REQUISITOS DE AMBIENTES DE ENGENHARIA DE SOFTWARE

Daltro José Nunes
Departamento de Informática
Curso de Pós-Graduação em Ciência da Computação
Universiade Federal do Rio Grande Do Sul
90000-Porto Alegre-RS

RESUMO

Neste trabalho descrevem-se os principais requisitos que deve ter um ambiente de engenharia de software, a necessidade do investimento em pesquisas para melhorar certas áreas consideradas ainda fracas e que refletem nas potencialidades dos ambientes e, finalmente, mostra-se rapidamente a arquitetura de um ambiente cujo protótipo está sendo desenvolvido na UFRGS.

1 INTRODUÇÃO

Não é novidade que o processo de construção de software é complexo e que, como em qualquer atividade de construção, a disponibilidade de ferramentas de apoio aumenta a produtividade e melhora a qualidade do produto final.

Os sistemas de apoio ao processo de construção de software, denominados de "Software-Engineering-Environment-System" (SEES), são ambientes organizacionais e apoiados por computador que compreendem todas as atividades de Engenharia de Software.

Os métodos , linguagens, ferramentas, orientações, "checklisten", e o modelo organizacional usados não devem estar um ao lado do outro senão totalmente integrados.

2 RELAÇÃO CUSTO BENEFÍCIO DE UM SEES

Abaixo estão listadas as razões da necessidade do desenvolvimento de pesquisas com vistas a melhoria da produtividade de software.

- aumento substancial na procura de software
- baixa disponibilidade de engenheiros de software
- aumento das exigências sobre ferramentas de apoio
- custo reduzido de hardware

O custo estimado para aquisição de um SEES [5], aos preços de 1982, classificados em quatro grupos conforme seu poder de apoio, é mostrado abaixo.

	Número de ferramentas	Custo
SEES I	13	$ 35 000
SEES II	24 + Banco de dados	$200 000
SEES III	33 + Banco de dados	$300 000
SEES IV	46 + Banco de dados	$3 000 000

Em [1] foi listada uma série de fatores que influenciam a produtividade de software, entre eles, evidentemente, a introdução de um SEES no processo de produção de software. Verificados os custos estimados para aquisição de um SEES e a necessidade do aumento da produtividade de software, o benefício que um SEES pode trazer a uma organização é da ordem de 1,49, quando traduzido em fator de produtividade. Isto significa que mantidos os demais fatores constantes, o processo de desenvolvimento de um projeto, sem o uso de ferramentas de software, acarreta 1,49 vezes mais esforço, do que

quando desenvolvido com ferramentas de apoio.

Boehm [2] fez uma análise sôbre o aumento da produtividade de software e chegou aos seguintes resultados.

- O aumento da produtividade exige a execução de um programa integrado com vistas a uma melhoria nas sequintes áreas: ferramentas de software, métodos de software, ambientes de trabalho, formação de pessoal, motivação pessoal, re-uso de software.
- Um SEES deve integrar ferramentas de software e serviços gerais de escritórios (processador de textos, etc.).
- Um programa integrado para a melhoria da produtividade de software pode ser muito rendoso. O aumento de produtividade de um fator 2, em 5anos, e de um fator 4, em 10 anos, são possíveis de serem alcançados.
- O aumento da produtividade exige um esforço contínuo.
- O maior aumento da produtividade é alcançado com o re-uso de software.
- Cada participante do projeto de software deverá possuir sua própria estação de trabalho.
- É exigido do SEES uma interface de comunicação homem-máquina padrão.
- É necessário desenvolver estratégias para que os usuários aceitem facilmente a introdução de um novo SEES.
- As redes de comunicação de dados devem apoiar os ambientes de trabalho distribuídos.
- O uso de salas privativas para cada participante do projeto aumenta a produtividade de software.

Como foi mostrado acima, é necessário o aumento da produtividade de software. Os SEES, como mencionado, são também responsáveis pelo aumento da produtividade. Para atingir-se estes objetivos é necessário um programa integrado que envolva pesquisas nas áreas acima citadas. O aumento de produtividade de um fator de 400% em 10 anos seguramente amortizará o capital investido.

3 REQUISITOS DE SEES

Os sistemas SEES, para atingir fatores de produtividade razoáveis, devem preencher uma série de requisitos. Dentre eles citamos: integração; interface homem-máquina padronizado; completeza; uso amigável; expansibilidade; portabilidade.

