UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
COMISSÃO DE COORDENAÇÃO DAS ATIVIDADES
DE PROCESSAMENTO ELETRÔNICO – CAPRE
COORDENAÇÃO DO APERFEIÇOAMENTO DE PESSOAL
DE NÍVEL SUPERIOR – CAPES

III SEMINÁRIO SOBRE DESENVOLVIMENTO INTEGRADO DE SOFTWARE E HARDWARE

Data: 26 a 29 de julho de 1976

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
COMISSÃO DE COORDENAÇÃO DAS ATIVIDADES
DE PROCESSAMENTO ELETRÔNICO – CAPRE
COORDENAÇÃO DO APERFEIÇOAMENTO DE PESSOAL
DE NÍVEL SUPERIOR – CAPES

III SEMINÁRIO SOBRE DESENVOLVIMENTO INTEGRADO DE SOFTWARE E HARDWARE

PROJETO 'MOSAICO'

DEFINIÇÃO DO PROJETO

por

Arndt von Staa
Departamento de Informática
Pontifícia Universidade Católica
Rio de Janeiro

RESUMO:

Descreve-se em linhas gerais o projeto MOSAICO definição e objetivos. Caracterizam-se produtividade e qualidade de programas. Definem-se modularidade e desenvolvimento por refinamento sucessivo. Descreve-se o estado do projeto em maio de 1976.

PALAVRAS CHAVE E FRASES CHAVE:

Modularidade, Compiladores, Linguagens, Qualidade de Programas, Produtividade de Programação, Laboratório, Ensino, Pesquisa, Implementação.

MOSAICO é um projeto que visa desenvolver um ambiente de programação com os seguintes objetivos principais:

a. MOSAICO deverá facilitar atingir-se altos índices de produtividade de programação;

b. MOSAICO deverá facilitar a criação de programas (sistemas)de alta qualidade.

Como consequência desses dois objetivos, tem-se que MOSAICO pretende dar soluções tanto a problemas econômicos quanto a problemas técnicos que ocorrem na atividade de programação e na de condução de projetos de programação.

Além dos objetivos principais citados acima, MOSAICO tem os seguintes objetivos secundários:

a. o projeto MOSAICO deverá servir como laboratório para pesquisa, desenvolvimento e aplicação de tecnologia de programação;

b. o projeto MOSAICO deverá servir como laboratório para pesquisa, desenvolvimento e aplicação de técnicas de administração de projetos de "software".

Como consequência dos objetivos secundários, o projeto MOSAICO produzirá um volume elevado de documentos tais como: relatórios de desenvolvimento, relatórios de pesquisa, teses, etc. Dessa forma o projeto MOSAICO caracteriza-se tanto como um projeto de desenvolvimento, quanto como uma ferramenta para o ensino e a difusão de metodologias de programação e condução de projetos de programação.

2. CARACTERIZAÇÃO DE PRODUTIVIDADE

É consenso geral que não existe atualmente uma medida precisa de produtividade de projetos de programação. Apesar disso nós procuraremos dar uma proposta de quantificação, cujo realismo deverá ser estabelecido no decorrer do projeto MOSAICO. É de nosso entender que produtividade deverá medir:

a. a produção de código (programas com comentários)

b. a produção de texto (especificações, manuais, etc)

c. os testes e certificações do programa produto (relação de testes, resultados, quantificações de confiabilidade e robustez, etc)

As medidas têm pesos diferentes e coeficientes corretores cujos valores (faixas de valores) deverão ser estabelecidos durante o decorrer do projeto MOSAICO.

As medidas de produtividade são posteriores à conclusão dos projetos, subprojetos ou etapas. Isto indica que os projetos deverão ser particionados para que se possa ter uma visão de produtividade no decorrer do projeto. Isto indica também que se deverá medir a produtividade de confecção de ferramentas apesar de que, como veremos mais tarde, essas ferramentas não participem explicitamente do cômputo geral de produtividade do projeto.

