1° CONGRESSO NACIONAL DE AUTOMAÇÃO INDUSTRIAL CONAI

Tema Automação e Desenvolvimento

Anais

Editores: Arylo G. Russo
Antonio T. M. Beraldo
Manuel J. Mendes

Promoção e Realização

SEI - Secretaria Especial de Informática
SUCESU - SP - Sociedade dos Usuários de Computadores e Equipamentos Subsidiários - São Paulo
ANDEI - Associação Nacional dos Dirigentes e Executivos de Informática
FAM - Forum das Américas
ABICOMP - Associação Brasileira da Indústria de Computadores e Periféricos

Data 11 a 15 de julho 1983

Local Maksoud Plaza
São Paulo

1º CONGRESSO NACIONAL DE AUTOMAÇÃO INDUSTRIAL CONAI

Tema Automação e Desenvolvimento

Anais

Editores: Aryldo G. Russo
Antonio T. M. Beraldo
Manuel J. Mendes

Promoção e Realização
SEI - Secretaria Especial de Informática
SUCESU - SP - Sociedade dos Usuários de Computadores e Equipamentos Subsidiários - São Paulo
ANDEI - Associação Nacional dos Dirigentes e Executivos de Informática
FAM - Forum das Américas
ABICOMP - Associação Brasileira da Indústria de Computadores e Periféricos

Data 11 a 15 de julho 1983

Local Maksoud Plaza
São Paulo

FERRAMENTAS DE APOIO AO DESENVOLVIMENTO DE SOFTWARE

Arndt von Staa
Departamento de Informática
Pontifícia Universidade Católica, RJ
Rua Marques de São Vicente, 225
Gávea
22453 Rio de Janeiro

RESUMO: É definido o que vem a ser um sistema embutido. São apresentados os requisitos gerais a serem satisfeitos pelas ferramentas de apoio ao desenvolvimento de software. É descrito o processo de desenvolvimento e manutenção de software, sendo apresentados exemplos de ferramentas dando suporte às diversas atividades deste processo.

1. INTRODUÇÃO

Este artigo tem por objetivo examinar, em linhas gerais, a contribuição que ferramentas de apoio ao desenvolvimento de software podem dar ao desenvolvimento de sistemas embutidos ("embedded systems"). Sistemas embutidos visam monitorar, supervisionar e/ou controlar um processo em operação, onde este processo por sua vez é componente de outro sistema, processo ou equipamento. Na figura 1 ilustramos de um modo genérico e simplificado, o que vem a ser um sistema embutido.

Do ponto de vista do propósito deste artigo é pouco interessante discutir-se o modo de realização do sistema embutido, o grau e a forma de interação deste sistema com o processo, etc. O que nos importa saber é que o sistema embutido:

i - relaciona-se intimamente com o processo através de uma diversidade de interfaces (sensores e atuadores);

ii - possui propriedades dependentes do processo;

iii - evolui à medida que o processo evolui (tempo real); o que exige do sistema embutido a capacidade de monitorar(obter dados relativos ao processo) em espaço de tempo exíguo, bem como exige a previsão de ações de emergência ou recuperação em caso de algum mal funcionamento.

Cabe observar também que a interface entre o sistema embutido e o processo é frequentemente difícil de determinar dado o grau de proximidade e interrelacionamento entre o sistema embutido e o processo, tornando o processo virtualmente inextricável do sistema embutido.

É notório que processadores digitais estão encontrando cada vez mais emprego como base de implementação de sistemas embutidos, desalojando, desta forma, outras tecnologias de controle, supervisão e/ou monitoramento anteriormente utilizadas. Tal evolução tem sido enormemente acelerada com o advento dos microprocessadores e dos circuitos de alto grau de integração. Estes componentes permitem solucionar de modo economicamente justificáveis problemas não antes solucionáveis, permitindo ainda atingir graus mais elevados de sofisticação, de desempenho ou precisão.

