ALGEBRAIC SOFTWARE STRUCTURING

Jean-Pierre LE PABIC
j-p.lepabic@telecom-paristech.org
Tél : 33 1 47 14 10 52

INPHONITY
20 avenue des Acacias
92500 RUEIL-MALMAISON
FRANCE

ORIGINE OF THE METHOD

- ➡PERIOD : FIRST COMPUTER-CONTROLLED PUBLIC TELEPHONE EXCHANGES
- ➡DIFFICULTIES : THE SPECIFICITIES OF EACH NEW ORDER AFFECTED THE WHOLE SOFTWARE
- ➡CONSEQUENCES : NEW ORDER => NEW SOFTWARE
- ➡DECISION : DEVELOPMENT OF A SOFTWARE STRUCTURING METHOD

WHY BRING OUT AN OLD METHOD?

- ➡COMMONLY USED SOFTWARE IS ALL BUGGED AND/OR POORLY DESIGNED (video recorders, DECT phones, GPS navigators, automobiles, etc.)
- ➡THIS METHOD HAS PRODUCED EXCELLENT RESULTS IN TERMS OF QUALITY (NO BUG), DEVELOPMENT TIME AND UPGRADABILITY
- ➡IT HAS BEEN TESTED IN COMPLEX REAL-TIME SYSTEMS : PUBLIC AND PRIVATE TELEPHONY, GSM NETWORK PLANNING TOOL

SPECIFICATIONS OF THE METHOD

1. SIMPLE EVOLUTION AS A RESULT OF :
   - ➡A CHANGE IN THE HARDWARE ENVIRONMENT
   - ➡THE ADDITION OR MODIFICATION OF FUNCTIONALITIES

2. OPERATING RELIABILITY
   - ➡NO REGRESSION IN CASE OF MODIFICATION
   - ➡LIMITED IMPACT OF SOFTWARE FAULT

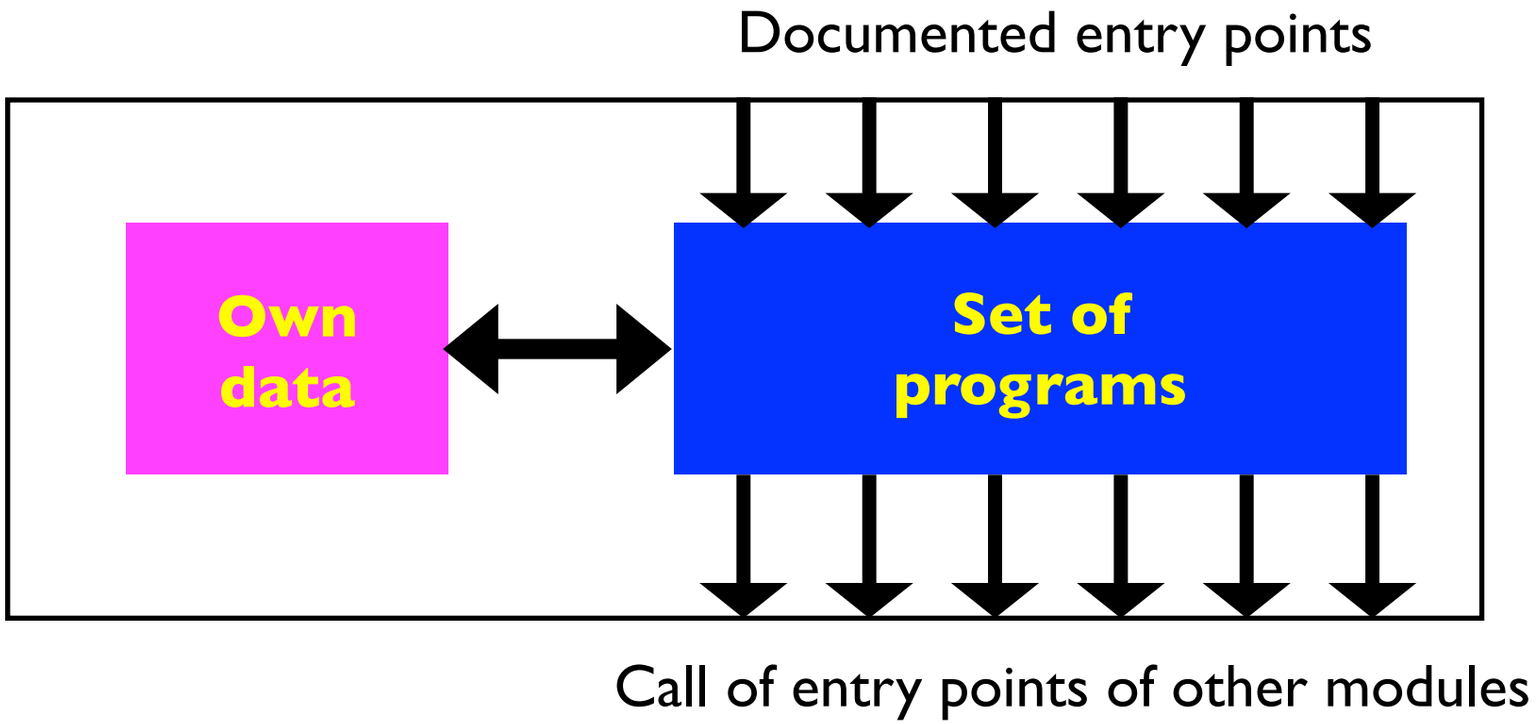
WHY « ALGEBRAIC SOFTWARE STRUCTURING » ?

