todos os componentes construídos.

Tabela 49 - Detalhamento da atividade Prototipar

Atividade: Prototipar.
Objetivo: Integração dos componentes para criação de um protótipo que representa um modelo do produto.
Entrada: Componentes, Modelo concreto.
Passos: • Integrar componentes formando um protótipo do produto.
Artefatos: Protótipo do produto pronto para ser testado.
Fase: Construção.
Papéis: Líder de cada área específica e desenvolvedor.

e) Disciplina de Validação

A validação é uma das disciplinas mais importantes da metodologia. Esta disciplina está presente em quase todas as fases da mesma: inicia-se com a definição dos critérios de aceitação do produto; passa pela definição dos testes que deverão ser realizados para validar o protótipo do produto; permite simular a modelagem conceitual; e gerencia a

execução efetiva dos testes no protótipo do produto sob a visão técnica e sob a visão do cliente final, gerando o modelo do produto que será enviado à linha de produção.

Tabela 50 - Detalhamento da atividade Definir critérios de aceitação

Atividade: Definir critérios de aceitação.
Objetivo: Definir quais os critérios que serão observados ao final do processo para que o produto seja considerado apto para ser entregue ao cliente.
Entrada: Casos de uso, Requisitos não funcionais.
Passos: • Analisar requisitos; • Definir, para cada requisito, os critérios de aceitação que serão submetidos ao produto no final do processo; • Definir o responsável pela avaliação do produto; • Validar critérios.
Artefatos: Critérios de aceitação.
Fase: Concepção.
Papéis: Líderes das áreas específicas, gerente e cliente.

Os critérios de aceitação constituem a primeira atividade da disciplina Validação. Logo no início da especificação são definidos critérios, se possível sob a forma de meta, para serem utilizados como avaliadores do produto final que será concebido.

Tabela 51 - Detalhamento da atividade Definir testes de componentes

Atividade: Definir testes de componentes
Objetivo: Especificar os casos de testes que deverão ser realizados para validação do componente.
Entrada: Modelo concreto, documento de detalhamento dos requisitos funcionais e não funcionais.
Passos: • Elaborar as possibilidades de teste para atender cada componente produzido; • Definir o estímulo e a resposta que deve ser obtida no teste; • Definir o responsável pelo teste.
Artefatos: Casos de teste.
Fase: Elaboração.
Papéis: Líderes das áreas específicas.

A atividade Definir testes de componentes engloba a validação dos componentes e subcomponentes produzidos ao longo do processo. A especificação gerada nesta atividade será utilizada pelos testadores, quando os componentes ou subcomponentes estiverem prontos, na realização de testes técnicos, ou seja, na validação de suas funcionalidades específicas e interfaces. Os testes definidos devem ser capazes de avaliar se os requisitos funcionais e não funcionais de responsabilidade do componente estão sendo atendidos.

Tabela 52 – Detalhamento da atividade Definir testes do Produto

Atividade: Definir testes do produto
Objetivo: Determinar os casos de testes que deverão ser realizados para validação do produto final sob a visão técnica e a visão do usuário.
Entrada: Modelo concreto, documento de detalhamento dos requisitos funcionais e não funcionais.
Passos: • Elaborar as possibilidades de testes para atender cada requisito (funcional e não funcional); • Definir o estímulo e a resposta que deve ser obtida no teste; • Definir o responsável pelo teste.
Artefatos: Casos de teste.
Fase: Elaboração.
Papéis: Líderes das áreas específicas.

Os testes do produto são especificados para serem submetidos ao protótipo do produto quando este estiver pronto. Embora sejam elaborados pelos líderes das áreas, serão utilizados também pelo cliente, pois ele será o principal testador do protótipo do produto.

A atividade Simular modelo concreto (Tabela 53), define os passos para a realização de uma simulação do modelo concreto do produto. Com o auxílio de um software de simulação, os componentes ou subcomponentes são representados com suas respectivas interfaces. Os testes definidos tanto para os componentes ou subcomponentes quanto para o produto são aplicados para avaliar os resultados gerados.

Tabela 53 - Detalhamento da atividade Simular modelo concreto

Atividade: Simular modelo concreto
Objetivo: Realizar uma simulação do modelo concreto (diagrama de componentes) para visualizar o funcionamento dos componentes. Validar as suas interfaces internas e externas para detectar possíveis problemas de definição.
Entrada: Modelo concreto, documento de detalhamento dos requisitos funcionais e não funcionais.
Passos: • Definir a ferramenta de simulação a ser usada; • Desenvolver o modelo de simulação; • Criar o modelo concreto no simulador; • Efetuar a simulação, aplicando os casos de teste; • Avaliar os resultados.
Artefatos: Modelo simulado.
Fase: Elaboração.
Papéis: Líderes das áreas específicas e especialistas.

Esta simulação não utiliza nenhuma forma de prototipação, pois o objetivo é testar as interfaces. Assim, o modelo de simulação é montado representando cada componente ou subcomponente com seus métodos (comportamento) e uma interface para se comunicar com os demais componentes ou subcomponentes. O processo de prototipação, *layout* de formas,

entre outros aspectos é um detalhamento de cada área específica de acordo com o método que por ela for utilizado para construir o objeto definido.

Além de avaliar a definição das interfaces, a simulação é importante também para validar se os requisitos não funcionais tais como tolerância à falhas, requisitos temporais ou de desempenho estão sendo tratados corretamente. Assim, é importante que este modelo permita simular, além dos requisitos funcionais, os requisitos temporais e que seja possível injetar defeitos no modelo para visualizar o comportamento especificado para tratamento das falhas.

Tabela 54 – Detalhamento da atividade Testar aspectos técnicos

Atividade: Testar aspectos técnicos
Objetivo: Testar os componentes e o protótipo do produto sob a visão técnica, para verificar se este atende aos requisitos definidos durante a fase Concepção.
Entrada: Casos de teste.
Passos: • Aplicar casos de teste.
Artefatos: Caso de teste aplicados e com as observações preenchidas.
Fase: Construção.
Papéis: Líder das áreas envolvidas e testadores.

Uma vez construídos os componentes, dois tipos de teste são realizados com igual importância: o teste técnico e o teste de uso. O primeiro deles é o teste da montagem dos componentes, onde é observado o funcionamento do componente montado, destacando-se a interface que permite a comunicação entre eles. O segundo teste é realizado no protótipo do produto. Mesmo que os testes individuais dos componentes tenham obtido êxito, o teste do produto abrange a integração dos componentes e faz-se necessário para observar a comunicação entre estes componentes e para validar se os requisitos estão realmente sendo atendidos.

Tabela 55 - Detalhamento da atividade Testar uso

Atividade: Testar uso
Objetivo: Testar o produto sob a visão do cliente.
Entrada: Casos de teste.
Passos: • Aplicar casos de teste definidos para o produto.
Artefatos: Caso de teste aplicado e com as observações preenchidas, e o modelo do produto.
Fase: Construção.
Papéis: Cliente e testadores.

f) Disciplina de Verificação

A MdpM indica a utilização de verificadores de modelos para realizar a verificação de partes críticas de um produto mecatrônico. Esta técnica pode ser aplicada realizando-se um mapeamento da especificação do produto em um modelo matemático formal e definindo-se propriedades, de acordo com os requisitos inicialmente identificados, que serão submetidos a este modelo.

Para realizar a verificação a MdpM especifica duas atividades, Identificar comportamento e Verificar modelos, cujos passos estão descritos nas tabelas a seguir.

Tabela 56 - Detalhamento da atividade Identificar comportamento

Atividade: Identificar comportamento.
Objetivo: Construir o modelo formal para as partes do produto que precisam de uma verificação formal e definir propriedades que possam ser submetidas a este modelo formal.
Entrada: Diagrama de estados, modelo concreto.
Passos: • Selecionar um verificador de modelos; • Mapear os diagramas de estados em autômatos; • Especificar propriedades em uma linguagem apropriada.
Artefatos: Autômatos e documento de propriedades.
Fase: Construção.
Papéis: Engenheiro de verificação com o apoio dos líderes das áreas envolvidas.

A primeira atividade desta disciplina consiste na identificação do comportamento produto, necessária para construir o modelo formal e definir as propriedades.

Também devem ser especificadas as propriedades que serão submetidas ao modelo para verificação.

Tabela 57 - Detalhamento da atividade Verificar modelo

Atividade: Verificar modelo
Objetivo: Verificar se o modelo está correto.
Entrada: Autômatos e documento de propriedades.
Passos: • Submeter propriedades ao verifiador; • Analisar os resultados.
Artefatos: Documento de propriedades com o resultado da verificação.
Fase: Construção.
Papéis: Engenheiro de verificação com o apoio dos líderes das áreas envolvidas.

A submissão das propriedades ao modelo pode ser realizada a partir de ferramentas automatizadas chamadas verificadores de modelos. Estes verificadores checam a propriedade e apresentam um resultado indicando se o modelo atende ou não a propriedade.

Existem trabalhos, a exemplo de [3], que utilizam diagramas gerados na linguagem UML para derivar automaticamente um modelo formal em uma linguagem de verificação. Estes trabalhos podem ser utilizados como ferramentas de apoio a esta atividade.

g) Disciplina de Documentação

Documentação é uma disciplina que está presente em todas as fases da metodologia, pois é formada pelos artefatos construídos ao longo do projeto.

Tabela 58 - Detalhamento da atividade Reunir e revisar modelos

Atividade: Reunir e revisar modelos
Objetivo: Reunir os modelos gerados em cada fase da metodologia em um documento único.
Entrada: Documentos e modelos gerados em todo o processo.
Passos: • Reunir os documentos; • Revisar documentos.
Artefatos: Documentação.
Fase: Construção.
Papéis: Líder de cada área envolvida.

Na fase Entrega, ela se torna um pouco mais expressiva porque é onde todos estes artefatos são organizados em uma documentação que acompanhará o modelo do produto.

Na atividade Reunir e revisar modelos são reunidos e revisados os modelos para montagem da documentação final.

Tabela 59 - Detalhamento da atividade Gerar documentação

Atividade: Gerar documentação.
Objetivo: Gerar documentação do produto.
Entrada: Documentação.
Passos: • Complementar documentação; • Anexar documentos ao modelo do produto para enviar à linha de produção.
Artefatos: Documentação.
Fase: Construção.
Papéis: Líder de cada área envolvida.

A atividade Gerar documentação é realizada quando há a necessidade de se executar algum trabalho adicional que será anexado à documentação do produto.

h) Disciplina de Gerência

A gerência do projeto está presente em todas as fases, permitindo controle e acompanhamento do processo.

Ao longo do projeto são realizadas reuniões multidisciplinares de controle e acompanhamento das atividades desempenhadas que precisam ser registradas. Qualquer modificação significativa deve ser aprovada nesta reunião.

Tabela 60 - Detalhamento da atividade Realizar reuniões

Atividade: Realizar reuniões
Objetivo: Avaliar o andamento do projeto.
Entrada: Solicitações dos papéis envolvidos.
Passos: • Reunir gerentes e líderes; • Acompanhar cronogramas; • Discutir projeto.
Artefatos: Ata de reunião.
Fase: Todas.
Papéis: Gerentes, líderes das áreas e cliente.

Para todo projeto deve ser elaborado um plano com as ações a serem seguidas, conforme a metodologia MdpM, e um cronograma deve ser elaborado com base no plano de projeto. Cronogramas devem fazer parte do produto desde o final da fase de concepção do sistema. À medida que o produto for sendo desenvolvido, ao final de cada fase, o cronograma deve ser atualizado e submetido aos gerentes do projeto.

Tabela 61 – Elaborar plano de projeto

Atividade: Elaborar plano de projeto.
Objetivo: Elaborar um plano que permita orientar a equipe no decorrer do projeto.
Entrada: Necessidades e requisitos.
Passos: • Identificar os principais passos para execução do projeto, segundo a metodologia MdpM; • Definir metas.
Artefatos: Plano de projeto.
Fase: Concepção.
Papéis: Gerentes e líderes das áreas e cliente.

Tabela 62 - Detalhamento da atividade Elaborar cronogramas

Atividade: Elaborar cronogramas.
Objetivo: Elaborar os cronogramas de atividades do projeto.
Entrada: Necessidades e requisitos.
Passos: • Identificar os recursos necessários; • Identificar as atividades segundo o plano de projeto; • Definir prazos; • Alocar recursos às atividades; • Atualizar cronograma à medida que as fases forem sendo elaboradas.
Artefatos: Cronogramas.
Fase: Todas.
Papéis: Gerentes e líderes das áreas e do cliente.

Um projeto pode necessitar de recursos sejam eles de recrutamento de pessoal, treinamento, compra de materiais, entre outros. Estes recursos foram previstos pelo plano de projeto e devem ser alocados de acordo com o cronograma de atividades definido. Por este motivo, a MdpM define uma atividade (Tabela 63) para gerenciar a alocação destes recursos que poderá ser realizada em várias fases da metodologia.