Se por um lado a evolução do hardware tem ampliado os horizontes de aplicação, por outro lado passou-se a depender do software responsável pelo comportamento exibido pelo sistema embutido. Sem este software, o sistema embutido nada mais é do que uma organização inerme de componentes. Por outro lado, varian

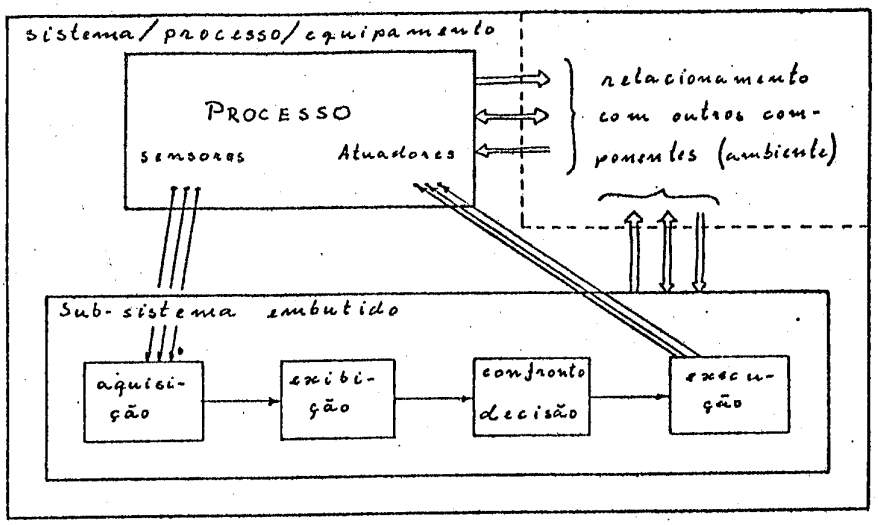

Figura 1. Visão genérica de um sistema embutido

do-se o software residente no sistema embutido pode-se alcançar drásticas mudanças de comportamento, tornado pois viável a construção de hardware para sistemas embutidos capazes de interagirem com amplas classes de processos. Consequentemente, por maior que seja o rigor do projeto e do controle de qualidade na implementação do processo e do hardware do sistema embutido, o resultado ficará muito aquém do esperado caso o software não seja implementado com o mesmo rigor de projeto e controle de qualidade.

Surge pois o problema: como construir software que equipe sistemas embutidos fazendo com que o conjunto processo/sistema embutido possua a qualidade e o desempenho desejado? Para tal precisamos:

a - selecionar, possivelmente desenvolver, e utilizar ferramentas de apoio às diferentes etapas do processo de especificação e construção;

b - selecionar, possivelmente formar e treinar, empregar e/ou consultar, pessoal competente e apto a resolver o problema em questão;

c - entender bem o problema, a ser resolvido em cada etapa documentando este entendimento sob a forma de especificações, utilizando para isto algum meio (ferramenta) de representação destas especificações;

d - controlar a qualidade dos resultados ao término de cada etapa do processo de especificação e construção;

e - gerir o processo de especificação e construção.

Nas seções a seguir examinaremos em mais profundidade as propriedades que ferramentas de apoio deverão satisfazer.

2. Propriedades gerais de ferramentas

Ferramentas de apoio ao desenvolvimento têm por objetivos:

a - viabilizar o desenvolvimento: ferramentas devem ser instrumentos de apoio à solução do problema a ser resolvido. Em muitas ocasiões ferramentas automatizadas complexas são mesmo indispensáveis dada a natureza e a complexidade do problema a ser resolvido, podendo até ser necessário desenvolver e/ou adaptar ferramentas automatizadas.

b - viabilizar a qualidade: ferramentas devem induzir as pessoas a fazerem corretamente as coisas certas. Ou seja, uma boa ferramenta deve auxiliar tanto no alcance de um elevado grau de qualidade de engenharia, como no alcance de um elevado grau de qualidade do produto em si.

c - aumentar a produtividade: ferramentas devem possibilitar o desenvolvimento requerendo menos pessoas e/ou menos tempo. Para tal devem impor poucos entraves, eliminar redundâncias de esforço, eliminar tarefas tediosas ou maçantes, etc.

d - reduzir os custos totais: ferramentas devem contribuir para reduzir o volume de recursos humanos, materiais, equipamento e financeiros necessários durante toda a vida do sistema. Tais recursos são consumidos durante o desenvolvimento a operação, a correção, a manutenção. São consumidos também ao estabelecer proteção e segurança, e ao monitorar e auditar o sistema. Finalmente podem ser desperdiçados em função de danos causados por erros, fraudes e acidentes.

