

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTAÇÃO E SISTEMAS

LANNA MAYRA SILVA OLIVEIRA

**CARACTERIZAÇÃO DO CONCEITO DE MODULARIDADE NO
DESENVOLVIMENTO DE LINGUAGENS DE PROGRAMAÇÃO**

João Monlevade

2017

LANNA MAYRA SILVA OLIVEIRA

**CARACTERIZAÇÃO DO CONCEITO DE MODULARIDADE NO
DESENVOLVIMENTO DE LINGUAGENS DE PROGRAMAÇÃO**

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Orientador: Leonardo Vieira dos Santos Reis

Coorientador: Elton Maximo Cardoso

João Monlevade

2017



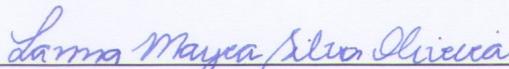
UFOP
Universidade Federal
de Ouro Preto

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

TERMO DE RESPONSABILIDADE

Eu, Lanna Mayra Silva Oliveira, declaro que o texto do trabalho de conclusão de curso intitulado "*Caracterização do Conceito de Modularidade no Desenvolvimento de Linguagens de Programação*" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 13 de abril de 2017


Assinatura do aluno

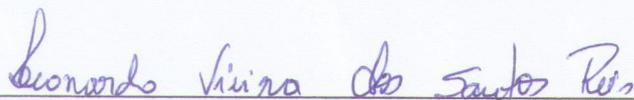
ATA DE DEFESA

Aos trinta e um dias do mês de março de dois mil de dezessete, às quinze horas e trinta minutos, na sala H102 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pela aluna **Lanna Mayra Silva Oliveira**, sendo a Comissão Examinadora constituída pelos professores: Prof. Dr. Leonardo Vieira dos Santos Reis, Prof. Me. Elton Máximo Cardoso, Prof. Me. Euler Horta Marinho e Prof. Me. Igor Muzetti Pereira.

O candidato apresentou a monografia intitulada: "*Caracterização do Conceito de Modularidade no Desenvolvimento de Linguagens de Programação*". A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 10 (Dez pontos), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

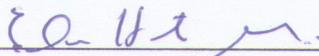
João Monlevade, 31 de março de 2017.



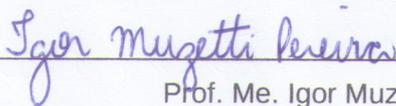
Prof. Dr. Leonardo Vieira dos Santos Reis
Professor Orientador/Presidente



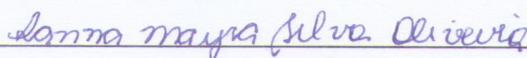
Prof. Me. Elton Máximo Cardoso
Professor Coorientador



Prof. Me. Euler Horta Marinho
Professor Convidado



Prof. Me. Igor Muzetti Pereira
Professor Convidado



Lanna Mayra Silva Oliveira
Graduando

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Caracterização do Conceito de Modularidade no Desenvolvimento de Linguagens de Programação

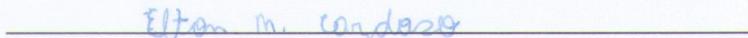
Lanna Mayra Silva Oliveira

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:



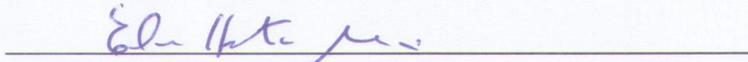
Prof. Dr. Leonardo Vieira dos Santos Reis

Departamento de Computação e Sistemas - DECSI



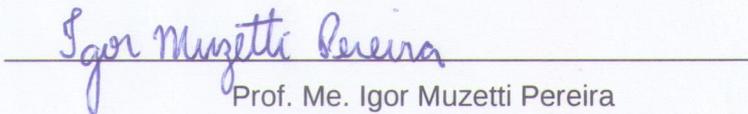
Prof. Me. Elton Máximo Cardoso

Departamento de Computação e Sistemas - DECSI



Prof. Me. Euler Horta Marinho

Departamento de Computação e Sistemas - DECSI



Prof. Me. Igor Muzetti Pereira

Departamento de Computação e Sistemas - DECSI

João Monlevade, 31 de março de 2017

*Este trabalho é dedicado à Francisco e Mayre,
os principais responsáveis pela concretização desse sonho.*

Agradecimentos

Agradeço primeiramente a Deus, por tornar os sonhos possíveis de serem sonhados e por dar força e determinação para realizá-los apesar de todas as dificuldades. Aos meus pais, Francisco e Mayre, por caminharem junto e por serem muito mais que pais, mas verdadeiros amigos. Aos meus irmãos Hugo e Rita, e aos meus primos Laísla e Daniel, pelo companheirismo e por sempre acreditarem em mim. À minha família pela constante presença e apoio, em especial à minha tia Mariluzza pelas palavras de incentivo e todo carinho de sempre. Aos amigos de longa data, e aos que conquistei durante a graduação, em especial Hannah, Hiuly, Ana Carolina, Thaís, Fernanda, Eduardo, Manoel, Bárbara, Danielle, Oto, João Pedro e Paulo Otávio. Ao Diego, pela total paciência, amizade, companheirismo, e por sonhar esse sonho comigo. Aos mestres que ensinaram com dedicação, (em especial ao Fernando que foi um grande amigo na reta final), tiveram paciência para ouvir todo choro e mostraram que aquilo também fazia parte do crescimento pessoal e profissional.

Em especial, agradeço ao meu orientador Leonardo Reis e ao meu coorientador Elton Cardoso, pelos conselhos, aprendizado adquirido, por toda paciência que tiveram para me ensinar e pela amizade criada durante o desenvolvimento deste trabalho. Enfim, agradeço a todos que direta ou indiretamente, presentes ou ausentes, contribuíram para a conclusão deste.

*“Por vezes sentimos que aquilo que
fazemos não é senão uma gota de água no mar.
Mas o mar seria menor se lhe faltasse uma gota.”
(Madre Teresa de Calcuta)*

Resumo

Desenvolver sistemas em módulos é uma das boas práticas atuais, pois a modularidade, baseada na filosofia “dividir para conquistar”, melhora o tempo de desenvolvimento e contribui com a flexibilidade e compreensibilidade do sistema (PARNAS, 1972). O conceito de Modularidade teve seus primórdios a partir da proposta de divisão de um sistema em módulos independentes que possibilitasse mudanças independentes (PARNAS, 1972). Entretanto, quando relacionada ao desenvolvimento de linguagens de programação, a modularidade não é definida criteriosamente a ponto de indicar técnicas e/ou características que de fato a provêm. Este trabalho tem como objetivo caracterizar o conceito de modularidade no desenvolvimento de linguagens de programação a partir da visão dos projetistas. Um conceito geral de modularidade é buscado para que seja possível alcançar o objetivo. A caracterização foi estruturada a partir da leitura de artigos, que foi marcada por uma Revisão Sistemática de Literatura na faixa de tempo de 1972–1978.

Palavras-chave: modularidade; desenvolvimento de softwares; linguagens de programação; revisão sistemática de literatura.

Abstract

Developing modular systems is one of today's best practices, since modularity, based on the "divide and conquer" philosophy, improves development time and contributes to system flexibility and comprehensibility (PARNAS, 1972). The concept of Modularity was born from the purpose of dividing a system into independent modules that enables to make independent changes (PARNAS, 1972). However, when it comes to the development of programming languages, modularity is not carefully defined to indicate techniques and/or features that actually supply it. This work aims to characterize the concept of modularity in the development of programming languages from the viewpoint of the designers. A general concept of modularity is sought for what the objective can achieve. The characterization of modularity was structured was based on reading of articles, which was marked by a Systematic Review of Literature in the time range of 1972–1988.

Key-words: modularity; software development; programming languages; systematic review of literature.

Lista de quadros

Quadro 1 – Busca Avançada(<i>Advanced Search</i>)	19
Quadro 2 – Artigos utilizados para leitura	45
Quadro 3 – Artigos excluídos da leitura	47
Quadro 4 – Características observadas no desenvolvimento	57

Lista de tabelas

Tabela 1 – Quantidade de artigos coletados a partir dos critérios definidos	21
---	----

Lista de abreviaturas e siglas

ACM	Association for Computing Machinery
IEEE	Institute of Electrical and Electronic Engineers
PL/M	Programming Language for Microcomputers

Sumário

1	INTRODUÇÃO	15
2	METODOLOGIA	17
2.1	Identificação da Pesquisa	18
2.2	Seleção de Estudos Primários	19
2.2.1	Filtragem de Artigos	21
2.3	Avaliação da Qualidade do Estudo	21
2.4	Extração e Monitoramento de Dados	22
2.5	Síntese de Dados	22
2.6	Limitações e Ameaças	22
3	DESENVOLVIMENTO	24
3.1	On the Criteria To Be Used in Decomposing Systems into Modules	24
3.2	Operating System Performance	24
3.3	Computerized Dispatching System	25
3.4	Organizing Computer Systems for Learnability and Useability	26
3.5	A modularized package of dual algorithms for solving constrained nonlinear programming problems	26
3.6	A pragmatic proposal for the improvement of program modularity and reliability	27
3.7	CAMAC Software for the Hot Fuel Examination Facility Real-Time Computer System	28
3.8	Modularization Around a Suitable Abstraction	28
3.9	Program design by a multidisciplinary team	29
3.10	The Effect of Certain Modular Design Principles on Testability	29
3.11	Programming Languages: The First 25 Years	30
3.12	A Balanced View of MUMPS	31
3.13	Possible futures and present actions	31
3.14	The use of Abstract Data Types to Simplify Program Modifications	32
3.15	Visibility and types	32
3.16	Notes on the design of Euclid	33
3.17	Programming languages for introductory computing courses: a position paper	34
3.18	Software engineering with standard assemblies	34
3.19	The modularity of MSC/NASTRAN	35
3.20	The Smalltalk-76 programming system design and implementation	35

3.21	Ocular motility test administration and analysis by computer in strabismus and amblyopia evaluation	36
3.22	A small-scale operating system foundation for microprocessor applications	36
4	ANÁLISE	38
4.1	Considerações	40
5	CONCLUSÃO	41
5.1	Trabalhos Futuros	41
	REFERÊNCIAS	42
	APÊNDICE A – ARTIGOS UTILIZADOS	45
	APÊNDICE B – ARTIGOS EXCLUÍDOS	47
	APÊNDICE C – CARACTERÍSTICAS OBSERVADAS	57

1 Introdução

O avanço constante da tecnologia, um maior acesso à informação e a necessidade de desenvolver novos softwares estão em crescimento constante na sociedade, fazendo com que novas linguagens de programação sejam desenvolvidas ou modificadas ao longo do tempo para atender essas necessidades. Seja na modificação ou no desenvolvimento de linguagens de programação e/ou softwares, a modularidade é um conceito que sempre está presente, entretanto, qual a sua definição?

Muito usada no desenvolvimento de softwares, sistemas complexos e conceituada na Engenharia de Software, a modularidade é conhecida pela filosofia popular *“dividir para conquistar”*. O conceito de Modularidade surgiu em meados da década de 1970 e está diretamente relacionado à estrutura, seja em softwares, circuitos, hardwares e outros diversos ramos de estudo. Relacionada ao desenvolvimento de sistemas, um dos primeiros artigos a citá-la, traz a modularidade como sendo a capacidade de dividir um sistema em partes independentes conhecidas como módulos, que podem ser modificadas individualmente sem informações adicionais das outras (PARNAS, 1972), o que em termos de alteração ou evolução em linguagens de programação e sistemas de softwares é muito importante, pois além de ser mais vantajoso alterar somente partes individuais do sistema, essa abordagem também permite desenvolvimento destes em paralelo e melhor compreensão dos módulos isoladamente. Além disso, para PARNAS utilizar modularidade pode apresentar vantagens como a melhoria da flexibilidade e compreensibilidade de um sistema, o que conseqüentemente resulta em um menor tempo de desenvolvimento deste.

Muitas características foram incorporadas ao conceito com o passar dos anos, e muitos trabalhos afirmam proverem modularidade, mas contradizem o conceito quando é necessário realizar alterações, pois precisam de informações adicionais de cada módulo para aplicar essas mudanças (MERNIK; UMER, 2005). Dessa forma entende-se que não existe um conceito bem definido de modularidade no contexto de desenvolvimento de linguagens de programação, uma vez que as características que são utilizadas para alcançá-la não são criteriosamente estabelecidas durante esse desenvolvimento.