1. COMMONLY USED METHODS RELATE MORE TO «ARITHMETIC» :
   - ➡ A MORE OR LESS COMPREHENSIVE UNDERSTANDING OF THE PROBLEM IS REQUIRED
   - ➡ AS SOON AS THE PROBLEM IS A BIT COMPLEX, IT BECOMES VERY DIFFICULT TO GRASP
2. WITH «ALGEBRAIC» SOFTWARE STRUCTURING :
   - ➡ THE PROBLEM IS BROKEN DOWN INTO SIMPLE ELEMENTS, EASY TO UNDERSTAND INDIVIDUALLY
   - ➡ THE DIFFERENT ELEMENTS ARE THEN «BROUGHT TOGETHER» INTO MORE COMPLEX FUNCTIONS AND SO ON
   - ➡ «ALGEBRA» MEANS INDEED «MEETING»

CORE PRINCIPLE

MODULES BUILT AROUND SIMILAR OWN DATA

- MODULE CUT-OUT USES THE SAME TECHNIQUES AS VALUE ANALYSIS
- EACH MODULE IS ONLY SEEN FROM THE OTHER MODULES THROUGH DOCUMENTED ENTRY POINTS
- EACH MODULE MANAGER DOCUMENTS THE ENTRY POINTS OF THE MODULE



