9º Workshop em Engenharia e Tecnologia Espaciais
15 e 16 de Agosto de 2018

Proposta de processo de desenvolvimento de software embarcado para satélites utilizando o padrão open-source cFS/NASA

MIRANDA, D.J.F. [1], FERREIRA, M.G.V. [1], KUCINSKIS, F. [1]

[1]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil

Aluno de Mestrado do curso de Engenharia e Gerenciamento de Sistemas Espaciais – ETE/CSE.

danilo.miranda@inpe.br

Resumo. *Nas primeiras missões espaciais, pouca complexidade era exigida do segmento espacial, e praticamente todas as tarefas computacionais eram alocadas ao segmento solo. Com a evolução da eletrônica e o aumento da confiança em operações mais autônomas de satélites, dentre outros fatores, observa-se que o software embarcado está cada vez mais complexo. Num esforço para lidar com tal complexidade, em 2015 o centro Goddard Space Flight Center, da NASA, tornou público o código do framework de software embarcado chamado cFS (core Flight System) utilizado em suas missões científicas. Tal framework se baseia no conceito de "linha de produtos de software", i.e., a organização criadora do sistema mantém e atualiza apenas o core do sistema, o qual será a base para a construção de produtos específicos para uma dada aplicação. O presente trabalho visa analisar os processos de desenvolvimento de software de uma missão CubeSat que optou por utilizar sistematicamente o padrão cFS em seu software de bordo. Tais processos serão organizados numa proposta de desenvolvimento de software, com critérios de validação a serem atingidos e cujo potencial desejado de reuso vai ser futuramente verificado.*

Palavras-chave: Software Embarcado; OBSW; OBDH; cFS; Software Product Line (SPL)

1. Introdução

Nas primeiras missões espaciais, praticamente todas as tarefas computacionais eram alocadas ao segmento solo. Em tais missões, o segmento espacial era meramente um executor de comandos enviados pelo operador em solo [Kucinskis, 2012]. Com a evolução dos computadores de bordo, da eletrônica e com o aumento da confiança em operações autônomas de satélites, mais responsabilidade foi gradativamente sendo dada ao software de bordo (OBSW). Dvorak [2009] aponta que a complexidade do software embarcado está crescendo rapidamente, com o número de linhas de código nas missões da NASA aumentando exponencialmente por um fator de 10 aproximadamente a cada 10 anos. Tal

aumento na complexidade, somado às especificidades da área espacial, tornam necessário um processo formal de desenvolvimento de software embarcado para atender às demandas complexas requeridas pelas missões espaciais atuais e futuras.

A despeito disso, Eickhoff [2012] relata que os tópicos de projeto do computador de bordo, do software de bordo e do conceito de operações de satélites são normalmente um desafio para instituições ingressantes no setor espacial, devido principalmente à carência de literatura especializada. Em aplicações terrestres de software, pelo contrário, a falta de conhecimento especializado ou de padrões de projeto de software já não representa mais um desafio às equipes de desenvolvimento.

É notório que nos últimos anos tem havido uma mudança de paradigma com respeito ao desenvolvimento tradicional de software. Ao invés de desenvolverem um software customizado para uma dada aplicação, é comum as organizações focarem seus esforços em desenvolver e manter um conjunto de aplicações "*core*", o qual será a base para a construção de produtos específicos para uma dada aplicação [Falvo Júnior et al, 2014]. Tal abordagem se chama "linha de produtos de software" (*software product lines, SPL*) e é bastante comum em aplicações de software para eletrônica de consumo.

No que se refere a área espacial, uma relevante iniciativa de SPL é o framework de software de bordo chamado cFS (*core Flight System*), projeto esse iniciado em 2003 no *Goddard Space Flight Center* (GSFC/NASA) nos EUA e que agrega a experiência da NASA ao longo dos últimos 30 anos em desenvolvimento de software para satélites.