A medida PLCC - produção de linhas de código corretas - mede a produtividade da atividade de programação propriamente dita. Esta medida é fortemente influenciada pela complexidade - dificuldade de criar o programa - do programa a ser feito. Além disso o número de linhas de código que foram criadas e que não figuram no programa produto não deveriam entrar como fator explícito na produtividade de programação. Isto sugere uma medida da seguinte forma:

$$PLCC = \frac{C_1 * LCC}{DTP}$$

onde: C_1 é um fator de correção para complexidade

LCC é o número de linhas corretas existentes no programa produto, ou seja, LCC não contém as linhas de código descartadas

DTP é a duração total do projeto (sub-projeto)

Note que nesta medida não entra como fator explícito a criação de ferramentas. Isto porque o código gerado para criar as ferramentas não é contabilizado na medida LCC. Isto é desejável uma vez que a influência de uma ferramenta deve ser de tal forma que o índice de produtividade com que o programa produto é criado seja aumentado, de outra forma a utilidade da ferramenta é questionável.

A medida PLTC - produção de linhas de texto corretas - mede a produção de manuais de especificação, de usuário e qualquer outra documentação que deverá acompanhar o programa produto. Essa medida é fortemente influenciada pelo grau de detalhe que é desejado na documentação final (qualidade de docu -

mentação desejável). Da mesma forma que em programação, serão geradas linhas de texto que não aparecerão na versão final da documentação. Isto sugere então uma medida da forma:

$$PLTC = \frac{C_2 * LTC}{DTP}$$

onde: C_2 é o fator de correção para detalhe de documentação
LTC é o número de linhas de texto no documento final
DTP é a duração total do projeto

A medida CLC - certificação de linhas de código - mede o esforço de correção de um programa durante a fase de testes. Idealmente o programa deveria ser criado sem que tenha que ser modificado durante a fase de testes. Dessa forma a produtividade de testes deveria ser inversamente proporcional ao número de linhas alteradas, introduzidas e/ou retiradas do programa entregue à fase de testes relativo ao programa produto liberado. Convém notar que as linhas de código que servem como ferramental para os testes não são consideradas como retiradas no programa produto, mas sim como comentários que constituem o programa produto. Os testes são influenciados pela complexidade e pelo grau de confiabilidade e robustez desejado no programa produto. Isto sugere então uma medida da forma

$$CLC = \frac{C_3 * LCC}{DTP * (p_1*LA+p_2*LR+p_3*LI+1)}$$

onde: p_1, p_2, p_3 são pesos de homogeneização para as medidas de modificação do programa
LA é o número de linhas alteradas
LR é o número de linhas retiradas
LI é o número de linhas introduzidas
C_3 é um coeficiente de correção para a qualidade do programa produto
LCC é o número de linhas corretas no programa produto final
DTP é a duração total do projeto

Resumindo, temos então como uma proposta para a medida de produtividade:

$$P = f_1 * PLCC + f_2 * PLTC + f_3 * CLC$$

onde f_1, f_2 e f_3 são pesos que relacionam as diversas medidas isoladas.

3. CARACTERIZAÇÃO DE QUALIDADE

A medida de qualidade também é subjetiva e muito menos quantificável ainda. Por esta razão não procuraremos dar uma quantificação para qualidade de programa produto. Os fatores que mais influência têm sôbre a qualidade do programa produto são:

a. exatidão - o programa produto deverá fazer tudo o que foi especificado. A exatidão é uma medida de correção teórica. Isto implica em MOSAICO possuir compiladores corretos e definições precisas das sintaxes e semânticas das linguagens que compõem o sistema MOSAICO.

b. confiabilidade - o programa produto deverá comportar-se adequadamente com relação a todos os dados de entrada corretos (permitidos). A confiabilidade difere da exatidão por ser uma medida experimental. Além disso a implementação introduz alterações na especificação, alterações essas que deverão ser mencionadas explicitamente e que deverão estar dentro da tolerância especificada e controlada pela exatidão. Isto implica em MOSAICO prover muitos recursos de teste e verificação de programas.

c. robustez - o programa produto deverá ser insensível a dados errados quando isto for possível ocorrer. Isto implica em MOSAICO ter capacidade de tratamento de excessões.