À vista disto, o objetivo deste trabalho é caracterizar o conceito de modularidade no desenvolvimento de linguagens de programação a partir da visão dos projetistas de linguagens. O primeiro passo para alcançar esse objetivo foi buscar compreender o conceito de modularidade, tendo em vista que este não é definido de maneira clara o suficiente quando relacionado à desenvolvimento de linguagens de programação. Para tentar compreender o conceito, foram analisadas características e definições tanto no desenvolvimento de linguagens de programação, quanto no desenvolvimento de softwares,

utilizando para tal uma Revisão Sistemática de Literatura ([Capítulo 2](#)), que “*identifica, analisa e interpreta todas as evidências relatadas e avaliadas para uma pesquisa específica em questão*” ([WOHLIN et al., 2012](#)). Optou-se pela Revisão Sistemática de Literatura posto que a ideia inicial de modularidade já existe a mais 40 anos, e muitas características foram atribuídas a ela, e portanto, a melhor forma de caracterizá-la é analisar os conceitos e características atribuídos a ela pelos projetistas.

De acordo com [WOHLIN et al.](#), uma revisão sistemática de literatura contém uma série de etapas a serem seguidas, ou seja, existe um protocolo para realizar a revisão. As etapas utilizadas são a identificação da pesquisa ([seção 2.1](#)), que se refere ao local e forma de coleta de dados; a seleção dos estudos primários ([seção 2.2](#)), em que são definidos os critérios para coletar os artigos; a avaliação da qualidade do estudo ([seção 2.3](#)), que é responsável por avaliar os critérios criados, bem como direção que a revisão está seguindo; a extração de dados e monitoramento destes ([seção 2.4](#)), em que os dados serão coletados; e a síntese de dados ([seção 2.5](#)), que é feito o agrupamento e análise dos dados para chegar a uma conclusão.

Foram selecionadas quatro bases de dados bem conceituadas na Ciência da Computação para a realização da Revisão Sistemática, sendo elas **ACM Digital Library**¹, **IEEE Xplore Digital Library**², **Science Direct**³ e **Springer Link**⁴. Os critérios utilizados para coleta de artigos foram palavras-chave, e os utilizados para seleção foram a abordagem do artigo, tema central e faixa de tempo. Os artigos utilizados foram estruturados em um quadro apresentado no [Apêndice A](#) por ordem cronológica, e os artigos que não foram aceitos pelos critérios de seleção foram descartados de acordo com os critérios definidos na [seção 2.2](#), e apresentados em ordem cronológica no [Apêndice B](#). Para esta revisão foram coletados 148 artigos, onde apenas 14,9% destes foram utilizados.

O monitoramento dos dados é feito a partir da leitura dos artigos. Esta foi iniciada partindo do artigo mais antigo até o mais recente dentro da faixa de tempo estipulada, pois é necessário entender como o conceito de modularidade foi sendo incorporado ao longo do tempo ([Capítulo 3](#)). Um breve resumo dos artigos foi apresentado, contendo uma definição e características explícitas ou implícitas no texto, de modo que agrupando essas definições ao final da leitura de todos, seja possível estruturar um possível conceito para a modularidade e assim apontar as características comuns, e a partir disso poder identificar quais as características de fato fazem parte do desenvolvimento de linguagens de programação e/ou softwares. Por fim, apesar de encontrar características que são comuns entre os trabalhos analisados, estas por vezes não são suficientes para alcançar a modularidade.

¹ dl.acm.org

² ieeexplore.ieee.org

³ sciencedirect.com

⁴ link.springer.com

2 Metodologia

O objetivo deste trabalho é caracterizar o conceito de modularidade no desenvolvimento de linguagens de programação a partir da visão dos projetistas de linguagens. Para alcançar esse objetivo, optou-se por desenvolver uma Revisão Sistemática de Literatura, pois é necessário compreender como o conceito de modularidade foi incorporado ao longo do tempo.

Uma Revisão Sistemática de Literatura *“identifica, analisa e interpreta todas as evidências relatadas e avaliadas para uma pesquisa específica em questão”* (WOHLIN et al., 2012). Para tornar possível a realização de uma Revisão Sistemática de Literatura, um planejamento deve ser feito. Este planejamento tem por objetivo identificar se existe necessidade de avaliação da pesquisa, ou seja, se já existem revisões sobre o tema em questão e, na condição de já existirem, verificar se estas são de fato válidas; apresentar a pesquisa em questão, e desenvolver um protocolo de revisão, de forma a definir minuciosamente quais passos devem ser seguidos, possibilitando assim que qualquer indivíduo que desejar avaliar ou conferir a revisão possa chegar exatamente nos mesmos resultados obtidos pelo revisor(a). O presente trabalho utilizou a abordagem de Revisão Sistemática de Literatura descrita por Wohlin et al. (2012), bem como seguiu os passos e regras apresentadas para sua realização. Estes são relatados nas seções deste capítulo.

O conteúdo sobre o tema é muito extenso, e a gama de artigos encontrados também, pois a modularidade está presente em diversos ramos de estudo, como por exemplo a Psicologia (BARRETT; KURZBAN, 2006). O foco deste trabalho é voltado para Linguagens de Programação e Engenharia de Software, que abordam a modularidade em seu conteúdo constantemente, seja no desenvolvimento ou alteração de softwares e/ou linguagens de programação, ou em outras de suas divisões.

Um dos primeiros trabalhos que aborda a divisão de módulos, atribui ao conceito de modularidade a capacidade de dividir um sistema em partes independentes conhecidas como módulos, que podem ser modificadas individualmente sem informações adicionais das outras (PARNAS, 1972). Entretanto, existem divergências quando o assunto é modularidade, pois muitos trabalhos utilizam abordagens que dizem prover modularidade, mas necessitam conhecer os detalhes ou alterar mais de um módulo quando mudanças são necessárias (MERNIK; UMER, 2005; JOHNSTONE; SCOTT; BRAND, 2014), contradizendo a ideia principal da modularidade que diz não precisar saber detalhes do módulo ou informações extras para realizar essas mudanças (PARNAS, 1972), pois se assume que cada módulo é independente.

Foi selecionada como metodologia para este trabalho uma Revisão Sistemática de

Literatura por ser necessário analisar e interpretar as diferentes definições e características da modularidade incorporadas pelos projetistas ao longo dos anos. A partir desta análise e interpretação, uma definição para modularidade é apresentada em um contexto geral, e assim são descritas as características em comuns apresentadas pelos projetistas.

Para realizar uma Revisão Sistemática de Literatura é necessário seguir uma série de passos que tornam possível sua validação, e para tal é preciso que exista um protocolo de revisão (WOHLIN et al., 2012). A Revisão Sistemática de Literatura realizada neste trabalho contém um protocolo de revisão que tem como passos: a identificação da pesquisa (seção 2.1), a seleção dos estudos primários (seção 2.2), a avaliação da qualidade do estudo (seção 2.3), a extração de dados e monitoramento destes (seção 2.4), e a síntese de dados (seção 2.5). Cada um desses passos foram explicados detalhadamente nas seções seguintes.

2.1 Identificação da Pesquisa

A fase de identificação da pesquisa é onde deve-se definir como será realizada a coleta de dados. É nessa etapa que a base de dados para a pesquisa é estruturada, ou seja, é definido através de onde os dados são coletados (artigos de periódicos, anais de conferência, livros, entre outros).

Para esta revisão optou-se por utilizar apenas artigos científicos como forma de coleta de dados devido à grande quantidade de conteúdo existente sobre o tema, e também pelo fato de que teses e dissertações quando bem redigidas são publicadas em revistas ou apresentadas em conferências em forma de artigos. Assim, entende-se que grandes trabalhos que possuem pesquisas extensas, podem estar resumidos em artigos.

Os artigos utilizados nessa revisão foram coletados nas seguintes bases de dados:

- ACM Digital Library¹;
- IEEE Xplore Digital Library²;
- Science Direct³;
- Springer Link⁴

A busca nessas bases de dados não foram feitas da mesma forma porque estas apresentam formas de pesquisas diferentes. Entretanto as pesquisas foram feitas de forma a ficarem o mais similar possível. Cada palavra-chave (“*modularity*”, “*encapsulation*” e

¹ dl.acm.org/

² ieeexplore.ieee.org/

³ sciencedirect.com/

⁴ link.springer.com/

Quadro 1 – Busca Avançada (*Advanced Search*)

Base	Pesquisar em	Tipo	Disciplina
ACM Digital Library	Title	-	-
ACM Digital Library	Abstract	-	-
ACM Digital Library	Author Keywords	-	-
IEEE Xplore	Document Title, Abstract or Author Keywords	Conference Publications, Journals & Magazines	-
Science Direct	Abstract, Title, Keywords	Journals	Computer Science
Science Direct	Abstract, Title, Keywords	Journals	Engineering
Springer	-	Article	Computer Science
Springer	-	Article	Engineering
Springer	-	Conference Paper	Computer Science
Springer	-	Conference Paper	Engineering

“*modularization*”) foi pesquisada individualmente em cada base. O Quadro 1 apresenta os critérios como os artigos foram encontrados em cada base de dados, utilizando para tal o método de *Advanced Search* (Pesquisa Avançada). Cada linha do Quadro 1 utiliza separadamente cada palavra-chave descrita.

2.2 Seleção de Estudos Primários

Essa fase consiste de criar critérios para filtrar os artigos encontrados. Esses critérios são importantes pois o tema “*modularidade*” não é específico das áreas de Engenharia de Software e/ou Linguagens de Programação, podendo também ser usado em várias outras áreas de pesquisa, como exemplo, o Desenvolvimento de Produtos (SCHUH; RUDOLF; VOGELS, 2014). Estes devem ser criados antes de começar a coleta de dados, e podem ser modificados no decorrer do trabalho desde que ainda estejam dentro das especificações.

Os critérios utilizados nessa revisão, consistem de buscar artigos que:

- Contenham “*modularity*”, “*modularization*” ou “*encapsulation*” em seu título, abstract e/ou palavras-chave;

A princípio, a ideia era coletar artigos que continham todos os sinônimos da palavra **modularidade**, ou seja, na busca também entrariam “*module*” (módulo) e “*modu-*

lar” (modular), bem como os sinônimos da palavra **encapsulamento**, sendo eles “*encapsulate*” (encapsulado) e “*encapsular*” (encapsular). Entretanto, a possibilidade foi descartada após várias tentativas de adaptação, devido à quantidade de artigos ser extremamente grande, inviabilizando a análise destes em tempo hábil, deixando apenas as palavras-chave citadas acima.

- Estejam diretamente ligados ao desenvolvimento/abordagem de softwares/linguagens de programação.

Pois, *modularidade*, *modularização* e *encapsulamento* são termos que estão presentes em diversas áreas de pesquisas, e neste caso, a pesquisa é voltada apenas para as áreas de Engenharia de Software e linguagens de programação. Para concluir se estes artigos de fato estão dentro desse contexto, os *abstracts* (ou resumos) foram previamente lidos, e partir disso, esses artigos foram incluídos ou não. Portanto, artigos que não possuíam abstract não entram no escopo do trabalho.

- Tenham **modularidade** como um dos temas centrais;

Foi feita uma análise prévia do artigo utilizando para tal a introdução e conclusão, e assim aqueles que apenas reportarem o termo “*modularidade*”, sem nenhum indicativo de que o assunto é realmente abordado, não entram no escopo do trabalho. Artigos que não possuem introdução não entram no escopo do trabalho.

A base de dados foi feita utilizando os critérios citados e o resultado foi de milhares de artigos, entretanto o tempo para o desenvolvimento do trabalho é curto, inviabilizando a análise de todos. Apesar de [Wohlin et al. \(2012\)](#) sugerir a utilização de trabalhos em diferentes anos de publicação, optou-se por utilizar uma faixa de tempo para que fosse possível analisar o conceito sem que nenhuma característica importante fosse desconsiderada.

A faixa de tempo estipulada foi 1972–1978. Os artigos foram analisados em ordem cronológica para que fosse possível compreender como o conceito foi estruturado ao longo dos anos. Deste modo, o menor valor do intervalo foi o ano em que o artigo utilizado como base para esse trabalho foi publicado e o maior é relativo ao ano em que foi possível realizar a leitura de acordo com o tempo de desenvolvimento.

