PMR 5020

Metodologia do Projeto de Sistemas

Aula 9: Systems Design and OO Systems Design

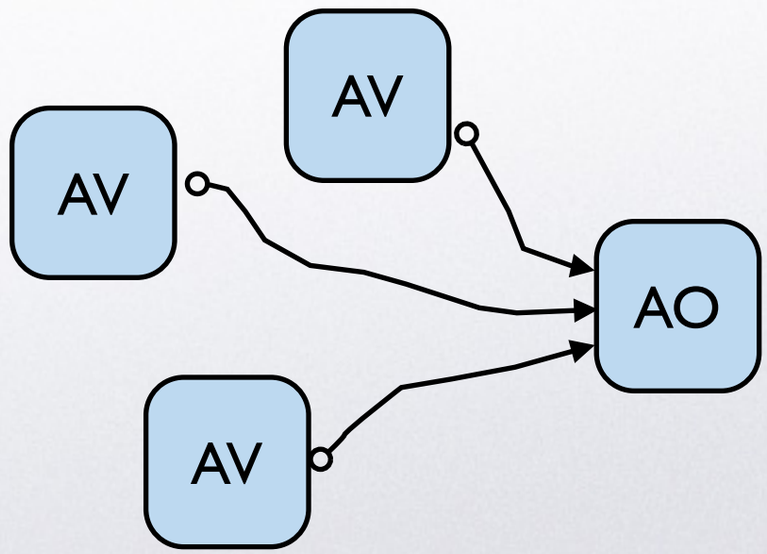
Prof. José Reinaldo Silva
reinaldo@usp.br

Alternativas para o Design de Sistemas

Até aqui vimos as seguintes alternativas para o design de sistemas:

I. investir na fase inicial, na eliciação de requisitos, análise e especificação;
II. a possibilidade de usar várias linguagens na fase de que vem após a especificação;
III. a *esperança* de que a formalização inerente ao processo não seja o motivo de perda de informação ou desvio dos requisitos iniciais;
IV. a possibilidade (especialmente para sistemas automatizados) de sofisticar o controle introduzindo sistemas inteligentes;

Existe de fato um processo de design formal?



A busca por uma Teoria Geral do Design

A discussão sobre a formalização do processo de design começou em 1981com a proposta de Hiroyuki Yoshikawa, e foi logo em seguida ampliada por Tetsuo Tomiyama, seu orientado. A polemica perdura até hoje e varia de alegações ao arcabouço teórico, à abordagem conceitual, até a perspectiva de aplicação.

Hiroyuki Yoshikawa

Tetsuo Tomiyama

Yoshikawa, H. (1981). General Design Theory and a CAD system, In: *Man-Machine Communication in CAD/CAM*, Sata, T. and Warman, E. (eds.), pp. 35–58, North-Holland, Amsterdam.

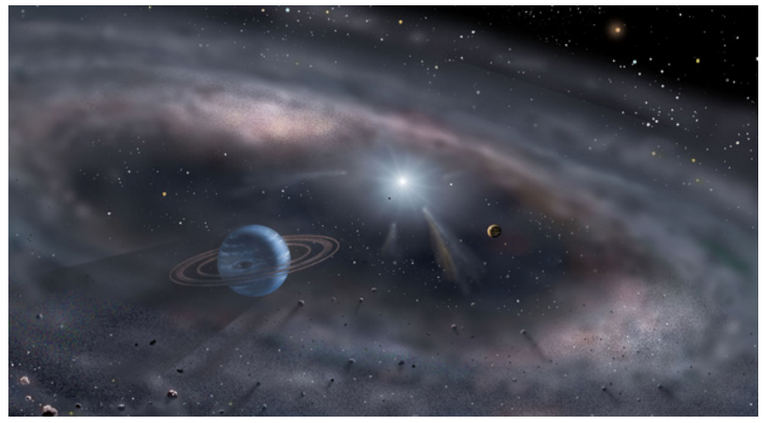
Axiom 1 (Axiom of Recognition) Any entity can be recognized or described by the attributes.

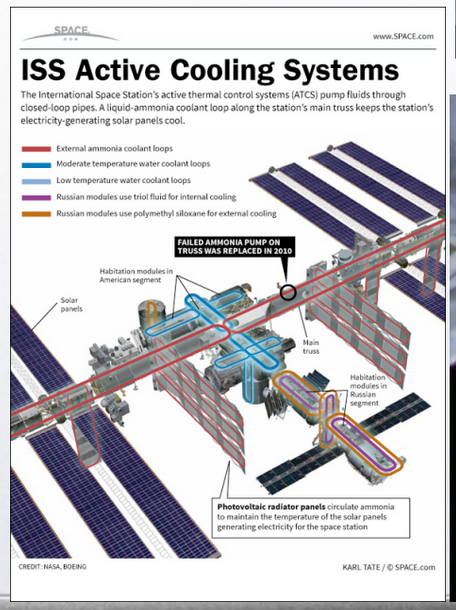
Axiom 2 (Axiom of Correspondence) The entity set S' and the set of concept of entity (ideal) S have one-to-one correspondence.

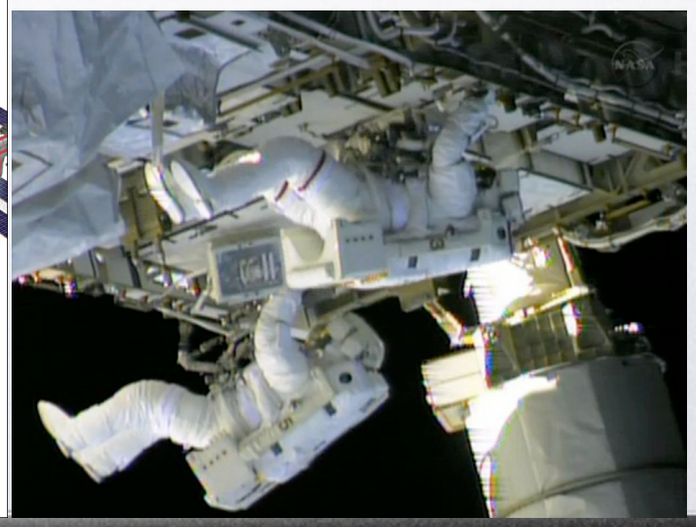
Axiom 3 (Axiom of Operation) The set of abstract concept is a topology of the set of entity concept.

The System of Systems Challenge

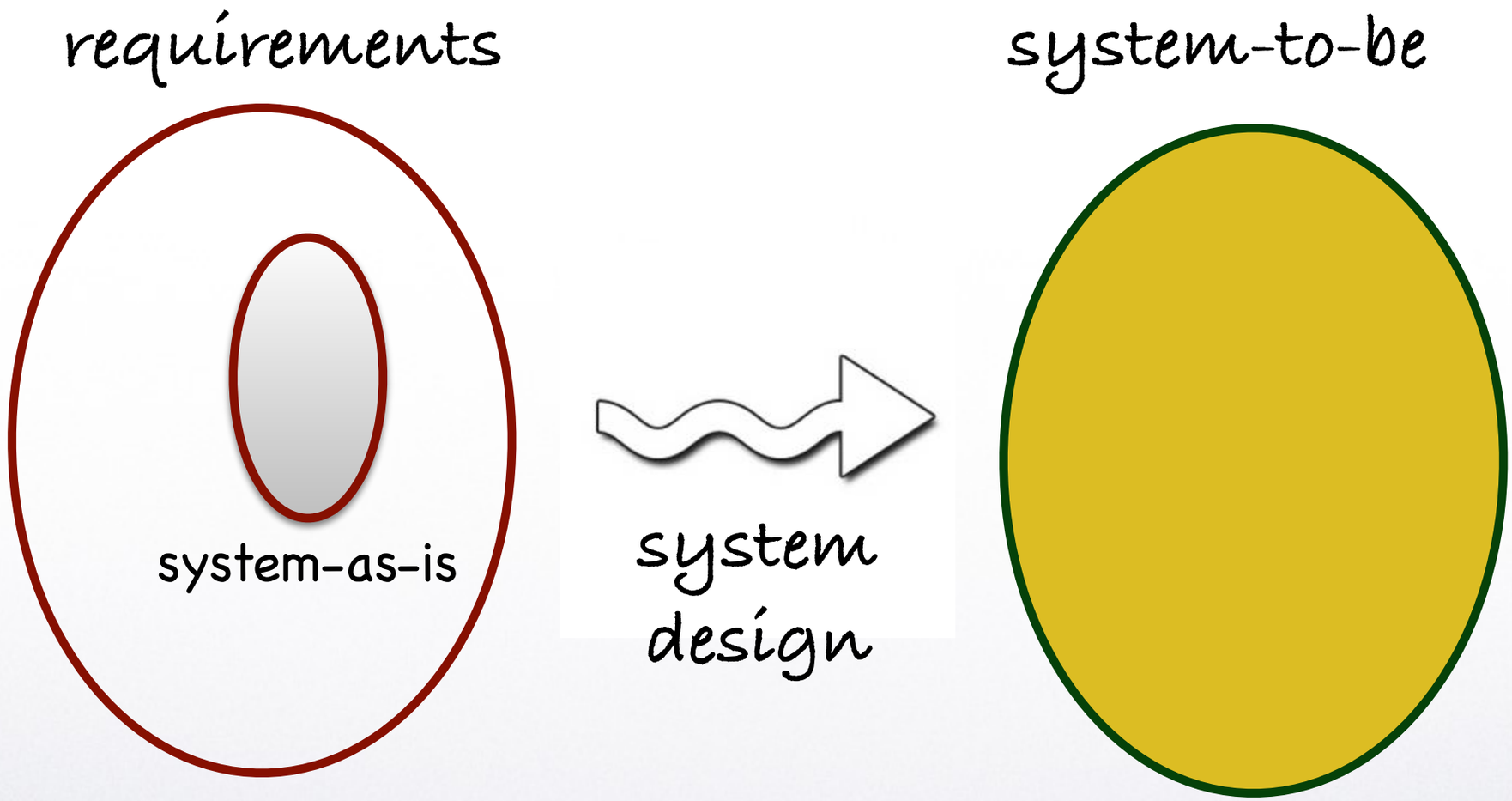
A practical obstacle to the formalization of design is the practical effectiveness of this approach, specially in this era of complexity. Generally, formal approaches do not fit the complexity of large systems (of systems).







