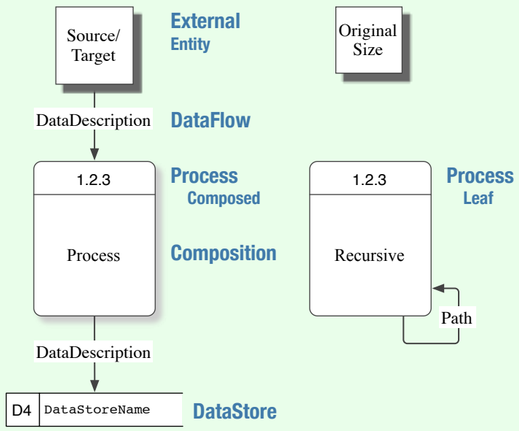
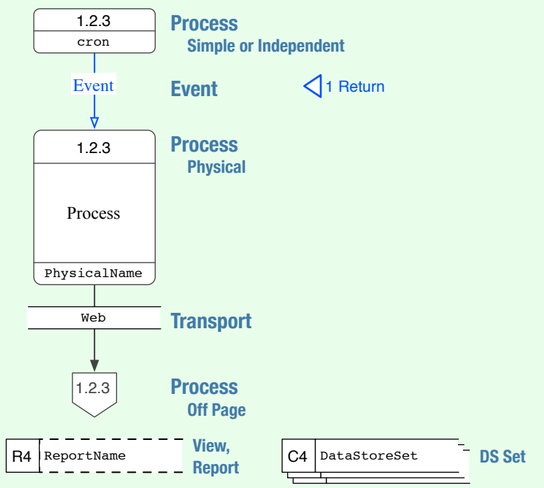
Introduction[Symbol & Explanation]

Structured Systems Analysis and Design Methodology

by Chris Gane & Trish Sarson 1979



SSADM Extension



Software Gems Pty Ltd
Derek Ignatius Asirvadem

Introduction
Unfortunately the book is out of print, but some suppliers provide used copies. Beware that there are many progressions and entire"methodologies" which are based on SSADM, they serve to impose massive documentary requirements, but the core simplicity is lost, and the Process Modelling method is buried.
- The main thing is to perceive the system only in terms of **Processes** only, and its needs, not as GUI windows or interaction.
- Of course, the Process Model goes hand-in-glove with a formal Data Model.

ExternalEntity
At the highest level, the Start or End point of *DataFlows* in the system.

DataFlow
- Single-ended arrow
- One end of a *DataFlow* must be a *Process* and the other end must **not** be a *Process*
- One label per true discrete *DataFlow* (Data, Document, Message, etc), it is a noun, not a verb (the *Process* is a verb)
- One arrow per group of *DataFlows* (same source and destination)

Process
- Application, System, Program, Procedure
- The *Process* performed on the input, in order to produce the output, expressed as a verb
- Try to draw a maximum of 5 to a page, an absolute maximum of 7

Process Composed
The shadow indicates that this *Process* is exploded on another page

Process Leaf
The leaf level in the process hierarchy, the lowest level of process definition is not a composition, it has no shadow

DataStore
File, Table or location (not necessarily electronic) where the data is persistent. Refer also to Data Dictionary.

The Standard dates from the 1970's, and has not been updated These are the extensions that I have developed over the decades.

Process Simple
Not requiring the same analytical focus as the other Processes on the page, but which must be identified (eg. cron). External systems are best treated as ExternalEntities, not as Processes, which should be limited to that which is within the modelled system.

Event or Trigger
Special form of DataFlow (such as an Execute request from cron, or an Event) that does not issue from an ExternalEntity. It is allowed to "call" another Process. Note the arrow is blue & hollow (faithful to IDEF0). For event-driven systems, a State Transition diagram is more appropriate.

Process Physical
The addition of the Program, Procedure or Object *name* (generally provided in the delivery documentation, after the system has been constructed)

Transport
In modern systems, it is sometimes important to show the boundaries that are crossed when a *DataFlow* is imported/exported to an *ExternalEntity*, eg. Web or TCP/IP. A *DataFlow* moves **through**, not to or from, a *Transport*

Process (Off-Page)
A *Process* defined elsewhere in the model. In essence, it is an **off-page connector**. (Included because it is a common extension, however it is a departure from Gane & Sarson's core principles and thus results in the very complications that SSADM eliminates. Therefore it is **not recommended**.)

