Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

ESPECIFICAÇÃO E VERIFICAÇÃO DE SISTEMAS AUTOMATIZADOS, UMA AVALIAÇÃO DO ESTADO DA ARTE

Ana Regina Cavalcanti da Rocha

Arndt von Staa

Departamento de Informática
Pontificie Universidade Católica do Rio de Janeiro
Rua Marques de Sao Vicente 209
Garea, Rio de Janeiro, Brasil

1. INTRODUÇÃO

Este trabalho tem como objetivo definir forma e conteúdo de especificações de sistemas automatizados. Parte-se de uma sistematização das características que determinam a qualidade de sistemas automatizados, procurando-se estabelecer critérios de criação e verificação de especificações visando o desenvolvimento racional de sistemas de elevada qualidade. Com isto pretende-se conseguir especificações que sejam realmente a base e o ponto de apoio para o desenvolvimento de sistemas automatizados de elevada qualidade, bem como espera-se estabelecer princípios para continuado controle de qualidade.

Qualidade de software é um problema que cada dia adquire maior importância. O custo de produção de software é muito alto e os resultados desta produção nem sempre são os desejados. Projetos se atrasam, estouram orçamentos, não atingindo os objetivos almejados. Surge então o conceito de integridade de sistemas.

Entendemos por sistema íntegro um sistema que faz as coisas certas e as faz quando são necessárias. Integridade de sistemas é um termo bastante amplo e subentende vários outros. Assim sendo, podemos dizer que sistemas íntegros possuem determinadas qualidades. Além disso, é reconhecido também, que o grau de integridade depende da aplicação, devendo

porém, ser determinado antes de partir-se para o desenvolvimento.

Dentre várias classificações destacamos a da "American Federation of Information Processing Societies", definida durante um seminário em 1977 [EDP ANALYZER DEZ 79]. Esta classificação estabelece que um sistema, íntegro deve ser:

- disponível: pronto para servir os usuários quando estes o desejem
- apropriado: produzido de modo que faça as coisas certas
- limitado: possui capacidade operacional e/ou funcional restrita, de forma a fazer apenas o que se espera que faça
- correto: o que faz, faz corretamente
- prognosticável: sempre fazendo as coisas da mesma maneira
- oportuno: faz as coisas no tempo certo, fornecendo resultados quando estes são necessários
- manutenível: não perde a integridade ao serem feitas correções ou ampliações, e,
- auditável: construído de modo que auditores possam continuamente verificar a integridade do sistema

Embora a necessidade de integridade seja um problema genérico para qualquer tipo de sistema, esta necessidade varia conforme o tipo de sistema automatizado. Se considerarmos, por exemplo, um sistema A para controle de tráfego aéreo e um sistema B para processamento comercial, vemos claramente que um erro no sistema A pode ter conseqüências bastante mais desastrosas do que um erro no sistema B. Além disso, para alguns sistemas há interesse em cuidar especialmente certos aspectos de integridade, enquanto para outros sistemas serão outros os aspectos a destacar. Mas qualquer que seja o nível de integridade desejado e os aspectos que se queiram destacar, é fundamental que haja uma atenção especial por parte dos desenvolvedores do sistema já desde do início do desenvolvimento, uma vez que integridade não é um fator gerado expontaneamente, mesmo quando o grupo de desenvolvedores é de alto nível.

Mas se por um lado reconhecemos a necessidade de obtermos níveis apropriados de integridade para cada sistema concreto, por outro lado defrontamo-nos com a dificuldade de produzirmos (ou construirmos) sistemas íntegros. Que fazer, então, para vencer as dificuldades que se apresentam ao desenvolver-se sistemas íntegros?

É fato conhecido que grande parte dos problemas com sistemas estão relacionados a erros, omissões e inadequações na definição dos requisitos e especificações. E o que é mais grave, estes erros muitas vezes tardam demasiado em serem detectados. Isto se torna extremamente sério se observamos que o custo de um erro aumenta de maneira dramática à medida que aumenta a distância no tempo entre a época em que o erro é cometido e a época em que é efetivamente detectado. Como proceder então para detectar erros, omissões e inadequações o mais cedo possível?

