PROCESSO DE DESENVOLVIMENTO DE SOFTWARE PARA APLICAÇÕES EM TEMPO REAL

Antônio SANTOS (1); Vicente F. de Lucena LUCENA Jr. (2)
(1) UFAM – FT – PPGEE – Universidade Federal do Amazonas – Faculdade de Tecnologia (FT) – Programa de Pós Graduação em Engenharia Elétrica. Campus da Universidade Federal do Amazonas - Setor Norte - Faculdade de Tecnologia - Av. Gal. Rodrigo Octávio Jordão Ramos, nº 3000. CEP 69077-000 – Bairro Aleixo – Manaus – AM – Brasil. Tel. (92) 3647 4495, Fax: (92) 3647 4494; e-mail: ajunior.analista@ig.com.br,
(2) UFAM, CETELI e CEFET-AM – Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação; Centro Federal de Educação Tecnológica do Amazonas, e-mail: vicente@ufam.edu.br

RESUMO

Aplicações em tempo real são aquelas que exigem que uma determinada informação seja recebida e processada obedecendo a requisitos pré-estabelecidos, em sua maioria relativos a tempo, o que envolve tanto o desenvolvimento de *hardware* como de *software*. Desenvolver este tipo de aplicação perfaz tarefa mais complexa do que desenvolver sistemas de informação convencionais, mas há um número razoável de técnicas e ferramentas que viabilizam sua implementação. O objetivo deste trabalho é apresentar um processo que use ferramentas e técnicas de desenvolvimento de *software* para aplicações em tempo real. Serão apresentadas em detalhes as metodologias utilizadas e os resultados alcançados, durante o desenvolvimento de um *software* de tempo real, cujo objetivo é fazer com que um robô Lego percorra um labirinto. A metodologia para o desenvolvimento deste trabalho consiste no estudo das ferramentas e técnicas existentes, entre elas a RT-UML e processos de desenvolvimento para aplicações de tempo real e sistemas embarcados, desenvolvimento do *software* citado utilizando as técnicas estudadas, apresentação do processo utilizado durante este trabalho e de todo o projeto de desenvolvimento de *software*. A principal contribuição é a apresentação do processo, o qual poderá ser utilizado por outros desenvolvedores de *software* que queiram desenvolver aplicações em tempo real.

Palavras-chave: Tempo real, RT-UML software embarcado, robô Lego.

1. INTRODUÇÃO

Em diversas áreas onde é possível automatizar processos industriais deve-se considerar o uso de sistemas de tempo real, desde que haja restrições de tempo, sejam essas restrições de duração de processamento, de tempo exato de execução ou tempo exato de término de processamento. O desenvolvimento de software para dispositivos que tenham restrições de tempo real não é algo trivial, pois exige análise de concorrência, de escalonamento, de tempo de entrada, tempo de processamento e tempo de saída dos dados necessários à aplicação (IAS, 2005a). Em geral, os sistemas em tempo real são embarcados em dispositivos para aplicações específicas (IAS, 2005a), que podem ser um caixa eletrônico que tenha restrições de tempo de resposta para solicitações, por exemplo, de saldo, ou um medidor de temperatura que faz medições a cada 1 minuto.

Para desenvolvimento de sistemas embarcados e em tempo real existem várias metodologias apresentadas na literatura, entre elas:

- Accord/UML, que possui um ciclo de processo de três estágios que garantem análise, prototipação e testes, onde o *software* é refinado até que atinja os objetivos propostos (TESSIER *et al*., 2003);
- ROLES, que possui um ciclo de processo evolucionário (espiral) com as fases de planejamento, análise, projeto, tradução (DOUGLASS, 1999a);
- *Real Time Development Process*, onde, para desenvolver um sistema de tempo real, deve-se identificar a resposta a um estímulo a ser processado, identificar as restrições de tempo para as respostas aos estímulos, agregar o processamento de estímulo e resposta em processos concorrentes (que serão considerados classes de estímulo e resposta), projetar algoritmos para processar cada classe de estímulo e resposta que atende a requisitos de tempo, projetar um sistema de escalonamento que irá assegurar que os processos são iniciados no tempo exato, atendendo seus tempos limites (IAS, 2005b).
- Para a modelagem de sistemas embarcados e em tempo real o OMG – *Object Management Group* – criou a RT-UML – *Real-Time Unified Modeling Language*, que contém, além das especificações da UML, mecanismos para representar sistemas em tempo real (DOUGLASS, 1999b).

O uso de aplicações em tempo real foi motivado em razão da primeira tentativa de desenvolver um robô Lego para percorrer um labirinto ter resultado em fracasso, ou seja, o robô não percorria o labirinto tendo um comportamento de girar sobre o próprio eixo sem sair do lugar ou sair da área do labirinto. Fazendo-se necessário realizar experimentos para identificar as causas. Durante esses experimentos, identificou-se que seria necessário considerar restrições de tempo para realizar a leitura correta dos valores de cores fornecidos pelo sensor de luz do robô. Sendo assim, fez-se uma pesquisa na literatura para identificar métodos de desenvolvimento de *software* em tempo real, os quais pudessem ajudar a encontrar uma forma adequada de fazer o robô percorrer o labirinto até encontrar o fim.