Tabela 63 – Detalhamento da atividade Alocar recursos

Atividade: Alocar recursos.
Objetivo: Alocar os recursos necessários para o desenvolvimento do projeto. Estes recursos podem ser de pessoal, compra de equipamentos, treinamento, etc.
Entrada: Plano de projeto
Passos: • Identificar os recursos necessários; • Alocar conforme cronograma de atividades.
Artefatos: Documento de registro dos recursos alocados.
Fase: Todas.
Papéis: Gerentes e líderes das áreas e cliente.

Assim como no RUP a MdpM utiliza o conceito de pontos de controle (PC), ao final de cada fase, para garantir que uma fase realmente foi finalizada e que é possível iniciar uma nova fase do projeto (Tabela 64).

Tabela 64 - Detalhamento da atividade Checar pontos de controle

Atividade: Checar pontos de controle
Objetivo: Avaliar o bom andamento do projeto de acordo com características pré-definidas.
Entrada: Modelos e documentos gerados na fase finalizada.
Passos: • PC1: acontece sempre ao final da fase Concepção e tem os seguintes passos: o verificar a validade dos requisitos. Se não existem outros requisitos para outros usuários; o verificar a consistência dos requisitos . Se os conflitos foram resolvidos; o verificar a viabilidade dos requisitos. Se é possível implementar o requisito com a tecnologia atual; o analisar a facilidade de avaliação de um requisito. Se é possível o cliente avaliar de maneira simples se o produto está atendendo aos requisitos; o Avaliar se o escopo definido para a versão é viável ao projeto sob vários aspectos diferentes, procurando atender aos seguintes questionamentos: ▪ as funcionalidades selecionadas agregam realmente o valor desejado ao produto? ▪ o cronograma inicial está de acordo com as expectativas? • PC2: acontece após a fase Elaboração. Considerando que todo o levantamento de requisitos foi finalizado e uma arquitetura foi definida, é necessário então observar se existe algum fator que possa impactar o sucesso do projeto, seja ele físico, técnico ou pessoal. Este ponto de controle deverá responder as seguintes perguntas: o os requisitos estão completamente entendidos e mapeados nos modelos do sistema? o todos os componentes definidos no modelo abstrato tem sua estratégia de implementação claramente definida? o os testes foram simulados e todos os problemas contornados? o existe concordância entre os líderes e gerentes quanto às soluções adotadas? • PC3: realizado após a fase Construção. Considerando que neste ponto o modelo do produto se encontra pronto, devem-se analisar as condições de desempenho, confiabilidade e tolerância a falhas. • PC4: aplicado quando o produto já está pronto para ser entregue à produção. Deve responder as seguintes perguntas: o a documentação está atualizada e de acordo com o produto construído? o é necessário iniciar as atividades de construção de outra versão do produto? o os critérios de aceitação foram atendidos?
Artefatos: Documento de checagem de pontos de controle (PC).
Fase: Ao final de cada fase.
Papéis: Cliente, gerentes e líderes das áreas.

APÊNDICE C – Experimento Prático

Este apêndice apresenta a documentação completa do experimento pratico descrito no capitulo 6. A documentação está organizada de acordo com as fases descritas na metodologia MdpM.

a) Concepção

A fase Concepção se inicia com uma descrição do produto que será construído. Para este experimento, o produto mecatrônico selecionado é um robô.

O robô será utilizado para exploração de ambientes que não permitam a presença humana. Como principal objetivo deve se locomover neste ambiente capturando imagens e enviando-as para um ambiente externo, representado na Figura 44 como o *Controle Remoto*(CR). Por exemplo, o robô pode ser utilizado para capturar imagens de um lugar afetado por um gás tóxico qualquer que não possa ser respirado pelo ser humano. Desta forma, ele deverá se locomover pelo ambiente, capturar as imagens e enviá-las para um lugar seguro onde possam ser analisadas.

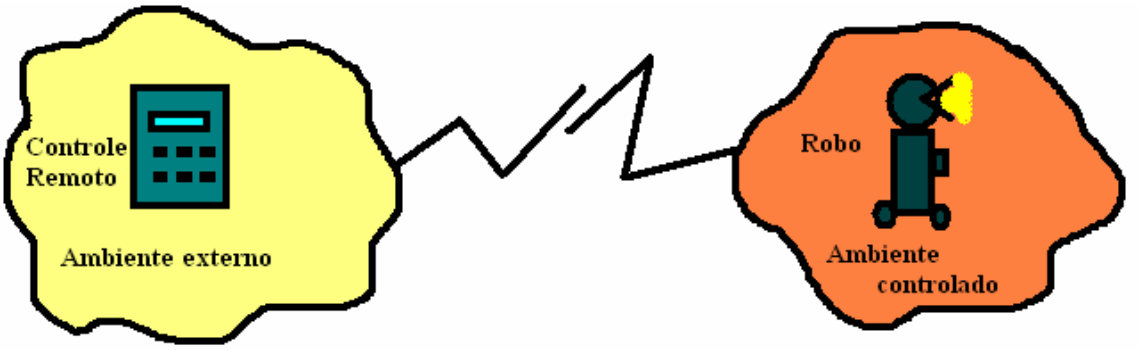


Figura 47 – Visão geral do sistema

O robô é controlado remotamente pelo controle remoto. Este controle pode ser realizado de duas formas: manual, ou automático, através de um sistema inteligente. Quando for controlado manualmente, uma pessoa será responsável por analisar as imagens transmitidas e guiar o robô pelo ambiente. Quando for controlado automaticamente, as imagens são transmitidas e servem de base para um sistema de navegação que indica a locomoção do robô. Desta forma, a partir das imagens capturadas o sistema de navegação

identifica obstáculos e define o caminho a ser percorrido. O robô fica em operação pelo controle automático até que este seja desativado.

O controle manual também permite efetuar ajustes na câmera para melhorar a qualidade das imagens capturadas tais como foco e distância (*zoom*).

A interface do controle remoto deve ser amigável, simples e flexível, pois se espera que este robô tenha um leque grande de aplicabilidade. O robô deve se locomover nas quatro posições (norte, sul, leste e oeste) com uma velocidade constante. Além disso, deve ser extensível à utilização de outros dispositivos, além do inicialmente projetado para filmagem, com poucas modificações.

Uma vez definida a descrição inicial do trabalho, as seguintes necessidades do cliente são destacadas, que devem ser consideradas na construção do robô:

- gerar imagens de um ambiente onde o homem não possa estar presente e enviá-las para um local remoto;
- permitir ajuste da câmera na captura da imagem;
- locomoção nas quatro direções: norte, sul, leste e oeste;
- ser controlado manualmente via controle remoto;
- ser controlado automaticamente via controle remoto;
- usar uma tecnologia que permita variedade de controladores remotos;
- usar uma tecnologia que permita adicionar novos dispositivos para serem controlados além do de filmagem;
- ter uma interface de comunicação simples;
- ter um custo acessível.

A partir destas necessidades iniciais são definidos os requisitos funcionais, representados pelos casos de uso ilustrados na Figura 48.

Como se pode observar, o diagrama de casos de uso apresenta dois atores, que representam as duas formas de controle identificadas para o robô: o controle manual representado pelo ator *controlador manual* e o controle automático representado pelo ator *controlador automático*.

O controlador manual possui algumas funcionalidades exclusivas tais como ligar e desligar o robô, ativar e desativar o controle remoto e controlar dispositivos.

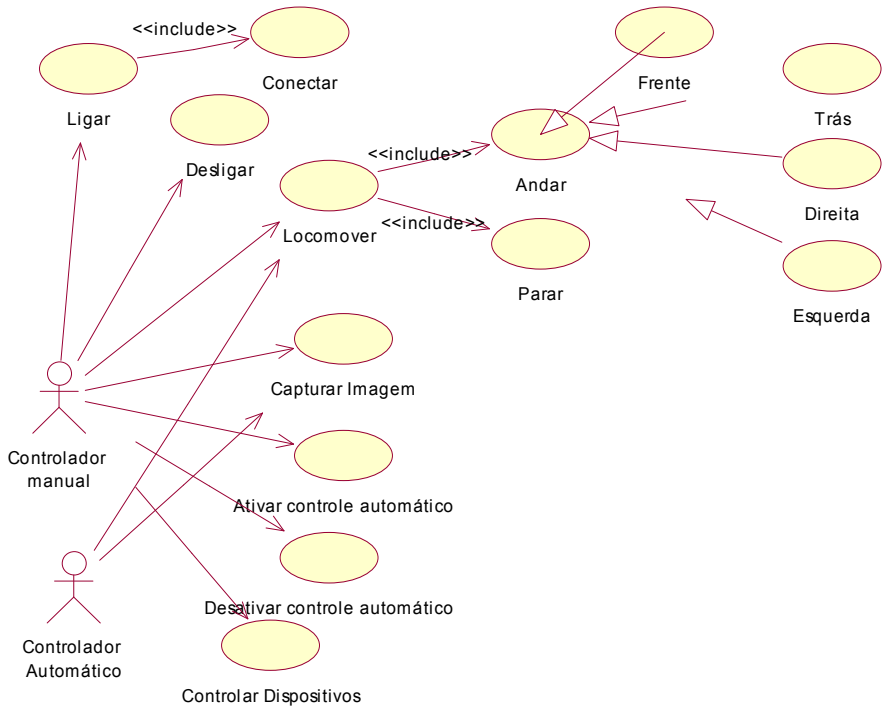


Figura 48 - Diagrama de casos de uso com as funcionalidades do robô

O robô é ligado a partir do caso de uso Ligar. Esta opção é responsável por estabelecer uma conexão entre o controle remoto e o robô, carregando configurações iniciais no robô e deixando-o pronto para receber novos comandos. Caso o controlador manual deseje ativar o controle automático ele deverá utilizar o caso de uso Ativar controle automático. A partir deste momento, o robô passa a ser controlado pelo sistema automático de navegação até que o caso de uso Desativar controle automático seja selecionado. Enquanto o robô estiver em operação é possível utilizar o caso de uso Locomover para andar (nas quatro posições) ou parar o robô. Também é possível, para o controlador manual, controlar dispositivos. O dispositivo previsto até o momento deve suprir apenas necessidades de filmagem, portanto é possível, por exemplo, ajustar o *zoom*. Sempre que o robô estiver se locomovendo, imagens estarão sendo capturadas e enviadas para o controle remoto. Finalmente, quando o controlador manual desejar, poderá desligar o robô a partir da opção Desligar, que encerra a conexão salvando a configuração atual para ser recarregada da próxima vez que o robô for ligado.

É importante destacar que a precisão na locomoção e captura de imagens do robô depende da velocidade com que as solicitações do controle remoto são atendidas. No caso do controle automático, esta depende também da velocidade de processamento e análise da imagem para guiar a locomoção.

Considerando que o robô estará sendo controlado à distância, é importante que este tenha um mecanismo de monitoria de conexão com o controle remoto. Em caso de perda de conexão, um estado de segurança deve ser projetado para preservar a integridade do robô até que a conexão seja restabelecida.

Para atender as necessidades descritas os seguintes requisitos não funcionais são considerados, distribuídos por categoria.

- Extensibilidade:
    - usar um sistema de comunicação que permita adicionar novos dispositivos com facilidade;
    - projetar um hardware com previsão de expansão de dispositivos.
- Usabilidade:
    - interface de controle amigável;
    - a imagem capturada deve ser nítida o suficiente para ser entendida pelo controlador manual ou pelo sistema de navegação.
- Falha:
    - qualquer situação de falha deve parar a locomoção do robô, inclusive a perda de comunicação entre a central e o robô, até que a falha seja recuperada.
- Tempo:
    - o deslocamento deve ocorrer em um intervalo de tempo *Td* específico para que o robô seja preciso em seus movimentos;
    - o envio da imagem deve ocorrer em um intervalo de tempo *Ti* específico para que estas imagens sejam válidas;
    - o tempo de processamento do sistema de navegação deve ser suficiente para que a atuação no ambiente tenha o resultado esperado.

- a parada do robô em caso de falhas ou em caso de situações anormais (obstáculos, por exemplo) deve ocorrer em um tempo *Tp* aceitável.

- Custo:
  - optar por tecnologias mais accessíveis, tanto no robô quanto no controle remoto;
- Gerais:
  - peso *P* específico
  - Comprimento *C* específico
  - Largura *L* especifica
  - Altura *A* específica
  - Consumo de energia

A partir destes requisitos, funcionais e não funcionais, é montada a matriz da casa da qualidade, definindo-se a importância de cada um destes requisitos para o cliente, identificando as relações existentes entre estes requisitos e comparando-os com os produtos similares de mercado, se for necessário (Figura 49).

