Viabilidade das Metodologias de Desenvolvimento de Software na Concepção e Validação Conjunta de Hardware/Software de Sistemas Ubíquos Embebidos

Alberto Sampaio, Gustavo Alves

Resumo. Neste artigo, afirma-se a necessidade de avaliar a adequação das metodologias de desenvolvimento de software, no contexto da concepção conjunta de HW/SW de sistemas ubíquos embebidos.

1 Introdução e Motivação

Os sistemas ubíquos são muitas vezes embebidos. Sistema embebido é um subsistema electrónico com uma aplicação específica, que pode ser usado em sistemas maiores [2], por exemplo o telemóvel. Trata-se portanto de uma arquitectura de hardware/software (HW/SW) em que estes dois componentes são desenvolvidos em "simultâneo" e o software é frequentemente específico para o hardware.

Assiste-se a uma cada vez maior necessidade e presença do software nos vários sistemas. Verifica-se a solicitação de cada vez mais funcionalidades, facilmente conseguidas através do software. Consoante o hardware se torna mais complexo, também o software requerido se torna. A computação reconfigurável e posteriormente a auto reconfigurável vieram dar uma ainda maior ênfase à componente software. Para a maior importância do software contribui ainda um cada vez maior nível de abstracção conseguido na simulação do hardware.

Consequentemente, ocorre uma maior intervenção humana directa em todo o processo o que constitui um factor adicional de subjectividade e incerteza no processo de desenvolvimento. Ao mesmo tempo, verifica-se um hiato entre o que as equipas conseguem conceber e as possibilidades oferecidas pela tecnologia. O nível de complexidade atingido pelos sistemas embebidos, associado à sua criticidade, levanta sérios desafios ao seu desenvolvimento. Tudo isto é, naturalmente, condicionado pela metodologia utilizada no desenvolvimento.

2 Alguns Problemas e Limitações

A concepção conjunta de HW/SW apesar de ter passado de disciplina emergente a uma tecnologia fundamental [6], não é um problema resolvido, quer pelos problemas que se encontram em aberto, quer pela evolução tecnológica que a afecta e levanta novos problemas. Um bom exemplo é o SoC ("System on Chip") que pode conter diversos CPUs e conter elementos heterógeneos e ainda sofisticados NoC ("Network on Chip"). Estas tecnologias tornaram possível desempenhos elevados, consumos reduzidos de energia com baixa dissipação da potência, e comunicação sem fios, tudo em sistemas de dimensões reduzidas, e assim facilitaram a computação ubíqua. No entanto, o desempenho exigido é cada vez maior, sempre com a preocupação de um consumo reduzido de energia. Acrecem preocupações com confiabilidade, testabilidade, segurança, adaptabilidade e auto-adaptabilidade.

Como implicação, o desenvolvimento desses sistemas tornou-se cada vez mais complexo, com o surgimento de cada vez mais ferramentas que visam permitir a sua automação. Como tem sido constatado no software, a adopção de ferramentas deve ser feita de forma criteriosa, o que requer um certo nível de maturidade da organização, a que subjaz a noção de processo de desenvolvimento/concepção e de qualidade do processo e do produto. Preocupações com o custo, o esforço e o tempo de colocação dos produtos no mercado tornam são importantes. Para além de se saber quais as ferramentas a adoptar, uma organização deve seguir uma metodologia de desenvolvimento que garanta a sua utilização no momento mais adequado.

A concepção conjunta de HW/SW tem consistido na partição do sistema em hardware e software, com uma equipa a desenvolver o software e outra a conceber a componente de hardware [2]. Isto levanta dois problemas [2]: problemas detectados no desenvolvimento do software não podem ser corrigidos na plataforma; o processo requer mais tempo com os custos decorrentes dessa morosidade. Para resolver estes problemas a indústria tem procurado passar para um processo concorrente.

Existem ainda outras limitações e problemas com que as metodologias actuais de concepção conjunta têm sido confrontadas. Por exemplo, os métodos clássicos de concepção não possuem uma generalização suficiente para serem eficientes num grande número de aplicações diversas permitidas por arquitecturas programáveis, nem oferecem uma extensão fácil dos sistemas embebidos para suportarem novas aplicações [1]. De forma resumida, algumas condicionantes e desafios em aberto na concepção conjunta de sistemas ubíquos embebidos são descritos em seguida.

- A ubiquidade levanta novos problemas como a adaptação e auto adaptação e a sensibilidade ao contexto. Este último ponto é evidente ao nível das interfaces que necessitam ser alteradas de acordo com as situações;
- Definição e redefinição de modelos computacionais para descrição conjunta dos sistemas de HW/SW [6];
- Exploração do espaço-de-concepção, onde têm sido aplicados métodos avançados como algoritmos genéticos [6];
- A sofisticação dos actuais sistemas embebidos com elementos possuindo vários processadores e elementos heterogéneos, o que levanta problemas como o da

escolha de processadores dedicados, ou programáveis [2] e torna inadequada a prática de simulação da exploração do espaço-de-concepção [5];
- Redes de sistemas embebidos (como acontece nas aeronaves e nos automóveis);
- Necessidade de fazer actualizações ao software existente em cada sistema;
- Utilização da mesma plataforma programável para diferentes produtos [2];
- O já referido problema de equipas separadas trabalhando em série para a criação das concepções de hardware e software, o que motiva problemas de qualidade e morosidade do processo;
- Também a computação reconfigurável requer um novo modelo de concepção [3]. E, mais ainda, a auto reconfigurável;
- Simultaneamente, existe uma preocupação cada vez maior em que os sistemas de software sejam autónomos e autonómicos.

