

## **Analysis and Simulation of Non-Functional Requirements Applied to an Air-Traffic Control System**

Wagner A. P. Coimbra (University of São Paulo, Brazil) – wagner.coimbra@poli.usp.br

Álvaro R. S. Fialho (University of São Paulo, Brazil) – alvaro.fialho@poli.usp.br

Marco T. C. de Andrade (University of São Paulo, Brazil) – marco.andrade@poli.usp.br

Edson Satoshi Gomi (University of São Paulo, Brazil) – edson.gomi@poli.usp.br

Colaboradores:

Reginaldo Arakaki (University of São Paulo, Brazil) – reginaldo.arakaki@poli.usp.br

Eduardo C. Giannotto (University of São Paulo, Brazil) – eduardo.giannotto@poli.usp.br

At the beginning of a system analysis, essential information is produced. Such information is mandatory for the system quality, in a way that the remaining development tasks will, somehow, depend on it. At the initial stage, the work subject is the knowledge (how to acquire it, how to organize it and how to verify it). Based on this context, this work presents a knowledge validation method through the usage of concept proof simulations. Such knowledge validation is achieved through a process of identification of intrinsic system architecture risks and its formalization as non-functional system requirements, which might be used by the next stages of the software engineering process. This process was applied in the analysis of an Air-Traffic Control System, and, as a result, it was possible to verify improvement points, to measure the enhancements obtained by each improvement proposal and to identify the critical functions of the system.

**Keywords:** Non-functional requirements, simulation, system architecture, system project, critical system.

### **Análise e Simulação de Requisitos Não-Funcionais Aplicada a um Sistema para Controle de Tráfego Aéreo**

No início da análise de um sistema, são produzidas informações responsáveis por guiar o desenvolvimento e pela aderência do produto final às necessidades da aplicação. Nesse momento inicial, o principal objeto de trabalho é o conhecimento, sendo necessário determinar como adquiri-lo, representá-lo e, principalmente, verificá-lo. Neste artigo é apresentado um método para a verificação desse conhecimento, baseando-se na prova de conceito por meio de simulações. Isso permite explicitar, a partir de um levantamento sobre os riscos intrínsecos à arquitetura de um sistema, requisitos não-funcionais, os quais são úteis para as demais etapas de engenharia. Esse processo foi aplicado na análise de um sistema hipotético para controle de tráfego aéreo e, como resultado, foi possível identificar pontos passíveis de melhoria, medir a eficiência de cada proposta de melhoria e identificar as funções críticas do sistema.

**Palavras-Chave:** Requisitos não-funcionais, simulação, arquitetura de sistemas, projeto de sistemas, sistemas críticos.

## 1. INTRODUÇÃO

Na Engenharia de Software, é relativamente comum a realização de análises voltadas à implementação, em detrimento dos conceitos. Normalmente, isso indica vícios de comportamento e lacunas de conhecimento sobre como realizar modelagem no domínio do problema (Arakaki, 2006: 4). Considerando que o conhecimento produzido na fase inicial de análise é essencialmente conceitual, além de ser potencialmente incompleto e incorreto, é grande a possibilidade do produto final não corresponder às expectativas iniciais.

Visando auxiliar a validação do conhecimento produzido pelas primeiras análises, este artigo apresenta um método para identificação de riscos intrínsecos à modelagem. São seguidos princípios semelhantes aos da ATAM – “*Architecture Tradeoff Analysis Method*” (Kazman *et al*, 1998: 1), mas de forma que, após a modelagem conceitual do sistema, é iniciada uma busca por falhas e suas soluções. Com isso é feita a simulação do impacto dessas falhas sobre o sistema, a fim de medir o grau de eficiência das soluções propostas. O resultado disso é a explicitação das limitações do modelo, na forma de requisitos sobre aspectos não-funcionais do sistema (restrições sobre o ambiente operacional e sobre a qualidade esperada dos algoritmos).

Considerando também que são comuns as interações e dependências sistêmicas, é necessário utilizar ferramentas de análise capazes de representar tanto aspectos estáticos (estruturais) como dinâmicos (comportamentais). Para tanto, foi aplicado o conceito de simulação, onde, na visão deste trabalho, componentes estruturais da modelagem são representados na forma de classes e a sua interação sistêmica é representada pela execução de métodos.

Foi escolhida como classe de aplicação o controle de tráfego aéreo, por se tratar de uma área com baixa tolerância a falhas. Dado esse contexto de aplicação, o aspecto de qualidade considerado mais importante foi a Precisão (ou Acurácia), uma sub-característica de Funcionalidade definida na norma NBR ISO/IEC 9126-1 (Associação Brasileira de Normas Técnicas, 2003: 7 e Koscianski *et al*, 1999: 38). A partir disso, foi modelado um sistema hipotético para controle de voo e realizada uma busca por falhas intrínsecas à modelagem, bem como possíveis soluções. Com esses dados foi implementado um simulador, sobre o qual foram aplicadas diferentes taxas de erro e identificados limites operacionais com base nas soluções propostas.

Nas próximas seções, os resultados deste trabalho de pesquisa são apresentados da seguinte forma:

1. O Controle de Tráfego Aéreo: aqui é descrito o domínio da aplicação, apresentado o processo de análise utilizado, o modelo de negócio, os principais casos de uso e funções do sistema, o modelo estático da solução (classes conceituais) e o modelo dinâmico (cenários);
2. Análise de Riscos: nesta seção é apresentado o processo de identificação de riscos, bem como os riscos identificados e propostas de soluções;
3. Simulação de Riscos Intrínsecos à Arquitetura do Sistema: este tópico consiste na apresentação do modelo do simulador, dos cenários simulados, dos resultados obtidos e das análises desses resultados;
4. Considerações finais.

## 2. O CONTROLE DE TRÁFEGO AÉREO

A partir da análise sobre trabalhos de validação de planos de voo e monitoramento de aeronaves em tempo real (Comando da Aeronáutica, 2000: 103 e Ministério da Aeronáutica, jun. 1999: 39), foi criado um modelo orientado a objetos representando a arquitetura de um sistema hipotético para controle de tráfego aéreo.

Para determinar o efeito de falhas sobre tal arquitetura, foi criado um programa de simulação com foco no subsistema mais crítico: o monitoramento em tempo real das aeronaves em voo, cujos riscos simulados foram levantados segundo os aspectos de Precisão da característica Funcionalidade, definida na NBR ISO/IEC 9126-1 (Gomes, 2000: 4).

Para verificar o comportamento da solução modelada, foram introduzidas, no simulador, seqüências de dados apresentando diferentes percentuais de erro. Para representar a ocorrência de falhas num ambiente real, tais erros são distribuídos no tempo de forma pseudo-aleatória, de acordo com uma distribuição de probabilidades. Com isso, foi criada uma base de conhecimento sobre as tendências comportamentais do sistema final, antes mesmo de qualquer implementação.

