Um Ambiente para o Desenvolvimento de Software Embarcado em Satélite

Primavera Botelho de Souza [*1], Tatuo Nakanishi [**2],João Bosco Schumann Cunha [**3]

(1)Área de Engenharia de Software

Divisão de Aeronomia

Instituto Nacional de Pesquisas Espaciais (INPE)

(2)Área de Engenharia de Software e Sistemas Distribuídos

Universidade de Mogi das Cruzes

(3)Área de Engenharia de Software

Universidade de Mogi das Cruzes

(*)Mestrado, e-mail: prima@laser.inpe.br (**) Orientadores

Resumo

Este trabalho tem por objetivo propor a organização de um ambiente para desenvolvimento de uma classe especial de software que desempenha funções críticas a bordo de um satélite. A estrutura do trabalho consiste na sistematização do processo de construção do software, através da abordagem estruturada, visando garantir um produto correto, confiável e preparado para reuso.

Palavras-Chave: sistemas de tempo real, sistemas embarcados em satélite, processo de desenvolvimento de software, garantia de qualidade de software.

Introdução

Os métodos tradicionais baseados na geração de código, ou seja, somente na atividade de programação, geram muitos erros durante o desenvolvimento e resultam em grandes períodos dedicados à sua correção, acarretando um aumento de custo e incerteza quanto à remoção efetiva das falhas. Exemplos recentes [5], [6] constatam que softwares embarcados designados para desempenhar funções críticas falham e que, mesmo com a utilização de modelos definidos pela Engenharia de Software, motivadas pela imaturidade das organizações no processo de desenvolvimento, as falhas podem resultar em insucesso no funcionamento ou em projetos longos e de custo elevado.

A seção 2 descreve de uma forma geral as características e os fatores diferenciadores do software embarcado. Em seguida é apresentado na seção 3 o problema existente em relação à forma como os softwares embarcados estão sendo produzidos no INPE, sob os aspectos de garantia de qualidade e reuso, bem como até que ponto o desenvolvimento é sistematizado e documentado. Por fim, na seção 4, são fornecidas diretrizes para aplicação ao processo de desenvolvimento do software do sistema a bordo do satélite CBERS.

Definição e descrição do sistema embarcado em satélite

O sistema embarcado em satélite é classificado como sistema de tempo real, ou seja, aquele que gera certa ação em resposta a eventos externos. Para cumprir essa função, o sistema realiza as atividades de controle e aquisição de dados em alta velocidade sob severas restrições de tempo e confiabilidade [8]. Além da característica de tempo real, cada sistema embarcado em satélite apresenta outras características que dependem da arquitetura do sistema embarcado definida de acordo com os requisitos da missão e as necessidades operacionais [7]. Estas características adicionais são classificadas como fatores diferenciadores [9] e estão relacionadas à distribuição do sistema, criticidade de resposta a ocorrência de eventos, grau de concorrência, tipo de desempenho, taxa de transferência de dados, forma, consumo, condições ambientais e manutenção. Por conseqüência, o software embarcado obedece as restrições impostas ao sistema embarcado, provenientes da sua própria natureza, tais como [9]: dependência do hardware em que é executado, pode ou não ser distribuído, é concorrente, deve obedecer restrições impostas ao seu tamanho, é complexo por ter interações com um ambiente complexo, deve utilizar eficientemente a capacidade de recursos do hardware (capacidade de processamento, memória e dispositivos E/S), deve cumprir os requisitos de confiabilidade, segurança e desempenho determinados para o sistema.

Fatores que interferem no processo de desenvolvimento do software embarcado no INPE

As recentes avaliações nos processos de desenvolvimento de projetos para aplicações espaciais indicam que apesar da existência de processos para desenvolvimento [7], alguns fatores interferem na construção do software e que, consequentemente, interferem na qualidade do produto de software gerado. Os principais fatores são decorrentes das modificações que ocorrem no projeto ao longo do processo, tais como: mudanças e falta de investimento na equipe, bem como a utilização de método para desenvolvimento baseado somente na atividade de programação resultam, na prática, em aumento expressivo no período dedicado à fase de teste, bem como uma documentação do software que, na maior parte dos casos, segue a engenharia de software reversa. Outros fatores são decorrentes da falta de recursos em ferramentas que auxiliem no desenvolvimento, tais como: documentação feita de modo manual, linguagens de baixo nível que não dispõem de maiores recursos para depuração, falta de uma ferramenta automática de teste e a necessidade de um equipamento para teste mais modular em software e hardware que possibilite o reuso [7].

