ARIS is a meta-model for systematic process modelling.

**Name:** Architecture of Integrated Information Systems (ARIS)

**Founder:** A.-W. Scheer (Saarland University)

**Idea:** Meta-model for process modelling (guidelines to model processes systematically)
ARIS divides processes in 5 different views.

Each process (process view)
- Is performed by an actor (organization view),
- Requires or generates data (data view),
- Is composed by several actions (function view)
- And has some kind of output (product/service view).

In other words: the model answers the following questions
- What is done? → Function view
- Who does it? → Organization view
- Why is it done? → Product/Service view
- Which information is needed/provided? → Data view
- How is it done? → Process view
The split in views reduces complexity and ensures a systematic approach.

- The process view connects all other views
- Every view is illustrated in a certain modelling notation
## THE ARIS METHOD

### CONTENT

<table>
<thead>
<tr>
<th></th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product / Service View</td>
</tr>
<tr>
<td>2</td>
<td>Function View</td>
</tr>
<tr>
<td>3</td>
<td>Organization View</td>
</tr>
<tr>
<td>4</td>
<td>Data View</td>
</tr>
<tr>
<td>5</td>
<td>Control / Process View</td>
</tr>
</tbody>
</table>
The product/service tree is a **hierarchical list** of all products/services.

→ It provides an **overview** of the entire product/service **portfolio**.
THE ARIS METHOD
CONTENT

1. Product / Service View
2. Function View
3. Organization View
4. Data View
5. Control / Process View
The function tree is a hierarchical list of all functions.

→ It provides an overview of the activities of a company.
# THE ARIS METHOD

## CONTENT

<table>
<thead>
<tr>
<th></th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product / Service View</td>
</tr>
<tr>
<td>2</td>
<td>Function View</td>
</tr>
<tr>
<td>3</td>
<td>Organization View</td>
</tr>
<tr>
<td>4</td>
<td>Data View</td>
</tr>
<tr>
<td>5</td>
<td>Control / Process View</td>
</tr>
</tbody>
</table>

© 2016 Software AG. All rights reserved.
The organizational chart is a *hierarchical list* of all organizational units/elements of a company.

→ It provides an *overview* of the *organizational structure* of a company.
Most used are “organizational unit” and “position”, but there are also other elements.

- Organizational unit:
  - Represents a department or sub-department (e.g. HR)

- Position:
  - Represents a job description (e.g. HR manager)
  - Each employee occupies exactly one

- Role:
  - Represents a task description (e.g. ARIS Trainer)
  - One employee can occupy none, one, or several (switch roles as necessary)

- Internal person:
  - Represents a specific person (e.g. John Doe)
THE ARIS METHOD
CONTENT

1. Product / Service View
2. Function View
3. Organization View
4. Data View
5. Control / Process View
An ERM is used to model data.

**Name:** Entity Relationship Model (ERM)

**Founder:** Chen

**Idea:** Modelling notation for (relational) databases
(Relational) databases are nothing more than (connected) tables.

→ ERM is used to model such a system of tables.
The model is composed of three basic elements which are linked to each other.

**Entity type**
Type of a real world object

**Relationship type**
Logical link between two entity types

**Attribute**
Properties of entity types or relationship types
A type is an abstraction of a real world object (e.g. „student“)

The concrete object is called an instance (e.g. „John Doe“)

An instance of the previous example would be: “John Doe borrows Economics 101”

However: modeling in ERM is always done at type level (!)
A key attribute identifies one entity (instance) exclusively.

- E.g. an ISBN identifies one book exclusively, whereas the name of the author most likely would indicate several books.
- E.g. “151430” identifies “Economics 101” whereas “Smith” would identify „Economics 101“ and “Advanced Economics”.
Any relationship type can be specified by one of the three multiplicities.

1:1  
one-to-one (1:1)  
E.g.: student owns mobile phone  
→ Every mobile phone belongs to exactly one student

1:N  
one-to-many (1:N)  
E.g.: student borrows book  
→ One student can borrow several books, but one book can only be borrowed by one student (at a time)

N:M  
many-to-many (N:M)  
E.g.: student enrolls in class  
→ One student can enroll in several classes, and one class is enrolled by several students
There are three ways to write those multiplicities down.