A partir desses critérios foram filtrados os artigos mais relevantes sobre o tema, buscando facilitar a identificação das principais características da modularidade. O [Apêndice A](#) contém os artigos utilizados e o [Apêndice B](#) contém os artigos descartados, bem como o motivo de descarte. No total foram analisados previamente 148 artigos, entretanto somente 22 destes foram utilizados, totalizando 14,9% da quantidade total. Essa informação pode ser melhor visualizada na [Figura 1](#), que mostra o gráfico com a relação de artigos analisados. A [subseção 2.2.1](#) apresenta os passos da coleta de artigos.

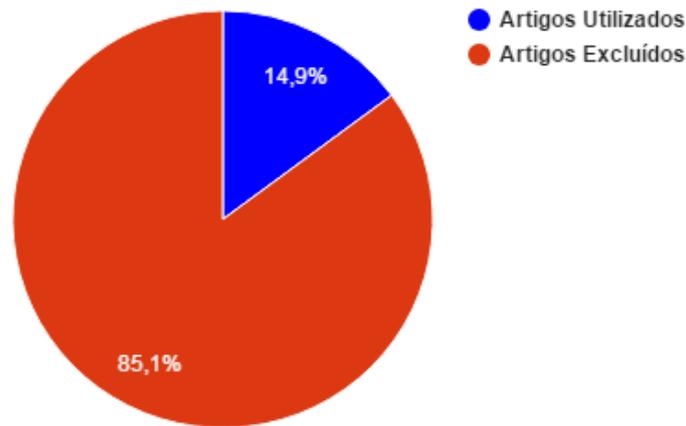


Figura 1 – Relação de artigos analisados

2.2.1 Filtragem de Artigos

Os artigos foram coletados a partir dos critérios mencionados na [seção 2.2](#). O intervalo de tempo foi estipulado após a coleta de todos os artigos. A [Tabela 1](#) mostra a quantidade de artigos coletados inicialmente, antes do intervalo de tempo ser estipulado, ou seja, a quantidade de artigos encontrados nas bases de dados entre 1972–2017.

Tabela 1 – Quantidade de artigos coletados a partir dos critérios definidos

Base de Dados	Modularity	Encapsulation	Modularization
ACM Digital Library	6414	1542	6414
IEEE Xplore Digital Library	4060	3333	1045
Science Direct	1477	1890	379
Springer Link	15049	10055	4477

Os artigos foram colocados em ordem cronológica e analisados durante o tempo de desenvolvimento, interrompendo a análise no ano de 1978. Foram analisados 148 artigos, dos quais foram selecionados apenas 22.

2.3 Avaliação da Qualidade do Estudo

Avaliar a qualidade dos estudos primários é importante porque a partir disso verifica-se o caminho que a revisão está seguindo, e assim se conclui se os critérios utilizados para realizá-la são realmente bons.

O foco do trabalho é caracterizar o conceito de modularidade no desenvolvimento de linguagens de programação, sendo assim, os critérios utilizados citados ([seção 2.2](#))

estão diretamente ligados ao tema. Os artigos coletados são publicados em conferências e revistas, e dessa forma entende-se que já passaram por algum tipo de revisão em algum momento, o que faz com que seus dados apresentem um certo tipo de consistência e sejam úteis para a realização da revisão, descartando de certa forma, a necessidade de realizar uma Avaliação da Qualidade do Estudo. Portanto, essa etapa da Revisão Sistemática de Literatura não foi realizada durante o desenvolvimento deste trabalho.

2.4 Extração e Monitoramento de Dados

O próximo passo para realizar a revisão é a extração e monitoramento de dados, que consiste de extrair os dados necessários para concluir a revisão. Para extração, os dados foram coletados a partir dos critérios definidos (seção 2.2).

Como a ideia do trabalho é caracterizar a modularidade de forma geral e em seguida caracterizá-la no contexto de linguagens de programação, o que se analisou em cada artigo foi uma possível definição de modularidade e as características necessárias para alcançá-la. É importante enfatizar que essa extração deve ser feita cuidadosamente, pois nem todos os artigos irão definir claramente qual é o real conceito de modularidade, fazendo com que seja necessário realizar a análise de algumas definições presentes para juntá-las e descobrir um possível conceito.

2.5 Síntese de Dados

A última etapa da revisão é a fase de síntese de dados, que consiste de reunir os dados coletados, analisá-los e chegar a alguma conclusão.

Após reunir diversas definições e características de modularidade, estas foram analisadas minuciosamente com o intuito de conceituar a modularidade, tanto num contexto geral, quanto no contexto de linguagens de programação, e assim definir de fato quais são suas características.

2.6 Limitações e Ameaças

Durante o desenvolvimento da Revisão Sistemática de Literatura foram apontadas algumas limitações e ameaças que podem interferir na qualidade do trabalho se não forem levadas em consideração. As limitações e ameaças observadas são:

- Tempo de desenvolvimento;

O tempo para a realização do trabalho é curto, e analisar todos os artigos coletados dentro desse tempo é inviável, pois como mostra a [Tabela 1](#) a quantidade de artigos existentes com os critérios definidos é muito extensa.

- Pesquisa nas bases de dados;

Algumas buscas em bases são estruturadas de modo remissivo, ou seja, quando uma palavra-chave é pesquisada, todos os artigos que contenham sinônimos desta palavra são retornados como resultado para o usuário. Neste trabalho, a base de dados da ACM Digital Library⁵ contém esse índice, e com isso foi necessário analisar cada artigo dessa base para observar se a palavra-chave utilizada na pesquisa realmente fazia parte do título, *abstract* ou palavras-chave de cada artigo.

- Quantidade de bases de dados.

A quantidade de base de dados é relativamente pequena, também devido ao tempo de desenvolvimento do trabalho. Esta poderia ser ampliada com o intuito de alcançar maiores quantidades de dados.

- Somente literatura inglesa;

O trabalho foi realizado utilizando somente artigos escritos em inglês, o que pode ser considerado uma ameaça porque a interpretação da escrita durante a leitura pode ser vista por diferentes pontos.

- Interpretação do texto para obter o conceito a partir da visão do autor;

O ato por si só já é uma ameaça considerável porque nem todos os artigos apresentam o conceito de forma explícita, e entender o que o autor quis expressar é muito complexo.

⁵ <http://dl.acm.org/>

3 Desenvolvimento

Como foi abordado no [Capítulo 2](#), o presente trabalho consiste da leitura de artigos de periódicos e conferências, realizando assim uma revisão sistemática de literatura. Cada artigo analisado contém um conceito de modularidade, seja ele explícito ou implícito, o qual será apresentado para que uma análise seja realizada e se torne possível caracterizar a modularidade no desenvolvimento de linguagens de programação a partir da visão dos projetistas de linguagens. Cada artigo analisado é abordado separadamente nas seções seguintes.

3.1 On the Criteria To Be Used in Decomposing Systems into Modules

[Parnas \(1972\)](#) foi o principal “incentivador” para o início das definições do tema e apresenta a ideia de que modularidade nada mais é do que a divisão de um sistema em módulos independentes que possam sofrer alterações sem precisar de informações extras dos outros módulos (*information hiding*), e sem que todo o sistema necessite mudar. Durante o artigo algumas abordagens da divisão de módulos são feitas, e é enfatizado que a modularidade tem o poder de melhorar a flexibilidade e compreensibilidade do sistema, diminuindo assim o tempo de desenvolvimento (ou manutenção) do mesmo.

Este artigo é referência para milhares de outros artigos¹ no que diz respeito à modularidade, e a ideia de tentar caracterizar o conceito a partir da visão dos projetistas de linguagem surgiu através dele.

As características observadas são:

- Ocultação de informações;
- Módulos independentes.

3.2 Operating System Performance

[Lynch \(1972\)](#) apresenta a avaliação do desempenho de sistemas operacionais, propondo também o desenvolvimento de um modelo global de sistema que irá causar impacto no que diz respeito à modularização. Para ele o desempenho individual de cada módulo, por apresentar pouca informação, não pode ser somado para encontrar o

¹ <http://dl.acm.org/citation.cfm?id=361623>

desempenho geral do sistema. São duas as formas existentes de modularização, sendo elas horizontal e vertical. A primeira também é conhecida como modularidade funcional, e leva em consideração a Lei de Conway, que diz que um sistema operacional será uma cópia uma-a-uma de cada uma de suas partes (LYNCH, 1972 apud MARTIN, 1967), e a segunda é gerada a partir da programação estrutural (LYNCH, 1972 apud WIRTH, 1971)[...], onde cada camada é criada a partir das funções da camada subjacente. O artigo se refere ao uso das modularizações em conjunto, pois separadas não contribuem tanto com o desempenho do sistema, e que o objetivo futuro é criar um modelo geral que possibilite a interação de módulos que sejam totalmente independentes entre si. Implicitamente, LYNCH descreve a modularidade como sendo a estrutura do software a nível de sistema (modularidade vertical ou programação estruturada) e a nível de seções (modularidade horizontal), utilizando módulos independentes.

As características observadas são:

- Módulos independentes;
- Modularidade vertical, ou programação estruturada a nível de camadas;
- Modularidade horizontal, ou modularidade funcional a nível de seções.

3.3 Computerized Dispatching System

Silverman (1973) apresenta o desenvolvimento de um sistema de despacho de veículos através de rádios, e para esse desenvolvimento utiliza-se modularidade de hardware e software, multi-compartilhamento e operações em primeiro e segundo plano. Esse tipo de sistema apresenta muitos problemas, como por exemplo, o fato da população estar sempre crescendo e a quantidade de usuários também, o que gera mais demanda ao sistema, e de acordo com SILVERMAN, utilizar modularidade, sendo os módulos interligados através de um link, pode resolver todos os problemas, uma vez que a implementação reduzirá custos de adaptação. A abordagem modular utilizada consiste de criar formulários – interligados através de uma base de dados em comum – para cada departamento que utiliza o sistema, que os recebe como entrada e retorna a saída de acordo. Utilizar essa abordagem permite alteração ou atualização destes formulários (módulos) sem alterar todo o sistema, ou torná-lo obsoleto. E por fim, a interação com o sistema se dá através do módulo de entrada de dados, que interage com o controlador do sistema e coordena os outros módulos. Implicitamente, entende-se modularidade como sendo um sistema dividido em módulos, controlados por um módulo que interage com os outros e retorna resultados a partir da saída destes.

As características observadas são:

- Módulos independentes;
- Módulos de interface para interação com o usuário;
- Módulo controlador dos outros módulos;
- Base de dados em comum.

3.4 Organizing Computer Systems for Learnability and Useability

[Anagnostopoulos \(1973\)](#) apresenta as técnicas structured, modular e soft que, quando utilizadas contribuem com a organização dos sistemas computacionais, tornando-os mais fáceis de aprender a manusear. A structured, ou estrutura, diz respeito a programação estruturada, ou diferentes níveis de um sistema ao qual o usuário tem acesso, e a técnica modular está relacionada a desenvolver sistemas utilizando modularidade, o que torna o manuseio destes mais simples e fácil de gerenciar. De certa forma, a modularidade e a programação estruturada são utilizadas juntas, pois a programação estruturada divide o sistema em níveis e a modularidade em seções. A técnica de soft, ou mutável, é utilizada para permitir que o compilador seja extensível. E por fim, utilizando essas técnicas o sistema Brown University Graphics System (BUGS) é desenvolvido, com módulos separados e independentes entre si. Enfim, entende-se que para [ANAGNOSTOPOULOS](#) a modularidade consiste da divisão de sistemas em níveis (programação estruturada) delimitando o acesso dos usuários, e em seções, utilizando módulos independentes.

As características observadas são:

- Ocultação de informações;
- Programação estruturada a nível de sistema;
- Modularidade a nível de seções.