Observa-se que são analisados os relacionamentos existentes entre as nove necessidades do cliente, representado pelas linhas da figura, e os requisitos mapeados para o produto, representados pelas colunas. O cliente atribui a cada necessidade um grau de importância como, por exemplo, a necessidade Envio remoto de imagens, que tem grau de importância cinco. A partir destas definições gera-se o grau de importância de cada requisito para o projeto, representado na última linha da Figura 49 em uma escala de um a doze.

No telhado da casa também é possível observar que alguns conflitos foram identificados. Por exemplo, para atender ao requisito de inclusão de novos dispositivos o produto terá que ser mais flexível e isso poderá deixar a interface menos amigável. Além disso, esta característica pode significar um aumento no custo do produto, pois os componentes que serão projetados devem atender a requisitos de expansão.

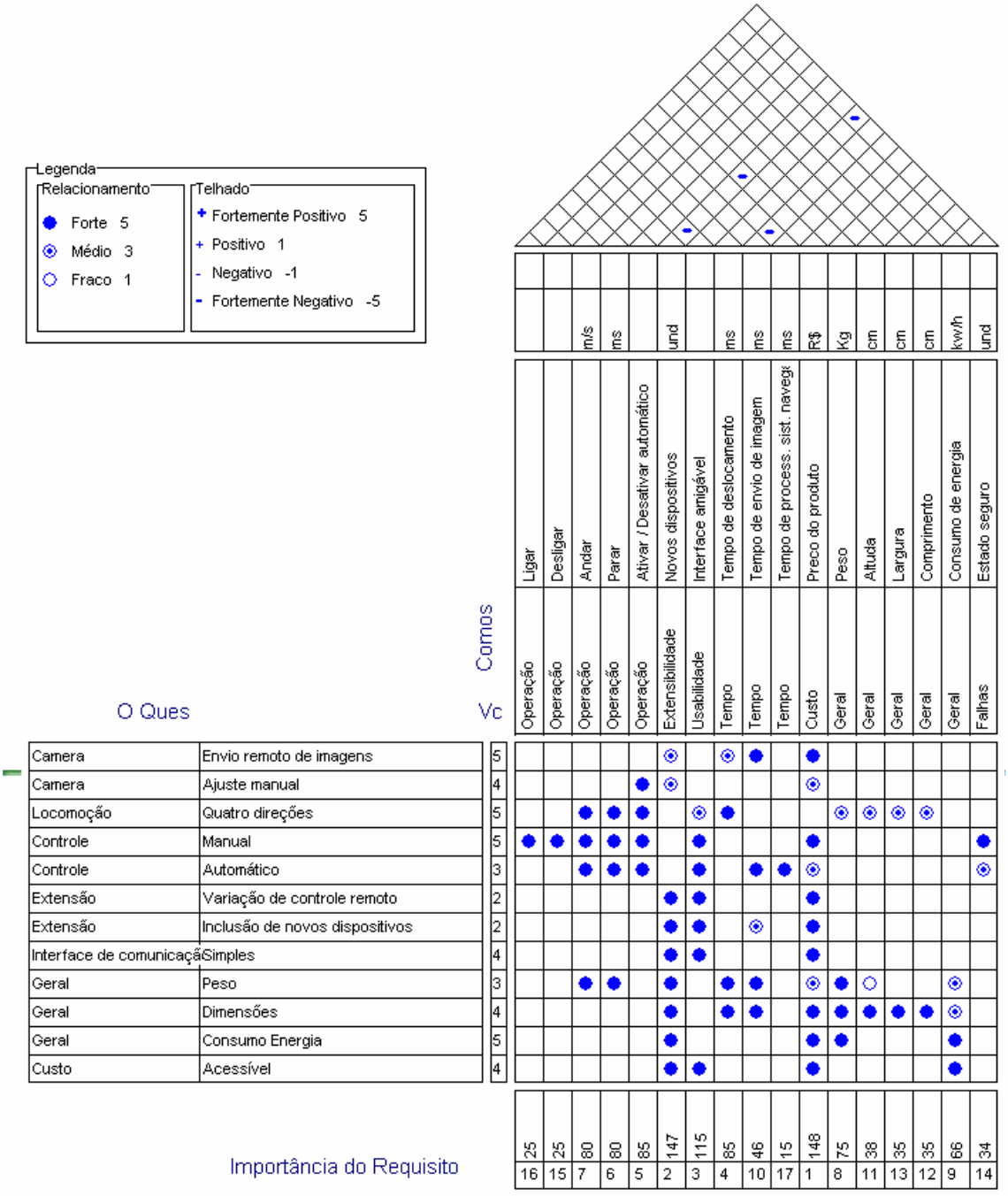


Figura 49 - Casa da qualidade para o robô

Para finalizar a fase Concepção, alguns fatores são considerados como de risco e, portanto, foram destacados para serem analisados de maneira a não comprometer o sucesso do projeto, são eles:

- tempo de execução das tarefas desde a solicitação de locomoção pelo controle remoto até o processamento das imagens enviadas pelo robô;
- tecnologia utilizada para conexão entre o controle remoto e o robô, que deve ser segura o suficiente para que não haja perdas constantes de comunicação.

Com as informações levantadas até o momento definiu-se que o escopo do projeto do robô deve conter todos os requisitos funcionais (representados pelos casos de uso) e não funcionais.

Como instrumento de avaliação do modelo do produto quando este já estiver pronto, são identificados também os seguintes critérios de aceitação:

- atender aos requisitos funcionais e não funcionais especificados;
- obter resultado positivo em 100% dos testes realizados;
- ter sido submetido a uma simulação real, piloto, com resultados positivos.

Ao longo da fase Concepção alguns artefatos foram gerados e validados. Esta validação faz parte da checagem de pontos de controle que acontece sempre ao final de cada fase e está ilustrada no documento de validação apresentado na Tabela 65.

Tabela 65 - Documento de validação da primeira fase de especificação do robô

Documento de Validação	
Data de início do projeto: 27/10/06	
Equipe Responsável	
Identificação	Nome
Gerente do cliente	Especificação de [37]
Gerente Técnico	Ana Patrícia
Líder Computação	Ana Patrícia
Líder Mecânica	Hermam
Concepção	
Data da validação	Artefatos
30/10/06	Casos de uso
02/11/06	Lista de requisitos não funcionais
04/11/06	Casa da qualidade e Soluções de conflitos
05/11/06	Fatores de risco
08/11/06	Documento de escopo

b) Elaboração

A fase Elaboração se inicia com o detalhamento dos requisitos do produto. Assim, para cada requisito funcional (caso de uso) é elaborada uma tabela que apresenta informações tais como: qual a pré-condição para que o caso de uso funcione; qual o ator responsável por iniciar o caso de uso; o objetivo do caso de uso; seu funcionamento normal detalhado; como deve funcionar em casos excepcionais; qual a pós-condição; e observações pertinentes.

Tabela 66 - Descrição do caso de uso Ligar

Caso de uso	**Ligar**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô desligado.		
Ator:	Controlador manual.		
Objetivo:	Ligar o robô. Estabelecer uma conexão entre a central e o robô.		
Caso normal			
Ator		Robô	
Selecionar opção Ligar.		Conecta com o controle remoto; Inicializa configuração; Liga o dispositivo de filmagem; Aguarda novos comandos.	
Exceção 1: Central não consegue se conectar			
Mensagem de conexão não efetuada.			
Pós – condições	Robô aguardando comando.		

A Tabela 66 descreve o caso de uso Ligar. Este caso de uso é ativado pelo controlador manual quando o robô estiver no está de desligado. Para isso é selecionada no controle remoto a opção Ligar. Sua função é estabelecer uma conexão com o robô para iniciar as atividades. Quando a conexão é efetuada com sucesso, o robô recebe a solicitação do controle remoto e restaura a última configuração utilizada. O dispositivo de filmagem também é iniciado e o robô fica aguardando os próximos comandos. Caso aconteça algum problema que impeça a comunicação inicial do controle remoto com o robô, uma mensagem é apresentada ao controle remoto e nada mais acontece.

Tabela 67 - Descrição do caso de uso Locomover

Caso de uso	**Locomover**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado.		
Ator:	Controlador manual ou Controle automático.		
Objetivo:	Locomover o robô fazendo-o andar ou parar.		
Caso normal			
Ator		Robô	
Selecionar opção de locomoção.		Chama casos de uso para andar ou parar.	
Exceção 1: Perda de conexão			
		Chama caso de uso *parar*.	
Pós – condições	Robô em movimento na direção solicitada ou robô parado.		

O caso de uso Locomover, descrito na Tabela 67, descreve a locomoção do robô através das opções andar e parar, definidas respectivamente nos casos de uso descritos nas Tabelas 68 e 69. As duas opções podem ser executadas independentes do tipo de controle que esteja sendo utilizado, manual ou automático.

Tabela 68 - Descrição do caso de uso Andar

Caso de uso	**Andar**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado.		
Ator:	Usuário controlador ou Controle automático.		
Objetivo:	Locomover o robô para frente, para trás, para a direita ou para a esquerda.		
Caso normal			
Ator		Robô	
Selecionar opção de locomoção (frente, trás, esquerda, direita).		Gira para a posição selecionada; Enquanto não receber outra instrução, locomove-se para frente.	
Exceção 1: Perda de conexão			
		Chama caso de uso Parar.	
Pós – condições	Robô em movimento na direção solicitada.		

O robô pode andar em varias direções diferentes. A partir da opção indicada pelo ator que dispara a ação (andar para frente, para trás, para direita ou para esquerda), o robô executa um giro e se locomove para frente. Observa-se também na Figura 67 que qualquer perda de conexão detectada pelo robô dispara o caso de uso Parar, para que o robô interrompa a execução até que a conexão seja restabelecida.

O caso de uso Parar, ilustrado na Tabela 69, é executado quando o robô está se locomovendo e deseja parar. Após a parada, o robô aguarda novos comandos.

Tabela 69 - Descrição do caso de uso Parar

Caso de uso	**Parar**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado e em movimento.		
Ator:	Controlador manual ou Controlador automático.		
Objetivo:	Interromper a locomoção.		
Caso normal			
Ator		Robô	
Selecionar opção de parada.		Interrompe a locomoção; Aguarda novo comando.	
Pós – condições	Robô ligado e parado.		

A Tabela 70 descreve o caso de uso Controlar dispositivos, responsável pelo ajuste dos dispositivos acoplados ao robô, a exemplo da filmagem. Este caso de uso só pode

ser executado pelo Controlador manual, pois requer a seleção dos parâmetros para ajuste do dispositivo.

Tabela 70 - Descrição do caso de uso Controlar Dispositivos

Caso de uso	**Controlar dispositivos**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado.		
Ator:	Controlador manual.		
Objetivo:	Controlar os dispositivos acoplados ao robô, a exemplo da câmera.		
Caso normal			
Ator		Robô	
Selecionar opção de controle de dispositivos; Informar parâmetros de controle, ex. posição e zoom da câmera.		Ativar o dispositivo solicitado; Executar a solicitação de acordo com os parâmetros. Ex. pára a câmera: Gira a câmera para a posição indicada; Ajusta o *zoom* de acordo com o indicado.	
Exceção 1: Perda de conexão			
		Chama caso de uso Parar.	
Exceção 2: Dispositivo com problema			
		Chama caso de uso Parar.	
Pós – condições	Dispositivo ajustado.		

Na Tabela 71 o caso de uso Capturar imagem é descrito. Quando uma solicitação de filmagem é executada, o dispositivo é ligado e a imagem começa a ser capturada. Para cada bloco de imagens, características importantes para transmissão são selecionadas e somente então transmitidas. Quando o controle remoto recebe a imagem, armazena para futuras análises.

Tabela 71 - Descrição do caso de uso Capturar Imagem

Caso de uso	**Capturar imagem**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado e câmera ligada.		
Ator:	Controlador automático, controlador manual.		
Objetivo:	Ligar a câmera e capturar imagem.		
Caso normal			
Ator		Robô	
Solicita imagem.		Liga a câmera; Efetuar filmagem; Extrair características importantes; Inicia a transmissão.	
Recebe e armazena a imagem.			
Exceção 1: Perda de conexão			
		Chama caso de uso Parar.	
Exceção 2: Câmera com problema			
		Chama caso de uso Parar.	
Pós – condições	Imagem enviada e armazenada.		

O caso de uso Desligar (Tabela 72) é executado pelo controlador manual para interrompe a locomoção, salva as configurações existentes, desliga o dispositivo de filmagem e finaliza a conexão com o controle remoto.

Tabela 72 - Descrição do caso de uso Desligar

Caso de uso	**Desligar**		
Data de criação:	27/10/06	Data de alteração	27/10/06
Pré-condição:	Robô ligado.		
Ator:	Controlador manual.		
Objetivo:	Desligar o robô.		
Caso normal			
Ator		Robô	
Comando de desligar.		Chamar caso de uso Parar; Salva configurações atuais; Desliga dispositivo; Finaliza a conexão.	
Pós – condições	Robô desligado.		