THE MODULES

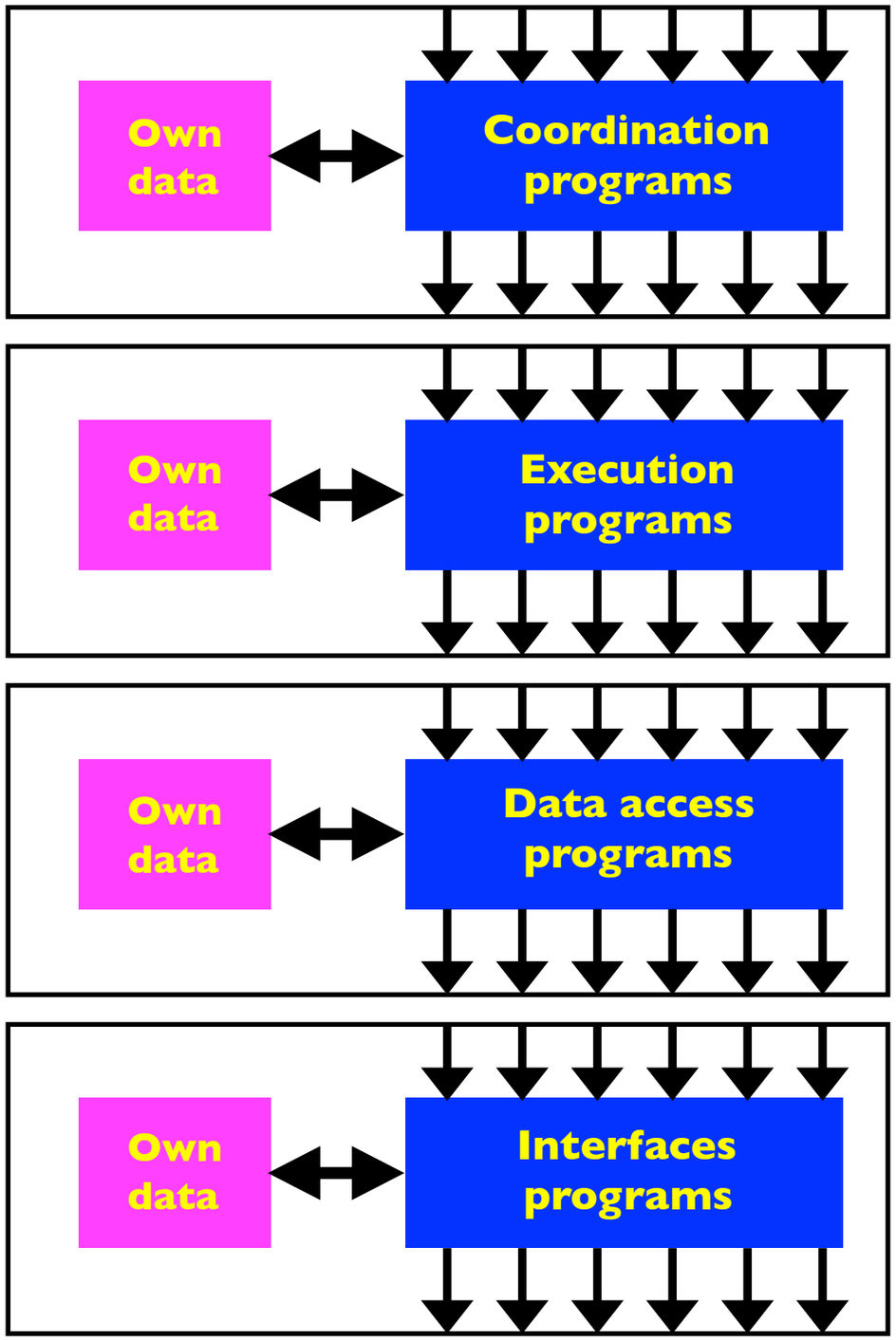
THERE ARE FOUR TYPES OF MODULES

1. COORDINATION MODULES

2. EXECUTION MODULES

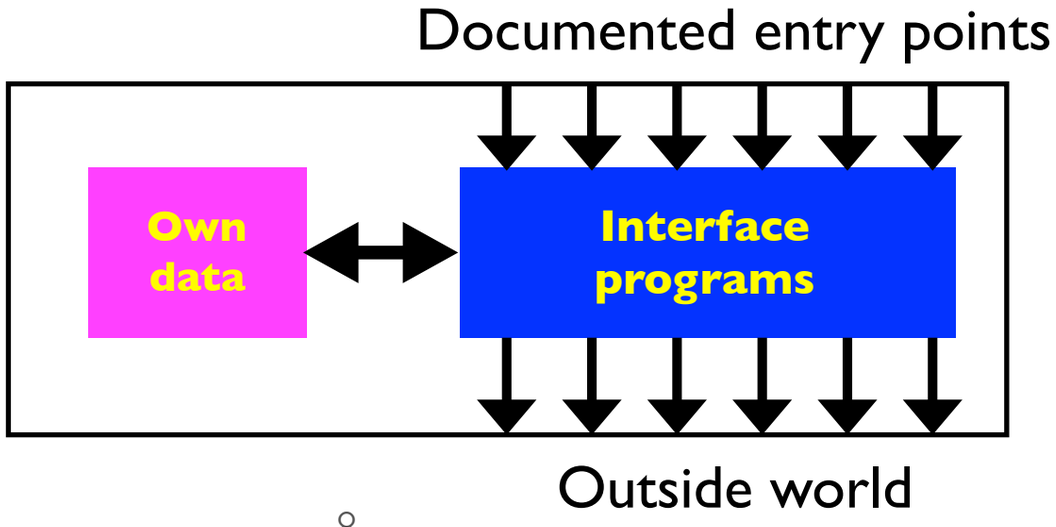
3. OPERATIONAL DATA MODULES

4. SURROUNDINGS INTERFACE MODULES



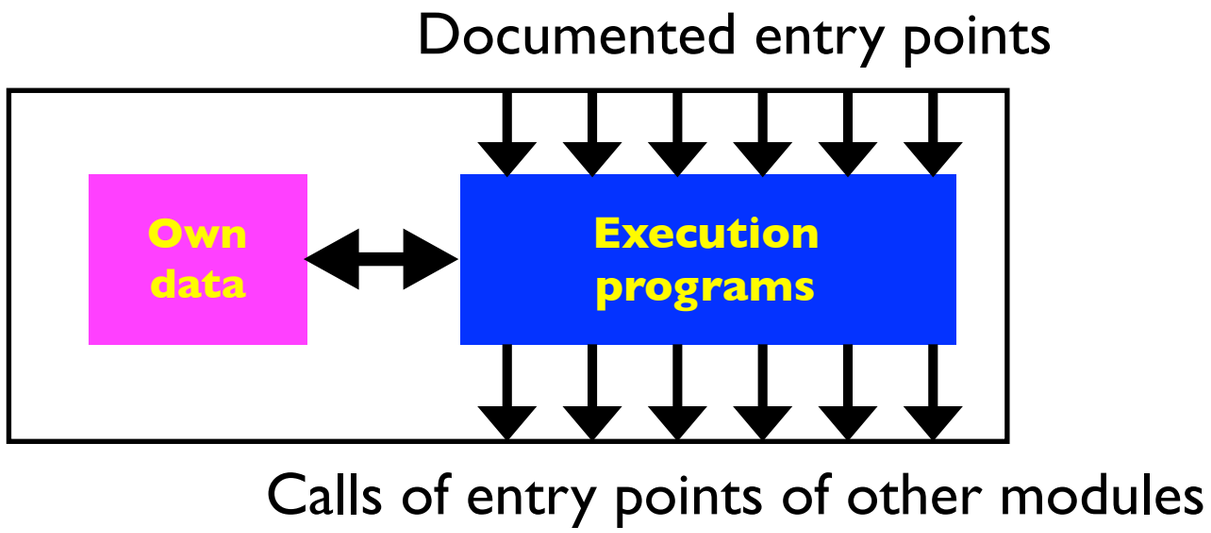
INTERFACE MODULES

- THE PURPOSE OF THE INTERFACE MODULES IS TO ISOLATE THE HEART OF THE SOFTWARE FROM THE OUTSIDE WORLD. WE DISTINGUISH:
- A FIRST LEVEL OF DECOUPLING WHEN THE INFORMATION IS MODIFIED ONLY IN ITS FORMS
  - In a car: reading the outside temperature, lighting a fire, controlling the accelerator in speed-controlled mode, etc.
- A SECOND LEVEL OF DECOUPLING WHEN THE INFORMATION NEEDS TO BE PROCESSED TO BE USABLE
  - Communication protocol change. Smart sensor in agile factory…