### 2.1. O processo para estabelecimento de requisitos não-funcionais

A fim de validar o conhecimento produzido na fase inicial de análise e explicitar limitações e requisitos não-funcionais do sistema, foi concebido o seguinte processo de trabalho:

1. Escolha de um domínio de aplicação;
2. Escolha de um modelo de qualidade de *software*;
3. Levantamento de dados conceituais no domínio de aplicação;
4. Identificação de casos de uso, para estabelecer o escopo da aplicação;
5. Modelo de Negócio em IDEF0 (identificando as principais funções do sistema);
6. Modelo Estático em Diagrama de Classes (arquitetura do sistema);
7. Análise Comportamental com Diagramas de Seqüência;
8. Identificação de riscos e possíveis tratamentos (segundo o modelo de qualidade escolhido);
9. Modelagem e construção de um simulador, com foco na arquitetura, em seus riscos e em possíveis soluções;
10. Análise dos resultados, confrontando o obtido com o esperado;
11. Estabelecimento de requisitos não-funcionais.

Nas etapas (1) a (8) é feita a modelagem de acordo com princípios bem conhecidos de análise. Na etapa (9) é criado um simulador de arquitetura, cujo objetivo é demonstrar o comportamento do sistema, o efeito de falhas e a eficácia de soluções. Na etapa (10) os resultados são avaliados, com eventuais retornos à fase (9), caso tenha sido necessário realizar alterações sobre a arquitetura. Ao ser obtido um comportamento adequado na etapa (10) torna-se possível iniciar a etapa (11), onde restrições sobre o ambiente operacional e sobre algoritmos são registradas na forma de requisitos não-funcionais.

## 2.2. O Modelo de Negócio

Para definir o escopo e as principais funções do sistema, as situações de trabalho foram registradas na forma de diagramas de casos de uso e a modelagem dos processos de negócio utilizou a técnica *Integrated Computer Aided Manufacturing Definition* (IDEF) em sua notação mais básica IDEF0 – *Integration DEFinition for Function (0)*.

O Controle de Tráfego Aéreo utiliza como principal ferramenta um documento chamado Plano de Vôo (Comando da Aeronáutica, 2000: anexo I). Trata-se de um formulário contendo informações sobre o vôo (aeroportos de origem e de destino, aeroportos alternativos de destino, escalas, horário de decolagem e de pouso, rota, altitude, etc) e sobre a aeronave (marca, modelo, categoria, velocidade máxima, altitude máxima, etc).

O Plano de Vôo deve ser preenchido pela tripulação da aeronave ou pela empresa aérea responsável pelo vôo, sendo posteriormente validado pela Torre de Comando, que observará erros de preenchimento e rotas conflitantes (rotas já em uso ou que apresentem potenciais de colisão). Não havendo tais erros, o plano é autorizado e o vôo monitorado pela Torre de Comando.

Foram identificados, com base na análise realizada, os seguintes casos de uso: "Insere Plano de Vôo", "Modifica Plano de Vôo" e "Monitora Aeronaves", disponíveis ao Operador da Torre; "Atualiza Posição", que ocorre quando aeronaves informam a torre sobre sua posição atual; e "Recupera logs e relatórios" disponível ao Supervisor da Torre para atividades administrativas. Na Figura 1 pode ser vista a relação entre tais casos.

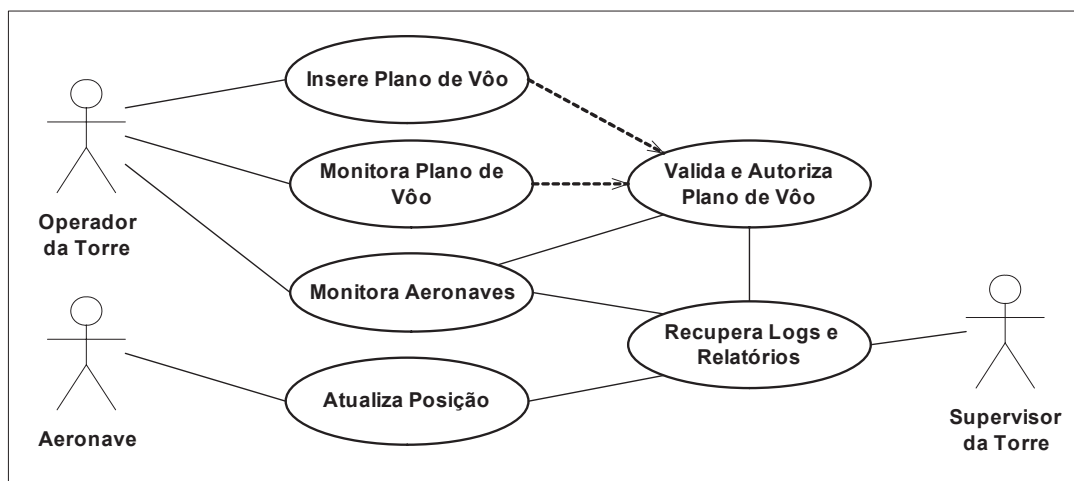


Figura 1 - Casos de Uso do Controle de Tráfego Aéreo

Considerando que o foco deste artigo é mostrar um processo para identificação de riscos e o conseqüente estabelecimento de requisitos não-funcionais, não foi necessário considerar os diferentes tipos de operador existentes na Torre de Comando. Além disso, foi assumido que ou existe um sistema de radar capaz de identificar aeronaves e suas posições, ou todas as aeronaves possuem mecanismos para obter e transmitir à Torre de Comando

dados de telemetria, tais como latitude, longitude, altitude, azimute, elevação, etc, utilizados na monitoração (Comando da Aeronáutica, 2002: 301-1).

A partir dos principais casos de uso levantados, foi construída a visão inicial dos processos de negócio. Isso permitiu identificar quatro processos fundamentais: "Valida Rota", "Localiza Aeronave", "Monitora Rota" e "Administração", representados na Figura 2.