The System Design Challenge



object-orientation

No.	Attribute	Values
1	paradigm	state machine, algebra, process algebra, trace
2	formality	informal, semi-formal, formal
3	graphical representation	yes, no
4	object-oriented	yes, no
5	concurrency	yes, no
6	executability	yes, no
7	usage of variables	yes, no
8	non-determinism	yes, no
9	logic	yes, no
10	provability	yes, no
11	model checking	yes, no
12	event inhibition	yes, no

Objects

Uma coisa visível e tangível com forma relativamente estável; uma coisa que pode ser percebida intelectualmente; uma coisa para qual o pensamento ou ação pode ser direcionada.

Randon College Dictionary

Um objeto tem identidade, estado e comportamento

Grady Booch

Um objeto é uma unidade de modularidade estrutural e comportamental que tem propriedades

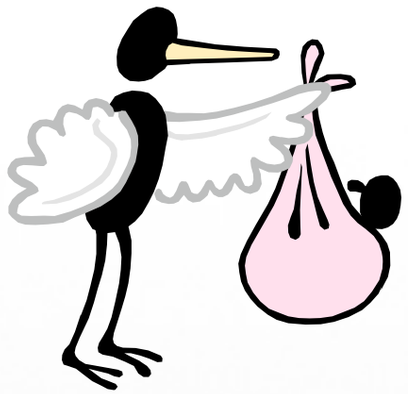
R. Buhr

Genesis dos objetos

Os objetos têm duas origens praticamente paralelas:

- estrutural : frames, Marvin Minsky, MIT, 1975
- programação : Simula 67

Objetos e programação



1966 Montagem da Simula 67: introdução do conceito de “information hiding” e encapsulamento.

1980 Aparecimento do Smalltalk 80 de Adele Goldberg

Features X Languages	Abstract Data Types	Inheritance Support	Dynamic Binding	Extensive Library
Simula	yes	yes	yes	no
CLU	yes	no	yes	no
Ada	yes	no	no	yes
Smalltalk	yes	yes	yes	yes
ObjectiveC	yes	yes	yes	yes
C++	yes	yes	yes	yes
CLOS	yes	yes	yes	no
Obj.Pascal	yes	yes	yes	no
Beta	yes	yes	yes	no
Eiffel	yes	yes	yes	yes
Actor	yes	yes	yes	no
Java	yes	yes	yes	yes

Conceitos Originais

Tipos abstratos de dados – David Parnas

Disciplina FIFO

Nome Estrutura de dados da fila
Insert(x, Fila) Retira(x, Fila)

Elements of Object-Oriented System

Let us go through the characteristics of OO System –

- **Objects** – An object is something that is exists within problem domain and can be identified by data (attribute) or behavior. All tangible entities (student, patient) and some intangible entities (bank account) are modeled as object.
- **Attributes** – They describe information about the object.
- **Behavior** – It specifies what the object can do. It defines the operation performed on objects.
- **Class** – A class encapsulates the data and its behavior. Objects with similar meaning and purpose grouped together as class.
- **Methods** – Methods determine the behavior of a class. They are nothing more than an action that an object can perform.
- **Message** – A message is a function or procedure call from one object to another. They are information sent to objects to trigger methods. Essentially, a message is a function or procedure call from one object to another.

Features of Object-Oriented System

An object-oriented system comes with several great features which are discussed below.

Encapsulation

Encapsulation is a process of information hiding. It is simply the combination of process and data into a single entity. Data of an object is hidden from the rest of the system and available only through the services of the class. It allows improvement or modification of methods used by objects without affecting other parts of a system.

Abstraction

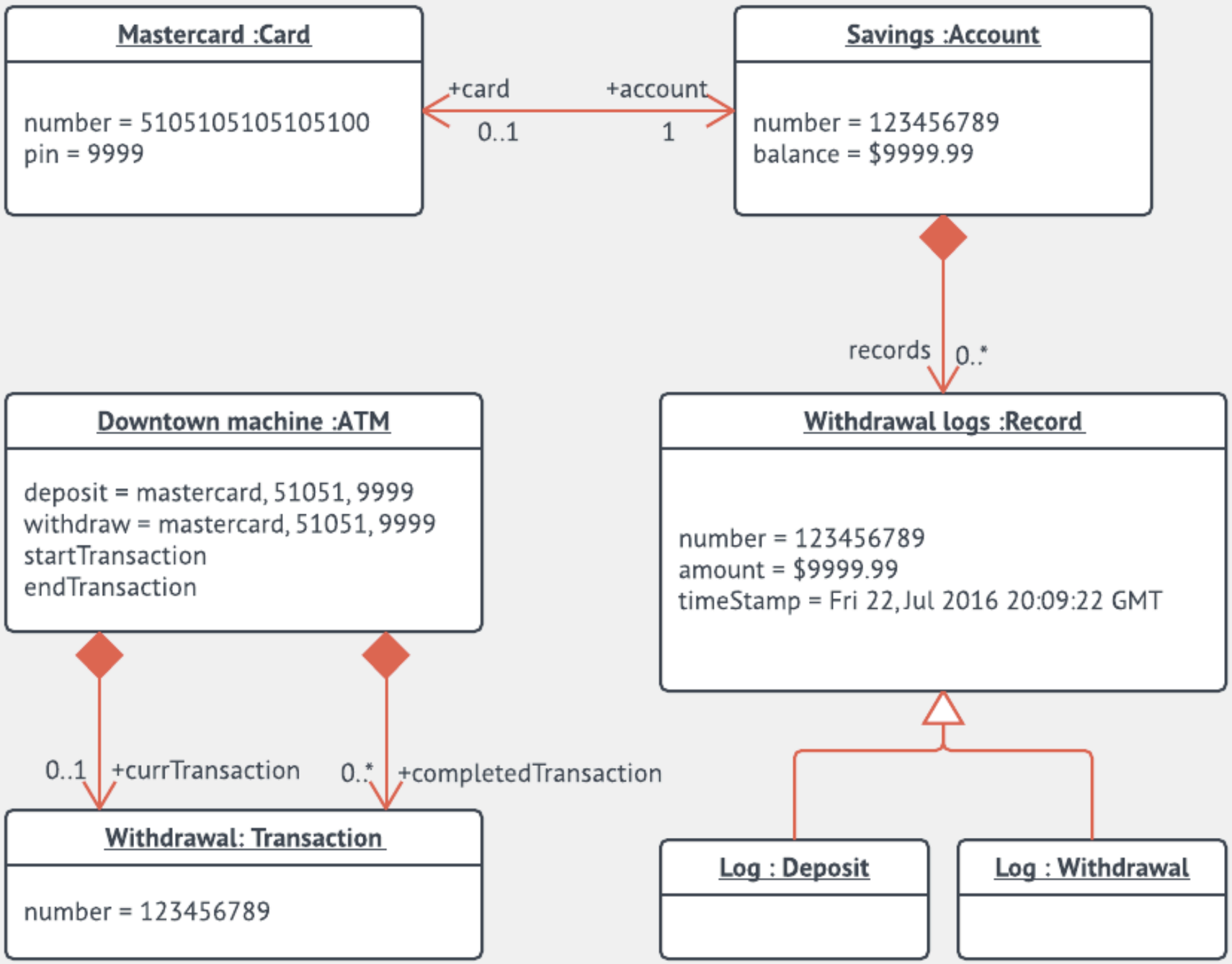
It is a process of taking or selecting necessary method and attributes to specify the object. It focuses on essential characteristics of an object relative to perspective of user.