Processo de desenvolvimento de software do sistema CTU

Com uma abordagem estruturada são fornecidas diretrizes para construção de um sistema embarcado em satélite. É utilizado, como exemplo, a construção do software para aplicação espacial do sistema Central Terminal Unit (CTU) que forma com os demais subsistemas o sistema On-board Data Handling (OBDH) a bordo do Satélite Sino-Brasileiro de Recursos Terrestres (China-Brazil Earth Resources Satellite – CBERS) [7]. A CTU centraliza o controle da operação dos demais subsistemas embarcados e realiza o tratamento de exceções por comando de solo ou recuperação automática.

O passo inicial consiste na definição do sistema, isto é, a criação de um modelo descritivo, um texto informal que deve anteceder a construção do projeto em si, pois serve de base para a etapa de elaboração do modelo conceitual (fase de análise) do sistema. Entretanto, ao se iniciar a fase da construção do modelo conceitual a partir do modelo descritivo, depara-se com uma grande dificuldade para a construção do diagrama de fluxo de dados (DFD), pois ainda pouco se conhece como a informação flui ao longo do sistema. Com o objetivo de suavizar a transição entre a fase de definição do sistema ou especificação de requisitos e a fase de elaboração do modelo conceitual ou análise, foi aplicado ao sistema CTU, o modelo *use case,* definido pela Linguagem de Modelagem Unificada *(Unified Modeling Language* - UML). Inicialmente, o modelo *use case* permite representar a especificação funcional do sistema como um todo, considerando o sistema como uma caixa preta [3] e definir as interfaces com o mundo exterior. Porém, a aplicação deste modelo não se restringiu a nível de sistema, pois pode também representar subsistemas ao longo das diversas fases de desenvolvimento [1], permitindo assim, obter-se valiosa informação sobre especificação funcional do sistema sem implicar em qualquer tipo de implementação física.

Com a construção do diagrama de fluxo de dados, definiu-se o modelo conceitual do sistema CTU e concluiu-se a fase de análise. Entretanto, uma série de dificuldades surgiram durante a transição da fase de análise para a fase de projeto físico, uma vez que as técnicas de mapeamento aplicadas ao diagrama de fluxo de dados não suportam adequadamente todos os problemas de um projeto de tempo real.

O sistema de tempo real é constituído de vários processos computacionais ou tarefas concorrentes. A concorrência se deve ao fato de existirem tarefas assíncronas executadas em velocidades diferentes. Entretanto, de tempos em tempos, as tarefas precisam comunicar e sincronizar umas com as outras [2]. Portanto, para simbolizar a comunicação e o sincronismo entre tarefas, foram desenvolvidas extensões às representações de fluxo de dados que oferecem mecanismos para o projeto de software de tempo real. A abordagem denominada *Método de Projeto para Sistemas de Tempo Real* (Design Approach for Real-Time System - DARTS) [2], [4] e [3], permite que projetistas de sistemas de tempo real adaptem técnicas de fluxo de dados às necessidades especiais de aplicações de tempo real [2],[4] e [8]. O método pode ser imaginado como uma ampliação do método clássico de análise e projeto estruturados ao criar uma abordagem baseada na divisão do sistema em tarefas, bem como um mecanismo para definir as interfaces entre elas. O método inicia-se com a aplicação dos princípios fundamentais de análise de software, ou seja, análise do domínio de informação e partição do problema aplicados no contexto da notação de fluxo de dados. Sendo assim, medidas de qualidade de projeto como modularidade e independência funcionais são reforçadas com métodos orientados para fluxo de dados.