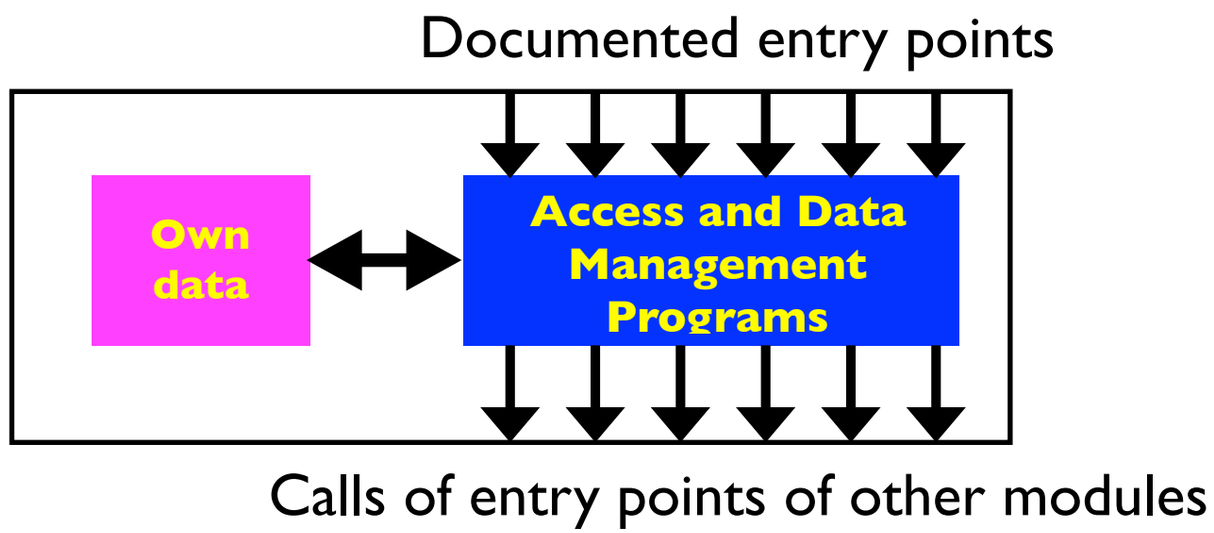
EXECUTION MODULES

- THEY CONTAIN PROGRAMS CARRYING OUT SIMILAR ANCILLARY FUNCTIONS
  - In telephony: Digital signal analysis to determine the dialed number or identity of the applicant, proceeding voice switching
- THEY MAY NOT EXIST



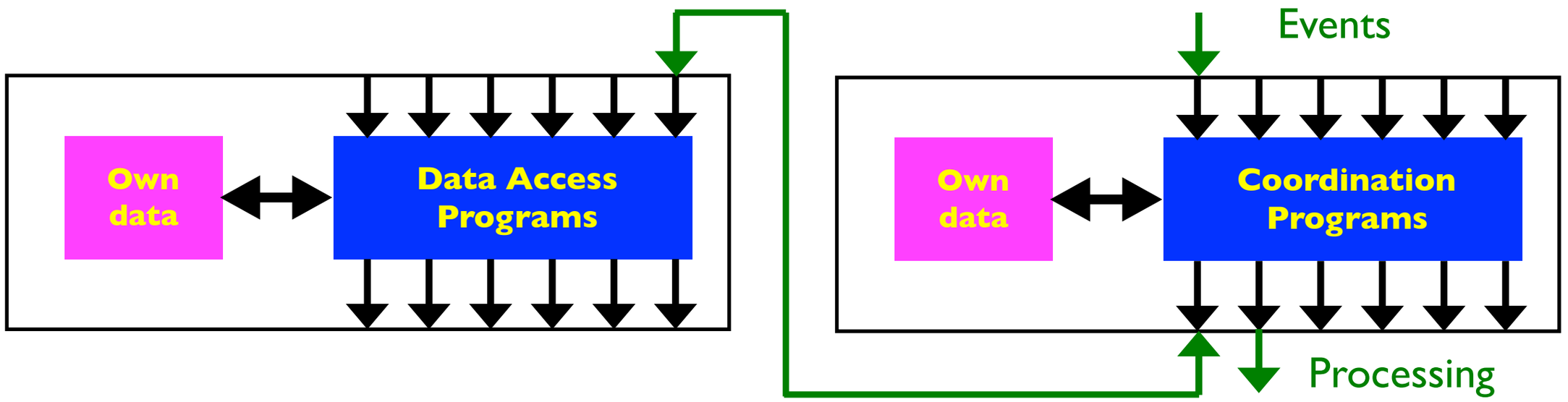
OPERATIONAL DATA MODULES

- THEY CONTAIN DATA THAT AFFECTS SYSTEM OPERATION AND WHICH DEPENDS ON EACH USE OF THE SOFTWARE (SEMI-PERMANENT DATA)
  - In telephony: Numbering Plan, Directory No<=> Equipment No translation.
- EACH COORDINATION MODULE CAN HAVE ITS DATA MODULE



COORDINATION MODULES

- ➡THEY CONTAIN THE ENTIRE CHAIN LOGIC
- ➡THEY RUN PROGRAMS ACCORDING TO THE EVENT, THEIR OWN LOGIC AND POSSIBLY DATA FROM THE DATA MODULES
- ➡THESE MODULES CAN CONTROL PROCESS FLOW OR MANAGE TIME CONSTRAINTS