2. O QUE SÃO ESPECIFICAÇÕES

Qualquer produto de software tem sua origem a partir de um conceito existente na mente de alguém. Este conceito geralmente pode ser realizado por um conjunto muito grande de programas alternativos de implementação. No entanto apenas algumas destas alternativas têm interesse prático. Esta situação é mostrada na figura 1. Neste caso o conceito é estabelecido informalmente e qualquer que seja a técnica utilizada para verificar a correção do programa, o resultado da aplicação da técnica também só poderá ser estabelecido em termos informais.

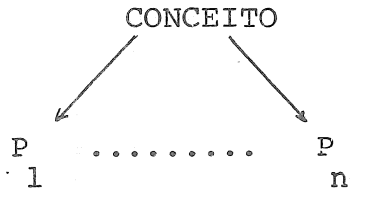

fig 1 - Um conceito e todos os programas que o implementam corretamente

Para atuar de maneira mais formal, deve-se interpor entre o conceito e os programas uma especificação. A especificação descreve o conceito de uma maneira formal e a correção do programa é mostrada provando-se ser ele equivalente à especificação. Neste caso as especificações também podem ser satisfeitas por uma classe de programas (figura 2) [LISKOV 77]. Também o conceito poderá ser realizado por diversas especificações alternativas.

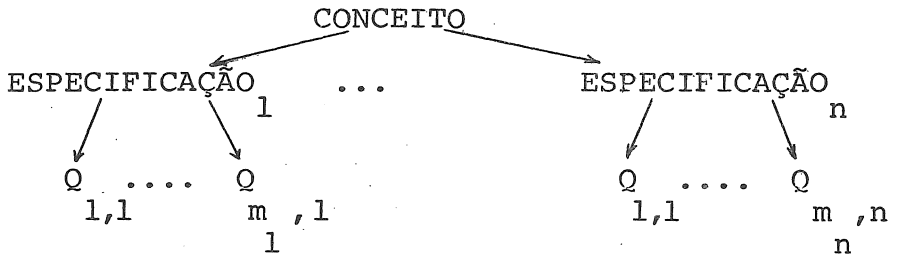

fig. 2 - Um conceito, suas possíveis especificações formais e todos os programas que podem ser provados equivalentes a estas especificações

Assim sendo uma especificação descreve o que faz a execução do programa ou do sistema, sem porém fixar como o faz. Pode ser vista também, como um contrato entre o desenvolvedor do produto de software e seu usuário [DIJSKTRA 77].

Outros autores vêem especificações como modelos. Consideram que ao escrever especificações estamos tentando transmitir a outros modelos de nossas imagens mentais. Modelos podem ser definidos da seguinte maneira: M é um modelo de um assunto A com um conjunto de questões Q, se M pode ser usado para responder questões em A com tolerância T. Para que se possam escrever especificações razoáveis são necessários vários e diferentes modelos. Estes modelos terão diferentes conjuntos de questões, pontos de vista e objetivos [HENINGER 79].

Resta o problema de verificar qual das especificações alternativas melhor resolve o problema em questão. Voltou-se então à origem, uma vez que, dada a definição informal do problema, a verificação da especificação também será informal. Podemos, porém atacar o problema de outra forma, definindo os requisitos e as restrições a serem atendidos pelo sistema e verificar as especificações com relação a estes requisitos e restrições. Não teremos uma solução formalmente verificável, porém teremos soluções cujos desvios do ideal serão ao menos toleráveis.

Através da confrontação de diversos modelos poderemos verificar o grau de confiabilidade destas restrições com relação aos problemas reais a serem resolvidos.

3. CONTEÚDO DAS ESPECIFICAÇÕES

Especificações surjem em duas modalidades:

a. especificação de definição (especificação de requisitos e funcional) onde é definido o problema

b. especificações de projeto (especificação lógica e física) onde são especificados a forma de resolver o problema e os componentes do sistema "solução"

Uma especificação de definição deve fornecer ao usuário todas as informações que ele necessita e nenhuma informação a mais. Assim sendo, a especificação de definição de um sistema deve consistir numa descrição do sistema proposto em termos dos problemas a serem resolvidos, indicando quais as funções pretendidas para o sistema, bem como as regras e restrições que deverão governar seu comportamento. Estas especificações também impõem restrições econômicas e de desempenho. Especificações de definição são então descrições do sistema de software orientadas para o problema.