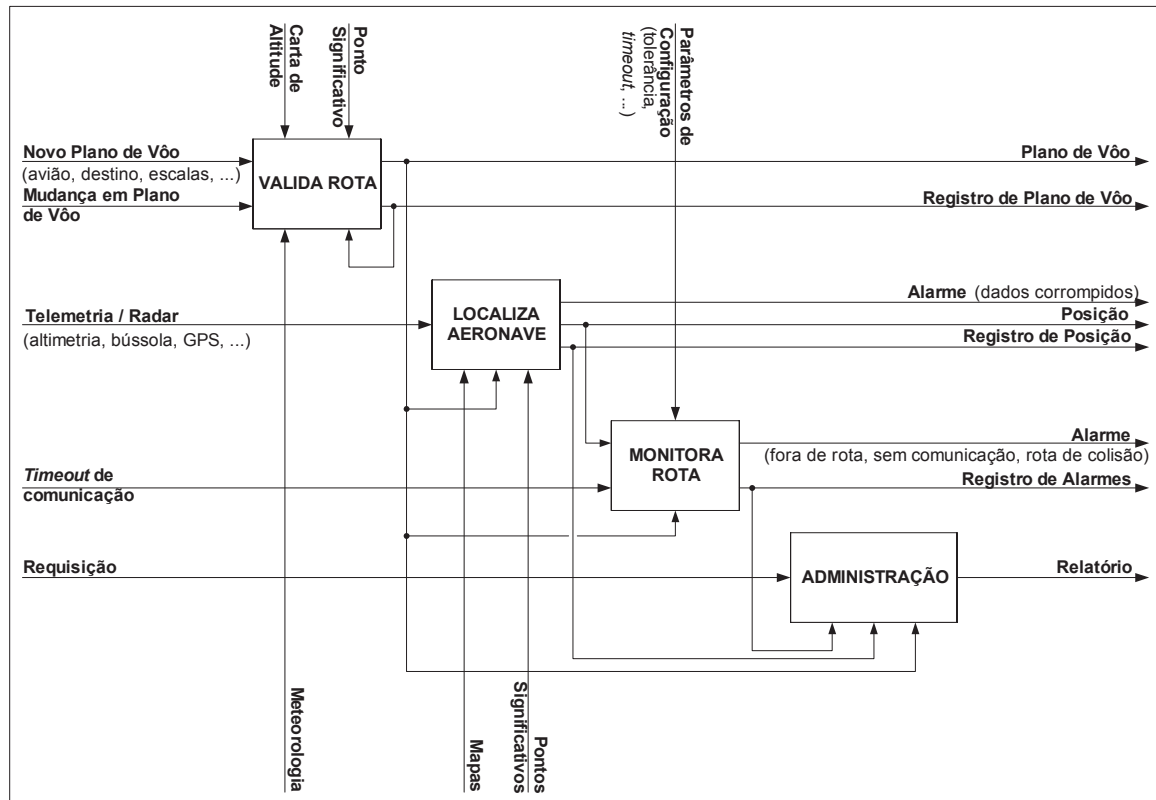


Figura 2 - Modelo de Negócio do Controle de Tráfego Aéreo (IDEF0)

### 2.3. Modelo Estático

De posse da visão do processo de negócio, foi iniciada a identificação das estruturas estáticas do sistema para Controle de Tráfego Aéreo.

O modelo estático de um sistema representa, de forma conceitual, os componentes estruturais dos processos de negócio. Podem ser utilizados diferentes níveis de abstração, como “somente componentes estruturais”, “componentes e operações” ou “componentes, operações e dados manipulados”. Nesta análise, foi utilizado o nível mais abstrato – somente componentes – na forma de um diagrama de classes, conforme é apresentado na Figura 3.

Esta concepção de estrutura estática dá suporte ao seguinte processo de negócio: uma “Torre” controla diversas “Regiões”. Cada “Região” possui um “Catálogo de Rotas” com todas as “Aerovias” que passam por determinada “Região”. Os “Planos de Voo” são registrados na “Região”. Os “Planos de Voo” interligam pelo menos dois “Aeroportos” utilizando uma ou mais “Aerovias”. A “Torre” também possui um “Radar”, que obtém e



necessários estão presentes, corretos e não-conflitantes. Também pôde ser derivado o cenário "Operador Insere novo Plano de Vô: Rota já está em uso", em que o sistema deverá informar a ocorrência de um erro ao operador e não permitir a autorização do vô.

Neste trabalho foi utilizado o cenário "Verifica Plano de Vô: Caso Ideal", concebido a partir dos casos de uso "Monitora Aeronaves" e "Atualiza Posição". Esse cenário representa a situação em que a posição de uma aeronave é atualizada com dados corretos e válidos. O cálculo de sua posição é feito de acordo com o esperado e obtém-se a confirmação de que a trajetória da aeronave está de acordo com as especificações de seu plano de vô (horário e localização corretos). O diagrama de seqüência da Figura 4 representa esse cenário.

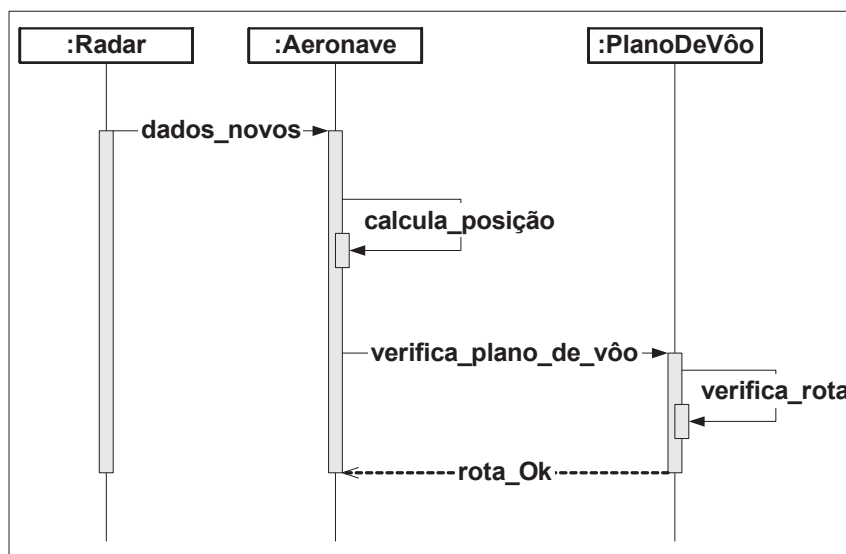


Figura 4 – Modelo Dinâmico: Diagrama de Seqüência do cenário "Verifica Plano de Vô"

Foram utilizadas três das classes presentes na modelagem inicial, as quais possuem a seguinte dinâmica: ao receber uma nova informação, o objeto “:Radar” gera um evento fazendo com que o objeto “:Aeronave” trate essa informação. O objeto “:Aeronave” inicia, então, os processos necessários para o cálculo da posição da aeronave. Com a posição calculada, o objeto “:Aeronave” utiliza um recurso oferecido pelo objeto “:PlanoDeVô” para verificar se tal posição está de acordo com o plano de vô estabelecido, o que é feito por meio de consultas a uma base de dados contendo os planos autorizados.

O diagrama da Figura 4 representa o caso de sucesso da operação de verificação de posição. No entanto, é possível derivar situações que evidenciem possíveis falhas no processo, o que corresponde ao levantamento de riscos inerentes à arquitetura do sistema.