Historicamente os SEES foram baseados na automação de métodos. Eles continham uma ferramenta ou um grupo de ferramentas e rodavam em máquinas diferentes. Não existia alguma troca de dados entre as ferramentas e muito menos entre as máquinas que continham ferramentas complementares. Assim, as ferramentas constituintes dos SEES formavam verdadeiras ilhas e o uso de uma ferramenta era acidental.

O primeiro passo para melhorar esta situação é estabelecer uma integração funcional entre as ferramentas, ou seja, colocar todas as ferramentas em uma única máquina.

O primeiro requisito de um ambiente de engenharia de software é a integração das ferramentas. Uma simples integração funcional, como mencionado acima, não é satisfatória. É necessária também uma

integração sintática entre as ferramentas. Isto significa que as ferramentas podem comunicar-se umas com as outras. O processo de comunicação entre as ferramentas podem ser classificados em: pipeline, banco de dados ou uma combinação dos dois anteriores. No processo pipeline uma ferramenta armazena seus dados em um arquivo que pode ser lido por outra ferramenta. Exemplo deste tipo de comunicação pode ser obtido através do sistema operacional UNIX. No processo de banco de dados, as ferramentas armazenam e leem seus dados de uma base de dados. Exemplos do uso deste tipo de comunicação são os sistemas APSE [6] e o sistema TRW [2].

O requisito dos SEES de conter uma interface de comunicação homem máquina homogênea visa facilitar o uso das ferramentas por parte dos usuários das ferramentas. Embora cada ferramenta possa ter sua própria interface que obedece a um critério comum, a melhor alternativa é definir uma interface comum para todas a ferramentas. Considerando que cada usuário venha a usar e dominar 10 a 20 ferramentas, o uso comum de todas as ferramentas de uma mesma interface vem facilitar o uso das mesmas.

Os ambientes de engenharia de software devem ser completos no sentido de que todas as atividades de todas as pessoas que participam do processo de produção de software devem ser apoiados através de recursos dos SEES. Isto, hoje, não é possível, razão pela qual fala-se em uma completeza mínima. Os recursos devem estar integrados semanticamente, ou seja, os recursos devem estar relacionados e sincronizados uns com os outros.

As atividades relativas ao processo de produção de software podem ser

divididas nas seguintes classes: produção, gerência e serviços de escritório.

O uso amigável de um SEES está fortemente relacionado com a aceitação do mesmo por parte do usuário. É importante tanto dispor de uma interface comum para todas as ferramentas como também de ferramentas amigáveis ao usuário. Entende-se por uso amigável a satisfação que o usuário tem em usar um SEES. Um SEES é amigável se ele contém os seguintes atributos: permite uso interativo, a interface é adaptativa (a interface adapta-se ou é adaptada ao usuário), contem componentes (ativos ou passivos) de ajuda, é tolerante a falha, possui a interface das ferramentas homogênea, é adaptável ao tipo de usuário (iniciante, usuário eventual e profissional), é eficiente, permite a integração de funções através de definições de macros e permite que cada usuário configure as ferramentas para o seu uso pessoal.

Considerando que o estado atual da arte não oferece ferramentas que cubram todas as atividades do processo de desenvolvimento de software, um SEES deve ter a capacidade de ser enriquecido com a introdução de novas ferramentas. As novas ferramentas podem ser introduzidas inicialmente na forma de protótipos e somente depois de obtidas experiências de aplicação é que se deve decidir pela sua produção. O SEES deve permitir também a agregação de ferramentas, ou seja, a partir de ferramentas pequenas, novas ferramentas, mais complexas, poderão ser construídas. Considerando que o SEES deve ser um sistema aberto, ele deve permitir que ferramentas construídas pelo próprio usuário possam ser integradas ao sistema.