Especificações de projeto descrevem o sistema em termos da solução proposta. Já não é uma descrição orientada para o problema e sim uma descrição orientada para a implementação [RIDDLE 78].

Uma especificação deve determinar:

(1) os requisitos de resultado, isto é, o que o produto deve fazer. Para isto deve responder às seguintes perguntas:
- faz o que eu quero que faça?
- o faz sempre?
- está disponível sempre que necessito?

(2) os requisitos de desempenho, isto é, uma definição dos padrões de qualidade e limitações da solução. Para isto deve responder às seguintes perguntas:
- executa de forma eficiente?
- utiliza recursos de forma eficiente?
- produz resultados em tempo hábil?

(3) os requisitos de utilizabilidade, isto é, as interfaces com o ambiente em particular com os usuários, operadores etc. Para isto deve responder às seguintes perguntas:
- sei operar?
- sei interpretar os resultados?
- sei alimentar o sistema com dados?
- sou capaz de corrigir erros?
- é protegido /seguro?

(4) os requisitos de mensurabilidade, fornecem elementos para a contínua avaliação. Para isto deve responder às seguintes perguntas:
- sou capaz de medir?
- sou capaz de auditar?

- sou capaz de testar?
- sou capaz de detectar a ocorrência de erros ou inadequações?

(5) os requisitos de evolução, fornecem elementos indicativos com relação à manutenção e à durabilidade do sistema. Para isto deve responder às seguintes perguntas:

- posso utilizá-lo em outro ambiente?
- posso interligar com outro sistema?
- posso expandir as funções?
- posso alterá-lo, adaptá-lo?

Especificações são produzidas ao longo do ciclo de vida do sistema automatizado. O processo utilizado é o de refinamentos sucessivos, partindo da visão mais abrangente (especificação de requisitos) para a mais detalhada (especificação física). De uma forma geral temos:

a- especificação de requisitos: - define a interface do sistema com o ambiente - resultados requeridos, quais são os usuários, restrições de solução

b - especificação funcional: - define os componentes funcionais e as interfaces entre eles, além de refinar e rever a definição da interface com o ambiente. Não existe preocupação alguma com possíveis modos de resolver o problema

c - especificação lógica: - é produzida uma especificaçao dos componentes e de suas interfaces, onde estes componentes dão uma solução técnicamente viável ao funcionalmente especificado, atendendo aos requisitos e às restrições

d - especificação física: - em que cada componente é especificado ("design") de forma a satisfazer à especificação lógica

4. COMO PRODUZIR ESPECIFICAÇÕES

4.1. Qualidades que deve ter uma especificação

Embora seja reconhecida a importância das especificações para que o sistema a ser desenvolvido venha a ser íntegro, o simples fato de que existam especificações não é suficiente para que sejam realmente úteis. Que qualidades deve então ter uma especificação para que seja realmente uma base útil e segura para a construção do sistema?

Diferentes autores fazem referência a diferentes

atributos. Baseados em Alford [ALFORD 76] fazemos aqui uma união dos atributos mais comumente referenciados:

- Comunicabilidade: é a habilidade de uma especificação comunicar requisitos e restrições de uma maneira explícita, inteligível e não ambígua às diversas classes de pessoas que as utilizarão, requerendo destas pouco esforço na interpretação dos documentos apresentados.

- modularidade- a possibilidade de poder-se alterar parte da especificação sem com isto afetar a validade e consistência do restante.

- estrutura - a possibilidade de poder-se determinar completa e claramente as porções antecedentes e consequentes de determinados itens chave em qualquer nível de detalhe da especificação.

- Capacidade de "trace", a possibilidade de localizar todas as referências a determinado assunto ou item chave , nos diversos níveis de especificação

- Consistência, a propriedade de não conter contradições entre quaisquer itens especificados.