3.5 A modularized package of dual algorithms for solving constrained nonlinear programming problems

[Minkoff \(1975\)](#) apresenta o desenvolvimento de um pacote de algoritmos para experimentação numérica que é modularizado e que permite alterações de cada módulo de forma independente. [MINKOFF](#) separa os módulos em lógico (ou módulo de interface), de código e auxiliares. O módulo lógico executa tarefas de manipulação de dados, interligando-se aos outros módulos. Os módulos de código estão um nível abaixo do módulo lógico e os módulos auxiliares conduzem as tarefas. Como vantagens da modularidade, o artigo traz: a execução de um módulo sem a necessidade de outros, a compreensibilidade do algoritmo

e a manutenção de bibliotecas. E como desvantagem apresenta o fato de quem nem todos os algoritmos são aptos a serem modulares e que a modularidade aumenta o tempo de computação, sendo que como solução para esse segundo caso basta utilizar um pré-processador para integrar todos os módulos em um único fluxo de código. Implicitamente, entende-se que para [MINKOFF](#) a modularidade consiste de módulos independentes, que são controlados e interligados por um módulo lógico, utilizando módulos auxiliares para tal.

As características observadas são:

- Ocultação de informações;
- Módulos independentes;
- Módulos separados entre módulo lógico, módulos abaixo dos lógicos e módulos auxiliares;

3.6 A pragmatic proposal for the improvement of program modularity and reliability

[Spier \(1975\)](#) apresenta que uma boa modularidade é facilmente reconhecida quando é vista, mas que não é fácil de alcançar, uma vez que não existe um algoritmo para tal. No entanto, a falta de meios para alcançá-la não garante que não se possa chegar a algo próximo dela, pois a modularidade é uma “ferramenta intelectual” que permite a decomposição de sistemas em níveis aninhados de abstração, dentro da compreensão humana, sendo esta tratada de forma hierárquica em relação ao fluxo de controle do sistema. Sendo assim, a modularidade acaba sendo uma questão de interesse administrativo, pois se o sistema é pequeno e compreensível suficiente, ele não precisa ser modularizado. Um sistema modular é composto por módulos independentes, onde cada módulo possui um conjunto de funções e é encapsulado para que se torne “imune” à influências externas. Enfim, dentre as vantagens que a modularidade pode trazer, estão a melhor compreensão de sistemas e a alteração de módulos – seja pelo seu próprio desenvolvedor ou por outro programador – sem a necessidade de todo o sistema ser alterado para tal. Assim, entende-se que para [SPIER](#) a modularidade pode ser alcançada, pois consiste de módulos independentes, que são um conjunto de funções e são encapsulados para não sofrerem influências externas.

As características observadas são:

- Ocultação de informações;
- Não existe algoritmo para modularidade;

- Módulos independentes;
- Módulos são um conjuntos de funções;
- Módulos são encapsulados de forma que não são influenciados por agentes externos.

3.7 CAMAC Software for the Hot Fuel Examination Facility Real-Time Computer System

Lau, Ramsthaller e Just (1975) abordam em seu texto o desenvolvimento de um sistema para controlar a verificação de combustíveis. Esse sistema afirma utilizar modularidade de software, tornando seu software mais flexível, bem como a resolução de problemas neste. Cada módulo deste software tem uma sub-rotina responsável pela sua execução, e em caso de módulos iguais, sub-rotinas gerais são utilizadas para evitar a redundância. Estas sub-rotinas se comunicam através de parâmetros, e cada uma possui uma entrada e uma ou mais saídas. A ordem de execução é independente, tornando possível a alteração do lugar dos módulos, bem como sua modificação. Implicitamente, entende-se modularidade como sendo módulos independentes, constituídos de uma sub-rotina, com entradas e saídas e que se comunicam através da passagem de parâmetros.

As características observadas são:

- Módulos independentes, porque a ordem de execução é independente;
- Cada módulo possui uma sub-rotina responsável por seu funcionamento, bem como entradas e saídas;
- Comunicação através da passagem de parâmetros.

3.8 Modularization Around a Suitable Abstraction

Zilles (1975) apresenta a modularidade como sendo uma das boas práticas de programação, tendo em vista que esta permite o desenvolvimento de sistemas de forma mais rápida pois possibilita o desenvolvimento de módulos paralelamente, bem como testes mais rápidos e manutenção do sistema de forma mais simples. O abstract mostra que, no que diz respeito à modularidade, muitos autores dão ênfase ao fluxo de controle, e que isso pode fazer com que um módulo tenha mais conhecimento do que ele necessita, o que consequentemente acaba gerando uma certa dependência entre módulos que podem não estar relacionados. Os módulos devem ser criados utilizando técnicas de abstração, de forma que apenas os aspectos necessários sejam apresentados, enquanto os outros detalhes de implementação sejam excluídos. A abstração processual é uma das técnicas utilizadas,

que permite a modularização de uma sequência de ações, contruindo uma interface de nível superior mais simples para o usuário. Enfim, [ZILLES](#) diz que para utilizar a modularidade é importante considerar a abstração de dados, de forma que o acesso à dados importantes seja limitado.

As características observadas são:

- Módulos independentes porque um módulo não pode estar relacionado a outro;
- Utilizar abstração no desenvolvimento de módulos.

3.9 Program design by a multidisciplinary team

[Voigt \(1975\)](#) apresenta a utilização de diagramas funcionais aninhados como forma de especificar um projeto de um Sistema de Elementos Finitos. Cada diagrama aninhado é utilizado para determinar uma função, tornando possível assim analisar o sistema desde as funções mais importantes até as mais simples (abordagem *top-down*). Esse tipo de diagrama foi utilizado devido à sua capacidade de limitar o contexto de atividade e de expandir de cima para baixo. Um módulo é um diagrama, ou um conjunto destes, onde todas as informações importantes são passadas, permitindo inclusive que outros desenvolvedores troquem informações sobre às especificações de outros módulos. A forma de comunicação entre os módulos é através de parâmetros, onde cada módulo possui entradas e saídas. Os testes são independentes para cada módulo e as variáveis globais são nomeadas e utilizadas para casos em que cruzem funções principais. Assim, entende-se que para [VOIGT](#) a modularidade consiste de um conjunto de módulos independentes, onde cada módulo é um conjunto de funções e que se comunica com outros módulos utilizando entradas e saídas, passagem de parâmetros e variáveis globais de forma que um módulo não interfira no outro.

As características observadas são:

- Módulos independentes porque permite testes independentes;
- Módulos são um conjuntos de funções;
- Comunicação através da passagem de parâmetros;
- Utilização de variáveis globais.

3.10 The Effect of Certain Modular Design Principles on Testability

[Edwards \(1975\)](#) fala de uma forma geral de modularidade, referenciando-a como modularidade controlada, que é uma junção de técnicas para decompor um sistema em

módulos. As técnicas utilizadas são a definição do problema, a quebra de funções em sub funções gerenciáveis (onde cada sub-função deve ter objetivos funcionais claramente definidos), o controle da complexidade e a utilização de partes reutilizáveis. Também é abordado, que na teoria é possível dividir qualquer função em partes menores, entretanto na prática é necessário observar por exemplo, a complexidade do módulo, para ter certeza se isso é praticável. Se praticável e a função é resultante de pequenas partes interligadas, deve ser possível especificar cada parte (entradas e saídas) individualmente, entretanto, essa especificação não garante que os módulos sejam independentes, pois um módulo pode chamar outro módulo. Outro ponto é em relação à comunicação entre os módulos. Esta é feita através de uma função de controle, que chama um módulo, executa suas funções, encerra e depois outro módulo é chamado. Essas chamadas são feitas através de módulos de ramificação, que apenas indicam se a comunicação deve ou não ocorrer, dependendo para tal, dos dados/resultados obtidos nos módulos anteriores. Enfim, funções de um módulo devem ser independentes da fonte de entrada, da saída e do histórico deste e sua precisão e confiabilidade devem ser garantidas. Portanto, entende-se que para [EDWARDS](#) a modularidade consiste de módulos independentes que possuem entradas e saídas, controlados por um módulo de controle e que acompanham o fluxo de controle do sistema.

As características observadas são:

- Módulo de Controle ou Função de controle para coordenar os módulos;
- Módulos possuem entradas e saídas;
- Fluxo de controle do sistema.

3.11 Programming Languages: The First 25 Years

[Wegner \(1976\)](#) apresenta o desenvolvimento de linguagens de programação nos primeiros 25 anos. Em relação à modularidade, o primeiro ponto destacado é sobre abstração. A abstração é muito importante no desenvolvimento de linguagens modulares, pois a ideia da modularidade é diminuir a complexidade dessas linguagens, dividindo-as em módulos abstratos, onde cada módulo desempenha uma função. Um breve histórico de linguagens modulares é apresentado, mostrando as características destas em relação à modularidade. Fortran utiliza modularidade através de sub-rotinas, enquanto Algol 60 utiliza módulos aninhados, que de certa forma não contribui para o uso de módulos independentes. A Simula, é uma generalização flexível de Algol 60, pois separa a citação e exclusão de instâncias de entrada e saída, preservando assim os dados. A linguagem Clu utiliza módulos que possuem atributos ocultos (estrutura de dados) e procedimentos exportáveis ([WEGNER, 1976](#) apud [LISKOV, 1974](#)), e a linguagem Alphard caracteriza

seus módulos como formulários (WEGNER, 1976 apud WULF; LONDON; SHAW, 1976) e utiliza a mesma ideia da Clu. A linguagem Módula utiliza monitores e módulos em Pascal (WEGNER, 1976 apud WIRTH, 1977), utilizando abstração em sua implementação, de modo que seja definida uma lista de objetos usados dentro e fora do módulo. Dos problemas enfrentados por todas essas linguagens, estão a definição de interface do módulo e a interconexão entre eles.

As características observadas são:

- Ocultação de informações;
- Módulos independentes e abstratos;
- Módulos são um conjunto de sub-rotinas ou formulários;
- Utilização de objetos externos para serem visíveis pelo resto do programa.

3.12 A Balanced View of MUMPS

Wasserman e Sherertz (1976) apresentam a linguagem MUMPS, que tem a modularidade como uma de suas características. Dentro desse contexto, o autor apresenta a modularidade como sendo uma técnica de grande valor num projeto de software (WASSERMAN; SHERERTZ, 1976 apud PARNAS, 1972). Implicitamente, o autor descreve modularidade como um conjunto de módulos que precisam ter um conjunto de entradas e saídas bem definidos, assim como a relação e ligação entre eles deve ser claramente visível. Outro ponto importante, é o fato de que algumas linguagens de programação, como por exemplo PASCAL, utiliza procedimentos ou até mesmo funções como forma de representação de módulos e que as entradas e saídas destes módulos são especificadas através de parâmetros, o que acaba não sendo uma boa escolha, pois muitas linguagens permitem o uso de variáveis globais, e formas diferentes de analisar os parâmetros (valor ou referência), e que uma alternativa, que a propósito é utilizada no desenvolvimento da linguagem MUMPS, é a utilização de sub-rotinas como forma de comunicação entre os módulos.

As características observadas são:

- Comunicação através de sub-rotinas.

3.13 Possible futures and present actions

Hopper (1976) apresenta o crescimento da tecnologia e afirma, que muito do que se tem hoje, será substituído e que cada ação a ser tomada está diretamente relacionada a linguagens de programação, modularidade e documentação. O texto também relata

o crescimento de sistemas, afirmando que estes crescem a tal ponto que é impossível modificá-los, tornando-se necessário repartí-los em partes menores, de forma a se tornarem mais gerenciáveis. Implicitamente, entende-se que para o [HOPPER](#), modularidade é a capacidade de dividir um grande sistema, em pequenas partes que se comunicam através de links, o que contribui para uma melhor compreensibilidade do sistema, e manutenção mais prática.

As características observadas são:

- Módulos são partes menores de um sistema;
- Comunicação através de links.

3.14 The use of Abstract Data Types to Simplify Program Modifications

[Linden \(1976\)](#) aborda o uso de tipo abstrato de dados como forma de modularizar um programa, mostrando o desenvolvimento de um software de números primos para tal. Um tipo abstrato de dados é definido como um conjunto de operações agrupadas ao redor de estruturas de dados comuns, que podem ser ocultas de operações que não fazem parte desse tipo abstrato de dados, pois permitem encapsulamento de estruturas de dados. O artigo fala que utilizar essa forma de modularidade permite um melhor crescimento e modificação do software, e que para isso, a modularidade deve fazer com que três requisitos sejam satisfeitos, sendo eles: módulos independentes, módulos pequenos (módulos grandes são complexos e difíceis de modificar) e as tarefas de cada módulo devem ser realizadas de forma mais direta possível. Enfim, a partir do texto entende-se modularidade por módulos independentes (onde cada módulo é um tipo abstrato de dados), pequenos, objetivos e encapsulados.