### 3. ANÁLISE DE RISCOS

Ao tratar um risco relacionado a uma dada característica de qualidade de *software*, interações sistêmicas ocorrem, fazendo com que aumentem ou surjam outros riscos, talvez associados a outras características de qualidade. Devido a isso, uma boa prática é garantir um equilíbrio entre as diversas ações de correção. Por exemplo, para aumentar Segurança (um aspecto de Funcionalidade da NBR ISO/IEC 9126-1) é provável um aumento no consumo de Recursos (relacionado à Eficiência do *Software*). O objetivo do analista deve

ser a obtenção de um equilíbrio entre todas as características de qualidade, o que será estabelecido de acordo com a aplicação.

Por esta razão, deve-se realizar a análise de riscos considerando-se as características de qualidade mais críticas do projeto (no caso deste trabalho, os aspectos relacionados à Funcionalidade – Precisão).

### 3.1. Identificação de Riscos

Partindo do pressuposto de que, dado um processo, os principais elementos sujeitos a falhas são os dados de entrada e as transformações realizadas pelo processamento em si (Figura 5), a partir do modelo dinâmico do sistema (Figura 4) foram mapeados os riscos associados aos processos e seus dados de entrada. Os riscos sobre os dados de saída não são considerados diretamente, mas indiretamente quando tais dados se tornam entradas para outros processos.

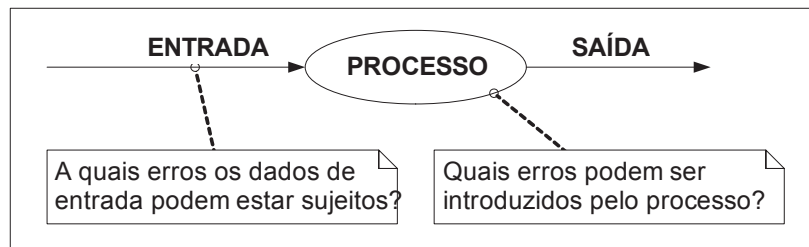


Figura 5 – Identificando Possíveis Falhas

Ao aplicar esta técnica, foram identificados os riscos apresentados na Figura 6.

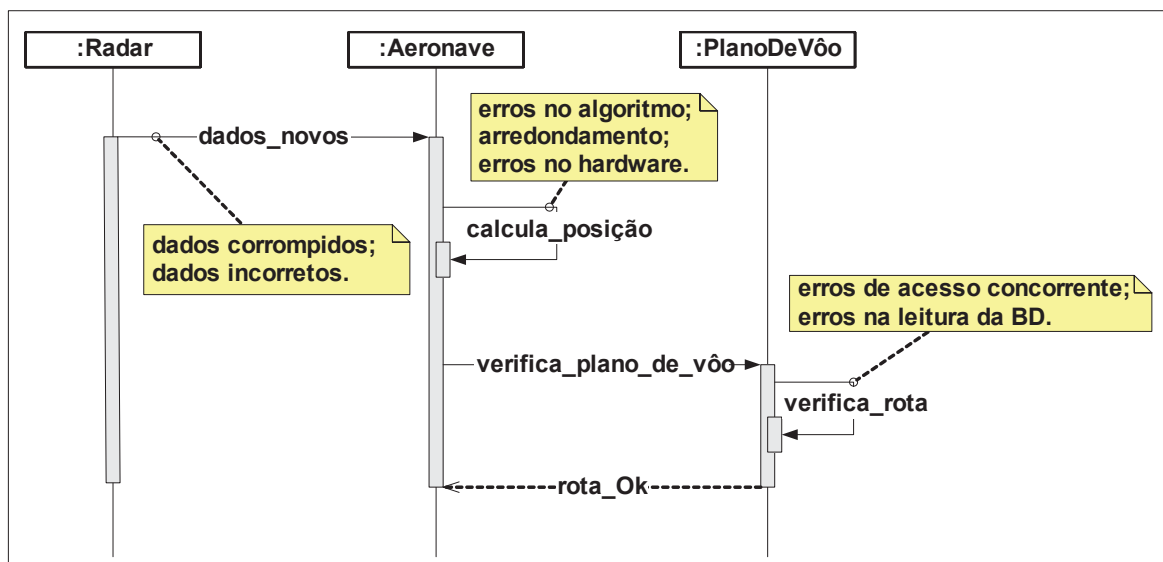


Figura 6 - Identificação de riscos sobre um cenário

Os riscos identificados são:

- **Ocorrência de dados corrompidos:** perda de precisão da informação ocasionada por falhas no sistema de comunicação ou a condições externas,

como altas taxas de interferência. Como consequência, pode ocorrer perda ou adulteração de informação;

- **Ocorrência de dados incorretos:** adulteração de dados decorrente de falhas em sensores ou falhas de processamento. Podem produzir informações inválidas, estimulando decisões incorretas;
- **Cálculo impreciso:** falhas em operações de arredondamentos, em algoritmos e até mesmo sobre o *hardware*, cujo impacto é a perda de precisão.
- **Acesso concorrente:** perda de precisão gerada pela manipulação de dados inadequados, obtidos durante acessos concorrentes a um banco de dados.
- **Acesso ao Banco de Dados:** recuperação de dados incorretos a partir do banco de dados.

Após identificar os riscos mais evidentes, foi iniciada a busca por soluções. É importante notar que a granularidade da identificação de riscos é também uma questão de bom senso relacionada com o domínio da aplicação.

### 3.2. Proposição de Soluções

Dos riscos identificados, foram considerados na simulação a “Ocorrência de Dados Corrompidos” e o “Cálculo Impreciso”.

Para a “Ocorrência de Dados Corrompidos”, as soluções podem envolver tanto melhorias sobre o *hardware* de telecomunicação como o uso de *software* para detecção e correção de falhas. Nesta última categoria foi identificada, como possível solução, a inclusão de redundâncias na transmissão de mensagens, atuando em conjunto com códigos para detecção e para correção de erros.

No “Cálculo Impreciso”, uma possível solução é o uso de um sistema de votação, em que os resultados de diferentes implementações de algoritmos para a execução de uma dada operação são confrontados em tempo de execução. Com isso, o resultado da operação é definido por meio de uma votação, cujo vencedor é o valor com maior frequência.

## 4. SIMULAÇÃO DE RISCOS INTRÍNSECOS À ARQUITETURA DO SISTEMA

Considerando que toda arquitetura de sistemas envolve aspectos funcionais (relacionados ao processo de negócio) e não-funcionais (relacionados a soluções de infraestrutura, como *hardware*, algoritmos, ambiente operacional, comunicação, etc), e que os requisitos não-funcionais são mais difíceis de serem estabelecidos, dada sua natureza sistêmica, foi decidido utilizar como ferramenta de teste um simulador de arquitetura, o que permitiu trabalhar sobre os riscos sem ter, *a priori*, um conhecimento completo sobre todas as interações possíveis.