O projeto do robô Lego *mindstorms* é descrito na seção 2, onde são apresentadas todas as etapas do desenvolvimento do *software*, bem como problemas encontrados e soluções tomadas. No capítulo 3 é apresentado o processo empírico de desenvolvimento de *software* baseado nas atividades que contribuíram de forma significativa para que o projeto alcançasse seus objetivos, bem como em técnicas descritas no processo *Real Time Development Process.* Na seção 4 são descritas as conclusões e recomendações.

2. PROJETO ROBÔ LEGO MINDSTORMS

Nesta seção será apresentado o projeto feito com o robô Lego *mindstorms*, que utilizou conceitos de desenvolvimento para sistemas embarcados e aplicações em tempo real, bem como as dificuldades encontradas durante o projeto, o processo proposto e o executado.

2.1. Objetivo do projeto

Desenvolver um robô, baseado no produto Lego *mindstorms* RCX (Lego, 2008), que consiga trafegar e sair de um labirinto, que contém linhas pretas representando os caminhos do labirinto (100% de escala de cinza), quadrados em escala de cinza indicando: intersecções (30% de escala de cinza), caminho sem fim (60% de escala de cinza) e fim do labirinto (80% de escala de cinza), de acordo com a Figura 1. O robô deverá seguir a linha preta, tomando a decisão de dobrar ou não quando encontrar uma intersecção, voltar quando

encontrar um fim de caminho e comemorar quando encontrar o fim do labirinto. Para isso é necessário montar o robô usando o kit Lego *mindstorms RCX*, usando sensores de luz, motores de passo, rodas, entre outras peças. Também é necessário embarcar no robô um *software* que implementará um algoritmo que o fará percorrer o labirinto no menor tempo possível. Deve-se ressaltar que a equipe que desenvolveu o projeto não tinha experiência em desenvolvimento de *software* para sistemas embarcados, mas sim para desenvolvimento de *software* para PC, e também não conhecia o funcionamento do robô. O labirinto da Figura 1 foi impresso em uma folha A0.

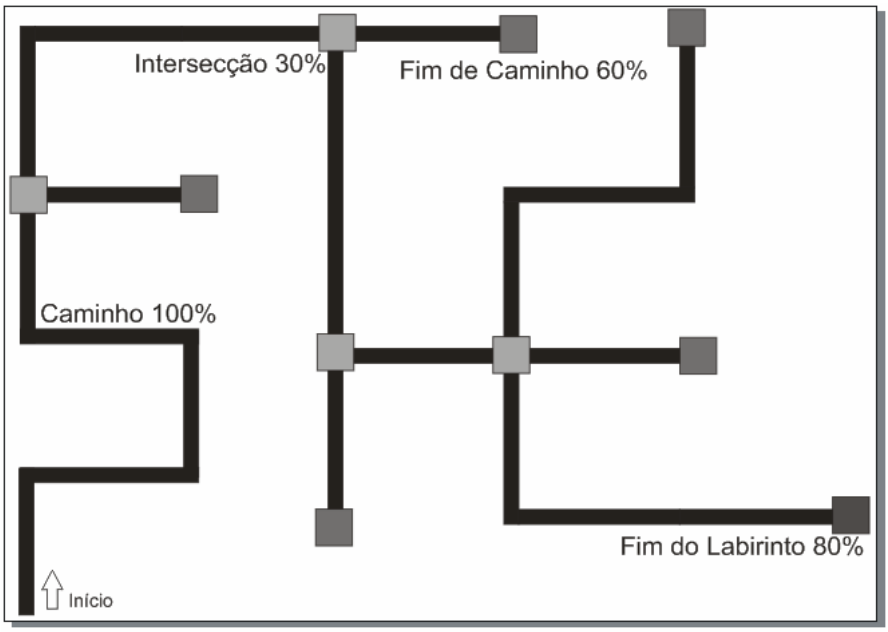

Figura 1 – Labirinto

2.2. O robô Lego mindstorms

Para construir o robô, foram usados componentes mecânicos e elétricos, bem como ferramentas de *software* que auxiliaram na implementação do algoritmo para fazer com que o robô percorra o labirinto. Estas ferramentas e componentes serão descritos nesta seção.

2.2.1. Componentes mecânicos e elétricos do robô

O robô foi montado, como mostrado na figura 2, com dois sensores de luz, cuja função é detectar os elementos do labirinto apresentado na Figura 1 através da leitura das cores; dois motores de passo, cuja função é movimentar as rodas direita e esquerda do robô; uma unidade de processamento, cuja função é processar os valores recebidos dos sensores de luz e comandar os motores de passo de acordo com o algoritmo contido no *software* embarcado em sua unidade de processamento; rodas, eixos e blocos, que formam a estrutura do robô.

(A) **(B)**

Figura 2 – robô montando com: (A) sensores unidos; (B) sensores separados

Na Figura 2-A, é apresentado o robô montado com uma configuração onde os sensores estão unidos, com esta forma de montagem, o robô segue o algoritmo descrito pelo diagrama de atividades mostrado na Figura 4 da sessão 2.3.3. Na figura 2-B os sensores estão separados a uma distância de 5cm e o robô segue o algoritmo descrito pelo diagrama de atividades mostrado na Figura 5 da seção 2.5.