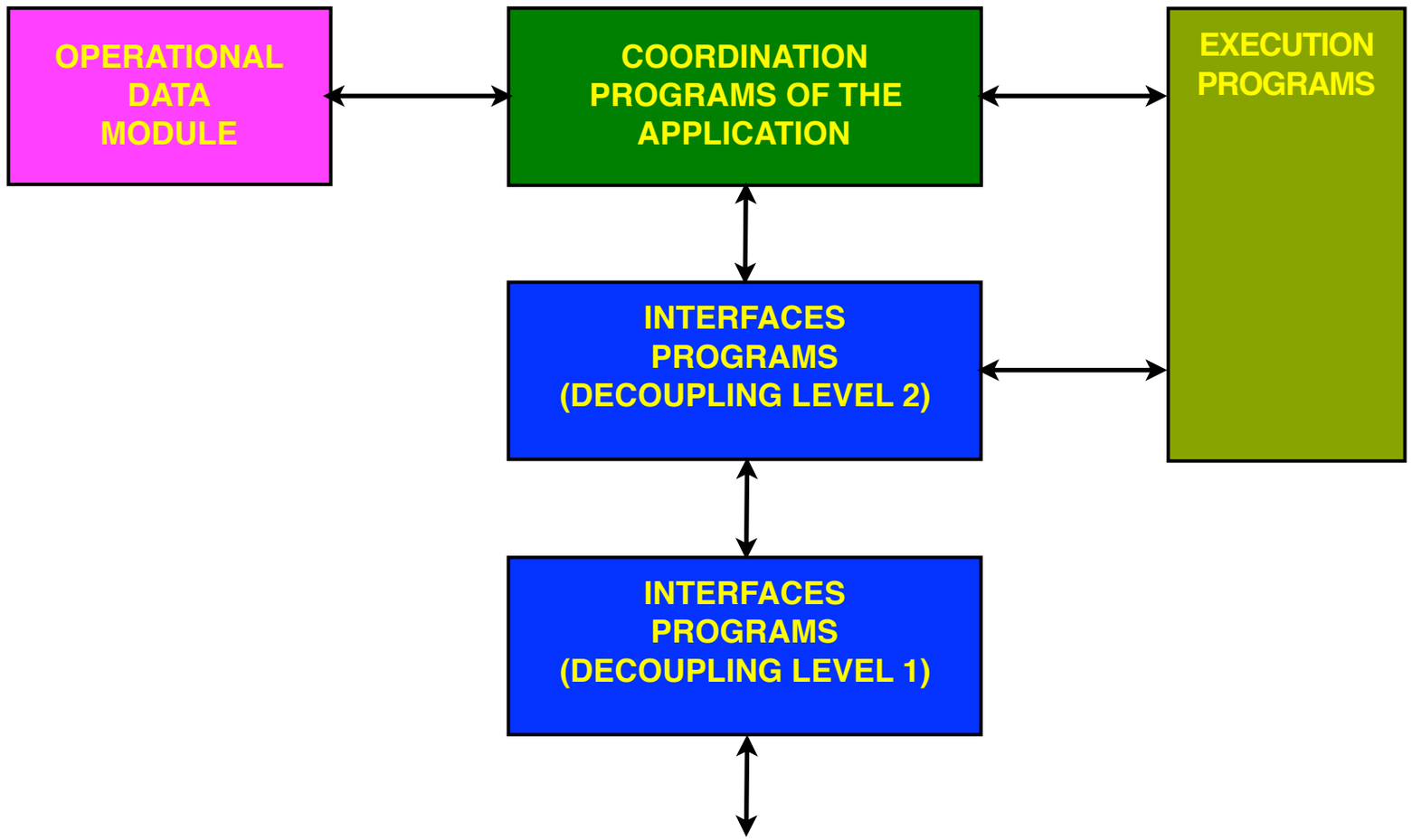


SUPER-FUNCTIONS

- THE SET OF PROGRAMS RELATED TO A SOFTWARE FUNCTIONALITY IS A SUPER-FUNCTION. THERE MAY BE SEVERAL, INCLUDING :
  - THE SUPER-FUNCTION APPLICATION
  - THE SUPER-FUNCTION OPERATIONS
  - THE SUPER-FUNCTION MAINTENANCE
  - THE SUPER-FUNCTION AUDITS
  - THE SUPER-FUNCTION COUNTS/OBSERVATIONS
  - THE SUPER-FUNCTION SEQUENCING

- OPERATIONS, MAINTENANCE, AUDITS, COUNTING AND SEQUENCING SUPER-FUNCTIONS CAN BE ANALYZED AND CODED AFTER COMPLETION OF THE APPLICATION SUPER-FUNCTION

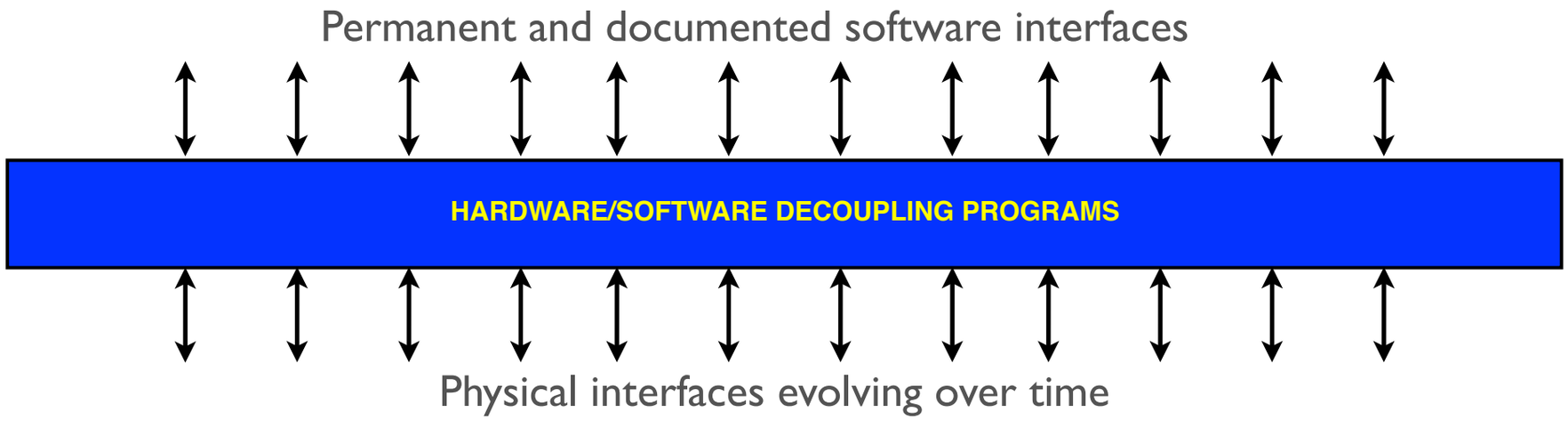
DIAGRAM OF A SUPER-FONCTION



EACH SUPER-FUNCTION INCLUDES AT LEAST COORDINATION AND INTERFACE PROGRAMS.