### 4.1. O Simulador

Como o objetivo deste simulador é permitir análises quantitativas sobre o impacto dos riscos levantados e a eficácia das soluções adotadas, sua análise e implementação visaram a máxima simplicidade, a fim de preservar a abstração da análise e evitar a inserção de falhas de simulação.

O primeiro passo para a construção do simulador foi a identificação dos componentes estáticos envolvidos nos cenários dinâmicos. Observando o cenário "Verifica Plano de Vôo" (Figura 4), as principais classes envolvidas foram: "Radar", "Aeronave" e "Plano de Vôo".

Para simular a ocorrência de falhas, foi desenvolvida uma classe para a introdução de erros no processo, chamada "Gerador de Erro". Essa classe é parametrizada, permitindo gerar erros de forma pseudo-aleatório, com base em uma distribuição de probabilidade. O diagrama de classes do simulador pode ser visto na Figura 7.

Para representar a qualidade da informação sobre a posição de uma aeronave foi utilizado o tipo de dado "booleano", com os estados lógicos representados pelos símbolos "V" (válido) e "I" (inválido). Essa estrutura afeta o simulador da seguinte forma:

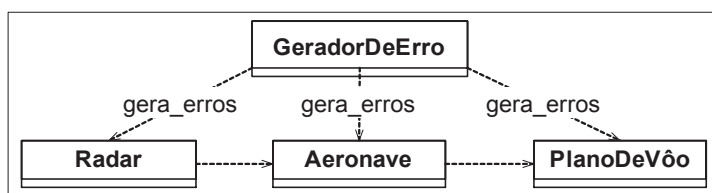


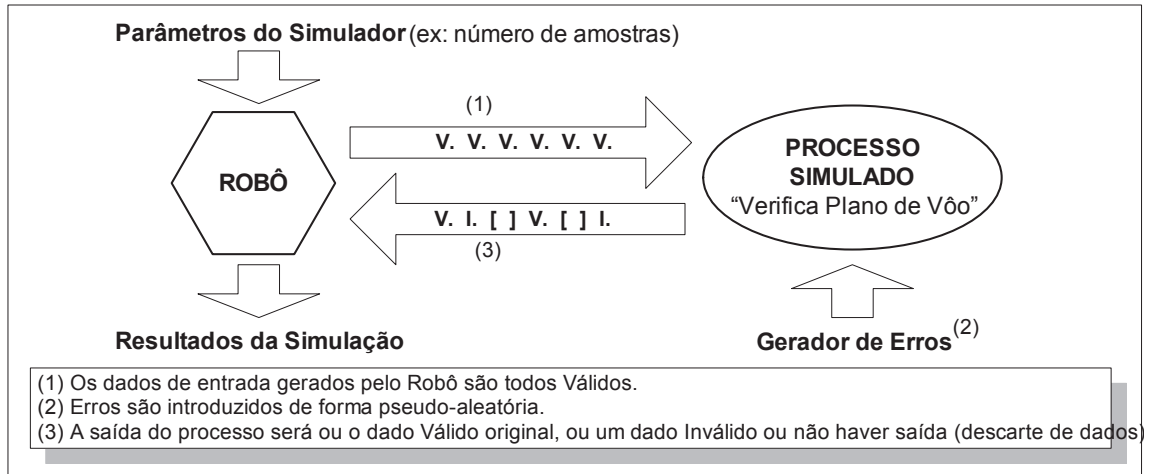
Figura 7 - Diagrama de classes do Simulador

- Se os dados de entrada são válidos (V) e o processo é executado sem falhas, a saída também é válida (V);
- Se os dados de entrada são inválidos (I), ou são válidos mas ocorre uma falha no processo, a saída é marcada como inválida (I);

Foi criada uma instância da classe "Gerador de Erro" para cada risco identificado. Essas instâncias são aplicadas aos métodos afetados pelos riscos, caracterizando a precisão intrínseca de cada método. A geração de erros é realizada segundo dois parâmetros: quantidade de erros e tamanho da amostra. A quantidade de erros indica o volume de falhas inseridas sobre a amostra (parâmetro "Erros por Amostra" na interface do simulador, apresentada na Figura 9). Esses erros são distribuídos de forma pseudo-aleatória sobre a amostra. Quando um erro é gerado, o valor do pacote que indica a qualidade do dado é alterado para inválido (I), independente de seu estado anterior.

Na classe "Radar", responsável por iniciar o processo de simulação, foram implementados métodos para simular a transmissão de dados e para registrar o resultado do processamento. A essa implementação foi dado o nome de "robô". O "robô" executa métodos da classe "Aeronave", passando como parâmetro de entrada somente dados válidos (V). Essa operação é realizada tantas vezes quantas determinadas pelo parâmetro "Amostras", sendo registrados tanto os dados de entrada como os de saída, o que possibilita a realização de uma análise posterior sobre a qualidade da arquitetura. Ao final da simulação é apresentado o total de "Acertos", "Erros" e "Descarte de dados", tanto de forma absoluta quanto em percentual.

Na Figura 8 pode ser visto um diagrama conceitual do simulador.



**Figura 8 - Diagrama conceitual do simulador**

Para efeitos de análise, foi assumido que um resultado inválido (ou erro) representa a confirmação de um risco que afeta o sistema, ou seja, comportamentos imprecisos ou fora das especificações de projeto. Foi considerado como resultado válido (acerto) a operação dentro das especificações de projeto. Uma terceira situação, a de descarte de dados, também foi considerada como acerto. O descarte ocorrerá ao ser detectado um resultado inválido em alguma parte do sistema e sua propagação interrompida, evitando a produção de saídas inválidas.

A partir dessas definições, foi implementado, em Java, um simulador específico para o cenário "Verifica Plano de Vôo". Sua entrada de dados consiste no número de amostras e na parametrização para geração de erros e aplicação de soluções. Sua saída consiste do total de erros, acertos e descartes. A interface com usuário pode ser vista na Figura 9.

dados\_novos (dados corrompidos) 20 / 100

dados\_novos (dados incorretos) 0 / 100

calcula\_posicao 0 / 100

Amostras 1000000

votação (calcula\_posicao) 3

detecção de erros (dados corrompidos)

repetição (dados corrompidos) 2

correção de erros (dados corrompidos)

filtro de dados (dados incorretos)

Acertos 992150 / 99,215 %

Erros 7850 / 0,7849999999999999 %

Descartes 0

Simular

**Figura 9 - Interface do simulador**

## 4.2. O Primeiro Cenário Simulado: Dados Corrompidos

Este cenário considera a perda de precisão no sistema, causada pela ocorrência de dados corrompidos durante a transmissão ou durante o processamento inicial. Quando é verificada tal ocorrência, o estado do booleano que representa a qualidade da informação é alterado para “inválido” (I), representando a falha gerada pela imprecisão nessa etapa.