- Completeza, a propriedade de definir tudo o que seja relevante no nível de detalhe correspondente à fase atual de elaboração da especifição

- Não condicionante, a propriedade de não antecipar se impondo condições cujo tratamento seria posterior ao nível de detalhe atual na elaboração da especificação

- Verificabilidade- a propriedade de poder-se determinar para cada item chave , requisito e/ou restrição, em qualquer nível de detalhe da especificação, ter o produto final satisfeito o especificado. A verificação poderá ser conduzida por testes e/ou provas formais.

- Correção, a propriedade do produto satisafazer a intenção de seus projetistas, caso satisfaça todos os itens chave, requisitos e/ou restrições especificadas

- Necessidade , a propriedade de cada requisito especificado contribuir para que o requisito originário seja atingido. Ou seja, se qualquer parágrafo de especificação subsequente for eliminado ou não for atingido pelo produto, a especificação originária ou a intenção dos projetistas também não será satisfeita.

- Viabilidade, a propriedade de que exista pelo menos um "design" para o produto que satisfaça as especificações. Esta existência porém não implica em que saiba determiná-lo.

4.2. Métodos de Especificação

Na seção anterior examinamos as propriedades que especificações devem possuir. É mister agora descrever como fazê-lo.

Os problemas encontrados na produção de especificações deram lugar nos últimos anos à realização de um número razoável de pesquisas nesta área. A maior parte delas tem sido no campo de linguagem de especificação [BALZER 78], [AMBLER 77], [BELL 76], [BELL 77], [ROSS 77], [EDP ANALYZER DEZ 79], [TEICHROEW 77], [DAVIS 77], [HENINGER 80], [IRVINE 77].

Especificações apresentam-se sob a forma de textos, diagramas, desenhos etc. São portanto redigidas em alguma linguagem ou conjunto de linguagens. Segundo Liskov e Zilles [LISKOV 77] tais linguagens de especificação, além de permitirem a produção de boas especificações (seção 4.1.), devem safistazer pelo menos aos seguintes requisitos:

- Formalidade - Uma linguagem de especificação deve ser formal, escrita numa notação que permita a verificação formal da especificação antes e após o desenvolvimento do produto.

- Constructibilidade - Deve ser possível construir especificações sem demasiada dificuldade desde que o especificador conheça a linguagem e entenda o conceito a ser especificado.

- Compreensividade - Uma pessoa treinada na notação usada deve ser capaz de ler a especificação e em seguida com um mínimo de dificuldade reconstruir (abstrair) o conceito que a especificação pretende descrever.

- Minimalidade - Deve ser possível, usando a linguagem de especificação, construir especificações que descrevam as propriedades do conceito que interessam e nada mais.

- Amplo campo de aplicabilidade - Quanto maior seja a classe de conceitos que possam ser facilmente descritos pela linguagem, maior sua utilidade.

- Extensibilidade - É desejável que uma pequena alteração em um conceito resulte em uma alteração similarmente pequena nas especificações.

- Detalhabilidade - É desejável que a família de linguagens de especificação possua membros adequados a cada nível de detalhe e que a evolução de um nível de detalhe para outro se dê de forma harmônica.

- Automatizabilidade - É a propriedade do processo de especificar/desenvolver poder receber auxílio automatizado para a sua execução. Um certo grau de automatismo é necessário para que se possa eliminar classes de erros atribuídos a falhas

humanas e/ou para poder verificar de modo mais formal o acerto das especificações e/ou terem estas sido satisfeitas pelo produto.

5. VERIFICAÇÃO E VALIDAÇÃO DAS ESPECIFICAÇÕES

Embora correção não seja a única qualidade desejável para um produto de software, ela é sem dúvida a mais fundamental. Se não existe correção todas as outras qualidades carecem totalmente de sentido. Além disso, já mencionamos que os custos de correção de um erro aumentam de maneir drástica à medida que aumenta o intervalo de tempo entre a época em que efetivamente foi cometido e a época em que foi detectado. Que fazer, então, para "garantir" a correção desde o início do projeto?