Tabela 73 - Descrição do caso de uso Ativar Controle Automático

Caso de uso	**Ativar controle automático**		
Data de criação:	01/11/06	Data de alteração	01/11/06
Pré-condição:	Robô ligado em modo de controle manual.		
Ator:	Usuário controlador.		
Objetivo:	Ativar o controle automático do robô.		
Caso normal			
Ator		Robô	
Comando de ativação do controle automático; Mudança do estado do robô para automático; Enquanto estiver no controle automático; Chamar caso de uso Capturar imagem; Calcula locomoção; Chama caso de uso Andar.			
Pós – condições	Robô em estado de controle automático.		

O robô pode ser operado em controle automático (Tabela 73), onde imagens são enviadas para serem analisadas pelo sistema de navegação. A desativação deste recurso (Tabela 74) devolve ao usuário o controle do robô.

Tabela 74 - Descrição do caso de uso Desativar Controle Automático

Caso de uso	**Desativar controle automático**		
Data de criação:	01/11/06	Data de alteração	01/11/06
Pré-condição:	Robô ligado em modo de controle automático.		
Ator:	Usuário controlador.		
Objetivo:	Desativar o controle automático do robô.		
Caso normal			
Ator		Robô	
Comando de desativação do controle automático.		Chamar caso de uso Parar.	
Interromper sistema de navegação automática; Mudar do estado de controle do robô para manual.			
Pós – condições	Robô em estado de controle manual.		

Além dos casos de uso, os requisitos não funcionais também foram detalhados conforme tabela a seguir. Características temporais relacionadas ao requisito também são detalhadas.

Tabela 75 - Descrição dos requisitos não funcionais

Requisitos não Funcionais		
Data:	27/10/06	
Categoria	**Requisito**	**Descrição**
Extensibilidade	Usar um sistema de comunicação que permita adicionar novos dispositivos com facilidade.	A comunicação entre o controlador manual e o robô será remota. Ela será implementada a partir de um protocolo de comunicação que permita trocar comandos, dados e imagens. Este protocolo deve ser flexível o possível para permitir que novos dispositivos acoplados ao robô sejam controlados também remotamente.
Usabilidade	Interface de controle amigável	A interface do controlador remoto deve ser simples, fácil de ser utilizada.
Tempo	O deslocamento deve ocorrer em um intervalo de tempo específico para que o robô seja preciso em seus movimentos.	A precisão no deslocamento do robô está diretamente relacionada ao intervalo de tempo existente entre a solicitação do comando de locomoção e a sua execução pelo robô. Assim, foram definidas as seguintes características: *deadline* 50ms*; t*empo de execução *40ms; e p*eriodicidade aperiódica.
	O envio da imagem deve ocorrer em um intervalo de tempo específico para que estas imagens sejam válidas.	A captura da imagem pela câmera e o envio desta para a central deve acontecer em um tempo suficiente para que estas imagens possam ser analisadas pelo controlador. Assim, foram definidas as seguintes características: *deadline 30ms;* tempo de execução *25ms;* periodicidade periódica.
	O tempo de processamento do sistema de navegação deve ser aceitável para que possa atuar no ambiente.	O tempo de processamento da imagem pelo sistema de navegação quando o robô estiver sendo operado em modo automático deve atender as seguintes restrições: *deadline 20ms;* tempo de execução *15ms;* e periodicidade periódica.
	A parada do robô em caso de falhas ou em caso de situações anormais (obstáculos, por exemplo) deve ocorrer em um tempo *T* aceitável.	O tempo de interrupção dos motores que locomovem o robô deve ser *50ms*.
Custo	Optar por uma tecnologia barata, tanto no robô quanto no controle remoto.	A tecnologia deve ser atender às expectativas de custo do projeto.
Falha	Qualquer situação de falha deve parar a locomoção do robô, inclusive a perda de conexão, até que a falha seja recuperada.	Caso ocorra algum problema de comunicação entre o controlador e o robô, este deverá ter suas atividades de locomoção suspensas. Isso deve ser feito em no máximo *50ms*.
Gerais	Peso, altura, largura e comprimento.	O robô deve ter um peso máximo de 15kg, uma altura máxima de 80m, uma largura máxima de 50cm e um comprimento máximo de 50cm.
	Consumo de energia.	O robô deve consumir um máximo de X de energia.

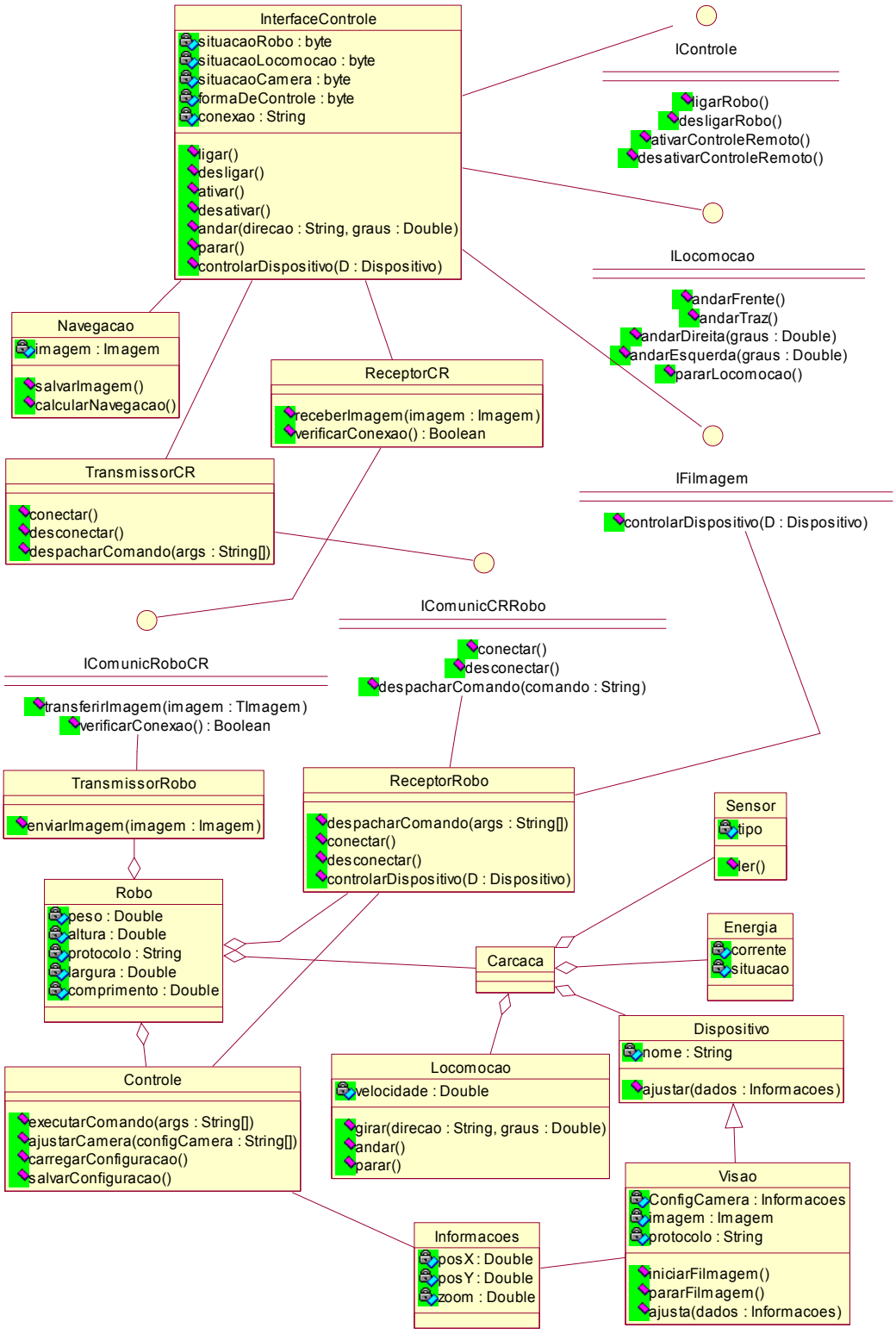


Figura 50 - Modelo estático do robô

O diagrama estático com a estrutura preliminar do robô ilustra as classes já identificadas para o produto, com seus atributos e métodos.

Pode-se observar a partir da Figura 50 que a estrutura do robô está dividida em duas partes que se comunicam através de um conjunto de interfaces. A primeira parte, formada pelas classes InterfaceControle, Navegação, TransmissorCR e ReceptorCR referem-se ao controle remoto (CR). A segunda parte formada pelas demais classes, refere-se ao robô propriamente dito. Nota-se então que o robô possui características gerais tais como o peso, e é composto por um conjunto de classes responsáveis pela comunicação com o CR (TransmissorRobo e ReceptorRobo), pelo controle local do robô (Controle) e pela sua estrutura física (Carcaça). É importante observar que a forma de modelar a visão do robô a partir de uma classe chamada Dispositivos já demonstra uma preocupação com a inclusão futura de novos dispositivos.

Para cada uma das classes do diagrama ilustrado na Figura 50, é elaborada uma descrição que detalha seu funcionamento, definindo cada atributo e representando as interfaces implementadas, conforme tabelas a seguir.

Tabela 76 - Descrição da classe InterfaceControle

Classe: InterfaceControle.			
Data da elaboração:	10/11/2006	Última alteração	26/12/2006
Descrição: Representa a classe principal do controle remoto do robô. É responsável por todo o controle seja ele manual ou automático. Relaciona-se diretamente com três outras classes, a ReceptorCR, a TransmissorCR e Navegacao.			
Atributos		**Descrição**	
SituacaoRobo: *byte*		Indica a situação atual do robô, se ligado (1) ou desligado (2).	
situacaoLocomocao: *byte*		Índica se o robô está parado (1) ou em movimento (2).	
situacaoCamera: *byte*		Índica se a câmera esta ligada (1) ou desligada (2).	
formaDeControle: *byte*		Indica se o robô está sendo operado manualmente (1) ou automaticamente (2).	
conexão: *String*		Comando de conexão com o robô.	
Métodos		**Descrição**	
ligar()		Liga o robô.	
desligar()		Desliga o robô.	
ativar()		Ativa o controle remoto.	
desativar()		Desativa o controle remoto.	
andar (direção: *String*, graus: *double*)		Locomove o robô na direção indicada.	
parar()		Pára o robô.	
controlarDispositivo(d: dispositivo)		Configura o dispositivo passado como parâmetro.	
Interfaces implementadas:			
IControle, IFilmagem e ILocomocao.			

A Tabela 76 mostra a descrição da classe InterfaceControle. Pode-se observar a existência de cinco atributos e sete métodos. O atributo situacaoRobo, por exemplo, indica a situação atual do robô, podendo assumir o valor 1 para ligado ou 2 para desligado.

Análogamente, a Tabela 77 mostra a descrição da classe Navegação. Observa-se que neste caso não existe interface associada a ela.

Tabela 77 - Descrição da classe Navegação

Classe: Navegação			
Data da elaboração:	26/12/2006	Ultima alteração	26/12/2006
Descrição: Esta classe representa o sistema de navegação que calculará a navegação do robô quando ele estiver sendo operado pelo controle automático.			
Atributos		**Descrição**	
imagem: Imagem		Guarda a imagem transmitida.	
direção: *String*		Define a direção para onde o robô deve se locomover; Se a direção estiver vazia, o robô deve ficar parado.	
Métodos		**Descrição**	
calcularNavegacao()		Calcula posição de locomoção de acordo com as imagens recebidas.	
Interfaces implementadas:			
Nenhuma.			

Seguindo o mesmo padrão a classe TransmissorCR é descrita na Tabela 78, responsável pelo envio de mensagens do CR para o robô.

Tabela 78 - Descrição da classe TransmissorCR

Classe: TransmissorCR			
Data da elaboração:	10/11/2006	Última alteração	10/11/2006
Descrição: Esta classe representa o transmissor do controle remoto. É responsável por estabelecer uma conexão com o robô para o envio de comandos e dados.			
Atributos		**Descrição**	
Métodos		**Descrição**	
conectar()		Estabelece uma conexão com o robô.	
desconectar()		Finaliza a conexão com o robô.	
despacharComando(arg: String[])		Envia um comando para ser executado pelo robô. Tem como argumento um vetor de string, onde o primeiro elemento índica o comando e os demais os argumentos, se existirem.	
Interfaces implementadas			
IComunicCRRobo			

Da mesma maneira, a classe ReceptorCR é descrita na Tabela 79. Ela e responsável pela recepção de dados e comandos enviados pelo robô ao controle remoto.

Como exemplo pode-se citar as imagens enviadas periodicamente para serem armazenadas e analisadas.