As características observadas são:

- Ocultação de informações graças ao encapsulamento;
- Módulos independentes;
- Módulos pequenos e diretos.

3.15 Visibility and types

[Koster \(1976\)](#) fala da ligação entre tipos abstratos de dados e ocultação de informação em grandes sistemas. O artigo aborda a divisão de um sistema em unidades. Uma unidade pode ser considerada um módulo e um objeto pode ser considerado um valor, tipo

ou sub-rotina. Essas unidades se comunicam através de objetos. Os objetos podem ser objetos de unidade, ou objetos de outra unidade, e através do uso de interface são apresentadas formas de restringir a visibilidade destes. Para tal, duas formas de estruturação de módulos são apresentadas, sendo elas a estrutura de blocos, onde cada módulo tem apenas uma seta de entrada, e a anti estrutura de blocos, onde cada módulo contém uma seta para apenas outro módulo, sendo que um módulo só conhece o que está abaixo de si. Outro ponto citado, é o fato de que utilizar estruturas aninhadas dentro de estruturas, ou seja, módulos dentro de módulos, é a forma mais lucrativa de visualizar grandes sistemas. E por fim, dentro dessa ideia foi proposto um modelo de sistema em camadas, que correspondem a níveis de abstração. No sentido vertical, o modelo representa interface de abstração e no sentido horizontal interface de extensão, sendo que no horizontal, menos restrições de visibilidade devem ser aplicadas, e a comunicação entre essas camadas é feita através de tipos. A partir disso, entende-se que para [KOSTER](#) a modularidade pode ser vertical, sendo essa a nível de sistema e horizontal a nível de seções, e que no primeiro caso, é de suma importância a ocultação de informações.

As características observadas são:

- Ocultação de informações;
- Módulos aninhados;
- Modularidade vertical a nível de sistema;
- Modularidade horizontal a nível de seções.

3.16 Notes on the design of Euclid

[Popek et al. \(1977\)](#) apresenta em seu conteúdo questões sobre a linguagem de programação Euclid. Uma das questões abordadas são os módulos desta linguagem. [POPEK et al.](#) apresenta módulo como um construtor de tipo que contém uma estrutura de dados, um conjunto de sub-rotinas, componentes de inicialização e finalização para sua manipulação e uma lista de exportação de nomes (exigência da linguagem). Os módulos compartilham uma base de dados comum e alocação de armazenamento é feita de modo que não haja confusão de tipos ou sobreposição de variáveis, garantindo assim a integridade destas. E em casos de variáveis de endereço fixo, existem os módulos chamados dependentes da máquina, que são utilizados para encapsular as características dessas variáveis.

As características observadas são:

- Ocultação de informações;

- Módulos são um conjunto de sub-rotinas, estrutura de dados, entradas e saídas, e nomes visíveis;
- Base de dados comum.

3.17 Programming languages for introductory computing courses: a position paper

[Solntseff \(1978\)](#) apresenta questões sobre linguagens de programação que são utilizadas em cursos introdutórios e aborda questões sobre modularidade e estruturas de procedimentos. De acordo com [SOLNTSEFF](#) um programador deve identificar objetos relacionados e a partir disso definir e nomear uma unidade de estruturas de dados independente de suas instâncias particulares. Uma unidade de estrutura de dados também é conhecida na linguagem Simula como classe e incorpora dentro de si componentes de dados e estruturas de ação. Uma extensão do conceito classe, os chamados tipos abstratos de dados, possuem um grupo de objetos, funções e operações em seu corpo e ocultam detalhes de implementação do usuário utilizando para tal procedimentos parametrizados, ou seja, especificando todas as operações que podem ser executadas nas instâncias das classes, fazendo com que assim o usuário tenha conhecimento apenas da “visão externa” de um objeto estruturado. Em outras linguagens, os tipos abstratos de dados são conhecidos como formulários, cluster e módulos. Enfim, entende-se modularidade nesse caso como sendo um conjunto de tipos abstratos de dados, cada qual com funções, operações e objetos definidos, que possuem procedimentos parametrizados e que ocultam informações do restante do sistema.

As características observadas são:

- Ocultação de informações;
- Procedimentos parametrizados;
- Módulos são tipos abstratos de dados, e possuem um conjunto de objetos, funções e operações.

3.18 Software engineering with standard assemblies

[Lanergan e Poynton \(1978\)](#) fala de técnicas de modularidade funcional que são utilizadas para preparação de módulos que podem ser utilizados em muitas aplicações empresariais. Os módulos são sub-rotinas, e algumas vezes são utilizados para funcionalidades específicas, fazendo parte de bibliotecas, e permitindo reutilização. Neste caso específico, os módulos, apesar de trabalhar em conjunto, foram separados por suas funcionalidades, sendo

eles: descrições de arquivo, descrições de registros, rotinas editáveis, rotinas funcionais, entrada e saída de dados, interface de módulos, argumentos de pesquisas e chamadas de procedimento. Em adição à esses módulos foram implementadas estruturas lógicas, que utilizam esses módulos e que realizam outras tarefas de aplicações empresariais, sendo estas divididas, em Divisão de Identificação pré-escrita, de ambiente, de dados e Divisão de Procedimentos. Assim entende-se que a modularidade são módulos separados por suas funcionalidades, compostos de sub-rotinas e são independentes.

As características observadas são:

- Módulos independentes, pois são divididos por funcionalidades;
- Módulos são sub-rotinas, e são separados de acordo com suas funcionalidades.

3.19 The modularity of MSC/NASTRAN

Gloudeman (1978) apresenta a modularidade do programa MSC/NASTRAN, que é um programa de larga escala. Esse programa é dividido em módulos funcionais, que são subdivididos entre sub-rotinas e possuem entradas, saídas e um conjunto de parâmetros. Os módulos funcionais, nem sempre são independentes, pois quando relacionados a um mesmo problema, podem comunicar-se. Em contra partida, se independentes, é possível adicionar, modificar ou corrigir erros nesses módulos sem que todos os outros módulos necessitem mudar, permitindo também a realização de testes individuais em cada módulo. E por fim, utilizar modularidade contribui na facilidade de uso, modificabilidade, manutenção e eficiência do programa.

As características observadas são:

- Módulos funcionais são um conjuntos de sub-rotinas, que possuem entrada, saída e conjunto de parâmetros, e podem ou não serem independentes.

3.20 The Smalltalk-76 programming system design and implementation

Ingalls (1978) apresenta um sistema baseado na comunicação de objetos, que oferece flexibilidade, modularidade e compacidade. No que diz respeito a modularidade, é dito que graças à referenciação e descrição de objetos, envio de mensagens e utilização de classes, é possível que os módulos sejam independentes. Os objetos são criados e manipulados através do envio de mensagens. A resposta de uma mensagem é implementada por um método que lê ou escreve em um arquivo de dados ou envia respostas adicionais para alcançar a resposta desejada. As classes contém instâncias, mensagens que podem ser respondidas e

métodos para calcular respostas, e são o que permite a modularidade, pois descrevem as mensagens externas, os detalhes internos dos métodos de respostas e a representação de dados nas instâncias, além de permitirem a existência de sub-classes que herdam de uma classe pai e são modificadas da forma que melhor couber. (INGALLS, 1978)

As características observadas são:

- Módulos independentes;
- Comunicação através de mensagens;
- Utilização de herança.

3.21 Ocular motility test administration and analysis by computer in strabismus and amblyopia evaluation

Simons, Moss e Reinecke (1978) apresenta um sistema de software e hardware utilizado por oftamologistas para análise de doenças oculares. De acordo com SIMONS; MOSS; REINECKE a modularidade do software permite a adição ou modificação da sequência de testes ou rotinas, pois podem ser utilizados em qualquer sequência de execução. O software é dividido em quatro partes, sendo elas o módulo executivo, que controla a execução de todos os outros módulos; o módulo de tempo real, que executa os testes e grava os resultados; o módulo de exibição, que captura os movimentos oculares e armazena, para ser exibido da forma que o operador desejar; e um conjunto de módulos que são responsáveis por executar análises dos movimentos oculares através de gráficos interativos na tela, e de rotinas auxiliares.

As características observadas são:

- Módulos independentes, porque a sequência de execução pode ser alterada;
- Módulo controlador;
- Módulos auxiliares.

3.22 A small-scale operating system foundation for microprocessor applications

Kahn (1978) apresenta um estudo de caso de um sistema de tempo real com o intuito de discutir metodologias de projetos de software, e enfatiza como utilizar generalidade e modularidade pode ser útil na produção de sistemas de software. O sistema apresentado foi desenvolvido utilizando *Programming Language for Microcomputers* - PL/M ou Linguagem

de Programação para Micro computadores, (KAHN, 1978 apud MCCRACKEN, 1978) e foi utilizado o conceito de tipo abstrato de dados para ocultar informações. Cada módulo é relacionado a uma estrutura de dados e fornece para outros módulos procedimentos públicos e variáveis, sendo que as únicas suposições que podem ser feitas por agentes externos são as especificadas por esses procedimentos. Os módulos são independentes porque mudanças podem ser feitas nessas estruturas sem afetar as outras partes do sistema. Segundo KAHN, usar essa estrutura contribui com a compreensibilidade do sistema, bem como facilita a manutenção deste. Entende-se que para KAHN a modularidade consiste em dividir um sistema em tipos abstratos de dados (ou módulos) independentes, que possuem procedimentos, e que possuem ocultação de informações.

As características observadas são:

- Ocultação de informação;
- Módulos independentes;
- Módulos são tipos abstratos de dados e possuem procedimentos públicos e variáveis, sendo relacionados a uma estrutura de dados.

4 Análise

Após a leitura dos artigos selecionados, fica evidente que o objetivo da modularidade em sistemas de software ou linguagens de programação é prover melhorias como a compreensibilidade do sistema (PARNAS, 1972; SPIER, 1975; KAHN, 1978), e a manutenção deste (GLOUDEMANN, 1978; HOPPER, 1976; LAU; RAMSTHALER; JUST, 1975). Entretanto, em relação às formas de alcançá-la, ainda há uma divergência de ideias a serem consideradas. As características observadas dos artigos analisados foram estruturadas em forma de quadro (Apêndice C).

Muito se fala em independência dos módulos, sendo esta talvez a característica mais desejada, pois é através dela que as melhorias citadas (compreensão do sistema e manutenção) são alcançadas, uma vez que, se os módulos são de fato independentes, a alteração ou adição de novos módulos não interferem no sistema (ou linguagem) como um todo, contribuindo com agilidade e praticidade no desenvolvimento ou manutenção. Lynch (1972) fala em utilizar módulos independentes futuramente, como sendo uma das formas de alcançar modularidade. Silverman (1973) aborda módulos independentes como formulários interligados a uma base de dados em comum. Minkoff (1975) separou os módulos em módulo lógico, módulos de código e módulos auxiliares, de forma que o módulo lógico coordene os outros, permitindo que estes sejam independentes entre si. Spier (1975) afirma que não existe um algoritmo para alcançar a modularidade, mas que se alcançada, os módulos (conjunto de funções) devem ser independentes. Lau, Ramsthaller e Just (1975) fala que a ordem de execução dos módulos é independente, deixando implícito que os módulos também são. Zilles (1975) fala de testes independentes para cada módulo, o que também deixa implícito que os módulos são independentes. Voigt (1975) fala da independência de testes, deixando subentendido que os módulos são independentes, e que possuem entradas e saídas e se comunicam através da passagem de parâmetros. Wegner (1976) aborda módulos como cada um responsável por uma função. Wasserman e Sherertz (1976) relata que a utilização de comunicação através de parâmetros pode quebrar a independência entre os módulos. Linden (1976) se refere a módulos como sendo tipos abstratos de dados que são um conjunto de operações agrupadas. Ingalls (1978) afirma que a independência dos módulos é realizada graças à referência e descrição de objetos, comunicação através de envio de mensagens e utilização de classes. Simons, Moss e Reinecke (1978) aborda a divisão de módulos de forma que o fluxo de controle não faça diferença e por fim Kahn (1978) fala que a independência do módulo se dá pela sua modificação individual sem que o sistema inteiro necessite mudar.