Tal framework tinha uso restrito à agência espacial norte-americana até que, em 2015, o Goddard tornou público parte de seu código fonte. O cFS tem TRL 9 e está em conformidade com os requisitos de desenvolvimento de software classe B da NASA (missões operacionais não tripuladas) [McComas, 2015].

Dentre as características do padrão cFS, se destacam sua modularidade, portabilidade, qualidade de software, suporte e manutenção do seu *core* pela NASA e potencial colaboração entre usuários.

O presente artigo apresenta uma visão geral sobre o trabalho em andamento, que tem por objetivo portar e utilizar o padrão cFS no software de voo de uma missão CubeSat, aqui tratada como "missão de referência", e derivar dessa implementação processos de desenvolvimento de software que possam ser reutilizados em missões futuras, em especial missões CubeSats e de pequenos satélites.

2. Metodologia

Os processos de software utilizados na missão de referência estão em conformidade com missões classe C/D da NASA (prioridade média-baixa, tolerância média-alta ao risco). Missões dessa categoria tipicamente incluem CubeSats e pequenos satélites, cujos *budgets* são menores, equipes técnicas normalmente reduzidas, e a aceitação ao risco é maior do que em missões operacionais e de alto custo, como por exemplo as missões CBERS e SGDC.

Em missões classe C/D tipicamente se gasta mais tempo com *design* e *engenharia de sistemas* do que com controle de configuração, qualidade e atividades de verificação e

validação [Johnson-Roth, 2013]. Ainda assim, as demais atividades do ciclo de vida de um software devem ser realizadas ao menos minimamente de forma a garantir a qualidade final esperada do produto.

Os autores observaram que a documentação que acompanha o padrão cFS é particularmente voltada ao código em si e à arquitetura de software, ambas atividades de design. A Figura 1 ilustra a arquitetura do cFS, mostrando com código de cores a origem de cada aplicação. Pouco ou nada se fala sobre as fases pré-design (necessidades, requisitos, conformidade às normas CCSDS típicas, etc) e pós-design (testes, verificação e validação, aceitação, operações, etc) de um software de bordo baseado em cFS. Outro aspecto particularmente interessante e pouco abordado na documentação do cFS são as interfaces solo-bordo e os impactos da utilização do padrão cFS no software de solo e na filosofia de operações.

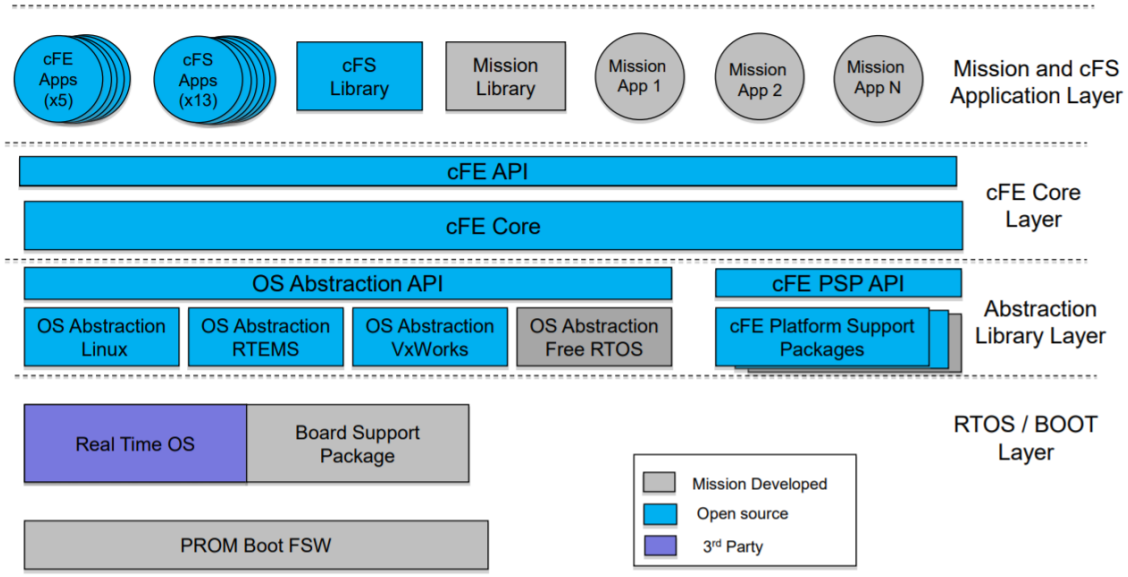

Figura 1. Arquitetura do Padrão cFS. [NASA, 2014]