3 Possíveis Soluções

Desde há algum tempo que para a concepção conjunta de HW/SW tem sido apontada a necessidade de novas abordagens [2]. A modelação conjunta de hardware e software envolve a descrição da aplicação, o que será de primordial importância para o desenvolvimento do sistema. Ao nível do sistema, restrições quanto ao domínio, como o de tempo real, dificultam a adopção de modelos orientados aos objectos. A investigação de métodos automáticos para análise da aplicação e derivação dos requisitos de hardware já foi empreendida [4], e baseia-se em diagramas UML.

Na concepção conjunta de modelos têm sido usadas notações como a SystemC (www.systemc.org), baseada em C++, para a concepção, a UML e a SysML (http://www.sysml.org/) mais usadas ao nível da especificação, e ainda a SpecC (http://www.specc.gr.jp/eng/index.htm), baseada em C, ao nível da especificação e concepção . Estas notações, e as ferramentas de apoio ao desenvolvimento devem ser enquadradas numa metodologia, que oriente a sua escolha e condução. Apesar de existirem propostas de integração de modelos como a MDA (“Model Data Architecture”), apenas dizem respeito a modelos, e parecem ser insuficientes [12], pelo que a escolha de uma metodologia continua a ser essencial.

Sabendo-se que o software ocupa uma importância crescente na concepção conjunta de HW/SW, e que na área da engenharia de software existem imensas metodologias de desenvolvimento (e.g., Espiral, Iterativo, Prototipagem, Crystal, RUP), onde os problemas da qualidade e da intervenção humana no desenvolvimento têm sido amplamente investigados, parece-nos lógico que em lugar de se proporem novos métodos para a concepção conjunta, se deveria começar por analisar a adequação dos que já existem na área do software ao contexto da concepção conjunta de HW/SW de sistemas ubíquos embebidos. Para isso, será necessária não só a análise individual de cada método, mas, também, a sua comparação tendo em atenção o contexto pretendido. Uma comparação pode, para além de avaliar a adequação dos métodos, facilitar a escolha do método a utilizar numa dada situação. Tanto quanto nos foi possível investigar, tal comparação nunca foi realizada.

4 Conclusões

Muitos dos sistemas ubíquos são sistemas embebidos, isto é, as duas componentes hardware e software são ambas importantes e devem ser consideradas no desenvolvimento dos sistemas. No entanto, essa não é sempre a posição observada. A nossa preocupação é com sistemas em que ambas as componentes têm de ser desenvolvidas, e não com sistemas tradicionais em que o projectista basicamente ignora o hardware, ou vice-versa.

Baseados na necessidade de considerar ambas as componentes e na importância crescente do software, foi afirmada a seguinte posição: necessidade de avaliar a adequação das metodologias existentes para o desenvolvimento de software, no contexto da concepção conjunta de HW/SW de sistemas ubíquos embebidos.

Para isso, propomo-nos analisar e comparar as metododologias actualmente existentes para o desenvolvimento de software. A abordagem a seguir deverá consistir na determinação dos requisitos da concepção conjunta e das especificidades dos sistemas ubíquos embebidos, no estudo individual de cada uma das metodologias, e finalmente na comparação destas tendo em atenção o contexto.

Os dois primeiros pontos já foram aqui abordados de forma preliminar. A comparação deverá considerar inicialmente o maior número possível de características, que irá sendo reduzido ao longo do processo, consoante forem determinadas as características mais importantes. Para a comparação propomo-nos utilizar, com ajustes, metodologias de comparação já usadas para a comparação de métodos de avaliação do processo e de métodos de concepção de software.

Referências

[1] El-Khoury, J., Redell, Törngren, A Tool Integration Platform for Multi-Disciplinary Development, Proceedings of the 31st Euromicro Conference on Software Engineering and Advanced Applications, Aug.30-Sep.3 2005, Portugal, pp.442-449, IEEE Press, 2005.

[2] Jerraya, A.A., Wolf, W., Hardware/Software Interface Codesign for Embedded Systems, IEEE Computer, pp.63-69, February, 2005.

[3] Hartenstein, R., A decade of reconfigurable computing: a visionary retrospective, Proceedings of the conference on Design, automation and test in Europe, Munich, Germany, pp 642-649, 2001.

[4] Lilius, J., Truscan, D., UML-driven TTA-based protocol processor design. In Proceedings of the 2002 Forum for Design and Specification Languages (FDL'02), Marseille, France, September 2002.

[5] Pimentel, A.D., Hetzberger, L.O., Lieverse, P., van der Wolf, P., Deprettere, E.F., Exploring Embedded-Systems Architectures with Artemis.

[6] Wolf, W., A Decade of Hardware/Software Codesign, IEEE Computer, April, 2003.