Relationships

All the classes in the system are related with each other. The objects do not exist in isolation, they exist in relationship with other objects.

There are three types of object relationships –

- **Aggregation** – It indicates relationship between a whole and its parts.
- **Association** – In this, two classes are related or connected in some way such as one class works with another to perform a task or one class acts upon other class.
- **Generalization** – The child class is based on parent class. It indicates that two classes are similar but have some differences.


Mastercard :Card
number = 5105105105105100
pin = 9999
+card
0..1
+account
1
Savings :Account
number = 123456789
balance = $9999.99
records
0..*
Downtown machine :ATM
deposit = mastercard, 51051, 9999
withdraw = mastercard, 51051, 9999
startTransaction
endTransaction
Withdrawal logs :Record
number = 123456789
amount = $9999.99
timeStamp = Fri 22, Jul 2016 20:09:22 GMT
0..1
+currTransaction
0..*
+completedTransaction
Withdrawal: Transaction
number = 123456789
Log : Deposit
Log : Withdrawal

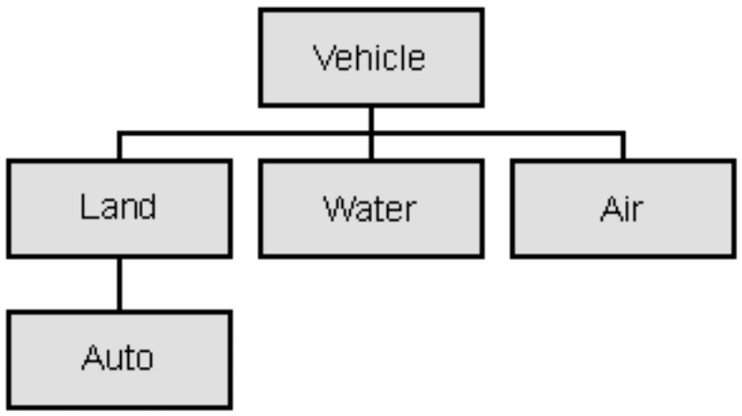
Inheritance

Inheritance is a great feature that allows to create sub-classes from an existing class by inheriting the attributes and/or operations of existing classes.

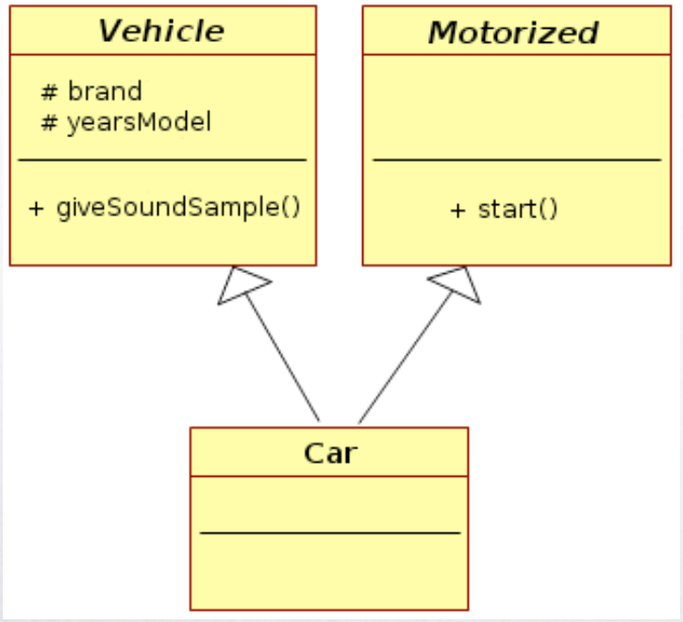
Polymorphism and Dynamic Binding

Polymorphism is the ability to take on many different forms. It applies to both objects and operations. A polymorphic object is one who true type hides within a super or parent class.

In polymorphic operation, the operation may be carried out differently by different classes of objects. It allows us to manipulate objects of different classes by knowing only their common properties.



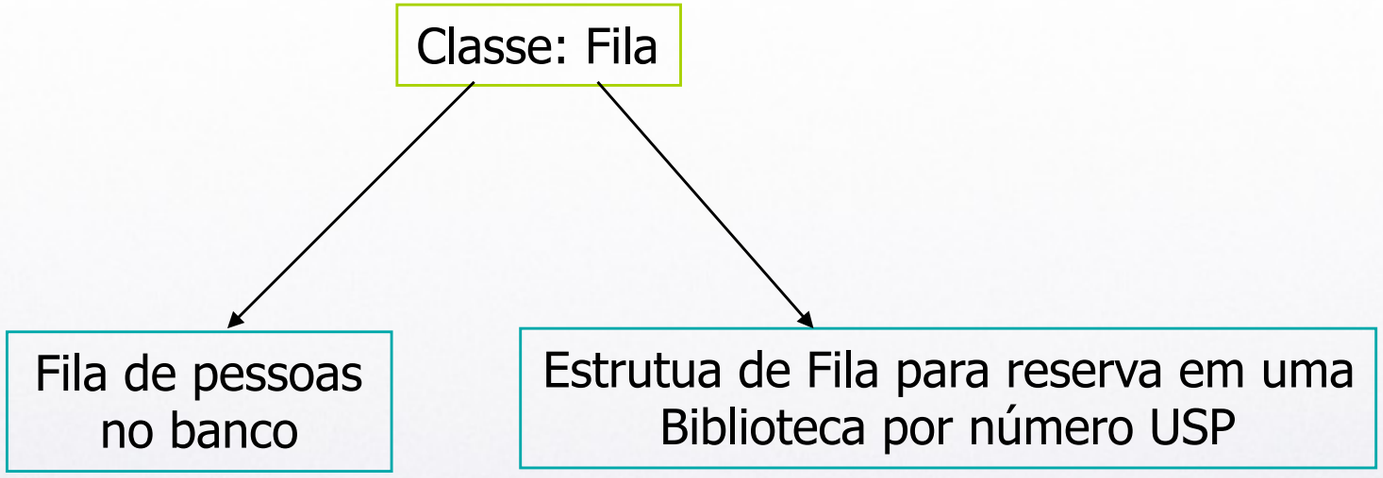
A herança simples deriva diretamente do conceito de classificação. Neste caso cada elemento ou instância de objeto tem um e somente um ancestral.