2.2.2. Ferramentas utilizadas para desenvolver o *software*

O software foi desenvolvido usando a linguagem de programação C++, em conjunto com as bibliotecas do sistema operacional Brickos (2008), o ambiente de compilação do código fonte e envio do código executável ao robô foi feito utilizando-se o simulador de ambiente Linux denominado Cygwin (2008). A compilação e envio do código executável ao robô foi feita de acordo com a documentação fornecida pelo grupo de desenvolvimento do sistema operacional Brickos (2008).

2.3. Planejamento

Durante o planejamento do projeto, as fases, os produtos de trabalho, os recursos e a análise e projeto do robô foram definidos e documentados, as subseções a seguir detalham como o planejamento foi realizado.

2.3.1. Fases do projeto

As fases planejadas para o projeto foram: concepção, detalhamento, implementação, testes de sistemas/manutenção e fechamento do projeto. As mesmas são baseadas do modelo clássico de desenvolvimento de *software*, o modelo cascata (PRESSMAN, 2002). Cada uma dessas fases será descrita a seguir.

- Concepção: visa estudar o problema, o funcionamento do robô e a plataforma (*software*); planejar o projeto, estimar custos e definir os requisitos.
- Detalhamento: visa detalhar os requisitos, planejar os testes, criar plano de gerência de configuração e revisar os planos.
- Implementação: visa escrever, testar e debugar o código, especificar os testes de sistema e revisar os planos.
- Testes de sistemas/manutenção: visa executar os testes, corrigir o código e revisar os planos.
- Fechamento do projeto: visa revisar a documentação e apresentar o produto funcionando.

2.3.2. Descrição dos produtos de trabalho

Os seguintes produtos de trabalhos (artefatos) foram usados durante o projeto.

- Plano de Projeto: Documento contendo os objetivos do projeto, hierarquia de atividades, cronograma, papéis e responsabilidades, produtos de trabalho, responsabilidades e pessoas.
- Especificação de Requisitos: Todos os requisitos são descritos e detalhados neste documento..
- Análise e projeto: Contém a arquitetura do sistema, componentes e interfaces e diagramas UML.
- Especificação de testes: define os testes de hardware, mecânicos e de software serem realizados.
- Código: Código fonte e executável que controla o robô.
- Robô montado: Robô montado com rodas e sensores de luz, contendo o software embarcado.

2.3.3. Análise e projeto do robô Lego

Os principais itens de análise e projeto que definem o funcionamento do robô são os casos de uso, que define as funcionalidades do *software*, e o diagrama de atividades, que define o algoritmo usado pelo robô para percorrer o labirinto, bem como a especificação de testes, que define os testes a serem feitos para identificar se o robô estava atendendo ao objetivo e aos requisitos. Os testes, para o caso do robô consistiam em identificar se o robô estava seguindo o comportamento do algoritmo. Informações sobre diagramas de casos de uso e de atividades, descritos a seguir, podem ser vistos em Booch (2000), e informações sobre especificação e execução de testes podem ser vistas em Lewis (2000).

- Diagrama de caso de uso do robô Lego: a Figura 3 apresenta as funcionalidades do *software*, com destaque ao caso de uso percorrer labirinto, que define de que forma o robô irá percorrer o labirinto e é descrito pelo diagrama de atividades da Figura 4 . O caso de uso inicializar robô define apenas que o usuário deverá apertar o botão "*Run*", do equipamento. O papel do usuário é posicionar o robô no início do labirinto e ligá-lo. A premissa é que o *software* já esteja embarcado no robô, juntamente com o sistema operacional Brickos (2008).

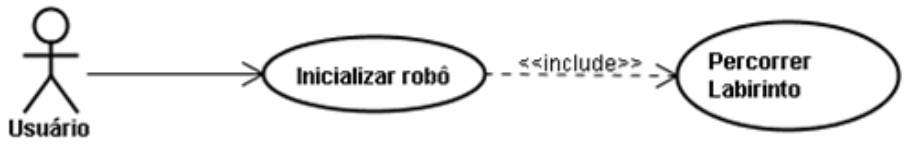

Figura 3 – Diagrama de Caso de Uso

- Diagrama de atividades do caso de uso percorrer labirinto: A Figura 4 apresenta o diagrama de atividades que mostra o algoritmo que guia o robô para percorrer o labirinto. Ele tem como premissa a montagem do robô usando dois sensores de luz posicionados à frente do robô e conectados, a Figura 2-A mostra essa configuração. A variável corEsquerda representa a cor lida pelo sensor posicionado na parte esquerda frontal do robô e a variável corDireita representa a do sensor direito. As cláusulas corCAMINHO, corBRANCA, corCRUZAMENTO, corPAREDE e corFIM, representam as cores dos elementos do labirinto em escala de cinza, que são, respectivamente, a parte branca do labirinto (0% de escala de cinza), intersecções (30% de escala de cinza), caminho sem fim (60% de escala de cinza) e fim do labirinto (80% de escala de cinza), representados na Figura 1. As ações a serem tomadas pelo robô, de acordo com as cores lidas pelos sensores são descritas nas elipses contidas na Figura 4.