Os critérios para determinar se as funções do DFD do sistema CTU deveriam ser definidas como tarefas separadas ou agrupadas com outras funções numa única tarefa, basearam-se em aspectos físicos como: dependência de E/S, funções críticas quanto ao tempo, requisitos computacionais, coesão funcional, coesão temporal e execução periódica [2].

A partir da concepção das tarefas, prosseguiu-se para análise do tipo de informação que deve fluir entre uma tarefa e outra, isto é, os fluxos de dados de entrada e de saída de cada tarefa devem permanecer inalterados em relação ao DFD construído. De forma a garantir este mapeamento, o método DARTS

proporciona um mecanismo para manusear as informações entre as tarefas ao definir duas classes de módulos de interface [2]: *o módulo de comunicação entre tarefas (Task Communication Module - TCM)* e *o módulo de sincronização de tarefas (Task Synchronization Modules – TSM).*

Uma vez definidas as tarefas, o próximo passo foi estabelecer a estrutura individual de cada tarefa, onde cada qual representa um programa seqüencial. Para cada tarefa, um DFD deve ser desenhado e a partir dessa estrutura, um fluxograma é derivado conforme método de projeto estruturado. Cada programa deverá interagir com os outros programas para que o sistema possa ter sua função executada, sob a forma de um conjunto de processos computacionais [8]. A seguir, deve-se definir um plano de implementação e testes para o sistema. O plano envolverá a adoção de uma estratégia para projetar os casos de teste para o software do sistema CTU, de forma a verificar se o sistema de software está cumprindo corretamente as suas funções dentro de um contexto maior, ou seja, se ele está operando de maneira correta quando colocado em conjunto com os elementos com que ele se relaciona.

Resultados alcançados

Esta metodologia é a parte concluída, até o momento, do ambiente voltado para o desenvolvimento de software para aplicação espacial. Consiste em um conjunto de métodos para todas as fases de desenvolvimento que interferem em medidas de qualidade de projeto tais como modularidade e independência funcionais, alcançando, parcialmente, o resultado esperado de sistematizar o processo de desenvolvimento para a produção de software correto, confiável e preparado para reuso.

Conclusão

Resta ainda, definir as atividades de gerenciamento e métodos de controle das fases de desenvolvimento, cuja abrangência se estende à melhoria do processo como um todo, através de normas, inspeções [10], revisões [4] e controle de configuração realmente eficientes, bem como apresentar uma ferramenta de teste mais modular para software que possibilite reuso. Enfim, a contribuição do trabalho pode tornar-se ainda mais significativa ao considerarmos que o ambiente a ser proposto não restringe a possibilidade de migrar da metodologia estruturada para orientada a objeto.

Referências

[1] Douglass, Bruce. **Real-time UML: developing efficient objects for embedded systems**. 2nd ed. Addison-Wesley, 2000.

[2] Gomaa, H. **A software design method for real–time systems**. Communications of the ACM, 27 (9), september 1984.

[3] Gomaa, H. **Designing concurrent, distributed, and real-time applications with UML**. Addison-Wesley, 2000.

[4] Gomaa, H. **Software development of real–time systems**. Communications of the ACM, 29 (7), july 1986.

[5] Leveson, N.; Turner, C. **An investigation of the Therac-25 accidents**. Computer, vol 26, n.7, July 1993, p.18–41.

[6] Nuseibeh, B. **Ariane 5: who dunnit?** IEEE Software, May 1997, p15-16.

[7] Pessota, F.A.; Alonso, J.D.D. **Supervisão de bordo**. Curso de Tecnologia de Satélite. Anais. São José dos Campos, Instituto Nacional de Pesquisas Espaciais, 1999.

[8] Pressman, R. **Software engineering: a Practitioner's Approach**.4ª. ed. New York, McGraw Hill, 1997.

[9] Spinola, M. M. **Diretrizes para o desenvolvimento de software de sistemas embutidos**. Tese. Escola Politécnica da Universidade de São Paulo, 1998. 250 p.

[10] Strauss, S.; Ebenau, R. G. **Software inspection process**. New York, McGrawHill, 1994. (System design and implementation series).