Tabela 79 - Descrição da classe ReceptorCR

Classe: ReceptorCR			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe representa o receptor do controle remoto. É responsável por receber dados vindos do robô a partir da conexão estabelecida pelo transmissor.			
Atributos		**Descrição**	
Métodos		**Descrição**	
receberImagem(imagem:Imagem)		Recebe uma imagem e chama o método SalvarImagem() para armazená-la.	
verificarConexao():boolean		Utilizado pelo robô para verificar se a conexão com o controle remoto ainda está ativa. Este método é chamado periodicamente pelo robô. Caso não envie resposta o robô entra em modo de segurança até que a comunicação seja restabelecida.	
Interfaces implementadas			
IComunicRoboCR			

A classe Robô representa a estrutura principal do robô e agrega todos os seus elementos. A descrição completa de seus atributos pode ser observada na Tabela 80.

Tabela 80 - Descrição da classe Robô

Classe: Robô			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe representa o robô. Especifica suas informações gerais e é composta por um conjunto de partes que representam necessidades específicas do robô.			
Atributos		**Descrição**	
peso: *double*		Peso máximo que o robô deve atingir.	
tamanho: *double*		Altura do robô.	
protocolo: *String*		Protocolo de comunicação com o CR.	
Métodos		**Descrição**	
Nenhum			
Interfaces implementadas			
Nenhum			

Análogamente, as classes TransmissorRobo e ReceptorRobo (Tabelas 81 e 82 respectivamente) são responsáveis pela comunicação do robô com o controle remoto. Assim, a classe ReceptorRobo recebe os dados do controle remoto, tais como os comandos solicitados pelo usuário, e direciona-os para a classe correspondente para execução. A classe TransmissorRobo envia dados do robô para o controle remoto, a exemplo das imagens captaradas no ambiente controlado.

Tabela 81 - Descrição da classe TransmissorRobo

Classe: TransmissorRobo			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe representa o transmissor do robô. É responsável por enviar dados para o controle remoto.			
Atributos		**Descrição**	
Métodos		**Descrição**	
EnviarImagem(imagem: Imagem)		Envia uma imagem para o controle remoto.	
Interfaces implementadas			
IComunicRoboCR			

Tabela 82 - Descrição da classe ReceptorRobo

Classe: ReceptorRobo			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe representa o receptor do robô. É responsável por receber comandos e dados do controle remoto e enviá-los para o cérebro do robô para que seja processado.			
Atributos		**Descrição**	
Métodos		**Descrição**	
despacharComando(args: *String*[])		Recebe um comando do CR e passa para o cérebro do robô para ser interpretado e executado.	
conectar()		Estabelece conexão com o CR.	
desconectar()		Finaliza a conexão com o CR.	
controlarDispositivo(d: Dispositivo)		Chama o cérebro do robô para ajustar a câmera.	
Interfaces implementadas			
IComunicCRRobo			

A classe Controle representa o cérebro do robô. Ela é responsável pelo processamento de dados locais no robô.

Tabela 83 - Descrição da classe Controle

Classe: Controle			
Data da elaboração:	10/11/2006	Ultima alteração	26/12/2006
Descrição: Esta classe é responsável pelo controle local do robô. Recebe os comandos do CR e executa-os.			
Atributos		**Descrição**	
Métodos		**Descrição**	
executarComando (args: *String*[])		Recebe um comando, interpreta e chama o método responsável pela sua execução.	
ajustarCamera(configCamera: *String*[])		Ajusta a câmera de acordo com os parâmetros passados.	
carregarConfiguracao()		Carrega as configurações salvas na última vez em que o robô foi desligado.	
salvarConfiguracao()		Salva as configurações atuais do robô para que possam ser recuperadas quando for novamente ligado. As informações são as de configuração da câmera.	
Interfaces implementadas			

A classe Carcaça (Tabela 84) representa a estrutura que abriga os componentes do robô. No diagrama de classes é visualizada como uma agregação das demais classes.

Tabela 84 - Descrição da classe Carcaça

Classe: Carcaça			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe é uma agregação de outras classes que juntas compõem o corpo do robô.			
Atributos		**Descrição**	
Métodos		**Descrição**	
Interfaces implementadas			

A classe Locomoção (Tabela 85) representa os elementos necessários para que o robô se locomova dentro do ambiente controlado.

Tabela 85- Descrição da classe Locomoção

Classe: Locomoção			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe é responsável pela locomoção do robô no ambiente.			
Atributos		**Descrição**	
velocidade: *doublé*		Velocidade de deslocamento do robô no ambiente. Esta velocidade será sempre constante.	
Métodos		**Descrição**	
girar(direcao: *String*, graus: *double*)		Gira o robô na direção especificada (norte, sul, leste ou oeste). O giro será em graus de acordo com o especificado.	
andar()		Locomove o robô para frente na velocidade especificada no atributo.	
parar()		Interrompe a locomoção do robô.	
Interfaces implementadas			

Tabela 86 - Descrição da classe Dispositivo

Classe: Dispositivo			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe é responsável pelo controle dos dispositivos conectados ao robô.			
Atributos		**Descrição**	
nome: String		Nome do dispositivo.	
Métodos		**Descrição**	
Ajustar(dados: Informações)		Método abstrato responsável pelo ajuste do dispositivo de acordo com as informações fornecidas pelo CR.	
Interfaces implementadas			

A descrição da classe Dispositivo é apresentada na Tabela 86. Ela representa os dispositivos que podem ser controlados pelo robô. Desta forma, novos dispositivos podem ser inseridos a exemplo do de Visão, representado na Tabela 87.

Tabela 87 - Descrição da classe Visão

Classe: Visão			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe é responsável pelo dispositivo que controla a visão do robô.			
Atributos		**Descrição**	
ConfigCamera: Informações		Informações de configuração da câmera.	
Imagem: Imagem		Imagem capturada.	
Protocolo: String		Protocolo utilizado para transmitir as imagens.	
Métodos		**Descrição**	
iniciarFilmagem()		Inicia a captura de imagens.	
pararFilmagem()		Interrompe a captura de imagens.	
ajustar(dados:Informações)		Configura a câmera.	
Interfaces implementadas			

A classe Informações (Tabela 88) é uma classe de configuração dos dispositivos acoplados ao robô.

Tabela 88 - Descrição da classe Informações

Classe: Informações			
Data da elaboração:	10/11/2006	Ultima alteração	10/11/2006
Descrição: Esta classe é responsável por definir o formato das informações enviadas para um dispositivo.			
Atributos		**Descrição**	
posX: *doublé*		Indica a posição x para posicionamento do dispositivo, considerando um eixo cartesiano.	
posY: *doublé*		Indica a posição y para posicionamento do dispositivo, considerando um eixo cartesiano.	
Zoom: *doublé*		Indica o grau de aproximação da câmera.	
Métodos		**Descrição**	
Interfaces implementadas			

Tabela 89 - Descrição da classe Sensor

Classe: Sensor			
Data da elaboração:	26/12/2006	Ultima alteração	26/12/2006
Descrição: Esta classe é responsável pelos sensores de segurança do robô, a exemplo dos sensores de presença.			
Atributos		**Descrição**	
Tipo: *String*		Tipo do sensor.	
Métodos		**Descrição**	
Ler()		Efetua uma leitura do ambiente.	
Interfaces implementadas			

O robô também prevê a utilização de sensores de segurança como, por exemplo, o sensor de presença. Estes sensores estão representados no modelo pela classe Sensor, descrita na Tabela 89.

Tabela 90 - Descrição da classe Energia

Classe: Energia			
Data da elaboração:	26/12/2006	Ultima alteração	26/12/2006
Descrição: Esta classe é responsável pela alimentação de energia do robô.			
Atributos		**Descrição**	
Corrente: Inteiro		Indica a corrente permitida (110/ 220).	
Situação: *boolean*		Ligado ou desligado.	
Métodos		**Descrição**	
Desligar()		Desativa o fornecimento de energia.	
ligar()		Iniciar o fornecimento de energia.	
Interfaces implementadas			

A classe Energia (Tabela 90) é responsável pelo controle da fonte de energia que alimenta o robô.

A partir do diagrama de classes modela-se a estrutura dinâmica do robô. Esta estrutura representa a maneira como as classes interagem entre si trocando informações. Os diagramas de seqüência ilustrados a seguir apresentam estas interações.

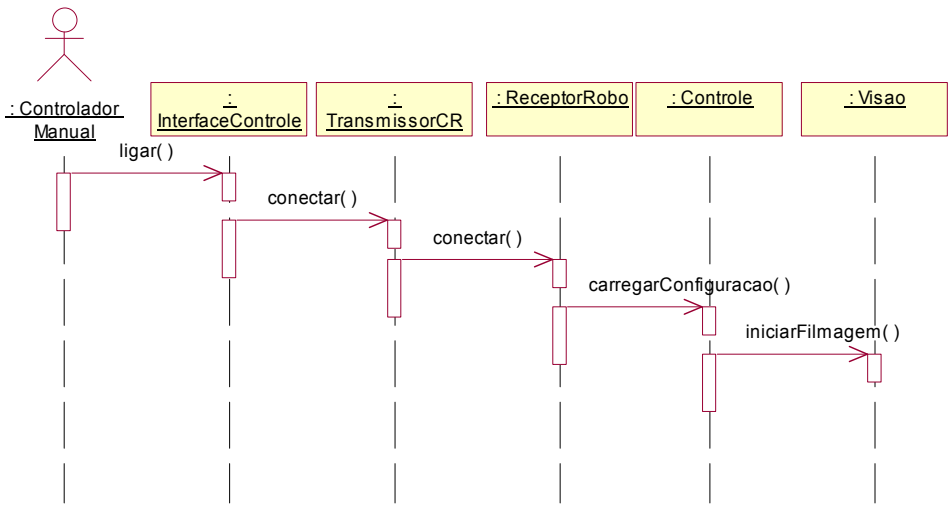


Figura 51 - Diagrama de seqüência para ligar o robô

Inicialmente a Figura 51 apresenta o comportamento do robô quando este é ligado. Pode-se observar que a ação se inicia a partir do ator controlador manual, solicitando à

interface de controle que o robô seja ligado. A partir desta solicitação então, é estabelecida uma conexão entre o transmissor de dados do CR e o receptor de dados do robô. A primeira atividade executada no robô quando ele é ligado é carregar as configurações que existiam quando ele foi desligado pela última vez e depois iniciar a filmagem enquanto aguarda novas instruções.

A Figura 52 apresenta o comportamento do robô quando a filmagem está sendo efetuada. Pode-se notar que o controlador do robô controla todo o processo. Primeiramente ele envia uma mensagem à classe Visão para que a filmagem seja iniciada. A partir deste momento, imagens são capturadas e tratadas, a partir da extração de características, e enviadas para o CR a partir da comunicação já estabelecida entre o transmissor do robô e o receptor do CR. Quando a imagem é recebida pelo CR ela é armazenada. Nota-se que todo este processo acontece de maneira periódica a cada intervalo de tempo *T*.

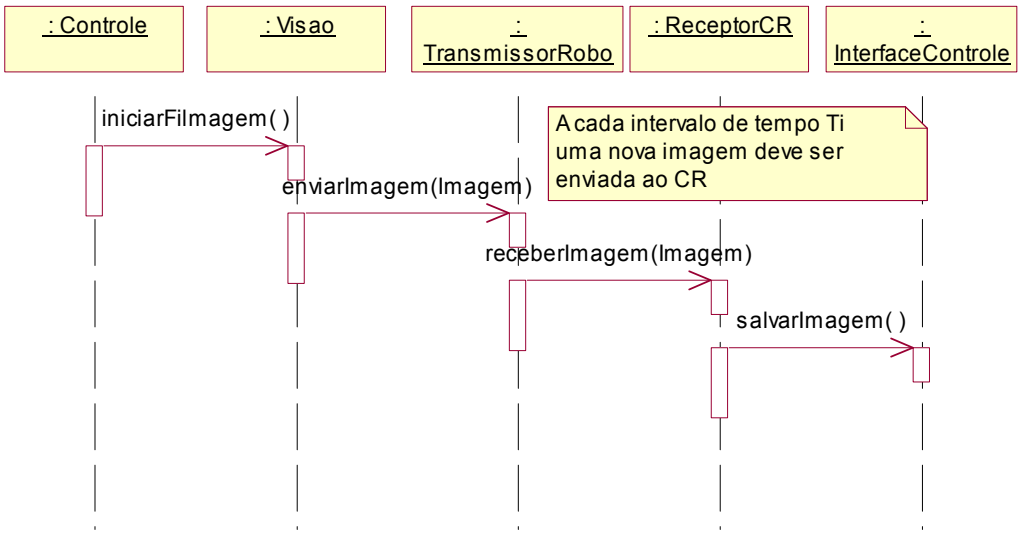


Figura 52 - Diagrama de seqüência da filmagem

Durante o seu funcionamento o robô pode também se locomover (Figura 53). O processo de locomoção envolve duas atividades principais: andar e parar.