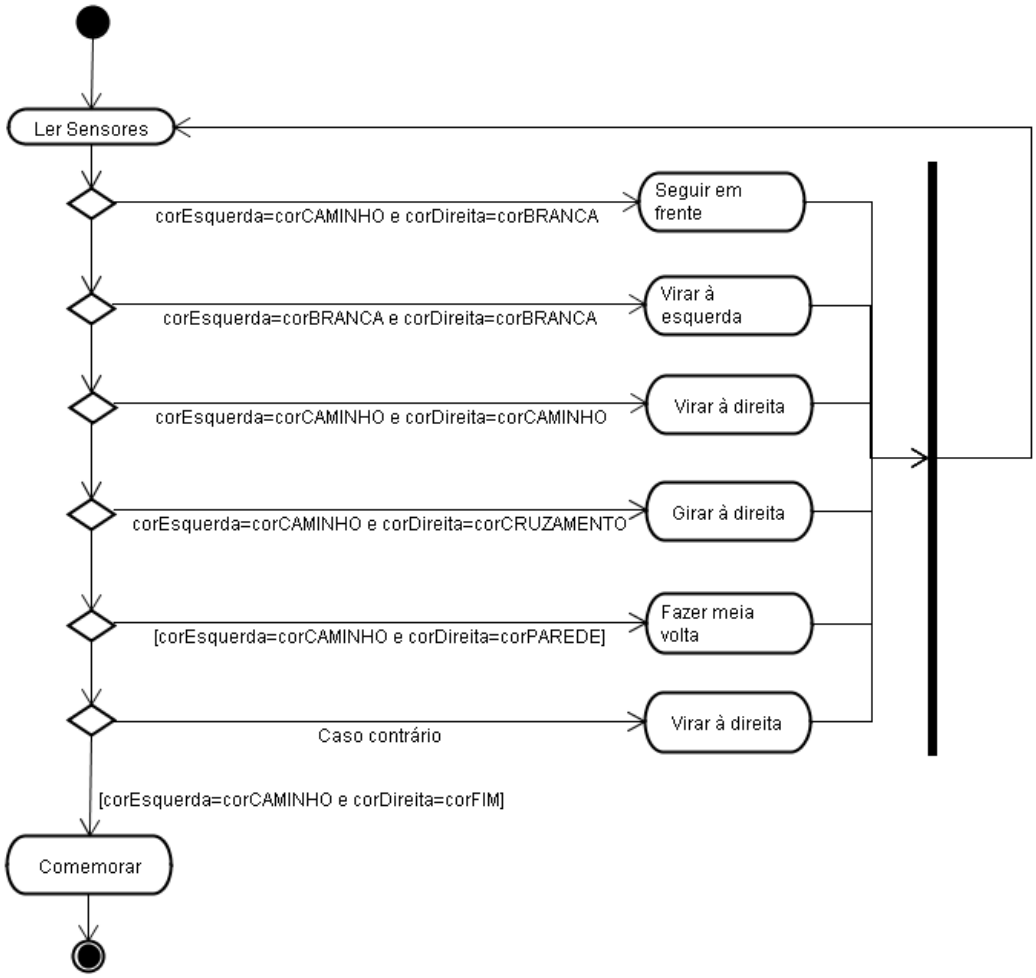

Figura 4 – Diagrama de atividades percorrer labirinto

2.4. Implementação do projeto

As fases de concepção e de detalhamento foram realizadas conforme o planejado. Quanto à fase de implementação, esta demorou mais tempo que o previsto, e deveria ter sido feita de acordo com o documento de análise e projeto, descrito na seção 2.3.2, mas o robô não estava percorrendo o labirinto usando essas premissas, mais especificamente, estava saindo da pista que deveria percorrer, não passando nos testes. Assim, o modelo clássico de desenvolvimento de *software* foi abandonado, pois, por este modelo, seria necessário voltar à fase de análise e projeto para pensar em outras alternativas, o que era inviável, já que a equipe deveria entender melhor o funcionamento do robô para fazê-lo funcionar. Um método empírico foi adotado, que consistia em, repetidas vezes, alterar o *software* e testar até atingir o resultado esperado. Usando-se essa abordagem, percebeu-se logo que o problema estava no tratamento dos valores fornecidos pelos sensores, pois os motores de passo funcionavam bem, mas as ações tomadas pelo robô após a leitura das cores pelo leitor não estavam corretas. Uma das primeiras suposições quanto ao problema seria que os sensores estavam fornecendo valores errados, mas após vários testes descartou-se essa suposição e passou-se a considerar que o algoritmo utilizado para fazer o robô percorrer o labirinto deveria ser modificado.