Como solução, foi incluída na simulação um sistema de repetição, visando melhorar a precisão por meio de retransmissões das mensagens. Caso uma das mensagens do bloco transmitido seja válida (V), o sistema propaga essa mensagem para as demais etapas do processo. O dado será considerado inválido somente no caso de todas as mensagens do bloco de transmissões serem inválidas. Essa situação de simulação é representada pelo diagrama de seqüência da Figura 10.

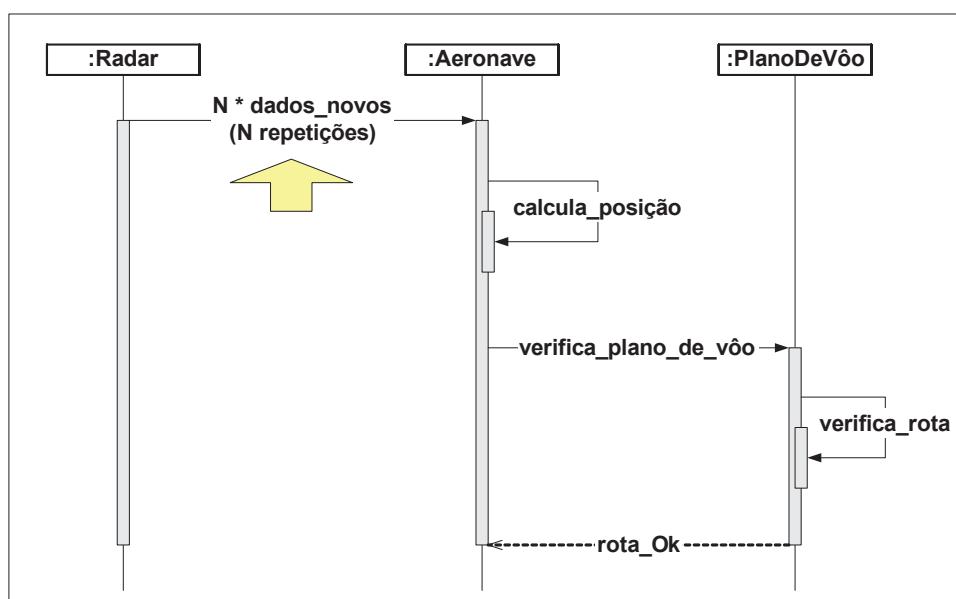


Figura 10 – Dados corrompidos e aplicação de um sistema de repetições

As simulações, cujos resultados são apresentados na Tabela 1, foram realizadas variando a taxa de erro, através do parâmetro "dados novos (dados corrompidos)", e variando a qualidade do tratamento de erros, através do parâmetro "repetição (dados corrompidos)". A representação gráfica do resultado dessa simulação pode ser vista na Figura 11.

Repetições	Percentual de erro sobre os dados (dados corrompidos na ENTRADA)									
	0	1	10	20	40	60	80	90	95	100
1	100,00	99,00	90,00	80,00	60,00	40,00	20,00	10,00	5,00	0,00
2	100,00	100,00	100,00	99,21	93,33	80,00	40,00	20,00	10,00	0,00
3	100,00	100,00	100,00	100,00	100,00	95,00	55,26	27,73	14,01	0,00
4	100,00	100,00	100,00	100,00	100,00	100,00	60,00	32,00	16,00	0,00
20	100,00	100,00	100,00	100,00	100,00	100,00	60,00	40,00	20,00	0,00
<b>Percentual de Dados Válidos após as Repetições (SAÍDA)</b>										

Tabela 1 - Teste de retransmissões para redução de dados corrompidos

As simulações mostram que, com o sistema a uma taxa de erro de até 40%, o processo de repetição de mensagens é capaz de melhorar a precisão de 60%, sem o uso de repetições, para 100%, ao utilizar três repetições. Esse cenário é ainda melhor quando adotadas quatro repetições, melhorando a precisão do sistema para 100% mesmo quando a taxa de erros é de 60%.

No entanto, há um limite para a eficiência desta solução (na marca de 60% de dados corrompidos), a partir do qual extrapolações sobre a quantidade de repetições não melhoram significativamente a precisão do sistema (como pode ser observado na curva de 20 repetições).

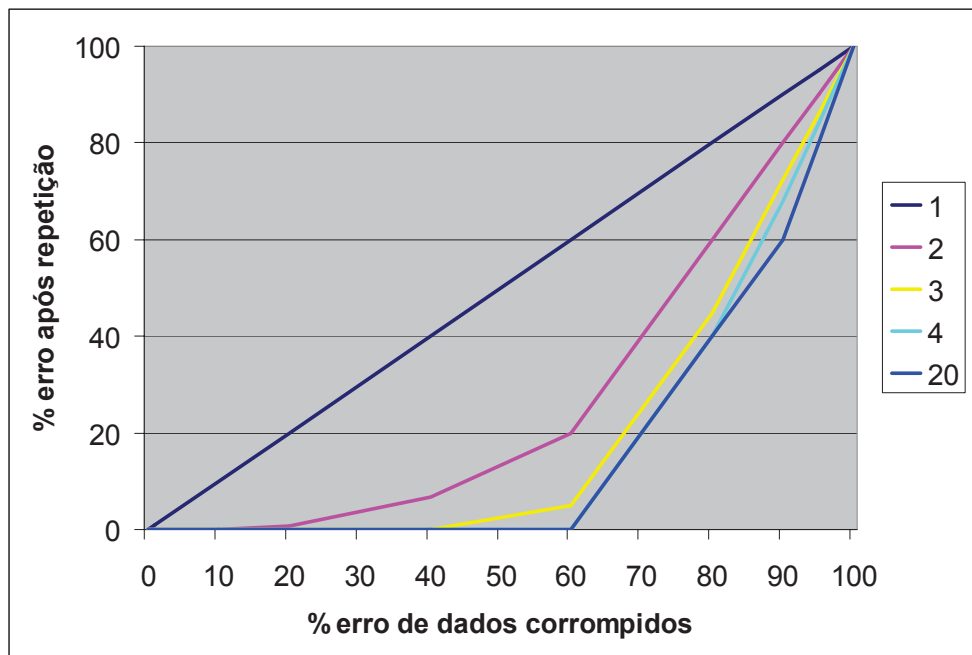


Figura 11 - Resultado da simulação com o sistema de repetições

### 4.3. O Segundo Cenário Simulado: Falhas em Cálculos

Este cenário considera a perda de precisão do sistema durante a execução de cálculos, o que pode estar relacionado à definição de algoritmos, sua codificação, arredondamentos sobre números fracionários e até mesmo a falhas de *hardware*. Todos esses fatores têm como consequência um resultado inválido (I) no cálculo de posição da aeronave. Como solução para tais falhas, foi implementado um sistema de votação, que escolhe o valor obtido com maior frequência dentre  $n$  resultados.