Quando o usuário envia uma mensagem para o robô andar, ele especifica a direção e a angulação que deseja que o robô se movimente, caso exista. Estes dados são então enviados pelo transmissor do CR e recebidos pelo receptor do robô. Todo comando recebido pelo receptor do robô é enviado à classe Controle para ser devidamente executado. No caso da

locomoção, é solicitada a classe de Locomoção que o execute. Nota-se que o robô, sempre que necessário, executa um giro para a direção solicitada e se move sempre para frente.

É importante observar que tanto o comando para andar quanto para parar devem respeitar a restrições temporais de execução. Por exemplo, quando é solicitado que o robô pare, ele deve parar sua locomoção em no máximo no intervalo de tempo *T*. A quebra desta restrição pode acarretar problemas ao robô.

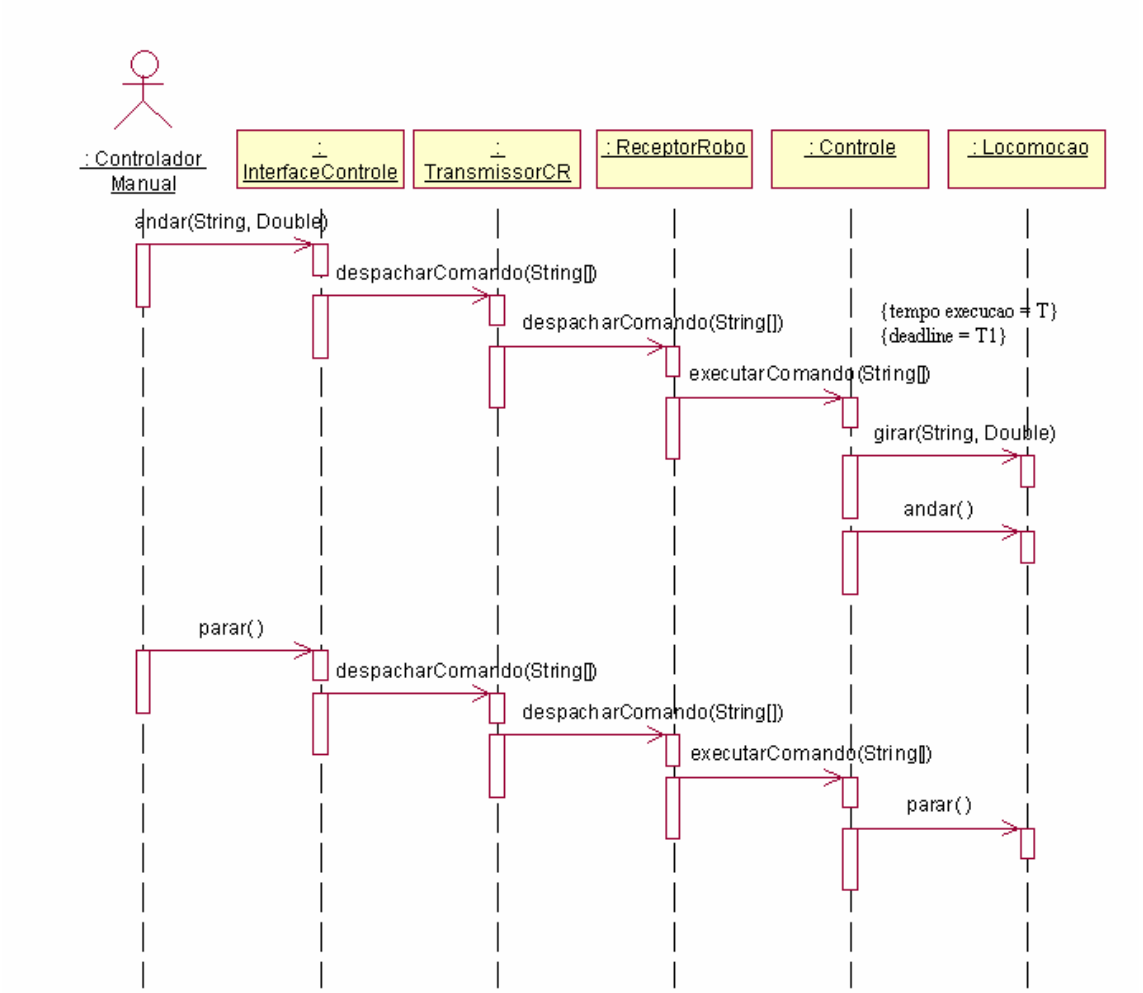


Figura 53 - Diagrama de seqüência da locomoção do robô

É possível que o usuário controlador deseje alterar a configuração da câmera durante a filmagem. O diagrama da Figura 54 ilustra este processo.

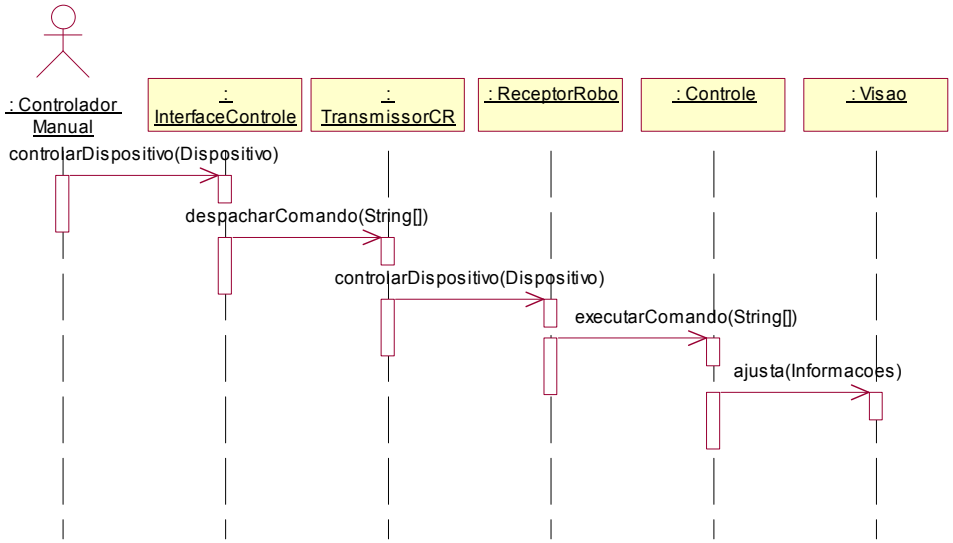


Figura 54 - Diagrama de seqüência de ajuste da câmera filmadora

Observa-se que a partir da InterfaceControle, o usuário especifica a configuração desejada para o dispositivo. Esta configuração é transmitida para o robô e a classe Controle se encarrega de enviá-la para o dispositivo de visão para que este seja ajustado.

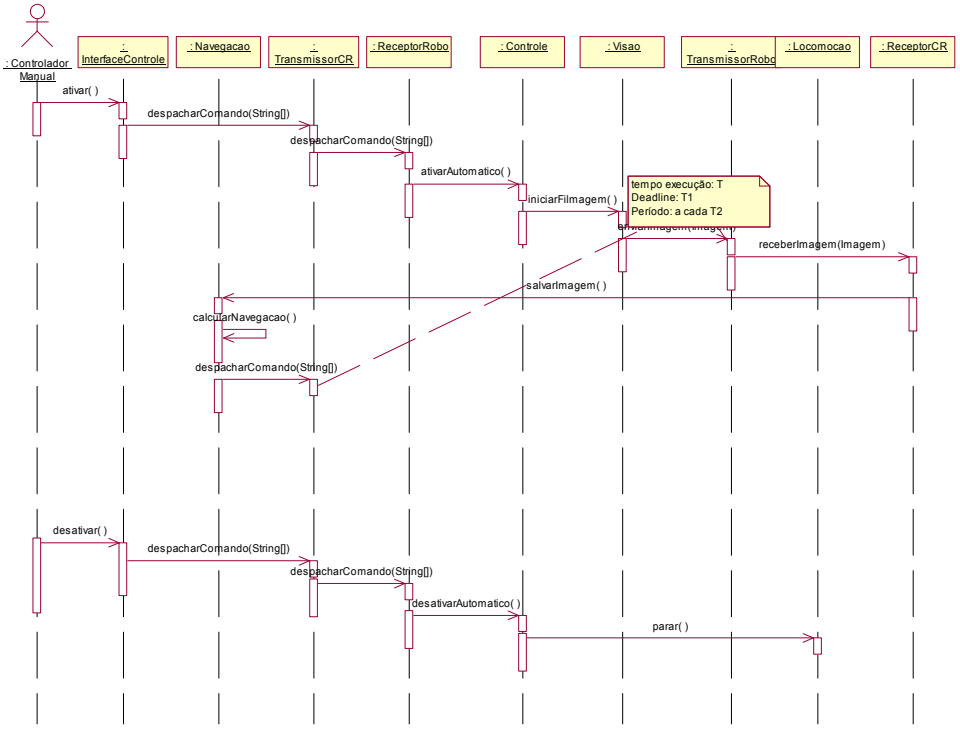


Figura 55 - Diagrama de seqüência para ativar/desativar controle automático

O controle do robô é executado sempre pelo CR, mas pode ser manual ou automático. Para que ele seja automático é necessário que esta opção seja ativada/desativada. O diagrama ilustrado na Figura 55 apresenta o processo de ativar / desativar o controle automático do robô.

Observa-se que quando o controle automático é ativado duas atividades principais acontecem: primeiro periodicamente, imagens são enviadas para o CR; segundo, estas imagens são analisadas por um sistema de navegação que calcula como o robô deve se locomover e envia o resultado deste cálculo para que o robô execute a locomoção planejada.

Este processo se repete até que o controle automático seja desativado. Neste momento, o sistema de cálculo de navegação é finalizado, pois este volta a ser controlado pelo usuário controlador.

Todo o processo de operação do robô, desde que ele é ligado, deve ser constantemente monitorado (Figura 56). Esta monitoria visa detectar interrupções na comunicação entre o CR e o robô e, nestes casos, colocar o robô em um estado seguro até que a comunicação (conexão) seja restabelecida.

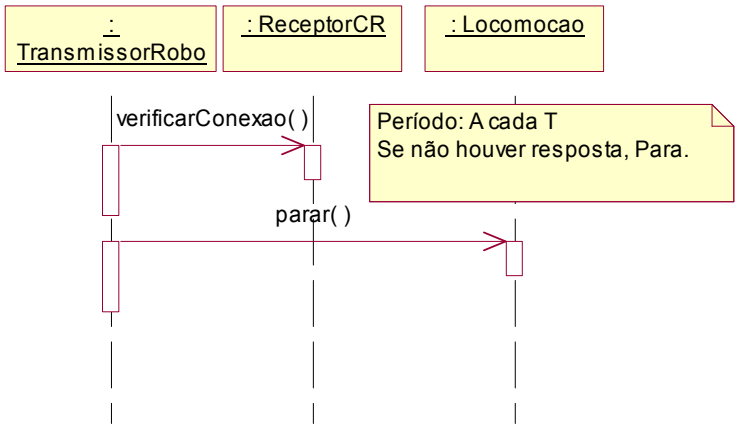


Figura 56 - Diagrama de seqüência de monitoria da comunicação com o robô

Ao final das atividades é necessário desligar o robô. O diagrama de seqüência da Figura 57 ilustra este processo.

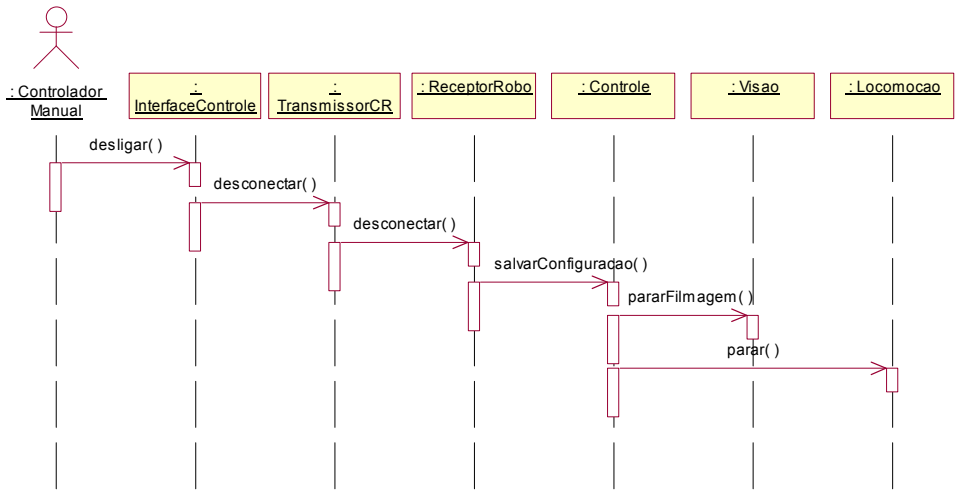


Figura 57 - Diagrama de seqüência para desligar o robô

O controlador manual envia um comando a partir da InterfaceControle para desligar o robô. Este comando é recebido pelo robô que solicita à classe Controle que o execute. Antes de finalizar as operações, as configurações atuais do robô são salvas para serem recuperadas quando o robô for novamente ligado. A câmera é desligada e o robô pára de se locomover.