Uma pesquisa em outros projetos de aplicação do robô Lego, entre eles Putten (2006) e Knudsen (1999), mostrou uma forma de montagem onde os sensores são posicionados na parte frontal a uma distância de 5 cm um do outro, o que pode ser visto na Figura 2-A. Nessa abordagem, os sensores são lidos um por vez e o robô percorria a linha preta em zig-zag, ou seja, o robô vira à direita até que o sensor da esquerda encontre a linha preta e, em seguida, vira à esquerda até que o sensor da direita encontre a linha preta, seguindo em frente. Essa nova forma de montagem foi implementada e o *software* foi modificado para funcionar de acordo. Desta forma, o robô passou a percorrer o labirinto, mas sem realizar o comportamento correto ao encontrar as intersecções, fim de caminho e fim de labirinto. Esse foi um grande avanço, mas não atingia os objetivos. Além disso, em determinadas ocasiões, ao ser lida a cor preta pelos sensores, o robô realizava comportamentos correspondentes a fim de caminho ou fim de labirinto, ou seja, dando meia volta ou comemorando, respectivamente. Para avaliar a razão destes problemas, foram planejados e realizados diversos experimentos para identificar o valor lido pelos sensores em várias posições do labirinto, ou seja, nas intersecções, nos fins de caminho e fim do labirinto, de acordo com a Tabela 1.

Tabela 1 – Experimentos realizados para identificar os valores lidos pelos sensores

#	Experimento realizado	Resultado(s) obtido(s)
1	Obter o valor de cor dos sensores em cada intersecção	Sensor da Esquerda: 2D,2E,2F; Sensor da Direita: 2A, 2B, 2C
2	Obter o valor de cor dos sensores em cada fim de caminho	Sensor da Esquerda: 29,2A,2C; Sensor da Direita: 26,27,28
3	Obter o valor de cor dos sensores na linha preta (vários trechos)	Sensor da Esquerda: 21,22,23; Sensor da Direita: 1D,1E,1F
4	Obter o valor de cor dos sensores nos trechos brancos	Sensor da Esquerda: 30,31,32; Sensor da Direita: 2D,2E,2F
5	Obter o valor de cor dos sensores no fim do labirinto	Sensor da Esquerda: 27,28,29; Sensor da Direita: 24,25,26
6	Obter o valor de cor dos sensores movimentando de um trecho de cor branca para um trecho de cor preta	Sensor da Esquerda : 21,22,23,24,25,26,27,28,29,2A,2B,2C,2D, 2E,2F,30,31,32; Sensor da Direita: 1D,1E,1F,20,21,22,23,24,25, 26,27,28,29,2A,2B,2C,2E,2F
7	Movimentar o robô para frente	Ambas as rodas giram para frente
8	Movimentar o robô para a direita	Roda da direita gira para trás e roda da esquerda gira para frente
9	Movimentar o robô para a direita sobre o próprio eixo	Roda da esquerda gira para frente e roda da direita se mantém parada
10	Movimentar o robô para a esquerda	Roda da direita gira para frente e roda da esquerda gira para trásq
11	Movimentar o robô para a esquerda sobre o próprio eixo	Roda da direita gira para frente e roda da esquerda se mantém parada
12	Movimentar o robô para trás	Ambas as rodas giram para trás.
13	Obter o número adequado de iterações e o tempo mínimo entre leituras de cores	Número de iterações = 80 e tempo mínimo entre leituras = 10 milissegundos

Os valores fornecidos pelo sensor são representados por números hexadecimais variando de 1D a 2D. Depois de realizados os experimentos, observou-se que os valores obtidos durante o experimento #6 correspondiam a todos os outros obtidos nos testes de 1 a 5, o que explicava o comportamento do robô em fazer meia volta ou comemorar enquanto apenas passava sob a linha preta. A solução adotada foi modificar o software para fazer várias leituras num curto intervalo de tempo de forma a garantir a leitura correta pelo sensor de um determinado valor válido de cor. Para encontrar os valores adequados, usou-se a definição e as técnicas escritas em IAS (2005a) para o cálculo do valor de tempo real, ou seja, o número de iterações deveria ser o mínimo possível de forma a ter certeza quanto ao valor da cor sendo lida e o maior intervalo de tempo possível que permita o robô se movimentar. Após várias repetições do experimento #13, observou-se que, para atender às restrições, 80 leituras com um intervalo de tempo de 10 milissegundos entre elas são números adequados. O *software* foi modificado para atender a essas restrições, fazendo com que o robô percorresse o labirinto até encontrar o fim, comemorando ao final, passando nos testes. A restrição de tempo caracterizou a aplicação como sendo levemente de tempo real, de acordo com as características de sistemas em tempo real definidas em IAS (2005a). A Tabela 2 compara as características à implementação do robô.

Tabela 2 – Caracterização da aplicação como sendo sistema em tempo real

Características de sistemas em tempo real	Características da aplicação do robô Lego que atendem
Sistema dirigido a eventos	O robô reage de acordo com as cores lidas pelos sensores
Estruturas de dados simples	Há apenas dois casos de uso e 3 classes implementados
Pouca quantidade de dados de entrada	Os dados de entrada correspondem apenas aos valores de cores lidos definidos na Tabela 1.
Dependente do *hardware*	O algoritmo para percorrer o labirinto depende da forma como o robô é montado.