d. eficiência - o tempo de execução de programas produzidos dentro do ambiente MOSAICO não deverá ser maior que o tempo de execução necessitado por programas equivalentes produzidos por compiladores comerciais de boa qualidade. Isto implica em que as linguagens existentes dentro de MOSAICO serem pouco complexas e serem bastante amenas à otimização.

e. legibilidade - o programa fonte deverá ser facilmente compreendido mesmo por programadores que não estiveram envolvidos no desenvolvimento deste programa. Isto implica em MOSAICO dever ter uma sintaxe simples e elegante.

f. manutenibilidade - o programa fonte deverá ser facilmente modificável para que possa ser adaptado a novas condições. Isto implica em MOSAICO permitir a concentração em um único local de todos os ítens logicamente correlacionados que poderão vir a ser afetados por uma modificação qualquer.

4. PRINCÍPIOS PRÉ-DEFINIDOS

O projeto MOSAICO partirá do princípio de que os seguintes dois ítens permitem atingir-se os objetivos principais do projeto:

a. a linguagem definida por MOSAICO deverá permitir a programação modular.

b. a administração do projeto MOSAICO será feita através de refinamentos sucessivos.

Apesar dos dois ítens acima serem assumidos dogmáticos para o projeto, daremos a seguir breves justificativas para cada um deles.

A modularidade deve permitir o aumento da produtividade por permitir o reaproveitamento de módulos já existentes e por permitir a criação de linguagens de nível muito alto. A modularidade deverá aumentar a qualidade do produto por permitir uma verificação mais detalhada de cada componente do produto final. Finalmente a modularidade deve ser propícia para o desenvolvimento de programas através de refinamentos sucessivos, por cada um dos módulos poder ser visto como um nodo de refinamento individual.

A administração por refinamentos sucessivos deve aumentar a produtividade por forçar uma constante observação do objetivo a alcançar e por eliminar grande parte de tentativas infrutíferas. Essa forma de administração deverá auxiliar também no aumento da qualidade do produto por permitir maior exatidão na definição de cada nível de refinamento

5. DEFINIÇÃO DE MODULARIDADE

Um módulo de programa é uma porção de texto de programa tal que o escôpo de todos os nomes textuais ou está completamente contido por este módulo, ou completamente contém este módulo. Uma linguagem de programação é dita permitir a programação modular se e somente se ela permitir a confecção de módulos com as seguintes características:

a. construtibilidade - a propriedade de se poder criar módulos e/ou programas a partir de vários módulos independentes, possivelmente pré-compilados.

b. não interferência sintática - é a propriedade que permite combinar-se módulos, através de composição ou chamada, sem que seja necessária a mudança de texto nos módulos combinados.

c. independência do contexto semântico - é a propriedade que permite a composição de módulos garantindo a integridade de funcionamento de todos os módulos do composto, mesmo que haja mudanças no ambiente externo de execução de um módulo. Devido a essa propriedade os módulos podem comunicar-se somente através de associação de parâmetros não globais.

d. generalidade de dados - a propriedade que permite módulos comunicarem informação entre si através de tipos abstratos (tipos definidos em termos do que fazem e não como fazem) independentemente das possíveis implementações.

e. transparência - é a propriedade de módulos serem conhecidos pelo que fazem e não como o fazem. Devido a essa propriedade, torna-se proibitivo o uso de parâmetros que possam ter influência direta e explícita sobre o funcionamento do módulo.

f. baixo acoplamento - é a propriedade de módulos fazerem uso efetivo de todos os parâmetros que são apresentados para intercomunicação.

g. força elevada - é a propriedade que estabelece que todas as funções de um módulo estão fortemente interrelacionadas logicamente, e que todas as funções fortemente interrelacionadas aparecem em um módulo.

6. DEFINIÇÃO DE REFINAMENTO DE ESPECIFICAÇÃO

Admite-se a possibilidade da definição global de um projeto. Admite-se, também, que projetos poderão ser particionados em vários subprojetos independentes, cada um desses subprojetos tornando-se por sua vez um projeto. Obtém-se dessa forma uma árvore, possivelmente um grafo acíclico, em que cada nodo é um projeto e cada conjunto de filhos de um nodo é o conjunto de subprojetos que constituem o projeto pai.