A modelagem comportamental do produto envolve também a definição dos possíveis estados que este pode assumir durante sua utilização. Para o robô foi importante definir três diagramas de estados, conforme ilustrado nas figuras abaixo.

Observa-se na Figura 58 o diagrama de estados do controle remoto do robô. O CR sempre é iniciado em modo manual, podendo ser colocado ou tirado do automático pelo controlador manual.

No modo manual, inicialmente o robô está desligado. Quando é solicitado que ele seja ligado, o CR fica no estado de Conectando até que a conexão com o robô seja estabelecida e ele passe para o estado Ligado. Caso não seja possível efetuar a conexão, volta para o estado Desligado.

Quando o robô está Ligado, o CR inicialmente fica no estado de Aguardando comando. Assim que o usuário controlador informa um comando qualquer, este comando é enviado para o robô (Enviando comando) e o estado retorna a Aguardando comando. Caso esteja sendo executada uma filmagem, o CR pode também receber do robô uma imagem, que será armazenada (Salvando imagem), depois retornar ao estado de Aguardando comando.

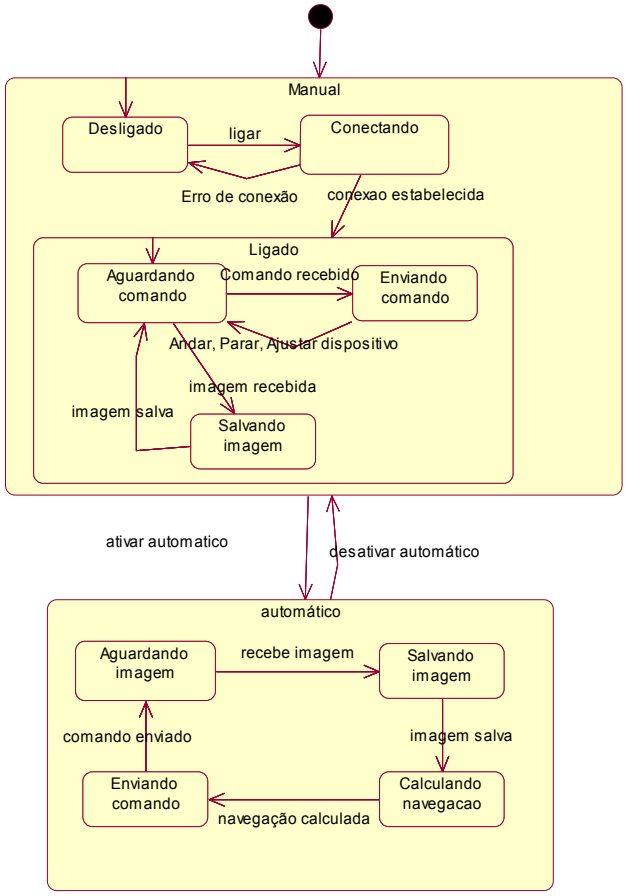


Figura 58 - Diagrama de estados do controle remoto

Quando o robô está sendo operado em modo automático, o CR fica sempre aguardando uma imagem. Uma vez que a imagem é recebida, ela é salva (Salvando imagem) e enviada para que seja efetuado o cálculo da navegação do robô (Calculando navegação). Depois que o cálculo é efetuado, um comando de navegação é enviado ao robô (Enviando comando) e o estado retorna ao Aguardando imagem para repetir todo o processo.

O diagrama de estados do robô é apresentado na Figura 59. Observa-se que o robô antes de iniciar as operações encontra-se no estado Desligado. Quando recebe um comando para ligar, ele passa para o estado Ligado e deste estado ele pode retornar ao estado Desligado. Quando o robô está ligado ele pode assumir dois subestados: Parado e Andando. Inicialmente ele está parado. Quando o usuário pede que ele ande, passa para o estado Andando e pode retornar ao estado Parado se receber um comando para parar.

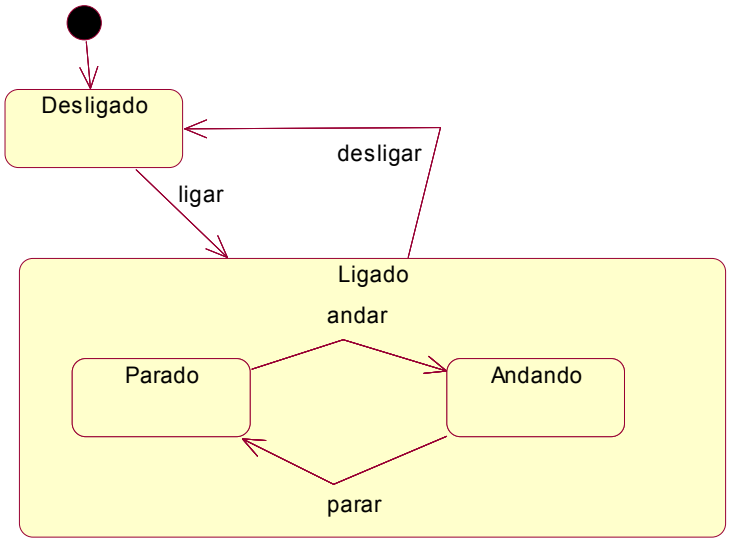


Figura 59 - Diagrama de estados do robô

A Figura 60 apresenta o diagrama de estados do processo de filmagem do ambiente pelo robô.

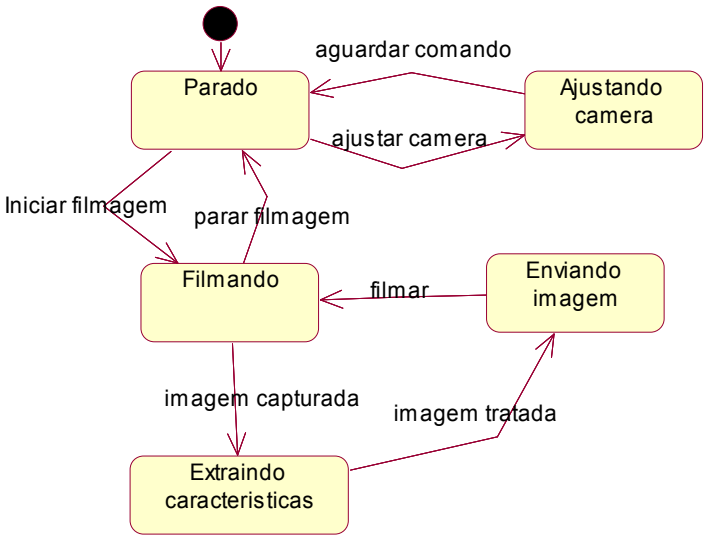


Figura 60 - Diagrama de estados com o processo de filmagem do robô

Inicialmente a câmera assume o estado de Parado. A partir deste estado ela pode receber um comando para ajustar a câmera e passar para o estado Ajustando câmera e depois retornar ao estado Parado. Pode também receber um comando para iniciar uma filmagem e passar para o estado Filmando. Neste caso, periodicamente ela enviará a imagem para a central. Antes de enviar a imagem ela extrai as características importantes para o envio

(estado Extraindo características). De posse da imagem já tratada, envia a imagem para a central, neste momento estará no estado Enviando imagem, e depois retornará ao estado Filmando. Quando receber um comando para interromper a filmagem, volta para o estado Parado.

A partir dos modelos estruturais e comportamentais definidos é desenvolvido o diagrama de componentes representado na MdpM pelo modelo abstrato do produto. A estratégia de definição dos componentes baseia-se no agrupamento das classes. Desta maneira um componente pode atender por completo a uma necessidade específica. O modelo, ilustrado na Figura 61, apresenta os componentes definidos para o controle remoto e para o robô. Nota-se que o componente Comunicação agrupa as classes TransmissorCR e ReceptorCR, provendo desta maneira todas os procedimentos necessários para efetuar a comunicação com o robô. De maneira análoga o componente ComunicacaoRobo agrupa as classes TransmissorRobo e ReceptorRobo, provendo esta funcionalidade para o robô.

Basicamente este projeto possui dois produtos, representados na figura como componentes compostos por subcomponentes, que são o controle remoto e o robô propriamente dito.

O primeiro componente, observado do lado esquerdo da figura, é o ControleRemoto. Ele compreende os subcomponentes InterfaceControle e Comunicação e possui cinco portas de comunicação com o meio externo. As três primeiras portas implementam as interfaces ILocomoção, IControle e IFilmagem, respectivamente, e permitem que um agente externo qualquer tenha acesso ao componente, pois representam os serviços disponíveis para serem utilizados pelo controlador manual na operação do robô. As duas outras portas implementam as interfaces IRecepcao e ITransmissao, e fornecem acesso direto ao componente Comunicação. O componente InterfaceControle se comunica com o subcomponente Comunicação através de duas portas. A primeira delas, EnviarComandos é responsável pelo envio dos comandos e dados que devem ser transmitidos para o robô. A segunda, ReceberDados, é responsável pelo recebimento dos dados enviados pelo robô para o controle remoto.

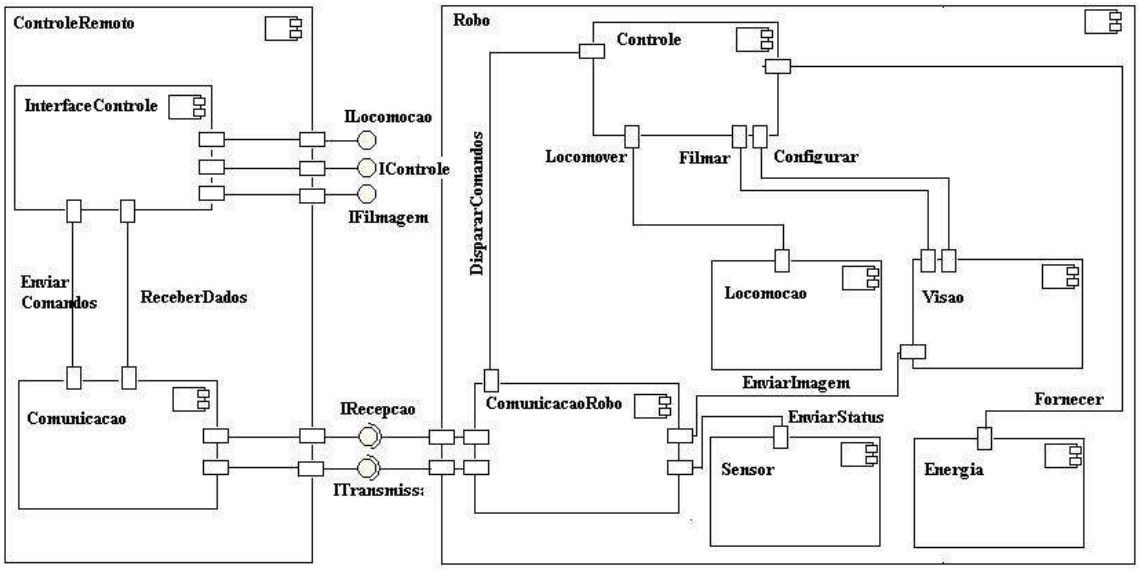


Figura 61 - Modelo abstrato do robô

O segundo grande componente é o Robô. Ele fornece duas portas com as interfaces requeridas para que haja a comunicação com o componente ControleRemoto. Desta forma, os subcomponentes Comunicação e ComunicacaoRobo se comunicam entre si a partir das interfaces definidas para eles, transmitindo dados e comandos. Cada comando recebido pelo subcomponente ComunicacaoRobo é enviado ao controle do robô a partir da porta DispararComandos para que seja traduzido e executado nos demais componentes que formam o robô. Existe uma comunicação direta entre o subcomponente Visão e o subcomponente ComunicacaoRobo que permite enviar as imagens capturadas diretamente para o componente ControleRemoto. Existe também uma comunicação direta entre o subcomponente Sensor e o subcomponente ComunicacaoRobo que permite enviar um estado com a situação atual do robô.

Pode-se dizer que o projeto do robô já atingiu um nível conceitual bastante detalhado neste momento. A partir de então, é iniciado o processo de definição da melhor solução tecnológica para implementação de cada um dos componentes.

A identificação da solução começa com a análise do modelo abstrato para construção da matriz morfológica ilustrada na Tabela 91. Observa-se que para cada um dos componentes identificados no modelo sugerem-se alguns tipos de implementação, nas diversas áreas envolvidas.