2.5. Fechamento do projeto

Durante a fase de fechamento do projeto, o documento de análise e projeto foi modificado de forma a usar os conceitos de RT-UML (DOUGLASS, 1999b), em especial o diagrama de atividades da Figura 5.

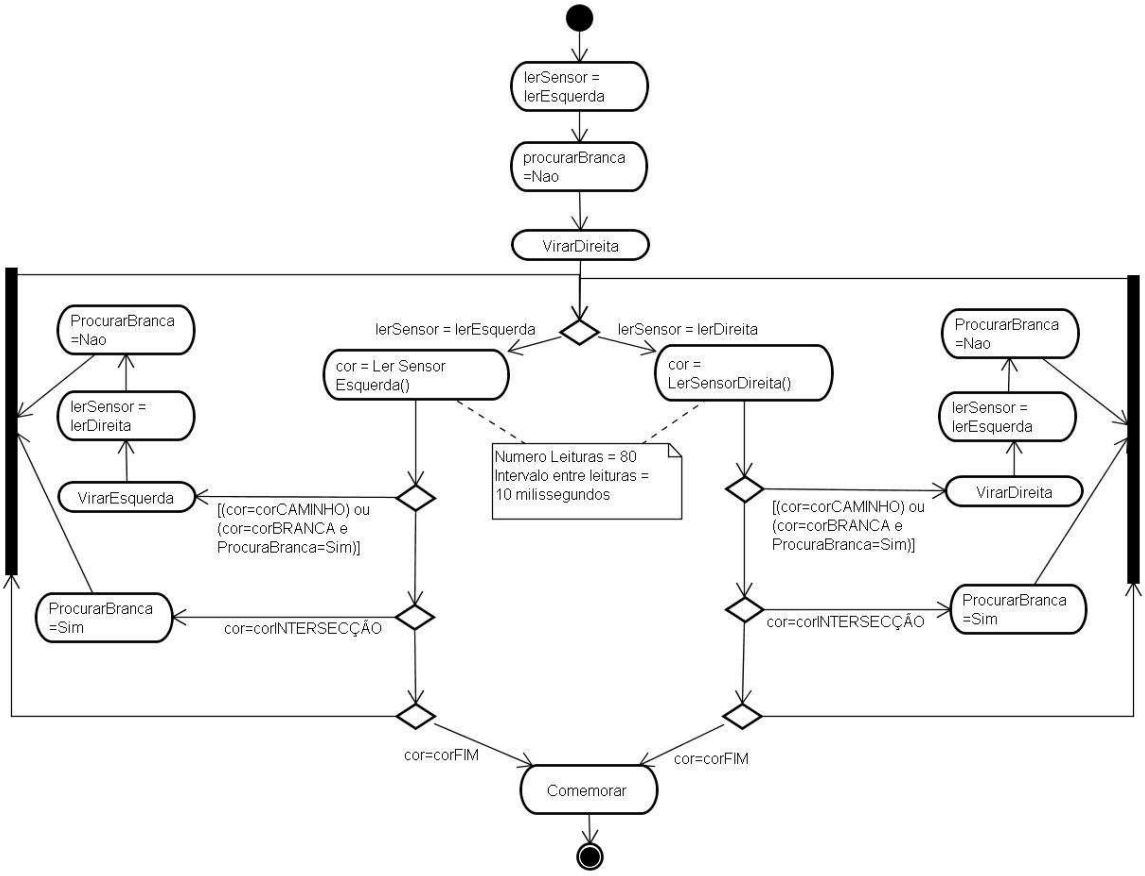

Figura 5 – Diagrama de atividades percorrer labirinto, com restrições de tempo

Na Figura 5, observa-se que durante a leitura das cores, é apresentada uma restrição de tempo real informando que a leitura será feita 80 vezes com um intervalo de tempo de 10 milissegundos entre elas. Essa restrição é representada através de uma nota, de acordo com a RT-UML (DOUGLASS, 1999b). Além da referida restrição, o algoritmo utilizado não é mais o mesmo descrito no diagrama de atividades da Figura 3, pois este novo algoritmo leva em conta o novo posicionamento dos sensores no robô, bem como o fato de que este passará a percorrer o labirinto em zig-zag. O documento de especificação de testes também foi modificado, de forma a atender a esse novo algoritmo, mais especificamente foi alterado o item referente à verificação correspondente à forma com que o robô deveria seguir a linha preta, pois antes ele deveria realizar essa operação seguindo em frente, e agora segue a linha em zig-zag.

3. PROCESSO SUGERIDO COM BASE NAS ATIVIDADES DESENVOLVIDAS

No projeto do robô Lego, os produtos de trabalho gerados pelas atividades da fase de concepção foram todos úteis, não havendo necessidade de modificá-los, mas os da fase de detalhamento dos requisitos precisaram ser alterados, em especial o algoritmo para percorrer o labirinto, que não foi bem elaborado devido, principalmente, ao pouco conhecimento do funcionamento das partes do robô, em especial o sensor de luz.