Verificação e validação são processos cujos objetivo é determinar e aumentar a integridade de produtos de software. Podemos definir verificação e validação como o processo sistemático de analisar, avaliar e testar o sistema, a documentação e o código com o objetivo de assegurar o máximo possível de integridade e satisfação das necessidades e objetivos do sistema.

Verificação é o processo cujo objetivo é determinar, a cada passo, se o produto obtido (especificação, programa) satisfaz os requisitos determinados no passo anterior. Deste modo cada passo no desenvolvimento fornece a base definitiva e verificada para a execução do passo seguinte.

Validação é o processo de executar e avaliar o software (sistemas, programas, módulos) após construido, comparandos resultados das medições (teste, provas formais, avaliações) com os requisitos e restrições requeridos [LEWIS 79].

Na figura mostramos de forma gráfica como interagem no tempo as atividades de especificar, implementar, verificar e validar (fig. 3)

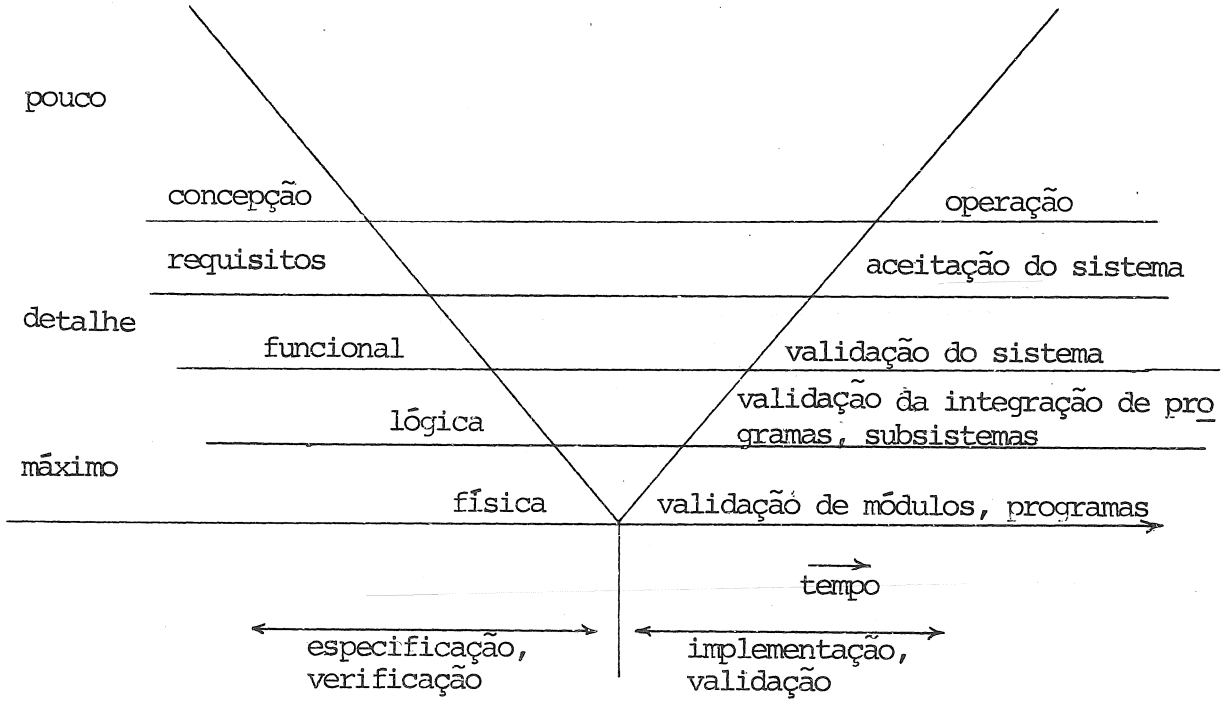

fig.3 - Visão pictórica das interações entre especificação, implementação, verificação e validação.