Considerando que um SEES pode ter uma vida maior que a vida da máquina

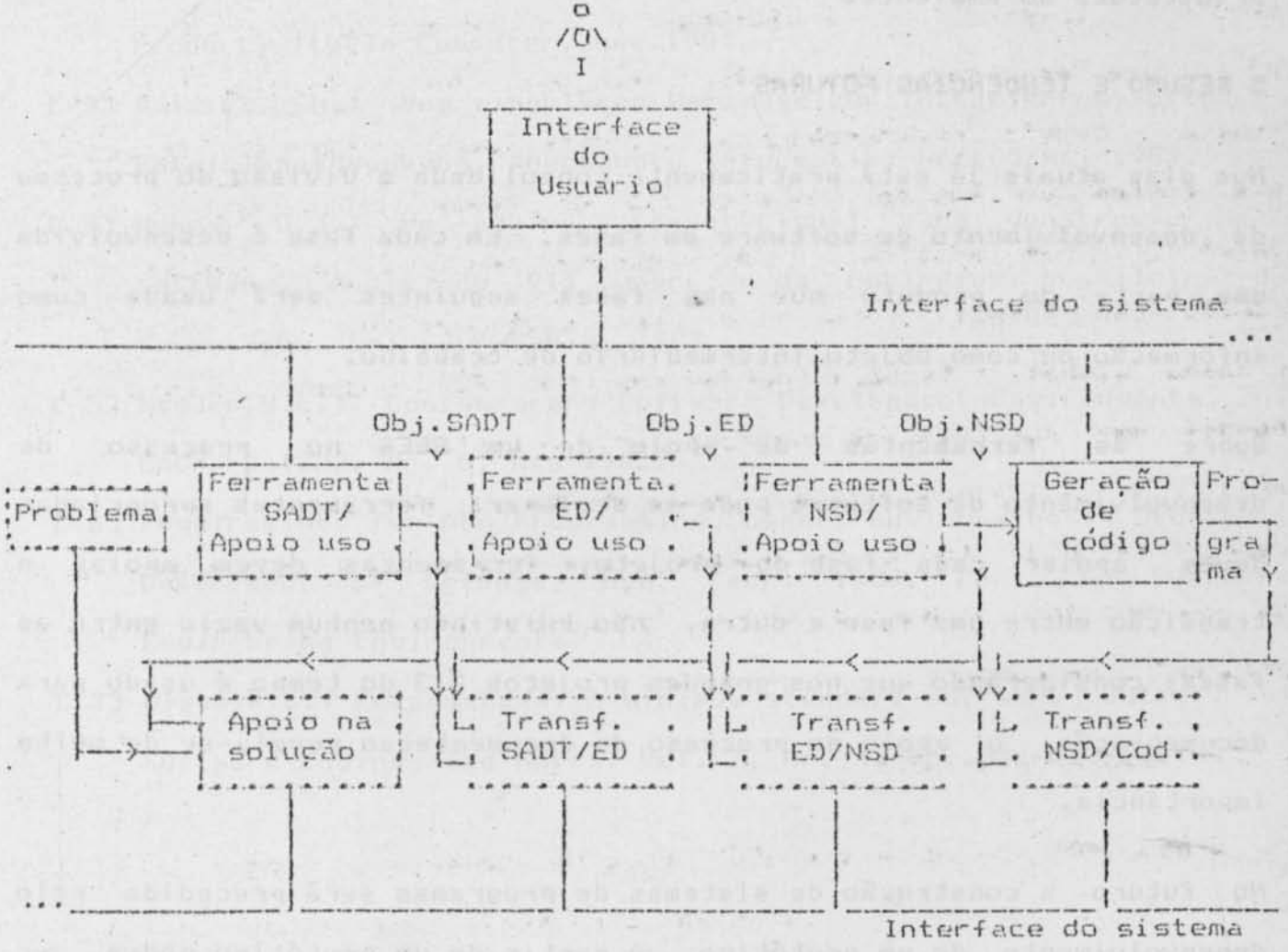
sobre a qual ele foi construído, ele deve poder ser transportado para outras máquinas. Uma maneira de concretizar este requisito é colocar em módulos separados aquelas funções que são dependentes da máquina.