O primeiro fator para que o projeto voltasse a percorrer o caminho certo para atingir os objetivos foi o reconhecimento de que o algoritmo descrito pelo diagrama de atividades da Figura 3 não era adequado. E para desenvolver um algoritmo adequado foi necessário realizar os experimentos descritos na Tabela 1 de forma a entender o funcionamento dos sensores. Considerando as atividades desenvolvidas que levaram ao atendimento dos objetivos do projeto, sugere-se o processo descrito na Figura 6 para desenvolvimento de aplicações em tempo real. A seguir, cada uma das atividades contidas na Figura 6 serão descritas, com destaque para planejar o experimento, realizar o experimento e analisar os resultados do experimento, que serão vistas e maiores detalhes.

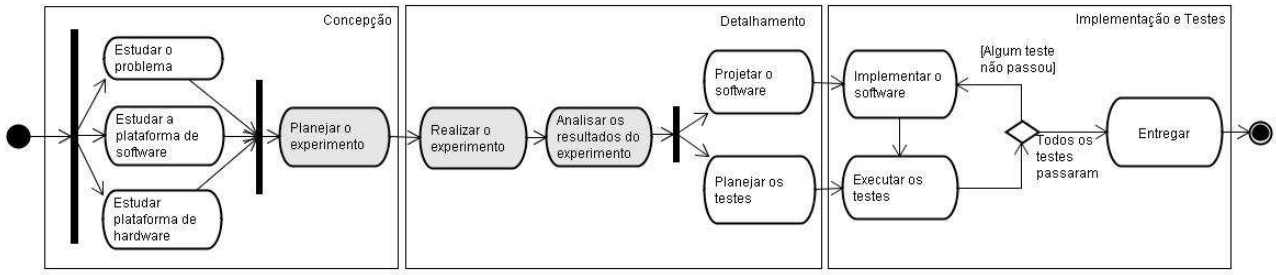

Figura 6 – Processo extraído do projeto para desenvolvimento de uma aplicação de tempo real

- Estudar o problema: consiste em entender os requisitos fornecidos pelo cliente e decompô-los em outros menos complexos. O resultado do estudo do problema é o documento de especificação dos requisitos e parte do documento de análise e projeto.
- Estudar a plataforma de *software*: consiste em identificar, obter e conhecer sistema operacional, *drivers*, linguagem de programação e bibliotecas necessárias ao desenvolvimento da aplicação. O resultado do estudo da plataforma de software é descrito tanto no documento de requisitos quanto no de análise e projeto.
- Estudar a plataforma de *hardware*: consiste em identificar, obter e conhecer os componentes físicos de produto necessários ao desenvolvimento da aplicação. O resultado desta atividade é descrito no documento de especificação dos requisitos.
- Planejar o experimento: com base no resultado dos estudos do problema, da plataforma de *hardware* e de *software*, planejar experimentos que permitirão entender o funcionamento das partes do dispositivo e também identificar restrições de uso destes dispositivos, bem como do ambiente em que eles são inseridos. Para sistemas em tempo real, esses experimentos são realizados para identificar se os dispositivos poderão ou não atender às restrições de tempo já determinadas. No caso do robô, experimentos foram planejados para identificar os valores retornados pelo sensor de luz, o funcionamento dos motores de passo, bem como o número adequado de iterações e o tempo mínimo de leitura de cores. A especificação dos experimentos está descrita na Tabela 1.

- Realizar o experimento: de acordo com o plano do experimento, as ações descritas são feitas e os resultados anotados para análise. Os valores obtidos pelos sensores e o comportamento dos motores de passo foram descritos na Tabela 1. Observa-se que nesta atividade pode haver implementação.
- Analisar os resultados do experimento: avaliar se os resultados obtidos são condizentes com a especificação; identificar se os resultados são consistentes ou não; identificar possíveis variações nos resultados obtidos mesmo que sejam valores válidos; definir um esboço do algoritmo que será utilizado para atingir os objetivos do projeto. Quanto à análise do experimento do projeto do robô, foi observada diferença nos valores retornados pelos dois sensores. Também foi percebida uma possível inconsistência nos valores retornados pelos sensores durante a mudança da cor sendo lida (quando o leitor fazia uma transição entre as cores preta e branca), o que durante os experimentos resultou na leitura de todas as faixas de valor das cores lidas, fazendo com que o robô tomasse decisões erradas. O resultado desta atividade foi descrito no documento de análise e projeto.

A definição das atividades projetar o *software*, planejar os testes, implementar o *software*, executar os testes e entregar podem ser vistas em (PRESSMAN, 2002). As atividades do processo apresentado relativas à experimentação são em parte similares às definidas do processo *Real-Time Development Process* (IAS, 2001b), tendo como principal diferença o não uso do escalonamento de processo de estímulo e resposta, em razão de não haver tratamento de processos concorrentes, o que não foi necessário.