A metodologia do presente trabalho consiste então em, com base nas normas e documentação aplicáveis, coletar os processos de engenharia de software tipicamente usados em missões espaciais, e adaptar tais processos com base nas restrições e condições de contorno impostas pelo framework cFS e pelo budget da missão.

3. Resultados Preliminares e Discussão

Os autores dispõem no momento de resultados preliminares, uma vez que o artigo em questão deriva de dissertação de mestrado em andamento.

O primeiro estudo feito foi uma análise comparativa entre o padrão cFS e outros padrões de software embarcado utilizados em satélites. Ao todo, sete padrões de software foram

analisados. As arquiteturas de software foram comparadas à luz de critérios tipicamente exigidos de um software de tempo real de aplicação crítica para sistemas espaciais. O padrão cFS se mostrou o mais promissor para missões rápidas e de baixo custo, por já abarcar característica de SPL e possuir código aberto, herança de voo e portabilidade atestada para sistemas operacionais de tempo real e diversas arquiteturas de processadores.

A fase seguinte consistiu em ponderar a conformidade do padrão cFS com as normas e protocolos espaciais aplicáveis, de forma a analisar as capacidades e limitações do padrão à luz das normas já aceitas pela comunidade espacial, conferindo um linguajar comum entre especialistas. As normas utilizadas foram:

- CCSDS
    - TMTC (CCSDS Space Packet Protocol (133.0-B-1), TM Space Data Link Protocol (132.0-B-2), TC Space Data Link Protocol (232.0-B-3))
    - Transferência de arquivos (CCSDS File Delivery Protocol – CFDP (727.0-B-4))
    - CCSDS SOIS (Spacecraft Onboard Interface Services - 850.0-G-2)
- ECSS
    - TM and TC Packet Utilization Standard (PUS - ECSS-E-ST-70-41A)
- NASA
    - Software Engineering Requirements (NPR 7150.2A)
    - Risk Classification for NASA Payloads (NPR 8705.4)

Tais normas foram consideradas representativas pelos autores. O framework cFS é *fully-compliant* com as normas CCSDS para TMTC e transferência de arquivos e parcialmente conforme com o padrão SOIS.

Com relação aos serviços PUS, o padrão cFS, no que se refere às aplicações disponíveis *open-source,* é parcialmente conforme com a maioria dos serviços.

No que se refere às normas da NASA, o framework cFS já foi utilizado em missões científicas classe B, conforme a norma NPR 8705.4. A presente missão de referência, entretanto, contempla maior aceitação ao risco, pertencendo à classe C ou D, e, dessa forma, o autor principal está se baseando no estudo de [Killough, 2013] para adaptar as exigências de processos de engenharia de software da norma NPR 7150.2A, que são muito demandantes, para uma missão CubeSat.

A arquitetura cFS é modular, sendo baseada em componentes ou aplicações, conforme ilustrado na Figura 1. Os autores estão trabalhando num *template* de uma aplicação de software *compliant* com o padrão cFS e regras de *design* a serem seguidas nas aplicações (camada *Application Layer*, da Figura 1). Busca-se assim simplificar o processo de desenvolvimento de software embarcado.

O fato de se adotar um framework de software simplifica não só a definição da arquitetura de software e elaboração subsequente do código, mas também a elaboração de requisitos e a filosofia de verificação e validação. Por exemplo, já se verificou preliminarmente que muitos

testes podem vir a ser substituídos por inspeção estática do código, i.e., sem executar o sistema, o que representa uma economia de tempo para a equipe de projeto.