View or Report
Refer to the note in Data Dictionary. The dashed line is carried over from IDEF1X/IDEF1R, indicating its virtual and denormalised nature.

DataStore Set
When it is necessary to show a collection: a cluster or bunch of reports.

Other Guidelines

Duplicate Symbol
- The original methodology does not allow crossed lines: that is a very important guideline, which improves clarity, and it is strongly recommended. It will elevate the clarity of your diagram. The relevance of this rule will be exposed as you gain experience. (I carry this rule into IDEF1R.)
- The original provided annotated *ExternalEntity* and *DataStore* symbols for objects that are duplicated on a page. However, they detract from the simplicity, and those symbols are excluded here. It is just as easy (demanded) to avoid duplicated symbols while reducing (if not eliminating) crossed lines.

Crossed Lines
Eliminate, if not minimise.

Composition/Decomposition
- Use legal numbering for *Processes*
- A single *Process* can be exploded onto another page (use the Action tool to open a separate canvas, document or URL), wherein the *Process* is exposed (exploded, decomposed) into its component *Processes*
- In this manner, an entire system can be (a) analysed and (b) modelled and defined, to whatever level of decomposition is required, in a single tree structure, the *DataFlow Diagram*
- The highest level (**Context**) of the *DataFlow Diagram* is a **single** *Process* identifying the system, its *ExternalEntities*, and its external *DataFlows*
- Single *Processes* (canvases, pages) can thus be viewed at the level of detail relevant to the area (scope) of analysis
- Buttons (enabled and disabled forms) are provided for buttons to *Return to previous level* on the Exploded pages (define an Action for it).

Extensions
Minimise: maintain the core concepts.

Shadow
Indicates explosion, do not use shadow to indicate anything else.

Colour
Generally avoid.
- If used, employ only light pastels, and use it to differentiate *Process* and *DataStore* **types** only.
- I use it only in the final edition, and only for the *DataStore number*, to match the colours in the Data Model.

Data Dictionary

The Data Dictionary is a single definition (progressing, as the Data and Process Models progress) which lists (early stage) or defines (final stage) all DataStores & ExternalEntities. It accompanies the Process Model (DataFlow Diagram, in the early stages it is list, as the models progress, each element is further defined, at the final stage it is a final definition, almost SQL/DDL. It is integrated with the Data Model, there is just one Data Dictionary for both, it progresses with the progression of the Data Model, parallel to the Process Model.

The DataStore *number* is an index, it is used in all instances:
- C*n* Database Cluster [of tables], usually BusinessInstrument; logical entity
- D*n* Database table
- F*n* File within the system
- P*n* Denotes a physical *thing* (outside the system) that is relevant to comprehension of the Process Model; library book; cash; etc. The DatFlows and DataStores are dotted lines. The purpose in the model is to register it in the Data Dictionary.
- Q*n* Event or Message queue
- R*n* Report (in modern systems, it is not necessary to document all reports and ResultSets, these is the few that are imported/exported to an ExternalEntity)
- X*n* External File (usually import/export)

Alternative • IDEF0

The primary purpose of a Process and Data Models is to provide a platform for communication (it is only as they progress that they become relevant to a system or implementation. SSADM is perfect for communicating with users, in order to clarify the processes and to obtain validation (eventually sign-off), It has just four symbols, and is easy to understand. For most modern systems, it is more than adequate for technical purposes.

For more intense systems (not "complex" because complexity is resolved by the simplicity of the design) such as factory automation or robotics, IDEF0 is recommended, it documents the **Controls** (eg. Legislature; SOP; Gauge; etc) and the **Mechanisms** or methods (eg. Open Support Ticket; Dam Gate). It is given here for comparison. Whereas the DataFlows in SSADM are top-to-bottom, in IDEF0 they left-to-right, forming a cascade.