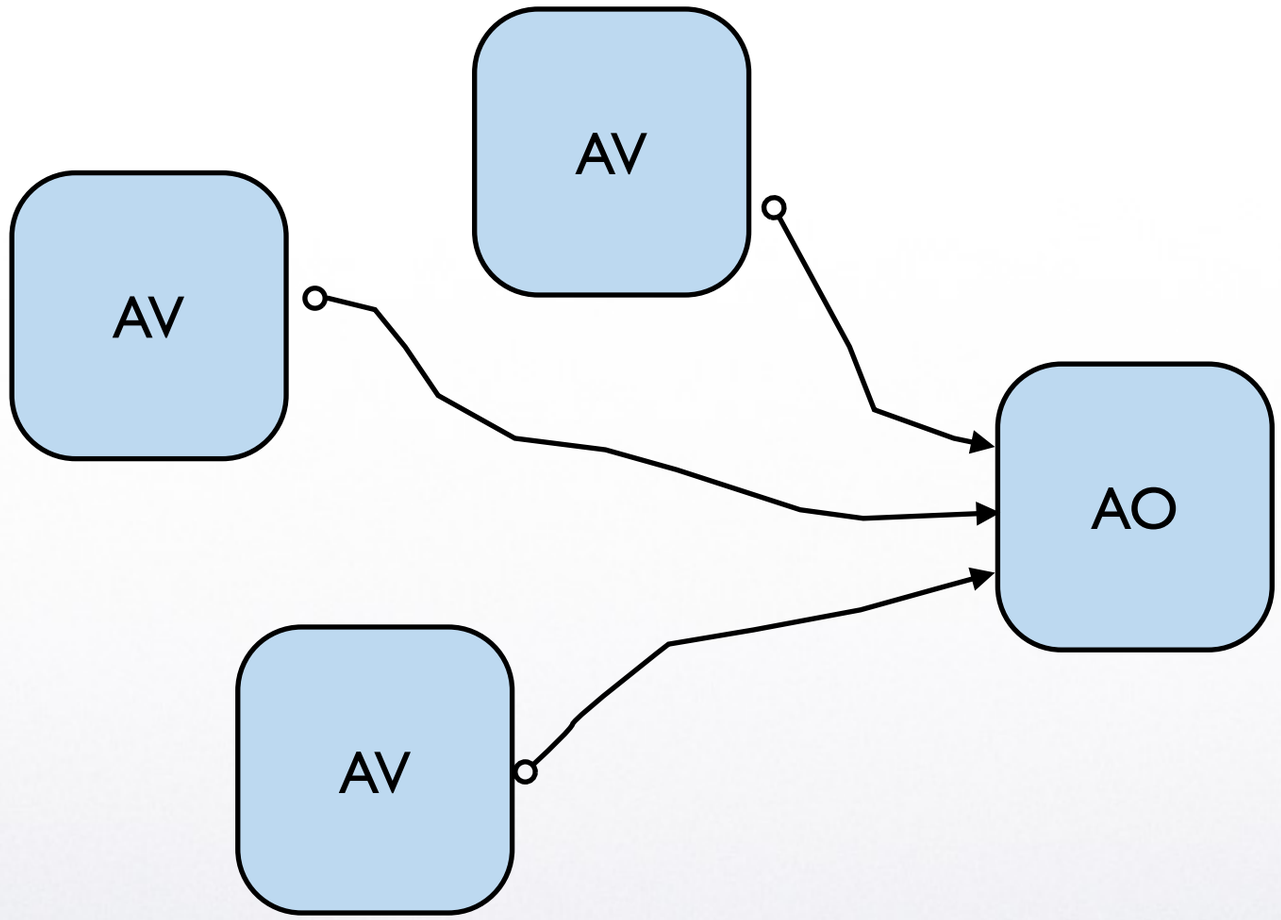


Na herança múltipla uma mesma instância pode herdar propriedades de "pais" distintos de forma composicional. Naturalmente esta implementação, mesmo em linguagens de programação é mais complexa.

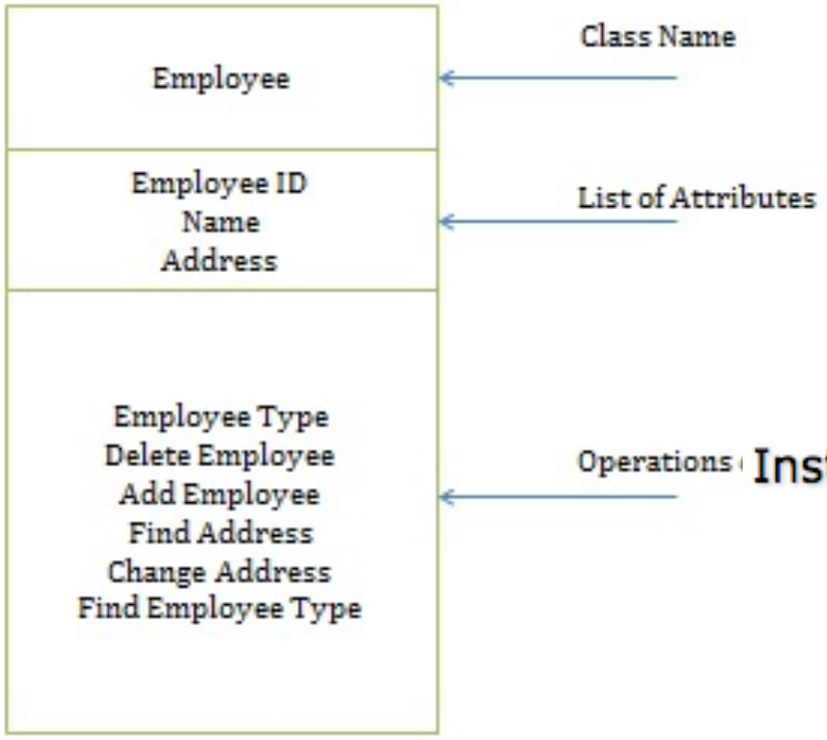
Um exemplo simples

Objetos e suas propriedades : herança (simples e múltipla), polimorfismo e Vinculação dinâmica