O sistema de votação foi simulado da seguinte forma: são pressupostas várias implementações do algoritmo para o cálculo de posição, sendo que cada uma é representada por uma instância da classe “Gerador de Erro”. Assumindo que todas as instâncias possuem a mesma taxa de erro, após a recepção de dados novos é disparado o cálculo de posição em cada instância. Ao término do último cálculo é realizada a votação, e o resultado será o valor obtido com maior frequência. Na ocorrência de 50% de resultados válidos e inválidos a saída do processo é considerada inválida. Essa situação operacional pode ser vista na Figura 12.

Foram realizadas simulações variando a taxa de erro por cálculo de posição e o número de implementações do cálculo (ver resultados na Tabela 2). A representação gráfica dos resultados obtidos pode ser observada na Figura 13.

Utilizando um sistema de votação composto por três implementações diferentes do cálculo de posição, foi possível melhorar a precisão do sistema de 90%, sem o sistema de votação, para cerca de 97%, a uma taxa de erro de 10% intrínseca a cada implementação do cálculo. Aumentando o número de votações para cinco, a precisão do sistema aumentou para cerca de 99%.

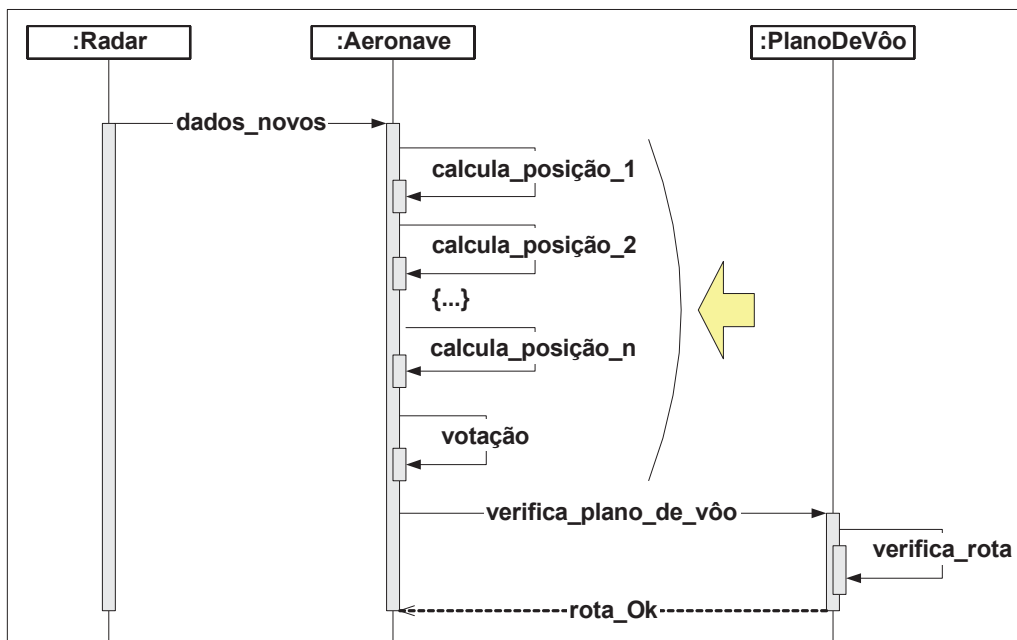


Figura 12 – Falhas de cálculo e aplicação de um sistema de votação

Também foi verificado que ao utilizar um número par de implementações do cálculo de posição, a precisão do sistema de votação diminuía. Isso é devido aos casos de empate serem tratados como resultados inválidos. Outra característica verificada foi a ineficiência do sistema de votação quando há elevadas taxas de erro, exigindo estudos sobre outras formas de solução para esses casos.

nº de implementações do cálculo de posição	Percentual de Erro no Cálculo de Posição (ENTRADA)									
	0	1	10	20	40	50	60	80	100	
1	100,00	99,00	90,00	80,00	60,00	50,00	40,00	20,00	0,00	
2	100,00	98,01	81,02	63,98	36,62	24,96	19,97	9,99	0,00	
3	100,00	99,97	<b>97,21</b>	89,61	64,44	50,00	40,00	20,00	0,00	
4	100,00	99,94	94,78	81,94	48,06	31,22	25,00	12,50	0,00	
5	100,00	100,00	<b>99,16</b>	94,23	67,37	50,00	40,00	20,00	0,00	
6	100,00	100,00	98,43	90,12	54,55	34,39	27,49	17,48	0,00	
21	100,00	100,00	100,00	99,91	76,68	50,00	40,00	20,00	0,00	
	Percentual de Dados Válidos após a Votação (SAÍDA)									

Tabela 2 - Teste de votações para correção do cálculo de posição

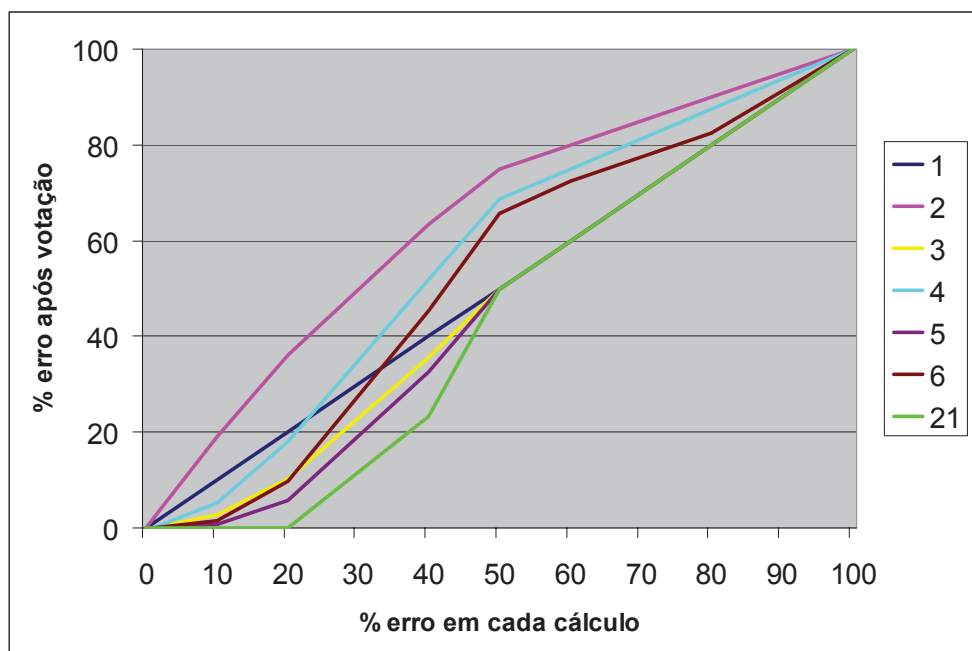


Figura 13 - Resultado da simulação do sistema de votação

#### 4.4. Discussão sobre os Resultados

Pela análise do primeiro cenário, conclui-se que a adoção de um sistema de repetição pode melhorar a precisão das comunicações quando as taxas de erro encontradas no ambiente forem de até 40%. Entretanto, como tal solução pode sobrecarregar os canais de comunicação, antes de estabelecer os requisitos finais para comunicação é necessário realizar um estudo complementar, relacionando número de repetições, tecnologia e viabilidade.