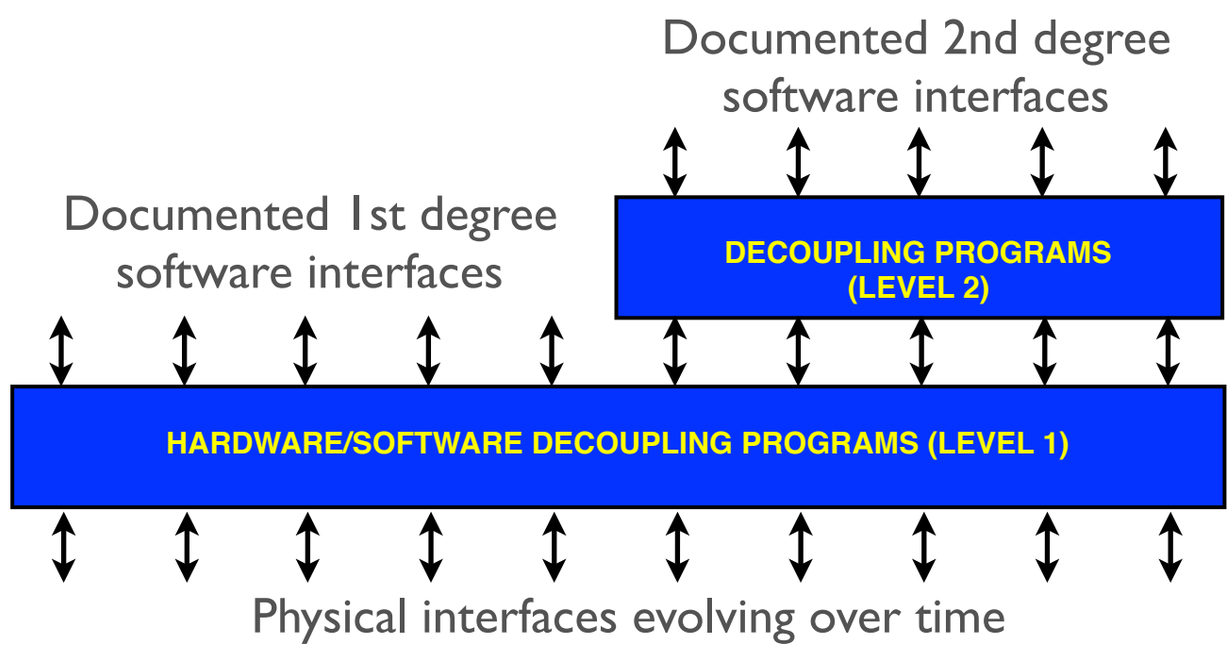
IMPLEMENTING

- ➡FIRST STEP : PHYSICAL INTERFACES
- ➡IDENTIFY ALL INPUT-OUTPUT AND GROUP THEM BY NATURE INTO DIFFERENT MODULES
  - ➡Ex : Translation: Physical address of a sensor + read value <=> «engine temperature» + value in degrees Celsius
- ➡DEVELOP EACH MODULE CONSISTING OF READING AND WRITING PROGRAMS
- ➡DOCUMENT ENTRY POINTS USED BY OTHER MODULES



IMPLEMENTING

- ➡SECOND STEP (EVENTUALLY) : LOGICAL INTERFACES
- ➡DEFINE FEATURES OF LOGIC INTERFACE MODULES
  - ➡Ex : In telephony, translate numbering pulses into numbers
- ➡DEFINE ELEMENTARY ACTION TABLES AS STATUS/EVENT PAIRS
- ➡DOCUMENT ENTRY POINTS USED BY OTHER MODULES



IMPLEMENTING

➡THIRD STEP : SEMI-PERMANENT DATA

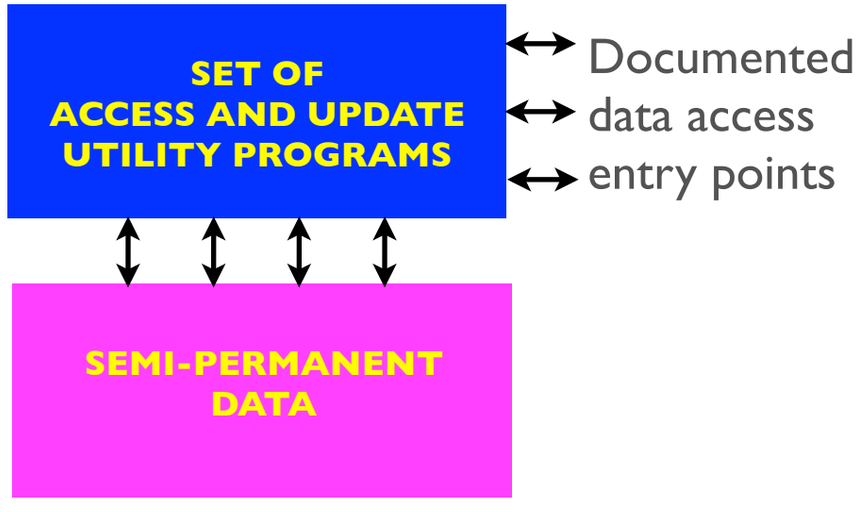
➡IDENTIFY SEMI-PERMANENT DATA

Ex : Translation table « directory numbers <=> physical addresses »; Engine temperature limit, etc.