e - integrar-se com outras de modo que o conjunto cubra todo o ciclo de vida: ferramentas são projetadas para darem apoio à resolução de uma determinada classe de problemas em determinado grau de detalhe (ou nível de abstração). Se fossem projetadas para darem apoio integral tornar-se-iam excessivamente complexas e custosas. Para evitar problemas de distorção, erros de conversão, etc. é claro que o resultado gerado por intermédio de uma determinada ferramenta deve servir diretamente como entrada à próxima ferramenta de construção ou de controle de qualidade. Não somente isto, durante o desenvolvimento e a vida útil do sistema ocorrerão alterações, estas devem ser realizáveis a um custo baixo e assegurando que a qualidade do sistema seja mantida ou aumentada.

Os objetivos acima apresentados interagem, requerendo, em vários casos, o estabelecimento de um compromisso com relação ao grau de alcance de cada um dos objetivos. O compromisso visa maximizar a satisfação geral no uso da ferramenta. Além destes problemas, é óbvio que ferramentas são sujeitas também a restrições técnicas comprometendo ainda mais o grau de satisfação alcançável pela ferramenta.

3. O processo de desenvolvimento de software

Nesta seção examinaremos algumas possíveis ferramentas, dando apoio às atividades do processo de desenvolvimento de software. Uma análise pormenorizada de ferramentas propostas e/ou existentes no mercado ultrapassa em muito o escopo deste artigo. Além disso é evidente que somente será possível avaliar caso sejam definidos critérios rigorosos de avaliação. É claro que tais critérios devem levar em conta os requisitos descritos na seção anterior. Devem levar em conta ainda o processo de geração e manutenção de representações (documentos, especificações, projetos, código fonte, etc,). Na figura 2 ilustramos este processo identificando as suas principais funções, insumos e resultados. Cabe salientar que o processo ilustrado é absolutamente geral, independendo de:

i - o modo de desenvolvimento ("bottom up", down" ou "ioio");

ii - o grau de inteligência mecanizada e apreensível por meios mecanizados ("knowledge based systems");

iii - a classe de software (suporte, embutido, aplicação).

Discutimos a seguir as diversas atividades do processo de desenvolvimento, dando, sempre que possível, exemplos de ferramentas mecanizadas apoiando estas funções.

Evolução esta função cria uma ou mais representações resultantes, a partir de uma representação de entrada caso exista. São exemplos de ferramentas:

i - sistemas de apoio à especificação, por exemplo PSL/PSA

ii - editores e formatadores inteligentes e projetados para o tipo de representação sendo criada, por exemplo editor/verificador de programas PASCAL

iii - tradutores capazes de converter a representação de entrada para a de saída por iniciativa própria (ex. compiladores) ou com apoio do usuário(ex. editores inteligentes).

Modificação esta função promove a alteração da representação de saída em função de solicitações de alteração pendentes e/ou de alterações realizadas na representação de entrada. Apesar de modificações serem muito frequentes, poucas são as ferramentas mecanizadas propostas para apoiá-la. São exemplo de ferramentas de apoio à modificação:

i - pacotes de programas dando suporte à gerência de configuração;

ii - sistema de apoio à reutilização de representações.

O pressuposto comum de que as ferramentas que apoiam a evolução apoiariam também a modificação é falacioso, uma vez que não interessa modificar toda a representação de saída mas somente a porção estritamente necessária. Torna-se necessário manter as "representações diferenciais" e também a "história de evolução/modificação". A representação diferencial descreve as porções alteradas por uma modificação efetuada na representação correspondente. Uma boa ferramenta deveria ser capaz de localizar quase automaticamente as porções a serem alteradas na representação de saída em função da representação diferencial. A história de evolução/modificação descreve as decisões efetuadas ao criar a representação de saída, possibilitando, pois, determinar as porções de representação de saída que deverão ser alteradas para acomodar uma solicitação de alteração e/ou uma representação diferencial.

reflexão esta função cria e/ou propõe alterações à representação de entrada em função de uma representação de saída alterada. Segue portanto o sentido inverso do processo normal de evolução e modificação. No caso ideal de desenvolvimento utilizando estratégias puramente descendentes("top dow") esta função não seria necessária. Porém a prática mostra que, via de regra o desenvolvimento não é puramente descendente. A prática mostra também que pode ser desejável integrar sistemas desenvolvidos sem a pretensão explícita de serem integrados.