Tabela 91 - Matriz morfológica do robô

Componente	Opções de Implementação					
	Software		Mecânica	Eletrônica		Mecatrônica
InterfaceControle	Software para PC.	Software para PalmTop.				
Comunicação	Internet.			Radio.	Rede s/ fio.	
ComunicacaoRobo	Internet.			Radio.	Rede s/ fio.	
Controle				Micro controlador.		PalmTop.
Visão			Sensor.			Câmera.
Locomoção			Receptor IR. Micro controlador. Motor. Carcaça.			
Sensor			Sensor de presença.			
Energia				Estabiliz ador.	Fonte.	

A partir da matriz morfológica apresentada acima, escolhe-se uma concepção de projeto para o robô, combinando as soluções sugeridas. A concepção escolhida para o projeto do robô é apresentada na Tabela 92.

Tabela 92 - Concepção de projeto para o robô

Componente	Implementação	Concepção escolhida
InterfaceControle	Construir	Sistema em um PC.
Comunicação	Adquirir	Rede sem fio.
ComunicacaoRobo	Adquirir	Rede sem fio (PalmTop).
Controle	Adquirir	PalmTop.
Locomoção	Construir	Receptor IR, micro controlador, motor e carcaça.
Visão	Adquirir	Câmera filmadora.
Sensor	Adquirir	Sensor de presença.
Energia	Adquirir	Estabilizador.

A partir das decisões de implementação, pode-se observar o modelo concreto conforme definido na Figura 62.

Nota-se uma simplificação do diagrama após a escolha da concepção de projeto que será produzido. Por exemplo, o modelo concreto unifica os componentes Controle e ComunicaçãoRobo com a utilização do PalmTop, que já provê recursos para atender a estas duas necessidades. Esta decisão não demandou revisão do diagrama de classes, pois o subcomponente PalmTop continua implementando as mesmas interfaces previamente definidas.

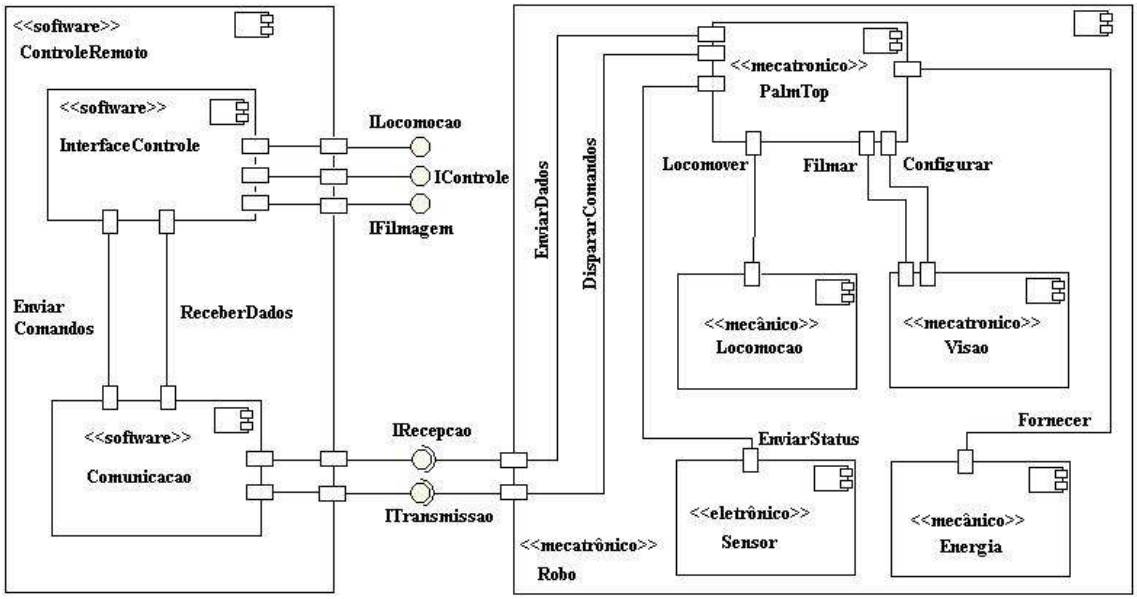


Figura 62 - Modelo concreto do robo

Com base nos componentes do modelo concreto, são definidos os testes que deverão ser realizados quando estes estiverem implementados. Para cada um dos componentes são especificados as situações que devem ser testadas.

Tabela 93 - Testes para o componente InterfaceControle

Componente:	**InterfaceControle**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Comando Ligar.	Chamar componente de comunicação para estabelecer uma conexão com o robô; Mudar o status situacaoRobo.
	Comando Desligar.	Chamar o componente de comunicação para finalizar a conexão com o robô; Mudar o status situacaoRobo.
	Comando Ativar controle remoto.	Iniciar sistema de navegação; Mudar o status de formaDeControle.
	Comando andar. Testar cada uma das direções.	Chamar o componente Comunicação com o comando de locomoção na direção e angulação solicitada.
	Comando Parar.	Chamar o componente Comunicação com o comando de parada.
	Comando desativar ControleRemoto.	Finalizar o sistema de navegação; Mudar o status de formaDeControle.
	Comando ControlarDispositivo com um dispositivo válido.	Enviar dados para o componente Comunicação e repassar para o robô.
	Comando ControlarDispositivo com um dispositivo inválido.	Mensagem de dispositivo inválido.

A Tabela 93 apresenta os testes do componente InterfaceControle. Nota-se que é identificado o papel responsável pelo teste, o que será testado e qual o resultado que se espera obter quando o teste for realizado.

Tabela 94 - Testes para o componente Comunicação

Componente:	**Comunicação**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber como entrada o comando ligar com uma conexão válida.	Conectar com o robô.
	Receber como entrada o comando ligar com uma conexão inválida.	Mensagem que não consegue se conectar.
	Estando conectado, receber como entrada um comando válido qualquer.	Enviar o comando para o robô.
	Sem estar conectado, receber como entrada um comando válido qualquer.	Mensagem informando que não está conectado ao robô.
	Receber uma imagem do robô.	Chamar o componente Interface para armazenar a imagem.
	Receber solicitação para verificar conexão.	Enviar mensagem de conexão.

A Tabela 95 mostra os testes do componente Comunicação e devem ser realizados pelo testador técnico. O primeiro teste analisa se o robô esta respondendo ao comando Ligar. Para que o funcionamento esteja correto, o resultado esperado é que o robô esteja conectado ao CR após o processamento do comando.

Tabela 95 - Testes do componente ComunicacaoRobo

Componente:	**PalmTop**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber como entrada uma solicitação para estabelecer conexão.	Carregar as configurações iniciais.
	Receber como entrada um comando válido qualquer.	Executar o comando.
	Recebe como entrada um comando inválido qualquer.	Nada acontece.
	Recebe como entrada um comando para controlar um dispositivo.	Executar o comando.
	Recebe um comando para desconectar.	Salvar as configurações atuais.
	Receber uma imagem.	Envia para o controle remoto.

Análogamente, as tabelas que se seguem descrevem os testes que serão aplicados a cada um dos componentes quando estes estiverem prontos.

Tabela 96 - Testes do componente Locomoção

Componente:	**Locomoção**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber um comando para andar estando parado ou em movimento.	Girar para a angulação correta; Iniciar locomoção na velocidade configurada.
	Receber um comando para parar.	Interromper locomoção em 50 milisegundo.

Tabela 97 - Testes do componente Visão

Componente:	**Visão**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber comando para iniciar filmagem.	Iniciar a filmagem; A cada 30 milissegundos, enviar a imagem para o componente ComunicacaoRobo.
	Receber comando para finalizar filmagem.	Parar a filmagem.
	Receber o comando para ajustar o dispositivo.	Substituir a configuração atual pela nova configuração informada.

Tabela 98 - Testes do componente Energia

Componente:	**Energia**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Receber uma corrente diferente de 110 e 220.	Desligar o robô.

Tabela 99 - Testes do componente Sensor

Componente:	**Energia**	
Quem realiza	O que testar	Resultado esperado
Testador Técnico	Colocar um obstáculo na frente do robô.	Indicação de status com obstáculo; Robô parado.
	Derrubar o robô.	Indicação de robô em situação anormal; Robô parado.
	Não colocar nenhuma anormalidade.	Nada acontece.

Além dos testes individuais dos componentes, também são projetados os testes do produto como um todo, conforme ilustrado na Tabela 100. Pode-se observar, por exemplo, o teste de ligação do robô, onde varias situações são testadas: ligar o robô enquanto ele estiver desligado, neste caso a conexão deve ser estabelecida; ou ligar o robô desligado, neste caso nenhum evento deve ocorrer; Outros testes também são especificados de maneira que todos os requisitos seja testados da maneira como deveriam funcionar ou de outra maneira qualquer que possa ser realizada pelo usuário para que o tratamento adequado seja efetuado.

Tabela 100 - Teste do produto

Teste do produto		
Quem realiza	O que testar	Resultado esperado
Testador Cliente / Testador Técnico	Ligar o robô desligado.	Conexão estabelecida e configurações iniciais carregadas; Mensagem de robô ligado aguardando comando.
	Ligar o robô ligado.	Nada deve acontecer, pois o robô já está em operação.
	Andar com o robô ligado.	Iniciar a locomoção no sentido selecionado; Receber imagens no tempo de 30 milisegundo.
	Andar com o robô desligado.	Nada acontece.
	Parar com o robô em locomoção.	Robô parado em no máximo 50 milisegundo.
	Parar com o robô já parado.	Nada acontece.
	Ajustar dispositivo com o robô ligado; Informar dados de parâmetro para o dispositivo.	Dispositivo ajustado corretamente.
	Ajustar dispositivo com o robô ligado; Informar dados errados ou em branco para os parâmetros.	Nada muda no dispositivo; Mensagem que os parâmetros estão errados.
	Ajustar dispositivo com o robô desligado.	Nada acontece.
	Ativar automático com o robô ligado.	Automático ativado; Sistema de navegação em operação.
	Ativar automático com o robô desligado.	Nada acontece.
	Desativar automático com o robô no automático.	Automático desativado; Robô parado aguardando comando.
	Desativar automático com o robô no manual.	Nada acontece.
	Colocar obstáculo perto do robô com o robô em movimento.	Robô parado.

Durante as atividades da segunda fase, validações foram realizadas conforme apresentado na Tabela 101.

Como ilustrado no documento de validação, são identificados os envolvidos no processo, com seus respectivos papéis. Devem ser identificados pelo menos o gerente técnico, o gerente do cliente e um líder para cada uma das áreas envolvidas.

Os artefatos gerados em cada uma das fases da metodologia devem ser listados com suas respectivas datas de aprovação em reuniões multidisciplinares.

Na prática, o documento de validação começa a ser construído na primeira reunião de formação da equipe do projeto e vai sendo atualizado à medida que as reuniões de validação dos artefatos vão acontecendo.

Tabela 101 - Documento de validação da fase Elaboração

Documento de Validação	
Data de início do projeto: 27/10/06	
Equipe Responsável	
Identificação	Nome
Gerente do cliente	Especificações de [37]
Gerente Técnico	Ana Patrícia.
Líder Computação	Ana Patrícia.
Líder Mecânica	Hermam.
Concepção	
Data da validação	Artefatos
30/10/06	Casos de uso.
02/11/06	Lista de requisitos não funcionais.
04/11/06	Casa da qualidade; Soluções de conflitos.
05/11/06	Fatores de risco.
08/11/06	Documento de escopo.
Elaboração	
Data da validação	Artefatos
10/11/06	Descrição dos casos de uso.
15/11/06	Modelagem estrutural (Diagrama de classes).
20/11/06	Modelagem comportamental (Diagrama de estados).
25/11/06	Modelo abstrato (diagrama de componentes).
26/12/06	Modelo concreto (diagrama de componentes particionado).
26/12/06	Casos de teste.

c) Desenvolvimento

O desenvolvimento consiste na construção, por cada uma das áreas específicas, dos seus respectivos componentes. Após a construção, estes componentes são integrados, prototipados e testados para gerar o modelo do produto que será enviado à linha de produção.

Esta fase não foi desenvolvida neste experimento prático por falta de recursos financeiros e de tempo.

d) Entrega

Nesta fase os modelos construídos durante a especificação do robô são reunidos e a documentação do produto é gerada.

e) Esforço realizado na especificação do Robô.

O desenvolvimento do robô gerou um conjunto de artefatos representados por documentos textuais, diagramas, tabelas, etc que guardam todas as definições realizadas pela

equipe durante o projeto. A Tabela 102 apresenta um resumo da quantidade de documentos gerados, como medida de esforço para avaliar o tamanho do projeto.

Tabela 102 – Esforço para a construção do robô

Elemento	Quantidade
Necessidades do cliente	9
Casos de uso	14
Requisitos não funcionais	15
Fatores de risco	2
Casa da qualidade	1
Tabelas de descrição de caso de uso	9
Tabela de descrição de requisitos não funcionais	9
Componentes	10
Matriz morfológica	1
Concepções de projeto	1
Tabelas de teste	7
Classes	15
Interface	5
Tabelas de descrição das classes	15
Diagramas de seqüência	7
Diagramas de estados	3