➡GROUP THEM BY NATURE INTO DIFFERENT MODULES

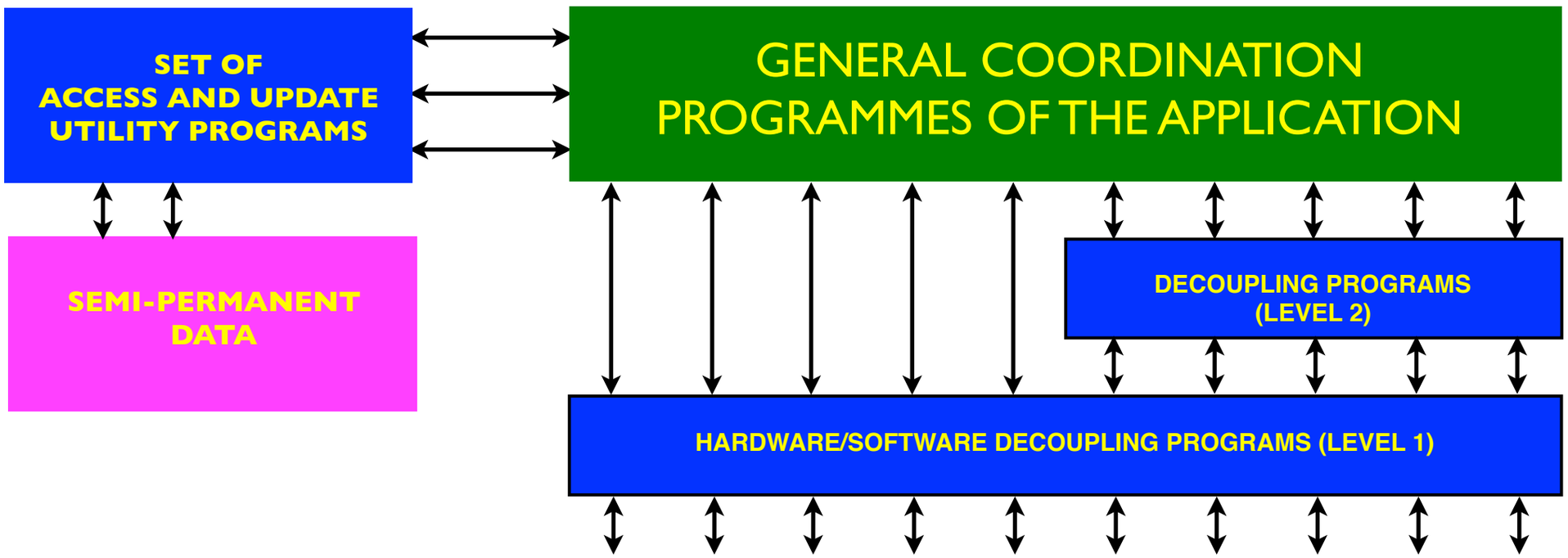
➡DEVELOP ACCESS AND UPDATE UTILITY PROGRAMS

➡DOCUMENT ENTRY POINTS USED BY OTHER MODULES



IMPLEMENTING

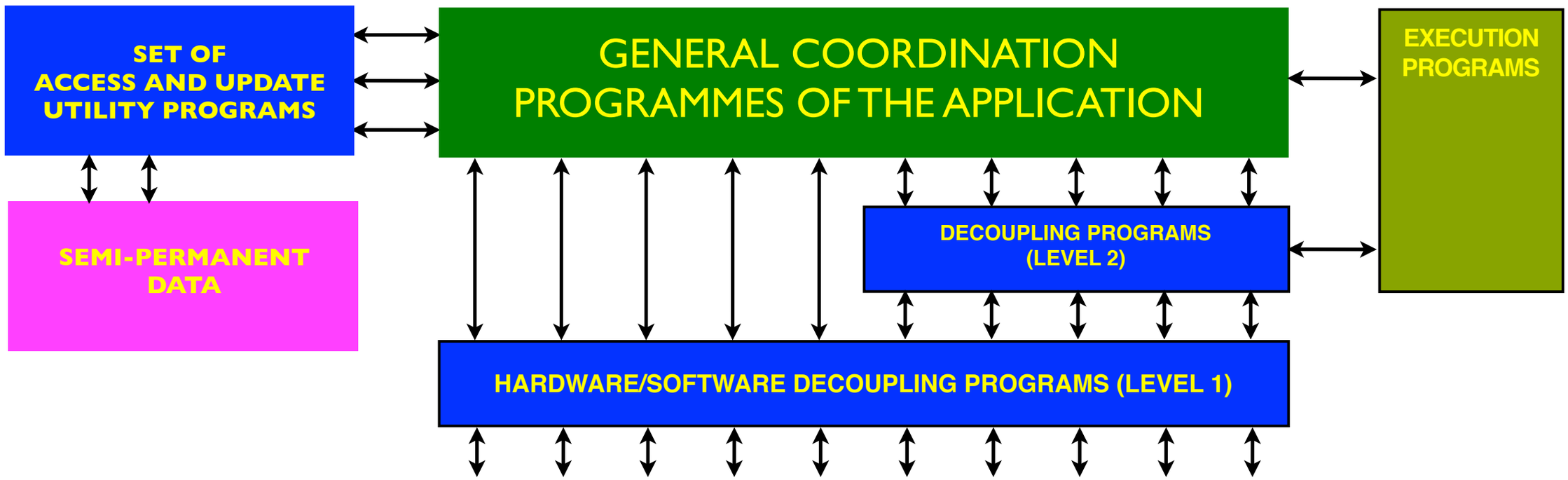
- ➡FOURTH STEP : COORDINATION OF THE PROCESS
- ➡DEVELOP THE CORE OF THE PROCESS IN THE FORM OF PROGRAMS TREATING EACH A STATE/EVENT CASE
- ➡DOCUMENT ENTRY POINTS USED BY OTHER MODULES