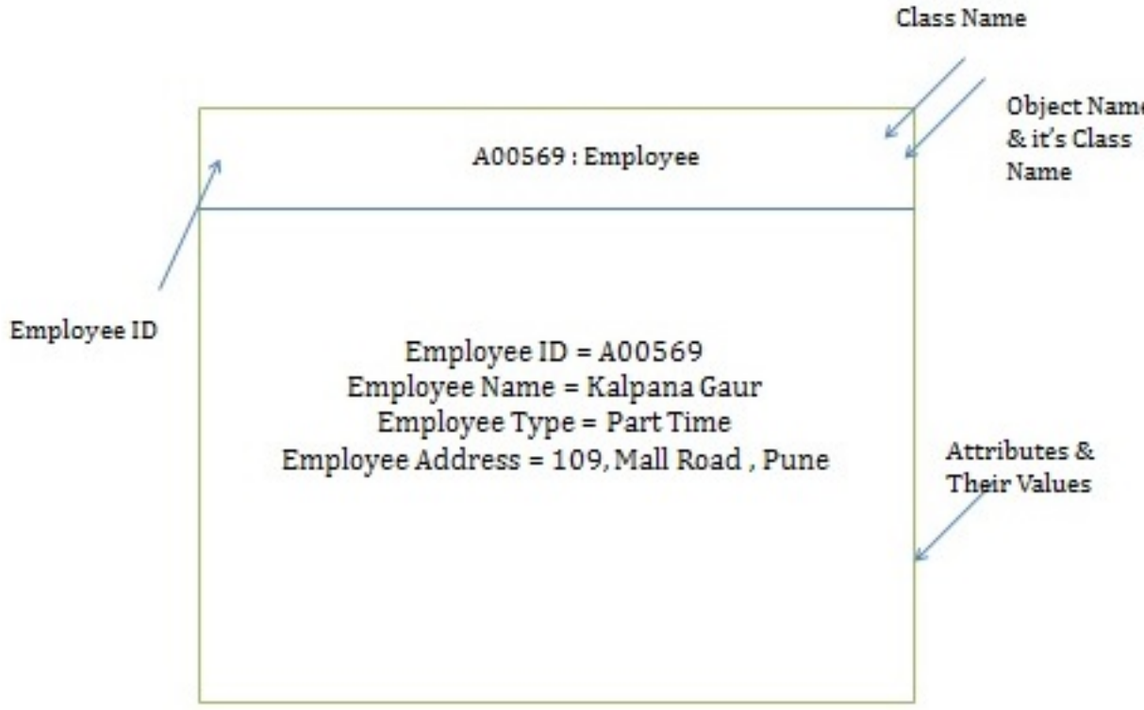



AV
AV
AV
AO

Example of UML Notation for class



Instance diagram-UML notation

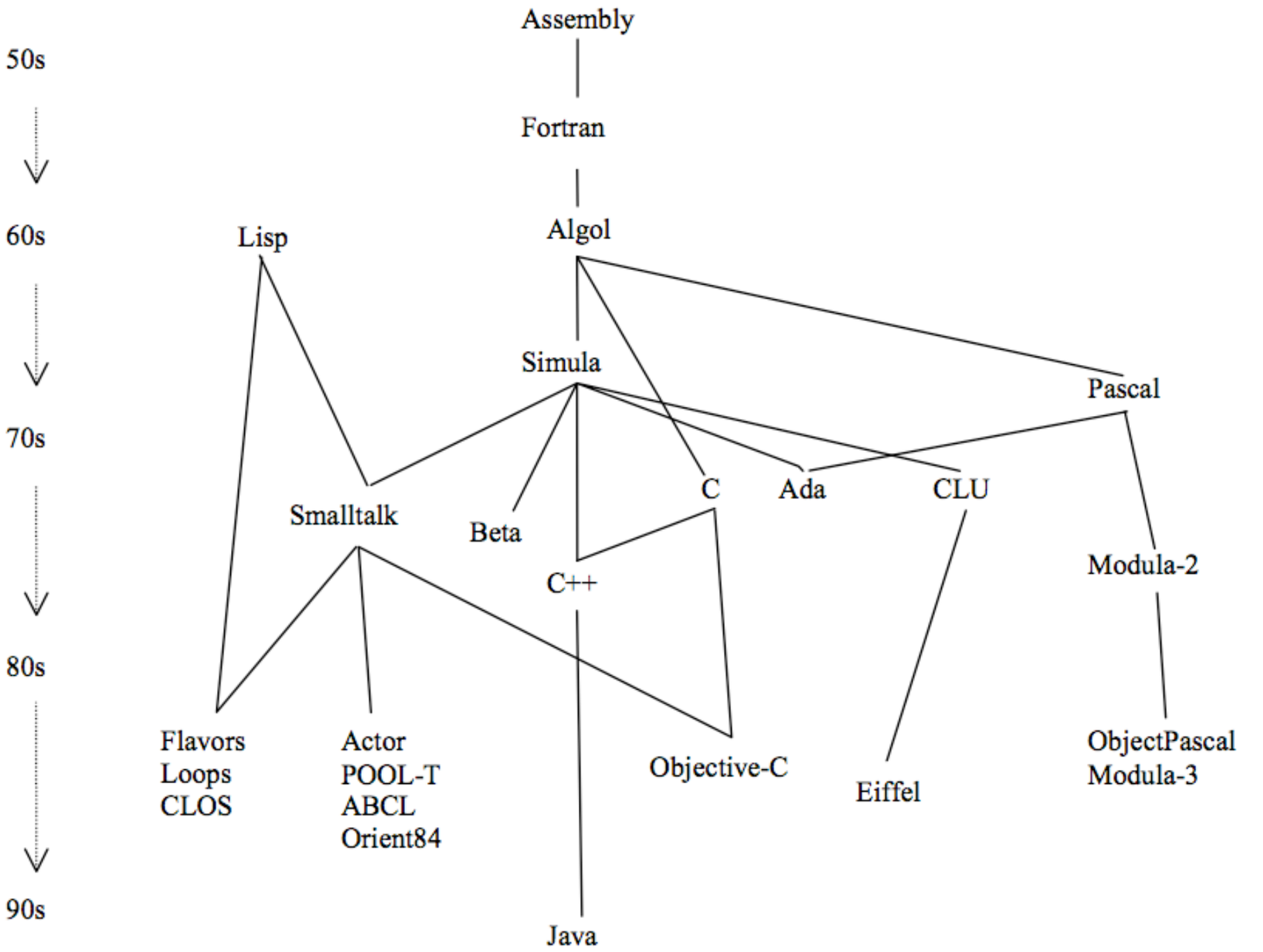


Operations Performed on Objects

The following operations are performed on the objects –

- **Constructor/Destructor** – Creating new instances of a class and deleting existing instances of a class. For example, adding a new employee.
- **Query** – Accessing state without changing value, has no side effects. For example, finding address of a particular employee.
- **Update** – Changes value of one or more attributes & affect state of object For example, changing the address of an employee.

dynamic binding - The property of object-oriented programming languages where the code executed to perform a given operation is determined at run time from the class of the operand(s) (the receiver of the message). There may be several different classes of objects which can receive a given message. An expression may denote an object which may have more than one possible class and that class can only be determined at run time. New classes may be created that can receive a particular message, without changing (or recompiling) the code which sends the message. A class may be created that can receive any set of existing messages.


50s
60s
70s
80s
90s
Assembly
Fortran
Algol
Lisp
Simula
Pascal
C
Ada
CLU
Smalltalk
Beta
C++
Modula-2
Flavors
Loops
CLOS
Actor
POOL-T
ABCL
Orient84
Objective-C
Eiffel
ObjectPascal
Modula-3
Java

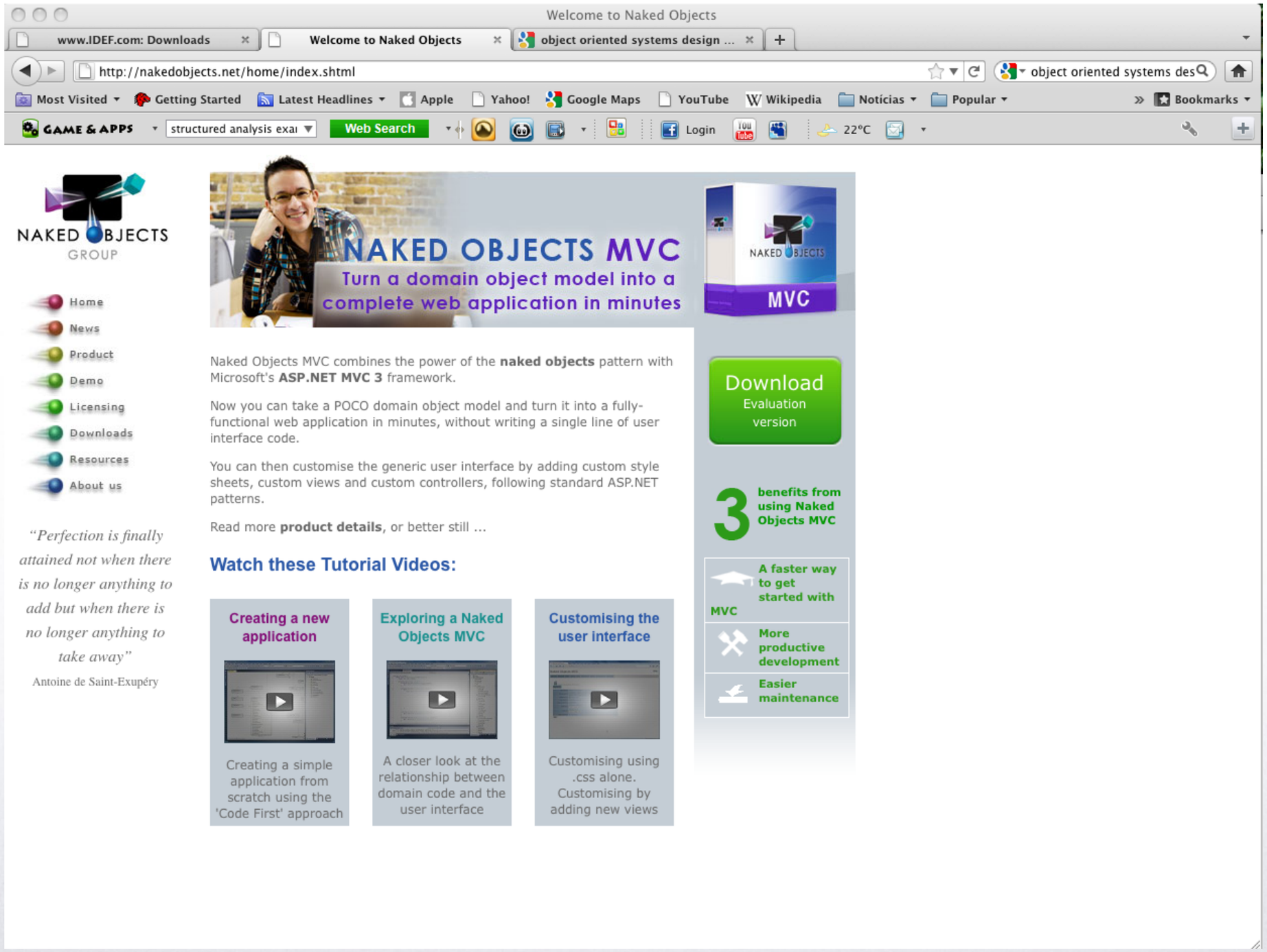
Object Oriented Design

- o sistema é composto por um conjunto de objetos
- o estado do sistema é dado pelos atributos de todas as instâncias de objeto
- uma transição no sistema se dá através de mensagens que por sua vez dispara um ou mais métodos.