A especificação mais atualizada de um projeto é dada portanto pelo arco que se origina na raiz e termina no nodo correspondente ao projeto em questão. Além disso, cada nodo sucessor nesse arco é definido mais precisamente que o nodo antecessor nesse mesmo arco. A fronte da árvore de especificação poderá ser vista então como sendo a especificação corrente mais precisa do projeto global.

O princípio do refinamento sucessivo pode ser aplicado tanto à geração de código de programas, quanto à geração de rotinas administrativas particulares, bem como quanto à geração de texto, especificações, etc. No projeto MOSAICO este princípio será adotado universalmente para todas as atividades a serem desenvolvidas durante o projeto.

À medida que as especificações vão sendo detalhadas, várias rotinas de controle deverão ser verificadas. Estas rotinas são:

a. especificação completa de um nodo - cada nodo sucessor deverá ser especificado completamente antes que trabalhos em subprojetos associados a esse nodo sucessor possam ser iniciados. Isto quer dizer que somente será possível a passagem de um nodo antecessor para um nodo sucessor, se o nodo sucessor tiver sido completamente especificado no nodo antecessor, ou seja, o esforço desprendido em cada nodo é a especificação completa de um ou mais sub-projetos e/ou a geração de texto ou código terminal.

b. aderência às especificações - sempre que o trabalho em um nodo sucessor tiver sido terminado, o resultado tem que ser verificado quanto à aderência às especificações dadas pelo nodo antecessor correspondente.

c. confiabilidade - para cada nodo deverá ser feito uma verificação experimental e/ou formal de que o nodo comporta-se adequadamente para todos os conjuntos de dados e/ou perguntas permissíveis.

d. robustez - para cada nodo deverá ser feito uma verificação experimental e/ou formal de que o nodo não produz resultados imprevisíveis para conjuntos de dados e/ou perguntas não permissíveis. A confiabilidade e a robustez são parâmetros de especificação.

e. flexibilidade - para cada nodo deverá ser estabelecida uma medida de modificabilidade e adaptabilidade. Essas propriedades são necessárias uma vez que o desenvolvimento de sub-projetos poderá indicar inadequações de especificações, bem como, durante a fase de manutenção, poderão ser sugeridas modificações em porções específicas do projeto global.

Convêm notar que a especificação de um nodo deverá indicar máximos para todas as possíveis restrições, por exemplo tempo de resposta máximo, volume de memória mínimo, custo máximo, etc. Dessa forma, poder-se-á verificar em nodos sucessores se estas restrições máximas não foram ultrapassadas.

Espera-se que a aplicação de refinamentos sucessivos para administrar o desenvolvimento do projeto MOSAICO contribua com os seguintes fatores de incremento de produtividade e qualidade:

a. minimização da propagação de erros de projeto. Devido ao controle por nodo e na passagem de um nodo antecessor para um nodo sucessor espera-se que possíveis erros sejam notados durante uma dessas duas atividades e não muito mais tarde durante o desenvolvimento, ou teste, de um sub-projeto distante de mais de um passo de especificação.

b. maximização da reutilização de trabalhos anteriores - uma vez um nodo tendo sido especificado completamente, pode-se verificar se existe um outro nodo com uma especificação compatível no grafo de refinamento. Dessa forma poder-se-á criar um grafo acíclico de refinamentos.

c. minimização do esforço burocrático - sempre que uma sistemática de controle é estabelecida, um certo volume de burocratização é estabelecido (por exemplo preenchimento de formulá

rios, criação de normas e controles de aderência a essas normas). Com o controle sendo limitado estritamente a nodos individuais e à passagem de um nodo a um nodo sucessor imediato, espera-se que o volume do esforço burocrático seja minimizado, e em particular que seja evitado o desenvolvimento de uma burocracia para controlar outra burocracia.

7. ESTADO ATUAL DO PROJETO

Atualmente, maio de 1976, está sendo definida uma ferramenta para o auxílio na implementação de compiladores - COMCOM2. Esta ferramenta constará de gerador de filtro léxico de gerador de reconhecedor sintático (gramáticas de dois níveis) rotinas de auxílio à geração de código e de suporte de ambiente de execução - descritores de estruturas. Espera-se ter estas ferramentas implementadas até dezembro de 1976.