Mas onde está o problema ao se tentar validar um produto de software? Em primeiro lugar podem ocorrer falhas na própria especificação. As especificações podem ter sido mal estabelecidas e ninguém consegue construir o produto a partir delas (falha da verificação). Pode também acontecer que embora as especificações sejam claras e não ambiguas, descrevem um produto não apropriado para o que se pretende (falha de verificação). Neste caso houve falha ao se estabelecerem as especificações. Mas as especificações podem estar perfeitas e haver falha na implementação. Neste caso, mesmo execuções corretas do programa não são capazes de atingir as especificações [DISJKSTRA 77].

A base para a construção de um software confiável está na produção de especificações confiáveis. Nenhuma metodologia ou técnica de programação poderá aumentar a confiabilidade de um produto de software se as especificações estiverem incorretas ou forem traduzidas de modo incorreto [BELFORD 76].

Se consideramos que as especificações são a única interface real entre o usuário e o desenvolvedor, a verificação das especificações adquire ainda maior importância. Pois,

sem uma base verificada sobre a qual trabalhar, pouco pode-se fazer para assegurar um entendimento comum entre os que vão usar e os que vão desenvolver o sistema. Além disso, especificações incorretas (ou inexistentes) levam a uma considerável perda em termos de homem-horas, devida ao tempo gasto no desenvolvimento de produtos de software que serão descartados depois de prontos devido à inerente inadequação ou mesmo inutilidade destes produtos.

Torna-se então evidente o caráter essencial da existência e verificação das especificações.

Em que consiste está verificação? Seu objetivo é assegurar que todos os requisitos e restrições documentados na especificação satisfazem aos requisitos determinados na seção 4.1 (claros, completos, corretos, possíveis de serem testados, etc.). É necessário, portanto, iniciar o projeto com requisitos verificados e testáveis, especificando critérios quantitativos que possam ser medidos para determinar se foram alcançados. Requisitos "testáveis" são requisitos específicos, não ambíguos e com um resultado claramente identificável quando atingido.

6. CONCLUSÃO

O problema formalidadeXinformalidade em especificações é um problema ainda sem resposta. Por um lado há os que insistem que a formalidade é um ingrediente necessário quando se fala em especificações. Estes autores valorizam o rigor e a possibilidade de verificação e validação automática. Por outro lado os defensores da informalidade alegam dificuldades na utilização de métodos formais enfatizando a imaturidade do estado da arte neste campo.

Outros decidem-se por um meio termo, utilizando o formalismo quando este for apropriado, e métodos informais quando da inexistência ou inadequação pragmática dos métodos formais.

Esta, portanto, é uma questão ainda em fase de pesquisa. Num artigo publicado na revista Datamation em janeiro de 1979 [LEHMAN 79] são relatados os resultados de uma pesquisa sobre gerência de projetos feita entre firmas da indústria aeroespacial. Entre os 57 projetos de software sobre os quais se obteve informação, nenhum tinha tido sua especificação de definição (requisitos e funcional) escrita numa linguagem de especificação. Isto mostra como a linguagem natural ainda é a linguagem de especificação mais comumente aceita pela indústria.

Por outro lado, já existem resultados experimentais animadores decorrentes do emprego de métodos mais formais,

tais como: HDM , SREM , SADT e CADSAT (PSL/PSA) [BAIL 79]. Mas qual destas técnicas será a melhor? Todas elas são potencialmente aplicáveis e todas possuem inadequações.

Se voltarmos aos critérios de avaliação de métodos de especificação propostos por Liskov e Zilles [LISKOV 77] podemos observar:

1. Os métodos propostos na literatura tendem a ser bastante formais.
2. Os objetivos de detalhabilidade constructibilidade e compreensividade nem sempre tem sido atingidos pelas metodologias propostas. Uma série de dificuldades são apontadadas pelos usuários destas metodologias [BAIL 79].
3. Minimalidade é um objetivo presente em todas as metodologias.
4. Os objetivos de extensibilidade e amplo campo de aplicabilidade são ainda difíceis de serem avaliados dado que as metodologias ainda foram pouco utilizadas. Como objetivo estão presentes em todas as metodologias propostas.