A abordagem de objetos

⇨ Completeza comportamental

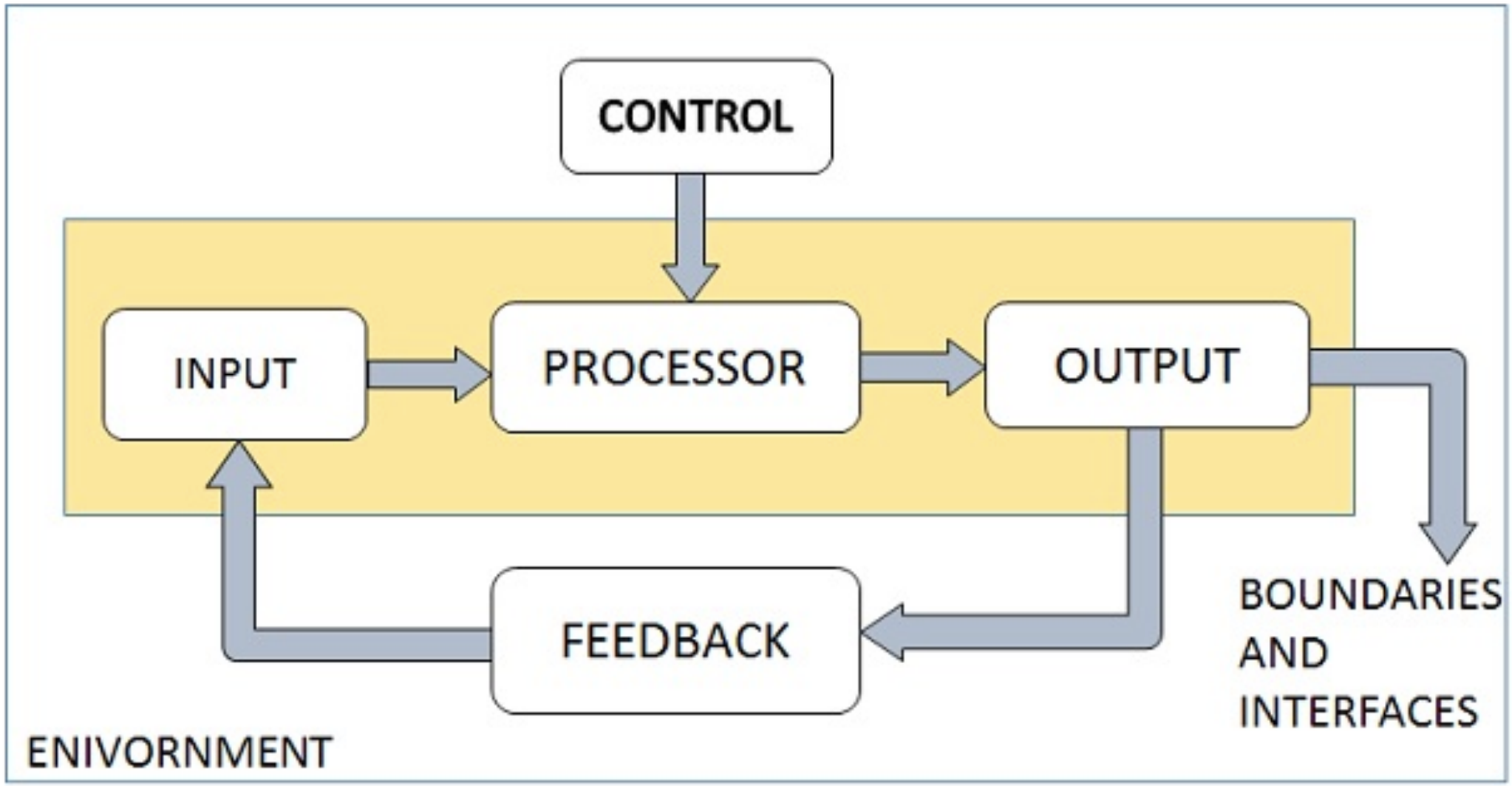
- separation of concerns
- encapsulation
- classification
- inheritance (single and multiple)
- polymorphism


Welcome to Naked Objects
www.IDEF.com: Downloads
Welcome to Naked Objects
object oriented systems design ...
http://nakedobjects.net/home/index.shtml
object oriented systems des
Most Visited
Getting Started
Latest Headlines
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
Notícias
Popular
Bookmarks
GAME & APPS
structured analysis exai
Web Search
Login
22°C
NAKED OBJECTS GROUP
Home
News
Product
Demo
Licensing
Downloads
Resources
About us
"Perfection is finally attained not when there is no longer anything to add but when there is no longer anything to take away"
Antoine de Saint-Exupéry
NAKED OBJECTS MVC
Turn a domain object model into a complete web application in minutes
NAKED OBJECTS
MVC
Naked Objects MVC combines the power of the naked objects pattern with Microsoft's ASP.NET MVC 3 framework.
Now you can take a POCO domain object model and turn it into a fully-functional web application in minutes, without writing a single line of user interface code.
You can then customise the generic user interface by adding custom style sheets, custom views and custom controllers, following standard ASP.NET patterns.
Read more product details, or better still ...
Download
Evaluation version
Watch these Tutorial Videos:
Creating a new application
Creating a simple application from scratch using the 'Code First' approach
Exploring a Naked Objects MVC
A closer look at the relationship between domain code and the user interface
Customising the user interface
Customising using .css alone. Customising by adding new views
3 benefits from using Naked Objects MVC
A faster way to get started with MVC
More productive development
Easier maintenance

Behaviorally Complete Objects
– or – Back to the Roots

- An Object models the (complete) behavior of the thing it represents
- An Object
  - knows something
    - →Properties and associations
    - →Fields
  - does something
    - →Methods

Structured Approach	Object Oriented Approach
It works with Top-down approach.	It works with Bottom-up approach.
Program is divided into number of submodules or functions.	Program is organized by having number of classes and objects.
Function call is used.	Message passing is used.
Software reuse is not possible.	Reusability is possible.
Structured design programming usually left until end phases.	Object oriented design programming done concurrently with other phases.
Structured Design is more suitable for offshoring.	It is suitable for in-house development.
It shows clear transition from design to implementation.	Not so clear transition from design to implementation.
It is suitable for real time system, embedded system and projects where objects are not the most useful level of abstraction.	It is suitable for most business applications, game development projects, which are expected to customize or extended.
DFD & E-R diagram model the data.	Class diagram, sequence diagram, state chart diagram, and use cases all contribute.
In this, projects can be managed easily due to clearly identifiable phases.	In this approach, projects can be difficult to manage due to uncertain transitions between phase.


CONTROL
INPUT
PROCESSOR
OUTPUT
FEEDBACK
BOUNDARIES
AND
INTERFACES
ENIVORNMENT

Systems Design Phases

Planning

Feasibility - id. resources - impact analysis

Requirements

Elicitation - Analysis - Validate - Formalize

Design

Transfer formal specs from Req. to Implementation

Prototyping

Implementation

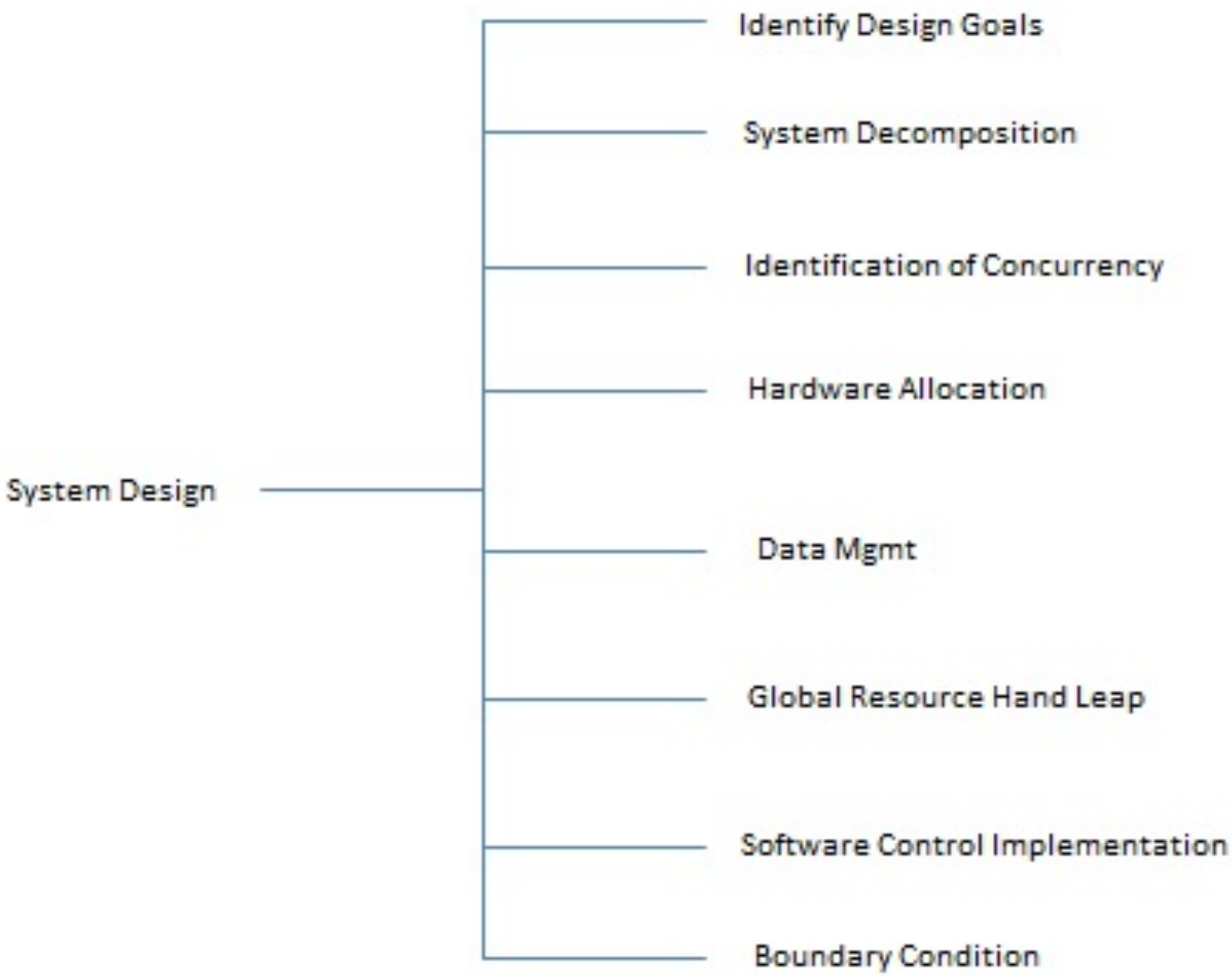
Develop-Test-Deploy-Operate-Maintain-Destroy