Physical interfaces evolving over time

IMPLEMENTING

- ➡FIFTH STEP : EXECUTION PROGRAMS
- ➡THEY PROVIDE SERVICES JUSTIFYING THE CREATION OF SPECIFIC MODULES. Ex : signal processing (decoding numbering frequencies, caller identity, etc.) or voice switching in telephony
- ➡DOCUMENT ENTRY POINTS USED BY OTHER MODULES



Physical interfaces evolving over time

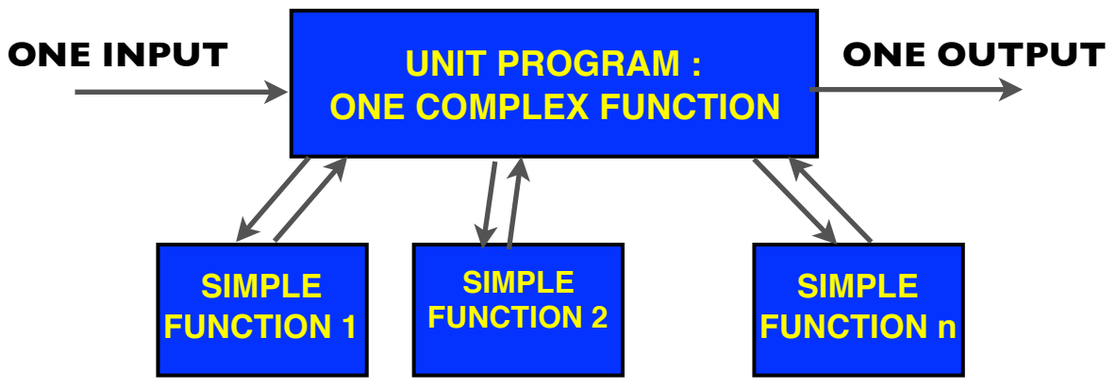
SIMPLICITY OF THE ELEMENTS

➡EACH PROGRAMME HAS ONLY A VERY LIMITED NUMBER OF PARAMETERS AND IS THEREFORE EASILY UNDERSTOOD IN ITS ENTIRETY BY THE ANALYST

➡Ex : Reading or writing data, control an actuator

➡IDEALLY: SHORT PROGRAM, ONE INPUT, ONE OUTPUT, ONE FUNCTION

➡A COMPLEX COORDINATION FUNCTION IS ACHIEVED BY SEQUENCING SEVERAL SIMPLE PROGRAMS



➡Ex : Switch the whole lighting in «night» mode in a car

SIMPLICITY OF STRUCTURE

- THE EXECUTION LOGIC IS DETACHED FROM THE SEQUENCE LOGIC
- THE LATTER IS CONTAINED EXCLUSIVELY IN THE COORDINATION MODULES
- "GOTO" OR "IF" BRANCHES ARE PROHIBITED.
- ONLY «CASE» PROCESSING ARE ALLOWED

- **STATE X**
  - **«CASE» Event a**
    - **Action x01**
    - **Action x02**
    - **Action x0p**
  - **«CASE» Event b**
    - **Action x11**
    - **Action x12**
    - **Action x1p**
  - **«CASE» Event c**

- **STATE Y**
  - **«CASE» Event a**
    - **Action y01**
    - **Action y02**
    - **Action y0p**

A HIGH LEVEL OF QUALITY

- ➡INITIAL BUGS ARE RARE BECAUSE EACH PROGRAM IS SIMPLE AND EASY TO DESIGN AND TEST
- ➡CORRECTIONS ARE EASY TO MAKE AS WELL AS LATER DEVELOPMENTS
- ➡THERE IS NO «BAD SURPRISE» DURING THE DEVELOPMENT
- ➡DEVELOPMENT TIME AND COSTS ARE OPTIMIZED
- ➡THE PROGRAM NO ONE DARES TO UPDATE NO LONGER EXISTS
- ➡THE STRUCTURE OF THE SOFTWARE IS EASILY UNDERSTOOD BY PEOPLE WITHOUT ANY COMPUTING SKILLS

DECISIVE COMPETITIVE ADVANTAGES

➡FLEXIBILITY OF THE SOFTWARE ALLOWS :

- AN OPTIMAL DEVELOPMENT TIME
- CHANGES AT MINIMUM COST
- POSSIBILITIES FOR IMPLEMENTING CUSTOMER REQUESTS THAT WERE PREVIOUSLY NOT POSSIBLE BECAUSE THEY WERE TOO EXPENSIVE
- SHORTER ADAPTATION TIMES THAN COMPETITION

➡UNDERSTANDING OF SOFTWARE STRUCTURE BY DECISION MAKERS IMPROVES:

- THE BARGAINING POWER
- REACTIVENESS TO CUSTOMER REQUESTS