4. CONCLUSÃO E RECOMENDAÇÕES

No presente artigo foi apresentado um processo para desenvolvimento de aplicações em tempo real, extraído a partir das atividades que funcionaram durante o desenvolvimento de um robô, cujo objetivo é percorrer um labirinto. O robô foi implementado em dois momentos. No primeiro, utilizou-se um processo baseado no modelo clássico que, durante a fase de detalhamento gerou um algoritmo que não funcionou, principalmente porque foi desenvolvido sem que se conhecesse o funcionamento do robô. Já no segundo momento, foi planejada e realizada uma experimentação para identificar o funcionamento dos sensores e dos motores de passo, e, após a análise desses dados, foi desenvolvido um algoritmo mais adequado. O uso de técnicas de desenvolvimento de *software* para sistemas de tempo real funcionou a contento tanto para identificar a restrição de tempo real – descoberta na experimentação – quanto para adaptar o algoritmo de forma a atender às restrições. A RT-UML também foi utilizada para representar as restrições de tempo e de número de iterações lidas pelo sensor, apresentada no diagrama de atividades representado na figura 5. O robô percorreu o labirinto sem se perder e encontrou o fim em todas as posições em que fora posto no labirinto, comemorando ao encontrar o final e passando nos testes especificados.

O processo apresentado manteve todas as atividades da fase de concepção, acrescentado à fase de detalhamento as atividades de experimentação, cujo objetivo é fornecer à equipe de desenvolvimento informações sobre o funcionamento de sensores e atuadores usados no dispositivo, bem como sobre as restrições de tempo real. Essas informações são fundamentais para as atividades de projeto de *software*, onde os algoritmos que controlam o dispositivo são definidos, e de teste de *software*, onde os testes são especificados. As restrições de tempo também podem ser verificadas e validadas durante os testes. No caso do robô, as atividades de experimentação contribuíram para, além de fornecer informações sobre o funcionamento dos sensores de luz e do motor de passo, identificar os valores adequados para o número de iterações e para o tempo entre leituras, usados para garantir tanto que as cores lidas pelo sensor fossem as corretas quanto o comportamento do robô fosse o adequado, sendo que os documentos de análise e projeto e especificação de testes foram atualizados na fase de fechamento do projeto de forma a manter a consistência com o *software* implementado.

Para trabalhos futuros, sugere-se avaliar o processo apresentado em outros projetos similares, ou seja, em sistemas embarcados e de tempo real, para identificar possíveis adaptações, de preferência usando-se técnicas de identificação de variáveis em processos empíricos definidos por CARVER e BASILI (2003). Também se sugere fazer o mesmo projeto apresentado usando-se outros processos, comparando os resultados obtidos, com os resultados aqui apresentados. Este último caso pode ser estendido definindo-se métricas tais como de avaliação de custo, prazo e de qualidade, para medir o desempenho desse processo comparado ao de outros processos. Algumas métricas para avaliação de processos e produtos podem ser vistas em ISO (2001).

REFERÊNCIAS

BOOCH, G. *et al*.; **UML – Guia do Usuário**. Rio de Janeiro. Campus. 2000.

BRICKOS at *Sourceforge*. Disponível em < http://brickos.sourceforge.net/>. Acessado em 08 Ago 2008.

CARVER J. e BASILI, V. ***Identifying Implicit Process Variables to Support Future Empirical Work*** In 17º. Simpósio Brasileiro de Engenharia de Software. Manaus, Amazonas, Brasil, 2003.

CYGWIN *Information and Installation*. Disponível em <http://www.cygwin.com/>. Acessado em 08 Ago 2008.

DOUGLASS, B. ***Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns***. Addison-Wesley Professional, 1999a.

DOUGLASS, B. ***Real-Time UML: Developing Efficient Objects for Embedded Systems***. 2 Ed. Addison-Wesley Professional, 1999b.

IAS – *INDUSTRIAL AUTOMATION AND SOFTWARE ENGINEERING INSTITUTE*, *University of Stuttgart;* ***Industrial Automation – Real-Time Programming***. *Stuttgart, Germany,* 2005a.

IAS – INDUSTRIAL AUTOMATION AND SOFTWARE ENGINEERING INSTITUTE, University of Stuttgart; ***Software Engineering for Real-Time Systems***. *Stuttgart, Germany*, 2005b.

ISO – INTERNATIONAL ORGANISATION FOR STANDARLIZATION. ***ISO 9126-1:2001, Software engineering - Product quality, Part 1: Quality model***. 2001.

KNUDSEN, J. ***The Unofficial Guide to LEGO® MINDSTORMSä Robots.*** O'Reilly & Associates, Inc, 1999.

LEGO Education – Mindstorms: RCX. Disponível em <http://www.Lego.com/eng/education/mindstorms/home.asp?pagename=rcx>. Acessado em 08 Ago 2008.

LEWIS, W. ***Software Testing and Continuous Quality Improvement***. 2 ed. CRC Press, 2000.

PRESSMAN, R. **Engenharia de Software**. 5 ed. Rio de Janeiro: McGraw-Hill, 2002.

PUTTEN, R. **Design of a maze solving robot using Lego MINDSTORMS**. Eindhoven University of Tecnology. Eindhoven, 2006. Disponível em <http://alexandria.tue.nl/repository/books/626828.pdf>. Acessado em 08 Ago 2008.

TESSIER, P. et al. **A Component-Based Methodology for Embedded System Prototyping** In Proceedings of the 14th IEEE International Workshop on Rapid Systems Prototyping (RSP'03), IEEE, 2003.