No momento, se prevê a validação dos processos de software elaborados em virtude da utilização do framework cFS por meio de duas estratégias:

- Teste *end-to-end*: A partir de simulação de operações representativas do *day-in-the-life* da missão durante 24h, será verificado se o software embarcado conclui as tarefas esperadas sem falhas. O objetivo com isso é a validação funcional de um OBSW conforme com o cFS e que segue os processos de desenvolvimento de software propostos pelos autores.
- Número de linhas de código: Contagem do número de linhas de código das novas aplicações escritas para a missão versus o número de linhas de código das aplicações herdadas do cFS. O objetivo é quantificar aproximadamente o esforço relativo para se elaborar o software de uma missão CubeSat com base nos processos de desenvolvimento propostos pelos autores.

O resultado final esperado é atestar se os processos de desenvolvimento de software elaborados em torno do framework cFS simplificam de fato o projeto do OBSW de uma missão de nanosatélite sem prejudicar a qualidade geral do software, e se tal estratégia possui potencial de reuso para missões satelitais similares.

4. Conclusão

Os processos de engenharia de software para o desenvolvimento de um OBSW tendo como premissa o framework cFS ainda estão em fase de elaboração. No entanto, os resultados parciais parecem promissores. O primeiro deles, que foi analisar se de fato o cFS, dentre os padrões de código aberto, era o mais conveniente para uma missão satelital *low-cost*, se apresentou favorável.

O segundo resultado parcial mostrou que o cFS é ao menos parcialmente conforme com as principais normas CCSDS, ESA e NASA relacionadas aos protocolos e serviços tipicamente esperados de um software embarcado. Foi possível verificar que mesmo as funcionalidades que não estão implementadas por *default* no cFS podem ser incluídas em novas aplicações de software que irão compor a camada *Application Layer* da Figura 1.

O conjunto de processos de desenvolvimento de software ainda está em elaboração, conforme o avanço da missão de referência.

Agradecimentos: *Ao Instituto Nacional de Pesquisas Espaciais (INPE), à equipe organizadora do WETE 2018 e à empresa Visiona Tecnologia Espacial S.A., pelo apoio dado na realização desse Mestrado.*

Referências

Dvorak, D. L. (2009). NASA Study on Flight Software Complexity. *In: AIAA Infotech@Aerospace Conference*. Seatle: Aerospace Conference.

Eickhoff, J. (2012). Onboard Computers, Onboard Software and Satellite Operations: An Introduction. 301 p. Springer Series in Aerospace Technology: Berlin, 2012.

Falvo Jr, V.; Duarte Filho, N. F.; Oliveira Jr, E. and Barbosa, E. F. (2014). Towards the Establishment of a Software Product Line for Mobile Learning Applications. In: *The 26th International Conference on Software Engineering and Knowledge Engineering*, pages 678-683. SEKE 2014: Vancouver, Canada.

Johnson-Roth, G. A. (2013). Key Considerations for Mission Success for Class C/D Mission. Contract No. FA8802-09-C-0001. 134 p. The Aerospace Corporation: AEROSPACE REPORT NO. TOR-2013-00294.

Killough, R.; Rose, D.; Rose, R. J. (2013). Software engineering processes for Class D missions. In: *SPIE Optical Engineering + Applications*, 2013. Vol. 8866. San Diego, California, EUA.

Kucinskis, F. N. (2012). Uma arquitetura de software embarcado do segmento espacial para habilitar a operação de missões baseada em objetivos. 194 p. Tese de Doutorado. São José dos Campos: INPE, 2012. (sid.inpe.br/mtc-m19/2012/03.01.14.50-TDI).

McComas, D.; Strege, S.; Wilmot, J. (2015). core Flight System (cFS): A Low Cost Solution for SmallSats. Available at https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20150018075.pdf.

NASA. (2014). core Flight System (cFS) Background and Overview. Available at https://cfs.gsfc.nasa.gov/cFS-OviewBGSlideDeck-ExportControl-Final.pdf.