Como no caso de modificação, também a reflexão carece de ferramentas de apoio. Na área de processamento comercial tem-se mostrado útil a instalação de um dicionário de dados corporativo, uma vez que este contribui para que os diversos sistemas sejam facilmente integrados.

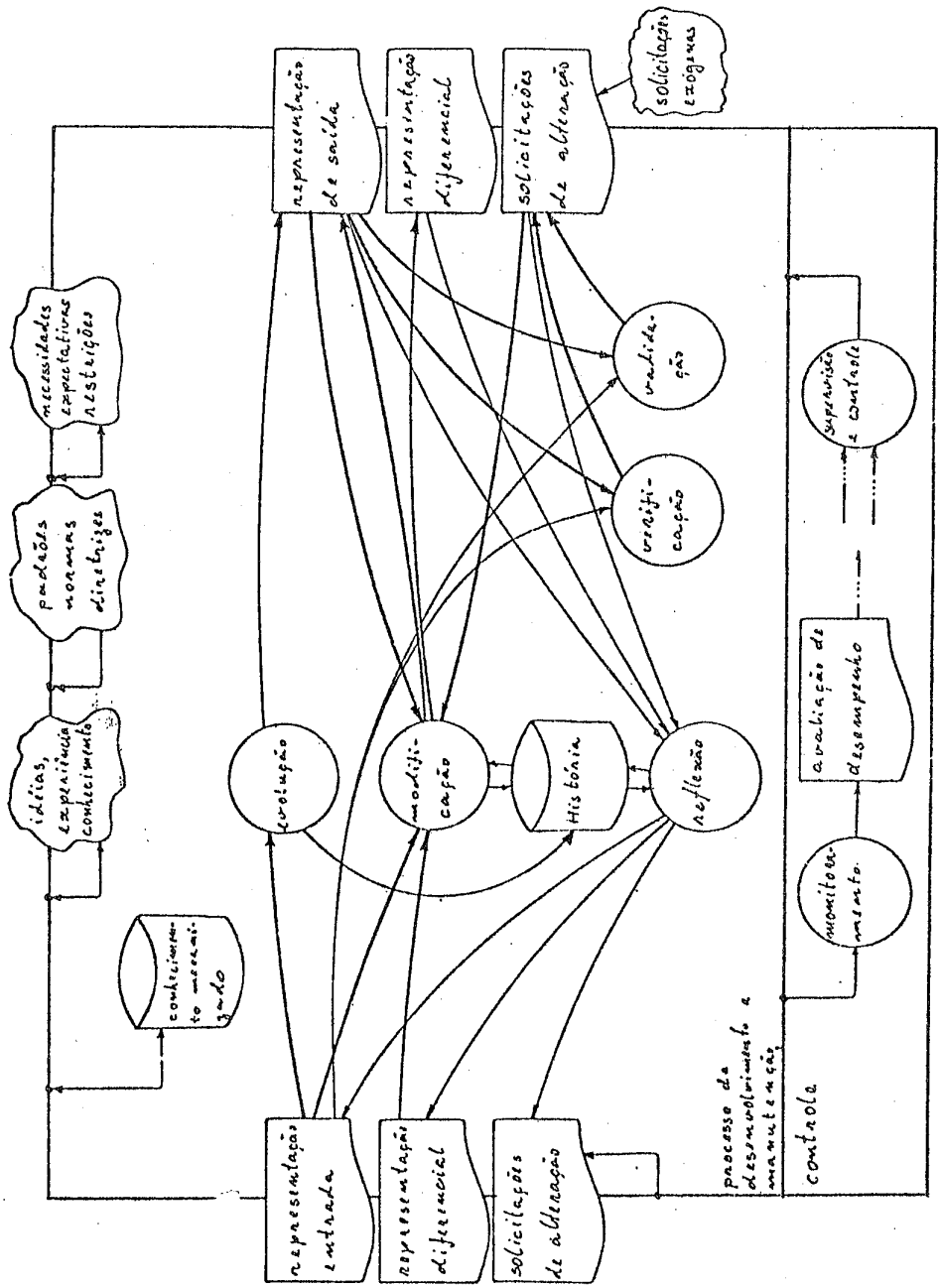

Figura 2 Atividades típicas do processo de desenvolvimento

verificação esta função controla a qualidade do resultado da evolução, de modificação e, por extensão, da reflexão. A verificação tem por objetivo assegurar que a representação de saída esteja em conformidade com a representação de entrada e com normas e padrões adotados. Exemplos de ferramentas são:

- i - geradores de dados teste
- ii - supervisores de teste
- iii - analisadores

validação esta função controla a qualidade do resultado da evolução e modificação com o objetivo de assegurar que este resultado

satisfaça às necessidades, expectativas e restrições do usuário e das interfaces do sistema em desenvolvimento. São exemplos de ferramentas.

i - geradores de protótipos

ii - simuladores

iii - analisadores semânticos

iv - dicionários de dados hierárquicos

No caso de sistemas embutidos em particular, é frequente ter-se que lançar mão de ferramentas. Por exemplo, ao desenvolver um monitor de determinado processo, torna-se necessário muitas vezes desenvolver um sistema simulador que simule o comportamento que o processo teria caso estivesse implementado. Desta forma pode-se testar condições de alto risco e/ou de difícil ocorrência, bem como pode-se repetir a ocorrência de determinadas condições dificilmente repetitíveis.

monitoramento

esta função coleta dados relativos ao desempenho do processo de desenvolvimento. O processo de desenvolvimento é, por sua vez, sujeito à supervisão e controle. Desejamos obter dados relativos ao desempenho do processo, à eficácia das ferramentas empregadas, à qualidade alcançável, etc. Com estes dados pode-se influir no processo de desenvolvimento com o intuito de otimizar e/ou corrigir o seu desempenho.

Também aqui existem poucas ferramentas, sendo o mais convencional o uso de formulários preenchidos (ex. "time sheet") pelos executores das diferentes funções. Infelizmente muito poucas ferramentas de apoio à evolução e à modificação foram projetadas para fornecer dados de acompanhamento.

Supervisão e controle esta função controla (gerencia) o processo de desenvolvimento. Como já foi mencionado, para obter um bom resultado é imprescindível uma boa gerência. A gerência se realiza através desta função. Mais uma vez temos que lamentar a escassez de ferramentas de apoio à supervisão e ao controle, exceto obviamente PERT e similares.

4. Conclusão

Este artigo procurou identificar propriedades que ferramentas devem possuir. Julgamos tal ser mais valioso do que simplesmente criticar e/ou comparar ferramentas, uma vez que, para ser objetivo, esta crítica e comparação necessita de critérios razoáveis e identificados de uma forma sistemática.

Vimos quais os requisitos básicos de ferramentas e examinamos as funções a que darão apoio. Com base nestes critérios e numa noção intuitiva das ferramentas existentes e/ou propostas na literatura podemos concluir:

a - o conjunto de ferramentas disponíveis forma um todo heterogêneo e desintegrado, impondo uma série de dificuldade de uso e de transição de uma etapa para outra.

b - a maioria das ferramentas apoia parcialmente a função a que se destina, contribuindo assim para dificultar a realização integral desta função, possivelmente forçando a sua realização parcial.

Uma possível razão para este quadro sombrio com relação a ferramentas é serem poucos os projetistas de ferramentas que se preocuparam em examinar o papel global destas ferramentas no contexto do processo de desenvolvimentos. O exemplo mais flagrante é a generalizada inexistência de suporte ao teste na maioria dos processadores de linguagem. Esta falta é indesculpável, uma vez que "dumps" simbólicos, perfis de execução, (ex. verificação da cobertura assertivas executáveis, etc.) todos são conhecidos há vários anos, antes mesmo do desenvolvimento de grande parte destes processadores de linguagem.

Especificamente com relação a sistemas embutidos cabe mencionar o projeto ADA que visa resolver diversos dos empecilhos mencionados através dos APSE e do projeto "Methodman". No entanto, também aí vemos uma série de inadequações por não estar sendo considerado com o rigor necessário o processo de desenvolvimento de software.

BIBLIOGRAFIA

Neste artigo optamos por não citar referências pois uma lista completa ou seria inócua (referenciariam algo que não interessa), ou seria excessivamente longa. Por outro lado uma lista parcial seria execessivamente tendenciosa e incompleta.