Outra característica mencionada é a ocultação de informação (*information hiding*), que é citada por Parnas (1972). Essa característica diz respeito ao conteúdo de cada módulo,

de modo que este não seja visível aos outros módulos e que mesmo assim a comunicação entre eles seja possível. Wegner (1976) fala da utilização de atributos ocultos e procedimentos exportáveis. Poppek et al. (1977) fala da utilização de encapsulamento de módulos que contem variáveis de endereço fixo. Spier (1975) se refere ao encapsulamento de módulos de forma a torná-los “imunes” à influências externas. Linden (1976) fala de encapsular as estruturas de dados de um tipo abstrato de dados (módulos), de forma que estas sejam ocultas às outras estruturas que não fazem parte desse tipo abstrato de dados e Kahn (1978) fala da utilização de tipos abstratos de dados como forma de ocultar informações.

Outras características foram definidas. Em relação à comunicação entre os módulos, esta pode ser feita através de uma base de dados em comum (SILVERMAN, 1973; MINKOFF, 1975; POPEK et al., 1977), através da passagem de parâmetros (LAU; RAMSTHALER; JUST, 1975; VOIGT, 1975; WASSERMAN; SHERERTZ, 1976; GLOUDEMAM, 1978), ou através do envio de mensagens, utilizando mecanismo de herança (INGALLS, 1978). No que diz respeito à composição de módulos, esta pode se dar por um conjunto de funções ou sub-rotinas (LAU; RAMSTHALER; JUST, 1975; EDWARDS, 1975; WEGNER, 1976; WASSERMAN; SHERERTZ, 1976; KOSTER, 1976; POPEK et al., 1977; LANERGAN; POYNTON, 1978; GLOUDEMAM, 1978), ou por tipos abstratos de dados (KOSTER, 1976; SOLNTSEFF, 1978; KAHN, 1978). Algumas características como modularização horizontal e vertical e/ou programação estruturada (LYNCH, 1972; ANAGNOSTOPOULOS, 1973), também são descritas, bem como a utilização de variáveis globais (WASSERMAN; SHERERTZ, 1976; VOIGT, 1975).

É visível que a divergência de ideias nessa época era muito grande, pois foi quando o conceito tornou-se “conhecido”, e quando os pesquisadores perceberam que é um método de desenvolvimento que promete benefícios considerados na engenharia de software, muito foi incorporado ao conceito. Portanto, baseando-se nas ideias apresentadas de cada autor é possível estruturar as características em comum que caracterizem o conceito de modularidade desta época (1972–1978).

Características estruturadas:

- Independência entre os módulos

Cada módulo é construído com sub-rotinas (ou funções), com um conjunto de entradas e saídas bem definidas, compartilhando uma base de dados. Um módulo também pode ser visto como um tipo abstrato de dados, compartilhando uma base de dados, que ainda assim não deixa de ser independente.¹ Utilizar essas características permite que um módulo seja desenvolvido e executado independente dos outros módulos do sistema.

¹ A utilização de módulos aninhados não contribui com a independência entre eles (WEGNER, 1976), pois um acaba servindo de base para o outro, portanto não pode ser levado em consideração quando a ideia é de independência entre os módulos.

- Ocultação de informações (*information hiding*)

Em relação à ocultação de informações, as ideias já possuem uma correlação maior. Entende-se que utilizando encapsulamento de dados e abstração, e deixando disponíveis os procedimentos que podem ser visualizáveis como “procedimentos exportáveis” para outros módulos, é possível alcançar a ocultação de informações (*information hiding*).

- Modularidade vertical ou Programação Estruturada

Esses termos estão relacionados à forma de modularidade em relação ao fluxo de sistema, e desde que a ordem de execução não interfira na adição ou remoção de um módulo, não oferece problemas ao restante do sistema.²

- Modularidade horizontal

Relacionado à formas de dividir um sistema em seções. Desde que não crie dependência entre os módulos e que o fluxo de execução não interfira, e não seja utilizada em conjunto com a Modularidade Vertical, esta também pode ser considerada como uma forma de alcançar a modularidade;

- Comunicação entre os módulos

Esta pode ser feita através de sub-rotinas ou utilizar um módulo de controle (módulo de interface) que realiza a comunicação dos outros módulos de forma que eles não se comuniquem entre si.^{3,4}

4.1 Considerações

As características estruturadas a partir da análise dos dados independem umas das outras, tornando possível a utilização de todas sem que uma anule a outra.

Deste modo, a partir da revisão sistemática de literatura utilizada, pode-se concluir que o termo modularidade está relacionado à estrutura de sistemas de softwares e linguagens (no contexto do trabalho), e à formas de facilitar a evolução, modificação ou manutenção destes. Cada autor cita algumas das características possíveis para alcançá-la, e as características propostas entram em consenso nas opiniões dos autores analisadas. E assim, conclui-se que não existe uma definição que pode-se dizer correta sobre modularidade, mas sim técnicas que podem ajudar a obtê-la, o que nem sempre é suficiente.

² A modularidade horizontal utilizada junto com a Modularidade Vertical pode ser relacionada à módulos aninhados, o que pode gerar dependência entre os módulos como já mencionado, portanto não é uma característica considerável.

³ A utilização de herança faz com que informações extras sejam solicitadas (MERNIK; UMER, 2005), quebrando o conceito de ocultação de informações, e portanto não é utilizada

⁴ A utilização de passagem de parâmetros não é uma boa opção devido a geração de dependência entre os módulos (WASSERMAN; SHERERTZ, 1976)

5 Conclusão

A modularidade teve suas primeiras características quando Parnas (1972) a descreveu como a capacidade de dividir de um sistema em módulos independentes, que permitiam mudanças independentes sem a necessidade de informações adicionais de cada módulo.

O conceito foi incorporado ao longo dos anos e muitos sistemas e linguagens de programação afirmam ser modulares, mas violam as características descritas por Parnas (1972) quando precisam de informações extras para manipular módulos (MERNIK; UMER, 2005). Sendo assim, não se sabe ao certo quais características devem ser levadas em consideração durante o desenvolvimento destas linguagens e softwares para que seja possível afirmar que realmente são modulares.

A partir do problema apresentado, foi feita uma revisão sistemática de literatura entre os anos 1972–1978. Através de critérios estabelecidos, foram analisados 22 artigos, e as características mais frequentes e que foram observadas em maior parte dos artigos são: módulos independentes através da utilização de sub-rotinas ou tipos abstratos de dados, base de dados comum e entrada/saída bem definidas; ocultação de informação através da utilização de encapsulamento e abstração de dados; e comunicação de módulos através de sub-rotinas.

Enfim, conclui-se que o termo modularidade é relacionado à estrutura e às formas de alteração, remoção ou adição de novos módulos em softwares ou linguagens de programação (como é abordado no contexto do trabalho), e sempre é acompanhado de técnicas para alcançá-lo, mas que assim como na realidade atual, nem sempre essas técnicas são suficientes para tal.

5.1 Trabalhos Futuros

Como a quantidade de conteúdo sobre o tema é muito extensa e uma análise completa até os presentes dias é inviável de se desenvolver no escopo do trabalho, a proposta de trabalhos futuros é, a partir dos requisitos descritos (estabelecendo um novo intervalo de tempo), procurar as características que foram incorporadas ao conceito de modularidade ao longo do tempo, compará-las com as que já foram descritas, e assim apresentar um conceito formado de modularidade e caracterizá-la no desenvolvimento de linguagens de programação a partir da visão dos projetistas de linguagens.

E por fim, outra sugestão de trabalho futuro é apresentar formas de medir a modularidade em um produto, ou de como gerenciar a modularidade de modo que esta não interfira no desempenho de um sistema.

Referências

- ANAGNOSTOPOULOS, P. C. Organizing computer systems for learnability and useability. *Conference record of the 6th annual workshop on Microprogramming*, ACM, p. 65–70, 1973. Citado 2 vezes nas páginas 26 e 39.
- BARRETT, H. C.; KURZBAN, R. Modularity in cognition: framing the debate. *Psychological Review*, American Psychological Association, p. 628, 2006. Citado na página 17.
- EDWARDS, N. P. The effect of certain modular design principles on testability. *SIGPLAN Not.*, ACM, v. 10, n. 6, p. 401–410, 1975. Citado 3 vezes nas páginas 29, 30 e 39.
- GLOUDEMANN, J. F. The modularity of msc/nastran. In: *Computer Software and Applications Conference, 1978. COMPSAC '78. The IEEE Computer Society's Second International*. [S.l.]: IEEE, 1978. p. 444–446. Citado 3 vezes nas páginas 35, 38 e 39.
- HOPPER, G. M. Possible futures and present actions. *SIGMINI Newsl.*, ACM, v. 2, n. 4-5, p. 6–8, 1976. Citado 3 vezes nas páginas 31, 32 e 38.
- INGALLS, D. H. H. The smalltalk-76 programming system design and implementation. In: *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. [S.l.]: ACM, 1978. (POPL '78), p. 9–16. Citado 4 vezes nas páginas 35, 36, 38 e 39.
- JOHNSTONE, A.; SCOTT, E.; BRAND, M. van den. Modular grammar specification. *Science of Computer Programming*, Elsevier, p. 23–43, 2014. Citado na página 17.
- KAHN, K. C. A small-scale operating system foundation for microprocessor applications. *Proceedings of the IEEE*, IEEE, v. 66, n. 2, p. 209–216, 1978. Citado 4 vezes nas páginas 36, 37, 38 e 39.
- KOSTER, C. H. Visibility and types. In: *ACM SIGPLAN Notices*. [S.l.]: ACM, 1976. v. 11, n. SI, p. 179–190. Citado 3 vezes nas páginas 32, 33 e 39.
- LANERGAN, R.; POYNTON b. Software engineering with standard assemblies. In: *ACM. Proceedings of the 1978 annual conference*. [S.l.], 1978. v. 2, p. 507–514. Citado 2 vezes nas páginas 34 e 39.
- LAU, L. D.; RAMSTHALER, J. A.; JUST, F. H. Camac software for the hot fuel examination facility real-time computer system. *IEEE Transactions on Nuclear Science*, v. 22, n. 1, p. 540–542, 1975. Citado 3 vezes nas páginas 28, 38 e 39.
- LINDEN, T. A. The use of abstract data types to simplify program modifications. In: *Proceedings of the 1976 Conference on Data: Abstraction, Definition and Structure*. [S.l.]: ACM, 1976. p. 12–23. Citado 3 vezes nas páginas 32, 38 e 39.
- LISKOV, B. *A note on CLU. Computation Structures Group Memo. 112*. [S.l.]: MIT, Cambridge, Mass, 1974. Citado na página 30.

- LYNCH, W. C. Operating system performance. *Communications of the ACM*, ACM, v. 15, n. 7, p. 579–585, 1972. Citado 4 vezes nas páginas 24, 25, 38 e 39.
- MARTIN, J. Design of real-time computer systems. Prentice-Hall, 1967. Citado na página 25.
- MCCRACKEN, D. D. *Guide to PL-M Programming for Microcomputer Applications*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1978. ISBN 0201045753. Citado na página 37.
- MERNIK, M.; UMER, V. Incremental programming language development. *Computer Languages, Systems and Structures*, Elsevier Science Publishers B. V., v. 31, p. 1–16, 2005. Citado 4 vezes nas páginas 15, 17, 40 e 41.
- MINKOFF, M. A modularized package of dual algorithms for solving constrained nonlinear programming problems. *Proceedings of the 1975 annual conference*, ACM, p. 159–162, 1975. Citado 4 vezes nas páginas 26, 27, 38 e 39.
- PARNAS, D. L. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, ACM, v. 15, p. 1053–1058, 1972. Citado 8 vezes nas páginas 8, 9, 15, 17, 24, 31, 38 e 41.
- POPEK, G. J. et al. Notes on the design of euclid. In: *ACM Sigplan Notices*. [S.l.]: ACM, 1977. v. 12, n. 3, p. 11–18. Citado 2 vezes nas páginas 33 e 39.
- SCHUH, G.; RUDOLF, S.; VOGELS, T. Development of modular product architectures. *Procedia CIRP*, p. 120–125, 2014. Citado na página 19.
- SILVERMAN, M. Computerized dispatching system. In: *24th IEEE Vehicular Technology Conference*. [S.l.: s.n.], 1973. v. 24, p. 136–146. Citado 3 vezes nas páginas 25, 38 e 39.
- SIMONS, K.; MOSS, A.; REINECKE, R. D. Ocular motility test administration and analysis by computer in strabismus and amblyopia evaluation. *Computers in biology and medicine*, Elsevier, v. 8, n. 2, p. 105–123, 1978. Citado 2 vezes nas páginas 36 e 38.
- SOLNTSEFF, N. Programming languages for introductory computing courses: A position paper. In: *Papers of the SIGCSE/CSA Technical Symposium on Computer Science Education*. [S.l.]: ACM, 1978. (SIGCSE '78), p. 119–124. Citado 2 vezes nas páginas 34 e 39.
- SPIER, M. J. A pragmatic proposal for the improvement of program modularity and reliability. *International Journal of Parallel Programming*, v. 4, n. 2, p. 133–149, 1975. Citado 3 vezes nas páginas 27, 38 e 39.
- VOIGT, S. Program design by a multidisciplinary team. *IEEE Transactions on Software Engineering*, IEEE, n. 4, p. 370–376, 1975. Citado 3 vezes nas páginas 29, 38 e 39.
- WASSERMAN, A. I.; SHERERTZ, D. D. A balanced view of mumps. *ACM SIGPLAN Notices*, ACM, v. 11, n. 4, p. 16–26, 1976. Citado 4 vezes nas páginas 31, 38, 39 e 40.
- WEGNER, P. Programming languages - the first 25 years. *IEEE Trans. Computers*, v. 25, n. 12, p. 1207–1225, 1976. Citado 4 vezes nas páginas 30, 31, 38 e 39.