O Conceito de Sistema e de "Modelo"

A mudança de paradigma de “produto" para sistema também nos leva a pensar em *modelos* ao invés de protótipos e assim nos desligar de uma funcionalidade intrínseca para buscar uma harmonia entre elementos distribuídos (objetos) capaz de produzir de forma flexível e adaptável comportamentos distintos, sejam estes resultado da contribuição direta dos seus componentes ou resultado da ação conjunta. Portanto também não se pode tratar este novo elemento como uma junção de pequenos produtos mas sim como sistemas de sistemas.

A dicotomia hardware/software e produto/sistema

Historicamente hardware e software tiveram o seu desenvolvimento separado e ao mesmo tempo "integrado", tipicamente uma abordagem produto-dominante. A automação como é vista hoje não poderia se desenvolver restrita a este paradigma.

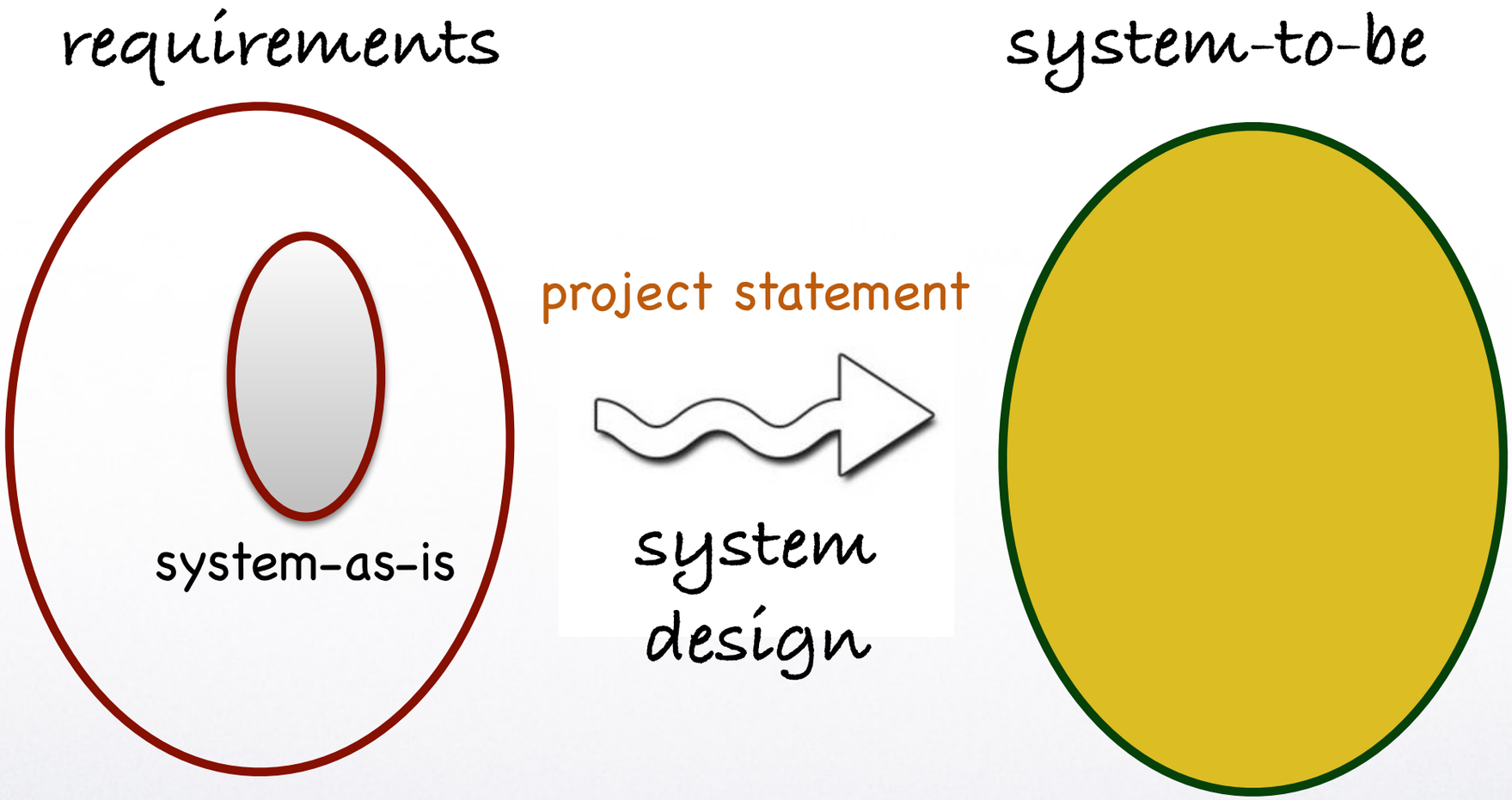

System Design
Identify Design Goals
System Decomposition
Identification of Concurrency
Hardware Allocation
Data Mgmt
Global Resource Hand Leap
Software Control Implementation
Boundary Condition

Inputs to System Design

System design takes the following inputs –

- Statement of work
- Requirement determination plan
- Current situation analysis
- Proposed system requirements including a conceptual data model, modified DFDs, and Metadata (data about data).

The System Design Challenge



Outputs for System Design

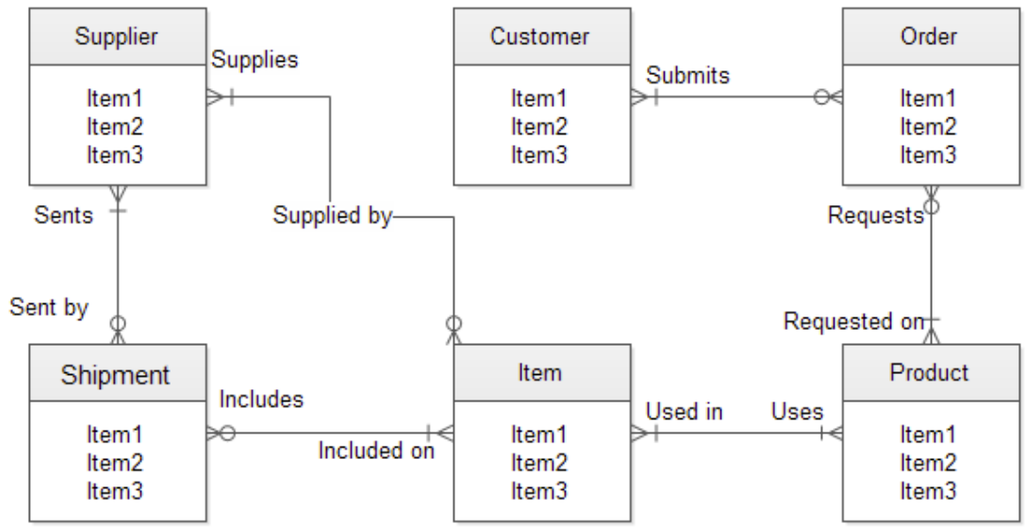
System design gives the following outputs –

- Infrastructure and organizational changes for the proposed system.
- A data schema, often a relational schema.
- Metadata to define the tables/files and columns/data-items.
- A function hierarchy diagram or web page map that graphically describes the program structure.
- Actual or pseudocode for each module in the program.
- A prototype for the proposed system.

Types of Systems Design

Logical Design

Physical Design



Architectural Design

Detailed Design

Conceptual Data Modeling

From Systems Design to OO-Systems Design

No final do século passado foi criado o Object-oriented System Design Method (OOSEM) que mudou significativamente a forma como se faz design de sistemas e se tornou um padrão internacional.