Analisando o segundo cenário, verificou-se que a etapa de cálculo de posição é crítica para a precisão do sistema, exigindo maior atenção na fase de desenvolvimento e na escolha da infra-estrutura, dada a baixa eficiência que tratamento de erros tiveram sobre esse cálculo. Um aspecto importante relacionado à etapa de cálculo de posição é que, além de ser dependente de requisitos (como tempo de processamento), ela própria pode ser considerada um requisito, do ponto de vista da aplicação (por exemplo, o sistema pode estar homologado para aplicações que aceitem taxas de probabilidade de erro de até 2%).

### 5. CONSIDERAÇÕES FINAIS

Na medida em que mudam as situações de trabalho, seja por aumento no risco de projeto, complexidade da área de aplicação, sigilo dos dados ou outro fator qualquer, novas técnicas de análise e projeto de sistemas tornam-se necessárias para proporcionar novas visões, tanto sobre o processo de negócio como sobre a própria atividade de análise e projeto em si, complementando as técnicas que já existiam.

O aumento do número de partes interagentes em sistemas aumenta a complexidade da análise, fazendo com que seja necessário considerar, além de aspectos funcionais, aspectos relacionados à arquitetura do sistema e ao ambiente operacional, a fim de garantir a qualidade do produto final.

Com isso, a principal contribuição deste trabalho consiste na apresentação de um método que permite explicitar riscos e simular seus efeitos sobre a arquitetura de um sistema, possibilitando estabelecer requisitos não-funcionais para as demais etapas de engenharia.

A simulação realizada foi baseada em classes representando tanto elementos do processo de negócio como do ambiente. Isso permitiu verificar tendências comportamentais do sistema, determinar a eficiência de cada ação de correção e identificar os pontos críticos do projeto, sobre os quais existem poucas condições de contorno na ocorrência de erro.

Além de ser uma representação expansível da arquitetura, o simulador também é uma prova de conceito, permitindo validar os conhecimentos adquiridos sobre o processo de negócio e detectar lacunas de conhecimento, pois sua construção exige que sejam respondidas perguntas como "quais erros podem ocorrer aqui?" ou "quais tratamentos de erro podem ser feitos aqui?".

Comparado à técnica de prototipação, o simulador pode ser visto como um protótipo comportamental voltado a demonstrar aspectos de qualidade definidos no modelo de qualidade de *software* escolhido para validar a aplicação.

Outro aspecto positivo do simulador é evidenciar o impacto que cada tipo de erro produz no sistema, proporcionando critérios objetivos para a construção de massas de dados adequadas para diversos casos de teste.

Considerando, então, que a partir da documentação inicial foi possível levantar riscos intrínsecos e transformá-los em requisitos de engenharia, estudos futuros poderão refinar a visão proporcionada pelo método exposto, na medida em que forem tratados temas como transição de sistema legado para novo, controle de qualidade da análise, independência entre equipes de desenvolvimento, natureza dos riscos, viabilidade, estruturas para representação da qualidade da informação, modelos de qualidade e relação entre diferentes categorias de qualidade de produto de *software*.

## REFERÊNCIAS

ARAKAKI, Reginaldo. Notas de Aula da Disciplina PCS5752 – Modelagem de Software Orientado a Objetos, Aula 3 – Identificação da Automação, Programa de Pós-Graduação da Engenharia Elétrica, Escola Politécnica, Universidade de São Paulo, 2006.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126-1: Engenharia de Software - Qualidade de Produto - Parte 1: Modelo de Qualidade. Rio de Janeiro, 2003.

COMANDO DA AERONÁUTICA. Manual Brasileiro de Inspeção em Vôo. Disponível em <<http://www.decea.gov.br/pame/publicacao/maninv/maninvbrasil.pdf>>. jan. 2002. Acesso em: 28 jan. 2007.

COMANDO DA AERONÁUTICA. Manual do Especialista em Informação Aeronáutica. Disponível em <[http://www.decea.gov.br/pame/publicacao/mca/mca\\_53-1.pdf](http://www.decea.gov.br/pame/publicacao/mca/mca_53-1.pdf)>. dez. 2000. Acesso em: 28 jan. 2007.

GOMES, Nelma da Silva. Qualidade de Software - Uma Necessidade. Disponível em <[http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade\\_de\\_Soft.pdf](http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf)>. maio 2000. Acesso em: 28 jan. 2007.

KAZMAN, Rick, KLEIN, Mark, BARBACCI, Mario, LONGSTAFF, Tom, LIPSON, Howard, CARRIERE, Jeromy. The Architecture Tradeoff Analysis Method, Fourth IEEE International Conference on Engineering Complex Computer Systems, 1998.

KOSCIANSKI, André, VILLAS-BOAS, André, RÊGO, Claudete M., “et al.”. Guia para Utilização das Normas sobre Avaliação de Qualidade de Produto de *Software* – ISO/IEC 9126 e ISO/IEC 14598. Associação Brasileira de Normas Técnicas – Sub-Comitê de *Software*, Paraná, Maio 1999.

MINISTÉRIO DA AERONÁUTICA. Estações Permissionárias de Telecomunicações e Tráfego Aéreo. Disponível em <[http://www.decea.gov.br/pame/publicacao/ima/ima\\_63-10.pdf](http://www.decea.gov.br/pame/publicacao/ima/ima_63-10.pdf)>. ago. 1999. Acesso em: 28 jan. 2007.

MINISTÉRIO DA AERONÁUTICA. Regras do Ar e Serviços de Tráfego Aéreo. Disponível em <[http://www.decea.gov.br/pame/publicacao/ima/ima\\_100-12.pdf](http://www.decea.gov.br/pame/publicacao/ima/ima_100-12.pdf)>. jun. 1999. Acesso em: 28 jan. 2007.