- WIRTH, N. Program development by stepwise refinement. *Communications of the ACM*, ACM, v. 14, n. 4, p. 221–227, 1971. Citado na página 25.
- WIRTH, N. Modula: A language for modular multiprogramming. *Software: Practice and Experience*, Wiley Online Library, v. 7, n. 1, p. 1–35, 1977. Citado na página 31.
- WOHLIN, C. et al. Systematic literature reviews. In: *Experimentation in Software Engineering*. [S.l.]: Springer Science & Business Media, 2012. cap. 4, p. 45–54. Citado 4 vezes nas páginas 16, 17, 18 e 20.
- WULF, W. A.; LONDON, R. L.; SHAW, M. Abstraction and verification in alphard: Introduction to language and methodology. In: *Alphard: Form and Content*. [S.l.]: Springer, 1976. p. 15–60. Citado na página 31.
- ZILLES, S. N. Modularization around a suitable abstraction. In: *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*. [S.l.]: ACM, 1975. (AFIPS '75), p. 279–279. Citado 3 vezes nas páginas 28, 29 e 38.

APÊNDICE A – Artigos Utilizados

Quadro 2 – Artigos utilizados para leitura

Base	Palavra	Ano	Título	Autor
ACM	Modularidade e Modularização	1972	On the criteria to be used in decomposing systems into modules	D. L. Parnas
ACM	Modularidade	1972	Operating system performance	W. C. Lynch
IEEE	Modularidade	1973	Computerized dispatching system	M. Silverman
ACM	Modularidade	1973	Organizing computer systems for learnability and useability	P. C. Anagnostopoulos
ACM	Modularização	1975	A modularized package of dual algorithms for solving constrained nonlinear programming problems	M. Minkoff
Springer	Modularidade	1975	A pragmatic proposal for the improvement of program modularity and reliability	M. J. J. Spier
IEEE	Modularização	1975	CAMAC Software for the Hot Fuel Examination Facility Real-Time Computer System	L. D. Lau et al.
ACM	Modularização	1975	Modularization around a suitable abstraction	S. N. Zilles
IEEE	Modularidade	1975	Program design by a multidisciplinary team	S. Voigt
ACM	Modularidade	1975	The effect of certain modular design principles on testability	N. P. Edwards
IEEE	Modularidade	1976	Programming Languages: The First 25 Years	P. Wegner

Continua na próxima página

Base	Palavra	Ano	Título	Autor
ACM	Modularidade	1976	A balanced view of MUMPS	A. I. Wasserman e D. D. Sherertz
ACM	Modularidade	1976	Possible futures and present actions	G. M. Hopper
ACM	Modularidade	1976	The use of abstract data types to simplify program modifications	T. A. Linden
ACM	Modularidade	1976	Visibility and types	C. H. A. Coster
ACM	Encapsulamento	1977	Notes on the design of Euclid	G. J. Popek et al.
ACM	Modularidade	1978	Programming languages for introductory computing courses: a position paper	N. Solntseff
ACM	Modularidade	1978	Software engineering with standard assemblies	R. Lanergan e B. Poynton
IEEE	Modularidade	1978	The modularity of MSC/NASTRAN	J. F. Gloudeman
ACM	Modularidade	1978	The Smalltalk-76 programming system design and implementation	D. H. H. Ingalls
Science	Modularidade	1978	Ocular motility test administration and analysis by computer in strabismus and amblyopia evaluation	K. Simons, et al.
IEEE	Modularidade	1978	A small-scale operating system foundation for micro-processor applications	K. C. Kahn

APÊNDICE B – Artigos Excluídos

Quadro 3 – Artigos excluídos da leitura

Ano	Título	Autor	Motivo da Exclusão
1972	Control Rod Signal Multiplexing	G. C. Minor e S. E. Moore	Eliminado pelo requisito 2.
1972	A collection of graph analysis APL functions	E. Girard, D. Bastin e J. C. Rault	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1972	A systematic approach to the design of digital bussing structures	K. J. Thurber et al.	Eliminado pelo requisito 2.
1972	Circuit considerations in the design of wide-band tunable transferred electron oscillators	D. Cawsey	Eliminado pelo requisito 2.
1972	Development of 1000 AMPERE RMS, 4 KHZ, 1500 volt thyristors and rectifiers	E. D. Wolley	Eliminado pelo requisito 2.
1972	Experiments concerning the measurement of floating-emitter potential	J. S. Bora	Eliminado pelo requisito 2.
1972	Glass-ceramic-copper microwave encapsulations and mountings	S. E. Davies	Eliminado pelo requisito 2.
1972	Microwave performance of glass-ceramic/copper encapsulations	D. J. Colliver	Eliminado pelo requisito 2.
1972	Reliability Aspects of Plastic Encapsulated Integrated Circuits	J. L. Flood	Eliminado pelo requisito 2.
1972	Reliability problems related to plastic encapsulation of integrated circuits	G. Andre e J. Regnault	Eliminado pelo requisito 2.
1973	Tools for modular programming: Finding out what's needed	J. B. Goode-nough	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1973	Modularity of Non-Sequential Programs	K.-P. Löhr	Não tem acesso.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1973	A model of a modular interactive system	D. A. Henderson	Eliminado pelo requisito 2.
1973	Adaptation of a gas chromatographic method for analysis of effluent from manufacturing processes	R. L. Hutchinson	Eliminado pelo requisito 2.
1973	Computing with multiple microprocessors	J. V. Levy	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1973	Fourth generation data management systems	K. M. Whitney	Eliminado pelo requisito 2.
1973	J-Band Transferred-Electron Oscillators	M. Dean	Eliminado pelo requisito 2.
1973	Learner-controlled course on the TICCIT system	C. V. Bunderson	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1973	Procedural encapsulation: A linguistic protection technique	S. N. Zilles	Eliminado pelo requisito 2.
1973	Self-Diagnosis and Self-Repair in Memory: An Integrated System Approach	S. A. Szygenda e M. J. Flynn	Eliminado pelo requisito 2.
1973	Semiconductor Radiation Probes for Nuclear Medicine and Radiobiology the State of the Art	M. Martini	Eliminado pelo requisito 2.
1973	Modularity in Design: Skit Registers and Counters Used as System Building Blocks	K. H. O'Keefe	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1973	The Computer in BWR Operations - Present and Future	W. B. Sweet e D. R. Conklin	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1973	On procedures as open subroutines	I H. Langmack	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1973	A methodology for producing reliable software systems	C. J. Lucena	Eliminado pelo requisito 3: apenas cita o termo modularidade.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1974	Encapsulation of High-Purity Germanium Detectors	G. A. Armantrout et al.	Eliminado pelo requisito 2.
1974	Modularity and multi-microprocessor structures	D. P. Siewiorek	Eliminado pelo requisito 2.
1974	“Software devices” for processing graphs using pl/i compile time facilities	A. Hansal	Eliminado pelo requisito 2.
1974	A Plastic Encapsulated MIC 17 GHz Mixer	G. H. Swallow	Eliminado pelo requisito 2.
1974	Encapsulation of high reliability devices	R. K. Slee e A. N. Magnus	Eliminado pelo requisito 2.
1974	Encapsulation of High-Purity Germanium Detectors	G. A. Armantrout	Eliminado pelo requisito 2.
1974	Feedback-free modularization of compilers	W. M. McKeeman e F. L. Deremer	Eliminado pelo requisito 2.
1974	Instabilities in Double Dielectric Structures	M. H. Woods	Eliminado pelo requisito 2.
1974	Large signal circuit characterization of solid-state microwave oscillator devices	M. J. Howes	Eliminado pelo requisito 2.
1974	Specifications for integrated circuits in telecommunications equipment	F. H. Reynolds	Eliminado pelo requisito 2.
1974	Structured operating system organization	M. V. Zelkowitz	Eliminado pelo requisito 2.
1974	Interactive programs for statistical computations and information display in chronic obstructive pulmonary disease research	N. H. McAlistter et al.	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1974	On procedures as open subroutines. II	H. Langmack	Eliminado pelo requisito 3: cita o artigo anterior que usa modularidade mas não utiliza em seu escopo.
Continua na próxima página			

Ano	Título	Autor	Motivo da Exclusão
1974	Encapsulation: an approach to operating system security	R. L. Bisbey,II e G. J. Popek	Eliminado pelo requisito 3: o tema modularidade não é abordado.
1974	On improving operating system efficiency through use of a microprogrammed, low-level environment	G. W. Cox e V. B. Schneider	Inilegível.
1974	Modularization of large econometric models: An application of structural modeling	A. M. Elmokadem et al.	Eliminado pelo requisito 2.
1974	CAMAC Applications in Nuclear Medicine at Vanderbilt: Present Status and Future Plans	A. B. Brill et al.	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1974	Modularity and multi-microprocessor structures	D. P. Siewiorek	Eliminado pelo requisito 2.
1975	A note on advanced software techniques in computer graphics	H. U. Lemke e A. P. Armit	Eliminado pelo requisito 3.
1975	A Review on Random Access MOS Memories	R. W. Mittler	Eliminado pelo requisito 2.
1975	A vacuum centrifuge for void-free potting of implantable hybrid microcircuits in silicone	P. E. K. Donaldson e E. Sayer	Eliminado pelo requisito 2.
1975	Arc plasma sprayed metallization on microwave ceramic substrates	D. H. Harris	Eliminado pelo requisito 2.
1975	Be implanted GaAs 1-x P x light emitting diodes	P. K. Chatterjee	Eliminado pelo requisito 2.
1975	Contaminant effects on semiconductor devices	T. Scharr	Eliminado pelo requisito 2.
1975	Encapsulation of ion-implanted GaAs using native oxides	B. J. Sealy e A. D. E. D'Cruz	Eliminado pelo requisito 2.
1975	Epoxy encapsulation compounds with improved thermal shock and reversion resistance	J. C. Bolger e S. L. Morano	Eliminado pelo requisito 2.
1975	JSYS traps: a TENEX mechanism for encapsulation of user processes	R. H. Thomas	Eliminado pelo requisito 2.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1975	Microwave field-effect transistors from sulphur-implanted GaAs	W. Kellner	Eliminado pelo requisito 2.
1975	Results of a 160 x 106 device-hour reliability assessment and failure analysis of TTL SSI integrated circuits, part 2 survey of dominant IC failure mechanisms and analysis of failure causes	A.P. kemeny et al.	Eliminado pelo requisito 2.
1975	Simulation of a real time micro-processor network	E. J. Passahow	Eliminado pelo requisito 2.
1975	The software engineering technique of data hiding as applied to multi-level model implementation of logical devices in digital simulation	E. W. Thompson e N. Billawala	Eliminado pelo requisito 2.
1975	OMAL: A Task-Oriented Micro-processor Applications Language	J. L. Hennessy et al.	Eliminado pelo requisito 3: modularidade de hardware.
1975	The structure of special computer in switching packets	R. Beaufiles et al.	Eliminado pelo requisito 2.
1975	A description of a parametrically controlled modular structure for speech processing	N. Dixon e H. Silverman	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1975	Architecture Considerations for Digital Automatic Flight Control Systems	S. Osder	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1976	Application of Aluminum Oxide to Integrated Circuits Fabrication	H. Harada et al.	Eliminado pelo requisito 2.
1976	Bias Influence on Corrosion of Plastic Encapsulated Device Metal Systems	B. Reich	Eliminado pelo requisito 2.
1976	A Complexity Measure	T. J. McCabe	Eliminado pelo requisito 2.
1976	Distributed Processing Within an Integrated Circuit/Package-Switching Node	C. Jenny e K. Kummerle	Eliminado pelo requisito 2.
Continua na próxima página			