**Chen** (1976)
Question: How many objects can be assigned to one object of the other side?
- How many students can be assigned to one book? → 1
- How many books can be assigned to one student? → n

**Schlageter/Stucky** (1977)
Question: With how many other objects can one object (of this side) be related to?
- With how many books can one student be related? → n
- With how many students can one book be related? → 1

**Min-Max-Notation** (ISO-Notation, 1982)
- States a range (minimum, maximum) of possible relations
- Instead of an ‘n’, a * is widely used

⚠️ Be careful: the UML class diagram uses the “old” Chen notation.
**ERM – FROM MODEL TO DATABASE**

1:n example

- Entity types represent tables, attributes represent data fields
- A data record can be identified by the primary key
- 1:n-relationships can be implemented using only two tables (one for each entity type)
- In the books-table “mat.no.” is called a **foreign key**

**Student**

<table>
<thead>
<tr>
<th>mat.no.</th>
<th>first name</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>John</td>
<td>Doe</td>
</tr>
<tr>
<td>12356</td>
<td>Jane</td>
<td>Doe</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Book**

<table>
<thead>
<tr>
<th>ISBN</th>
<th>name</th>
<th>author</th>
<th>mat.no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>151430</td>
<td>Economics 101</td>
<td>Smith</td>
<td>12345</td>
</tr>
<tr>
<td>151540</td>
<td>Economics 101</td>
<td>Marx</td>
<td>12345</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
• If the student buys the books, the relationship changes into n:m
• An additional attribute (number of copies) can be added to the relationship type
• n:m-relationships are modelled using 3 tables (one for each entity type and relationship type)
• The primary key in the relationship-table is a composed key of the two foreign keys
### THE ARIS METHOD CONTENT

<table>
<thead>
<tr>
<th></th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product / Service View</td>
</tr>
<tr>
<td>2</td>
<td>Function View</td>
</tr>
<tr>
<td>3</td>
<td>Organization View</td>
</tr>
<tr>
<td>4</td>
<td>Data View</td>
</tr>
<tr>
<td>5</td>
<td>Control / Process View</td>
</tr>
</tbody>
</table>
An EPC is used to model processes.

**Name:** Event-driven Process Chain (EPC)

**Founder:** A.-W. Scheer (Saarland University)

**Idea:** Modelling notation for processes
There are three basic elements.

- **Functions** are activities carried out by a real person
- **Events** trigger these activities or are the result of them
- The control flow represents the sequence of activities

Rules:
- Every EPC **starts** and **ends** with an event.
- Events and functions **alternate**
  (a function cannot be followed by another function and an event cannot be followed by another event)
- Exception: **trivial events** must not be modelled
  (events who only state the termination of a function without adding any useful information, e.g. “check invoice” → “invoice checked”)

**Diagram:**
- **hungry**
- **eat**
- **full**
Operators split and merge the control flow.

There are three operators:

- **AND** – all succeeding paths will be executed (simultaneously)
- **OR** – one or several of the succeeding paths will be executed (simultaneously)
- **XOR** – only one of the succeeding elements will be executed (exclusive OR)

Rules:
- splitting operator = joining operator
Events cannot ‘decide’.

- Only use the AND-Operator to split the control flow after an event
- Apart from these two, there are no restrictions on how to use operators
  - e.g. use one operator to split in four different paths
  - e.g. use an OR-Operator to merge two events
Loops are also possible.

- Introduced by XOR
- Joining occurs before Splitting
- Avoid ‘infinite loops’!

If you don’t know the password, you might be stuck forever!
Process Interfaces are used to link independent processes.

- Process Interfaces are used to signal:
  - that the process triggers the execution of another process (like in the example)
  - that the process is triggered by another process

- Process Interfaces refer to a process that is not part (logically) of the original process
Assignments are used to outsource details.

- With assignments, functions can be described in more detail in another EPC
  - Outsource detail in another EPC
  - Reuse sub-processes in different EPCs

Rules:
- The same events are used both in the high-level and the low-level process ("hungry" / "food is ready")
- Has to link to an EPC with the same name ("cook")
- Starts and ends with a process interface, to indicate that the process does not stand alone
To each function, different roles (according to RACI*) can be assigned.

**Responsible:** Role is responsible for the execution of the function; execution may be handled by that person also, or delegated to somebody else (still the responsibility remains with the role)

**Accountable:** Role to whom “R” is accountable; who must sign off and approve; commercial responsibility

**Consulted:** Role has information and/or capability necessary to execute the function

**Informed:** Role must be notified of results; has the right to ask for information; but does not need to be consulted during execution

---

*The RACI Matrix is the most prevalent tool for clarifying roles and responsibilities in cross-functional projects and processes.*
EEPC – Extended Event-driven Process Chain

- An eEPC includes all relevant information to the process.
- Integrates all the other views:
  - Roles (Organization)
  - Product/Service
  - Data Input / Output
ARIS is able to automatically transfer any EPC into a BPMN.
LEARN MORE

Students / Universities

→ University Relations
   www.softwareag.com/university

✉ World-wide mail account
   university@softwareag.com

Customers / Partners

→ Global Education Services
   www.softwareag.com/education

✉ Regional mail accounts
   Please refer to “contact us” section on the webpage mentioned above.