Em [7] foi descrito o conjunto mínimo de ferramentas que deve compor um SEES. Em [3] foi mostrada uma tabela contendo 16 SEES e seus conjuntos de ferramentas, divididas em: ferramentas para a definição de requisitos, para o projeto, para a implementação, para o "Checkout", para a manutenção, para a gerência, para a documentação e para propósitos gerais. O sistema DoD APSE, com 16 ferramentas, é o maior de todos: para a fase de definição de requisitos contém um seguidor ("tracing") de requisitos e uma linguagem de requisitos. Ele possui uma ferramenta para apoiar o projeto. Para a implementação ele contém o compilador/assembler e um linker/loader. Para o "Checkout" ele contem um monitor de execução de comando, um simulador de interface, um analisador do programa fonte e um sistema de verificação formal. Para a manutenção ele possui uma referência cruzada global e temporizador/desempenho. Para o gerenciamento ele possui uma gerência de configuração, o controle de projeto e o relatório de falhas. Para a documentação ele possui um gerador de especificador de etapas. Finalmente, para propósitos gerais ele possui um gerenciador da base de dados, um editor de textos, uma "pretty printer" e um comparador de arquivos.

4 ARQUITETURA DE UM SEES

Na UFRGS desenvolve-se o protótipo de um ambiente de engenharia de software[4], que contém algumas das características citadas acima. A interface de comunicação com o usuário é comum a todas as ferramentas.

A linguagem de comando é traduzida em menus. Os objetos da engenharia de software como diagramas, estrutogramas, componentes, fluxo de dados, módulos etc. possuem representação gráfica e são exibidos em janelas especialmente abertas para este fim. A cada janela é associada uma ferramenta, denominada de ATO (Ambiente de Tratamento de Objeto), correspondente ao objeto exibido na janela. O ATO tem a função de, por determinação do usuário, realizar as operações necessárias sobre o objeto. O usuário pode operar simultaneamente vários objetos. A introdução ou remoção de ATO's é feita de maneira bastante simples.



A funcionalidade do ambiente está sendo testada com a introdução de três ferramentas: uma para a fase de requisitos SADT; uma para o projeto, diagramas NSD, e uma para a geração semi-automática de código Pascal. O ambiente assim descrito possui uma integração funcional, sintática e semântica. A integração semântica de recursos é realizada através de "verificadores" que analisam o processo de transformação de objetos. Dado que os métodos escolhidos pertencem a classe dos chamados métodos semi-formais, a verificação é realizada através de heurísticas de transformações. A figura acima mostra a arquitetura do ambiente.

5 RESUMO E TENDENCIAS FUTURAS

Nos dias atuais já está praticamente consolidada a divisão do processo de desenvolvimento de software em fases. Em cada fase é desenvolvida uma parte do produto que nas fases seguintes será usada como informação ou como objeto intermediário de trabalho.

Sobre as ferramentas de apoio de um SEES no processo de desenvolvimento de software pode-se declarar: ferramentas apropriadas devem apoiar cada fase do projeto; ferramentas devem apoiar a transição entre uma fase e outra, não existindo nenhum vazio entre as fases; considerando que nos grandes projetos 2/3 do tempo é usado para documentação, o apoio no processo de documentaçao revela-se de muita importância.

No futuro a construção de sistemas de programas será precedida pelo desenvolvimento de um protótipo. A partir de um protótipo podem ser também discutidas alternativas para o modelo de desenvolvimento de

software. Parece promissor um apoio mais acentuado na implementação a partir de protótipos, particularmente um apoio maior nas implementações a partir de especificações formais. Os sistema baseados em conhecimentos e os sistemas especialistas apoiarão melhor o produtor de software.

6 LITERATURA

[1] Boehm,B.W.: Software Engineering Economics. PRH 1981.

[2] Boehm.B.W., Penedo, M.H.Stukle, E.D., Williams, R.D.B, Pyster, A.B.: A Softwre Development Environment for improving Productivity.In Computer,June 1984.

[3] Balzert, H.: Vom singulären Werzeuge zur Integrierten Software-Entwiclungsumgebung. Angewandte Informatik, Heft 5/Mai 1987.

[4] Nunes, D.J.: Um ambiente computacional para construção de software. Anais do VII Congresso da Sociedade Brasileira de Computação, Salvador-Bahia, 1987

[5] Howden,W.E.: Contemporary Software Development Environments. In: CACM, Vol.25, Nr. 5, May 1982.

[6] Requirements for ADA Programming Support Environments. Stoneman, Department of Defense, USA, Febr. 1980. Conf. on Software Engineering Environments.

[7] Glass,R.L.: Recommended: A minimum standart software toolset. In: Software Engineering Notes, Vol. 7, Nr. 4, Oct.82, p. -13.