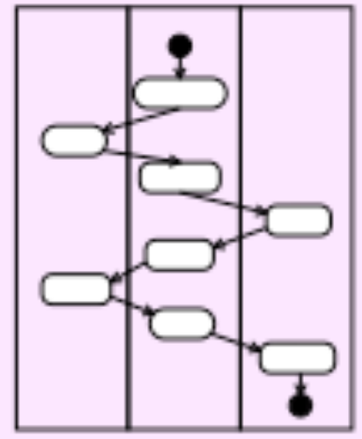
SE Practices for Describing Systems

Past

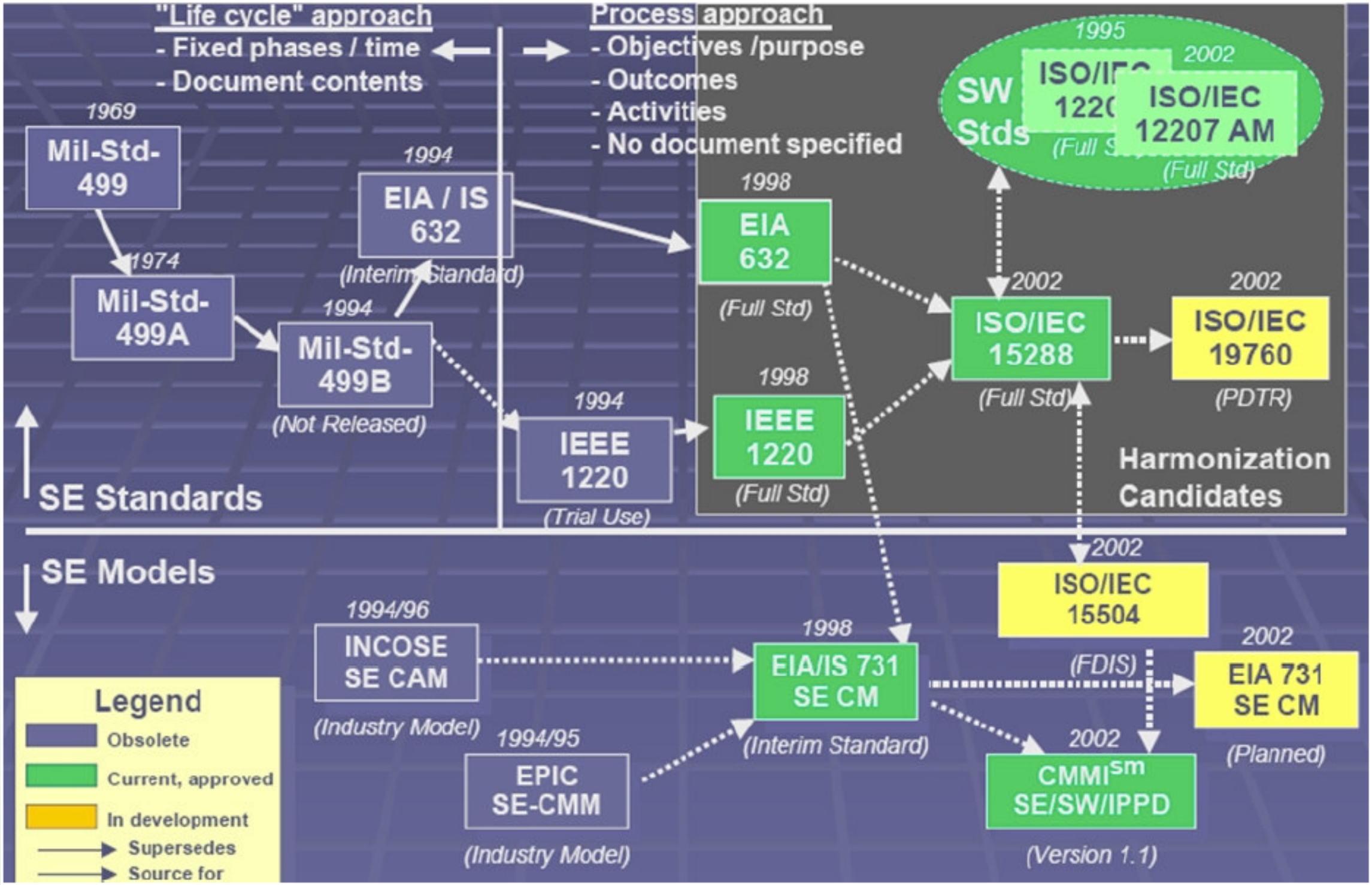
Text

- Specifications
- Interface requirements
- System design
- Test plans
- Analysis & Trade-off

Future



Model


"Life cycle" approach
- Fixed phases / time
- Document contents
Process approach
- Objectives /purpose
- Outcomes
- Activities
- No document specified
1969
Mil-Std-499
1974
Mil-Std-499A
1994
Mil-Std-499B
(Not Released)
1994
EIA / IS 632
(Interim Standard)
1994
IEEE 1220
(Trial Use)
1998
EIA 632
(Full Std)
1998
IEEE 1220
(Full Std)
1995
2002
SW Stds
ISO/IEC 12207 AM
(Full Std)
2002
ISO/IEC 15288
(Full Std)
2002
ISO/IEC 19760
(PDTR)
Harmonization Candidates
SE Standards
SE Models
1994/96
INCOSE SE CAM
(Industry Model)
1994/95
EPIC SE-CMM
(Industry Model)
1998
EIA/IS 731 SE CM
(Interim Standard)
2002
ISO/IEC 15504
(FDIS)
2002
EIA 731 SE CM
(Planned)
2002
CMMI[sm] SE/SW/IPPD
(Version 1.1)
Legend
Obsolete
Current, approved
In development
Supersedes
Source for

Fundamentação matemática para o MBSE

A. Wayne Wymore (T3SD)

1927-2011

Wymore, A. Wayne, A Mathematical Theory of Systems Engineering: The Elements , John Wiley & Sons: New York, NY, 1967.

Wymore, A. Wayne, Model-Based Systems Engineering , CRC Press, Inc.: Boca Raton, FL, 1993.

Wymore, A. Wayne, “Contributions to the Mathematical Foundations of Systems Science and Systems Engineering,” Systems Movement: Autobiographical Retrospectives, The University of Arizona, Tucson, AZ, 2004.

Os Comitês de Estudo para o Design de Sistemas

INCOSE

OMG

IEEE

Uma Norma para o Design de Sistemas

A norma ISO/IEC 15.288 foi lançada em Novembro 2002, editada por Stuart Arnold e arquitetada por Harold Lawson;

Em 2004 foi adotada pelo IEEE e passou a ser uma norma ISO/IEC/IEEE;

A última revisão foi publicada em Maio de 2015.

Uma Norma para o Design de Sistemas

Até aqui olhamos o Projeto de Sistemas do ponto de vista técnico, sempre privilegiando a formalização do processo após a fase de requisitos. Para chegar a um "projeto real" teremos que incluir a aspecto de negócios (business process) assim como os aspectos de gestão do próprio processo.

Technical processes		Technical management processes	Agreement processes	Organizational project-enabling processes
Business or mission analysis process	Integration process	Project planning process	Acquisition process	Life cycle model management process
Stakeholder needs & requirements definition process	Verification process	Project assessment and control process	Supply process	Infrastructure management process
System requirements definition process	Transition process	Decision management process		Portfolio management process
Architecture definition process	Validation process	Risk management process		Human resource management process
Design definition process	Operation process	Configuration management process		Quality management process
System analysis process	Maintenance process	Information management process		Knowledge management process
Implementation process	Disposal process	Measurement process		
		Quality assurance process		

Definition of Systems

real

"…are man made, created and utilized to provide products or services in defined environments for the benefit of users and other stakeholders"

model

"…an integrated set of elements, sub-systems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements." (INCOSE)

System and System of Systems

A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possess two additional properties:

1. Operational Independence of the Components: If the system-of-systems is disassembled into its component systems, the component systems must be able to usefully operate independently. That is, the components fulfill customer-operator purposes on their own.

2. Managerial Independence of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems. (Maier 1998, 271)

Maier, M. W. 1998. "Architecting Principles for Systems-of-Systems". Systems Engineering, 1(4): 267-84.

Classification of SoS: US Dept. of Defense

According to US DoD systems of systems could be classified into:

Virtual
Collaborative
Acknowledged
Directed

DUS(AT). 2008. Systems Engineering Guide for Systems of Systems," version 1.0. Washington, DC, USA: Deputy Under Secretary of Defense for Acquisition and Technology (DUS(AT))/U.S. Department of Defense (DoD).

Other System Architectures

Federation of Systems

A Federation of Systems (FoS) is a loosely coupled set of collaborative and distinct institutions (systems) with a week structuring control that "voluntarily" contributes to some social goal. This set could be closed or open.

Family of Systems

A family of systems is set of systems that have some common characteristics and also a loosely coupled association but that could collaborate to achieve some social goals and share facilities.

Obrigado

Reinaldo