No que se refere a verificação de especificações podemos concluir que esta é uma área de pesquisa cujos resultados são ainda mais incipientes. O desenvolvimento de metodologias e ferramentas para construção e verificação de especificações apresenta-se assim, como um campo aberto para pesquisas.

REFERÊNCIAS BIBLIOGRÁFICAS

[ALFORD 76] - Alford, M.W.; Burns, I.F." R-Nets : a graph model for real-time software requirements", Proceedings of the Symposium on Computer Software Engineering, New York 1976

[AMBLER 77] - Ambler, Allen L.; Good, Donald I.; Browne, James C.; Burger, Wilhelm F.; Cohen, Richard M.; Hoch, Charles G.; Wells, Robert E. "GYPSY: A language for Specification and Implementation of verifiable programs", Proceedings of an ACM Conference on Language Design for Reliable Software, março 1977, editado por David B. Wortman

[BAIL 79] - Bail, William G. "User experience with specifications tools", Panel from Specifications of Reliable Software

Conference, ADM SIGSOFT, Software Engineering Notes, vol 4. nº3, julho 1979

[BALZER 78] Balzer, Robert: Goldman, Neil; Wile, David "Informality in Program Specifications" , IEEE Transactions on Software Engineering,vol SE-4, nº 2 março 1978

[BELFORD 76] Belford, P.C.;Taylor D.S. "Specification Verification, a Key to improving software realiability", Proceedings of the Symposium on Computer Software Engineering, New York 1976

[BELL 76] Bell,T.E.; Bixler D.C."A flow - oriented requirements Statemente Language" Proceedings of the Symposium on Computer Software Engineering, New York 1976

[BELL 77] Bell, T.E.;Bixler, D.;Dyer,M."An extendable approach to Computer-Aided Software Requirements Engineering", IEEE Transactions on Software Engineering,vol SE- 3 , nº 1, janeiro 1977

[DAVIS 77] Davis, G.;Vick, Charles R."The Software Development System", IEEE Transactions on Software Engineering,vol SE-3, nº 1, janeiro 1977

[DIJKSTRA 77] Dijkstra,E.J. "A position paper on software reliability",ACM SIGSOFT, Software Engineering Notes, vol 2, nº 5 , outubro 1977

[EDP ANALYZER 79] "Progress Toward System Integrity" , EDP ANALYZER vol 17, nº12, dezembro 1979

[HENINGER 79] Heninger,K."Limits to Specifications why not more progress?" Panel at IEEF Conference on Specifications for Reliable Software, ACM SIGSOFT,Software Engineering Notes, vol 4,nº 3, julho 1979

[HENINGER 80] Heninger,K."Specifying Software Requirements for Complex Systems: New Techniques and their aplications", IEEE Transactions on Software Engineering, Vol SE-6 nº 1, janeiro 1980

[IRVINE 77] Irvine,C.A.;Brackett,J.N."Automated

Software Engineering through structured data Management", IEEE Transactions on Software Engineering,vol SE-3, nº 1 , janeiro 1977

[LEWIS 79] Lewis,Robert D."Software Verification and Validation" in Software Quality Management, ed Cooper, John D. e Fisher, Matthew J.; Petrocelli Books Inc 1979

[LISKOV 77] Liskov, Barbara;Zilles, Stephen "An introduction to formal Specifications of data anstractions", in Current trends in Programming Methodology, vol I, ed Yeh, Raymond T., Prentice-Hall Inc, New Jersey 1977

[RIDDLE 78] Riddle, W.E.; Wileden, J.C."Languages for representing software specifications and designs" ACM SIGSOFT, Software Engineering Notes, vol,nº 4 outubro 1978

[ROSS 77] Ross,D.T. "Structured Analysis (SA): A language for Communicating Ideas" IEEE Transactions on Software Engineering , vol SE- 3, nº 1, janeiro 1977

[TEICHROEW 77] Teichroew,D.;Hersley,E.A.III" PSL/PSA: A Computer Aided Technique for Structured Dcoumentation and Analysis of Information Processing Systems", IEEE Transactions on Software Engineering, vol SE-3 nº 1 , janeiro 1977