Foi definido um manual de especificação de programas que está sendo utilizado na indústria e está em fase de revisão e crítica como esforço de um curso de engenharia de software em andamento neste semestre. Espera-se ter uma versão depurada final até final de julho de 1976.

Foram propostas as seguintes teses de mestrado:

a. experimentação com tipos abstratos que está em fase de conclusão.

b. definição da organização da informação de interface de módulos e definição das operações sobre esta informação. Esta tese encontra-se em progresso devendo estar concluida até final de dezembro.

Finalmente está em fase de conclusão um trabalho de pesquisa sobre tratamento de excessões.

O fundamento teórico do projeto MOSAICO está contido em [v.Staa 74]. Este trabalho não define uma linguagem, nem tampouco fornece todos os detalhes necessários para o desenvolvimento do projeto MOSAICO. Os detalhes serão fornecidos à medida que o projeto for avançando e sentir-se a falta deles.

Além do trabalho acima mencionado uma série de outros trabalhos servirão de respaldo para o projeto MOSAICO. Estes trabalhos estão mencionados na seção de referências bibliográficas.

BIBLIOGRAFIA E REFERÊNCIAS BIBLIOGRÁFICAS

Casanova, M.A.; Simões, P.C.; v. Staa, A.
Construção de Compiladores, PUC/RJ, Departamento de Informática, Relatório Técnico (Livro) em progresso.

Cowan, D.D.; Lucena, C.J.P.; v. Staa, A.
On the concept of modules in programming systems.
University of Waterloo, Computer Science Dept., CS-76-05; Jan 1976.

Azevedo, F.L.F.
Necessidade e operações de informação de interface de módulos; PUC/RJ, Departamento de Informática, tese de mestrado em progresso.

Gries, D.
Compiler Construction for Digital Computers; John Wiley & Sons; 1971.

Hoare, C.A.R.
'An Axiomatic Basis for Computer Programming';
Communications of ACM, vol 12, nº 10; Oct 1969; pp 576-580, 583.

Hoare, C.A.R.
'Hints on Programming Language Design' SIGACT/SIGPLAN Symposium on Principles of Programming Languages; Oct 1973.

Knuth, D.E.
'Structured Programming with go to Statements'; Computing Surveys ACM; vol. 6; nº 4; Dec 1974, pp 261-302.

Lauterbach, C.; v. Staa, A.
On the Tree Organization of Abstract Data Types; PUC/RJ; Deptº de Informática; MCSCA - 5/76 em impressão.

Lucena, C.J.P.
Manual para o Projeto e Documentação de Sistemas de Programação; PUC/RJ; Deptº Informática; Relatório Técnico em revisão.

Lucena, C.J.P.; Cowan, D.D.

Toward a System's Environment for Computer Assisted Programming; University of Waterloo, Computer Science Dept., CS-76-06; Jan 1976.

v. Staa, A.

COMCOM, Compilador de Compiladores; PUC/RJ, Rio Datacentro, Grupo de Aplicações, 1970.

v. Staa, A.

Data Transmission and Modularity Aspects of Programming Languages; University of Waterloo, Computer Science Dept., CS-74-17; Oct 1974.

v. Staa, A.

On the Design of Languages for a Disciplined Use of Pointers, PUC/RJ, Deptº Informática MCSCA - 2/76, Dez 1975.

v. Staa, A.

On Monitoring the use of Data, PUC/RJ, Deptº Informática, MCSCA-3/76, em impressão.

v. Staa, A.

Generator Functions in Modular Programming, PUC/RJ, Deptº Informática, MCSCA- 4/76 em impressão.

v. Staa, A.

COMCOM2 - Sistema de suporte à construção de compiladores: Definição; PUC/RJ; Deptº Informática; Relatório Técnico em preparação.

v. Staa, A.; Lucena, C.J.P.

Implementação com generalidade, da transmissão da estrutura de dados entre módulos de programas. IBM, Revista Brasileira de Sistemas, vol. 1, nº 1, a aparecer.