Ano	Título	Autor	Motivo da Exclusão
1976	Dynamic Permeability Method for EPOXY Encapsulation Resins	N. Vanderkooi e M. N. Riddell	Eliminado pelo requisito 2.
1976	The Encapsulation of Microelectronic Devices for Long-Term Surgical Implantation	P. E. K. Donaldson	Eliminado pelo requisito 2.
1976	A simulator generator based on formal descriptions of architectural, load, and operating system models	H. Kerner e W. Beyerle	Eliminado pelo requisito 2.
1976	A network-oriented multiprocessor front-end handling many hosts and hundreds of terminals	W. F. Mann et al.	Eliminado pelo requisito 2.
1976	A Study of Parasitic MOS Formation Mechanism in Plastic Encapsulated MOS Devices	Y. Wakashima	Eliminado pelo requisito 2.
1976	Ageing in ZnSe:Mn light emitting diodes	J. I. B. Wilson e J. W. Allen	Eliminado pelo requisito 2.
1976	Control of the encapsulation material as an aid to long term reliability in plastic encapsulated semiconductor components (PEDs)	J. C. Harrison	Eliminado pelo requisito 2.
1976	Encapsulation of Integrated Circuits Containing Beam Leded Devices with a Silicone RTV Dispersion	D. Jaffe	Eliminado pelo requisito 2.
1976	High reliability Ta/Al thin-film hybrid circuits	R. G. Duckworth	Eliminado pelo requisito 2.
1976	Interactive graphics in the analysis of neuronal spike train data	R.J. Scabbassi et al.	Eliminado pelo requisito 2.
1976	Microwave Heating of Malignant Mouse Tumors Encapsulated in a Dielectric Medium	J. E. Robinson	Eliminado pelo requisito 2.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1976	Remreed switching networks for No. 1 and No. 1A ESS: Remreed network electronic control	D. Danielsen e W. A. Liss	Eliminado pelo requisito 2.
1976	Short Range Precision Navigation and Tracking System	W. Sternberger e L. LeBlanc	Eliminado pelo requisito 2.
1976	The Encapsulation of Microelectronic Devices for Long-Term Surgical Implantation	P. E. K. Donaldson	Eliminado pelo requisito 2.
1976	Thermally Stimulated Discharge Effects in Polymer Encapsulants	J. B. Woodard	Eliminado pelo requisito 2.
1976	On the Implementation of Data Generality	A. von Staa e C. J. Lucena	Não tem acesso.
1976	A data type encapsulation scheme utilizing base language operators	M. B. Wells e F. L. Cornwell	Eliminado pelo requisito 3: o tema modularidade não é abordado.
1976	Toward a systems environment for computer assisted programming	C. J. Lucena e D. D. Cowan	Eliminado pelo requisito 2.
1976	A new architecture for process control computers NARCIS project	P. Vernel et al.	Eliminado pelo requisito 2.
1976	Asynchronous Interlock Units for Speed-Independent Multiprocessor Systems	P. Corsini	Eliminado pelo requisito 2.
1976	Modularization in the pilot compiler and its effect on the length	J. Ingojo	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1976	Development of On-board Space Computer Systems	A. E. Cooper	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1976	A pipeline architecture oriented towards efficient multitasking	F. Romani	Eliminado pelo requisito 2.
1976	Dynamic software engineering: An evolutionary approach to automated software development and management	J. S. Greene, Jr	Eliminado pelo requisito 3: apenas cita o termo modularidade.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1976	Distributed Computer Systems in the Laboratory Environment	B. Zacharov	Eliminado pelo requisito 3: modularidade de hardware.
1977	On unit element structures for wave digital filters	J. -P. Thiran	Eliminado pelo requisito 2.
1977	A comparison of the performance of plastic and ceramic encapsulations based on evaluation of CMOS integrated circuits	M. J. Fox	Eliminado pelo requisito 2.
1977	A general linear continuous model for design of power conditioning units at fixed and free running frequency	A. J. Fossard et al.	Eliminado pelo requisito 2.
1977	Assessment of Silicone Encapsulation Materials: Screening Techniques	A. Christou e W. Wilkins	Eliminado pelo requisito 2.
1977	Encapsulation of integrated circuits containing beam leaded devices with a silicone RTV dispersion	D. Jaffe	Eliminado pelo requisito 2.
1977	Encapsulation, sectioning and examination of multilayer ceramic chip capacitors	G. J. Ewell e W. K. Jones	Eliminado pelo requisito 2.
1977	Fabrication and characterization of thin-film silicon solar cells on alumina ceramic	T. Warabisako	Eliminado pelo requisito 2.
1977	In-circuit characterisation technique for barritt diodes and other 2-terminal negative-resistance oscillator devices	R. D. Pollard	Eliminado pelo requisito 2.
1977	Performance of New Copper Based Metallization Systems in an 85°C 80-Percent RH Cl 2 Contaminated Environment	N. Sbar	Eliminado pelo requisito 2.
1977	Reliability assessment of hybrid encapsulants: Fluorinated network polymeric materials and silicones	A. Cristou	Eliminado pelo requisito 2.
Continua na próxima página			

Ano	Título	Autor	Motivo da Exclusão
1977	Hardware engineering and software engineering	B. R. Gaines	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1977	Language design methods based on semantic principles	R. D. Tennent	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1977	Aspen language specifications	T. R. Wilcox	Eliminado pelo requisito 3: o termo modularidade não entra no escopo.
1977	A forecast of the future of computation	C. Hammer	Eliminado pelo requisito 2.
1977	Seismic loads in modularized and unmodularized large pools located on hard or intermediate hard sites	R. G. Dong	Eliminado pelo requisito 2.
1977	Solubility and Diffusion of Sulfur in Polymeric Materials	B. S. Berry	Eliminado pelo requisito 2.
1977	A Data Flow Multiprocessor	J. Rumbaugh	Eliminado pelo requisito 2.
1977	The design of an integrated burner-boiler system using flue-gas recirculation	J. G. Meier e B. L. Volterrin	Eliminado pelo requisito 2.
1977	The SARA system for computer architecture design and modelling	G. Estrin	Eliminado pelo requisito 2.
1977	Design of a modular systems applications microcomputer	J. Gallacher	Eliminado pelo requisito 3: modularidade de hardware.
1977	Synchronization in actor systems	R. Atkinson e C. Hewitt	Eliminado pelo requisito 3: modularidade de sincronização
1977	A framed painting: The representation of a common sense knowledge fragment	E. Charniak	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1977	Dynamic modelling in development planning	L. Grandinetti et al.	Não tem acesso.
1977	A top-down, laboratory based operating system course	D. H. Grit e D. D. Georg	Eliminado pelo requisito 2.

Continua na próxima página

Ano	Título	Autor	Motivo da Exclusão
1977	Implementation and application of a function data type	M. B. Wells	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1977	Conservation of software science parameters across modularization	L. Hunter e J. Ingojo	Eliminado pelo requisito 2.
1977	Changing technology in switching system software	E. Bierman	Eliminado pelo requisito 2.
1977	A Multi Radar Tracking simulation using ALGOL 68'R	P. R. West	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1978	The Effect of LSI Technology on the Theory of Modular Computer Design	A. D. Friedman e L. Simoncini	Eliminado pelo requisito 2.
1978	A modular representation and analysis of fault tress	L. Wolf	Eliminado pelo requisito 3: modularidade de árvore de falha.
1978	Dynamic graphics using quasi parallelism	C. Hewitt e K. M. Kahn	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1978	On A System For Adaptive, Parallel Finite Element Computations	I. Babuska e W. C. Rheinboldt	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.
1978	How to write information mapping: Robert E. Horn	E. Schlesinger	Eliminado pelo requisito 2.
1978	Conditions for the Equivalence of Synchronous and Asynchronous Systems	E. A. Akoyunlu, et al.	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1978	An LSI Modular Direct-Execution Computer Organization	Y. Chu	Eliminado pelo requisito 3: modularidade de hardware.
1978	The design of a corporate planning system simulator	E. A. Stohr e M. Tanniru	Eliminado pelo requisito 3: apenas cita o termo modularidade.
1978	The Euclid Language: a progress report	R. C. Holt, et al.	Eliminado pelo requisito 3: não tem modularidade como um dos temas centrais.

APÊNDICE C – Características Observadas

Quadro 4 – Características observadas no desenvolvimento

Artigo	Características observadas
On the Criteria To Be Used in Decomposing Systems into Modules	Ocultação de informações e Módulos independentes.
Operating System Performance	Módulos independentes, Modularidade vertical(ou programação estruturada) e Modularidade horizontal
Computerized Dispatching System	Módulos independentes, Módulos de interface, Módulo controlador e Base de dados em comum.
Organizing Computer Systems for Learnability and Useability	Ocultação de informações, Programação estruturada e Modularidade a nível de seções.
A modularized package of dual algorithms for solving constrained nonlinear programming problems	Ocultação de informações, Módulos independentes e Módulos separados entre módulo lógico, módulos abaixo dos lógicos e módulos auxiliares;
A pragmatic proposal for the improvement of program modularity and reliability	Ocultação de informações, Não existe algoritmo para modularidade, Módulos independentes, Módulos são um conjuntos de funções e Módulos são encapsulados
CAMAC Software for the Hot Fuel Examination Facility Real-Time Computer System	Módulos independentes, Cada módulo possui uma sub-rotina, bem como entradas e saídas e a Comunicação é através da passagem de parâmetros.
Modularization Around a Suitable Abstraction	Módulos independentes e Utilizar abstração no desenvolvimento de módulos.
Program design by a multidisciplinary team	Módulos independentes, Módulos são um conjuntos de funções, Comunicação através da passagem de parâmetros e Utilização de variáveis globais.
The Effect of Certain Modular Design Principles on Testability	Módulo de Controle ou Função de controle, Módulos possuem entradas e saídas e Fluxo de controle do sistema.
Continua na próxima página	

Artigo	Características observadas
Programming Languages: The First 25 Years	Ocultação de informações, Módulos independentes e abstratos, Módulos são um conjuntos de sub-rotinas ou formulários e Utilização de objetos externos para serem visíveis pelo resto do programa
A Balanced View of MUMPS	Comunicação através de sub-rotinas.
Possible futures and present actions	Módulos são partes menores de um sistema e Comunicação através de links.
The use of Abstract Data Types to Simplify Program Modifications	Ocultação de informações, Módulos independentes e Módulos pequenos e diretos.
Visibility and types	Ocultação de informações, Módulos aninhados, Modularidade vertical e Modularidade horizontal.
Notes on the design of Euclid	Ocultação de informações, Módulos são um conjunto de sub-rotinas, estrutura de dados, entradas e saídas, e nomes visíveis, e Base de dados comum.
Programming languages for introductory computing courses: a position paper	Ocultação de informações, Procedimentos parametrizados, Módulos são tipos abstratos de dados.
Software engineering with standard assemblies	Módulos independentes e são sub-rotinas.
The modularity of MSC/NASTRAN	Módulos funcionais são um conjuntos de sub-rotinas, que possuem entrada, saída e conjunto de parâmetros, e podem ou não serem independentes.
The Smalltalk-76 programming system design and implementation	Módulos independentes, Comunicação através de mensagens e Utilização de herança.
Ocular motility test administration and analysis by computer in strabismus and amblyopia evaluation	Módulo controlador e Módulos auxiliares.