JACKSON STRUCTURED PROGRAMMING (JSP)

Jackson Structured Programming was developed in the 1970's by Michael Jackson (1976), and became a widely used design method, especially in Europe.

JSP is a method for structured programming based on correspondences between data stream structure and program structure. JSP structures programs and data in terms of sequences, iterations and selections, and as a consequence it is applied when designing a program's detailed control structure. The method applies to processing of any data structure or data stream that is describable as a hierarchical structure of sequential, optional and iterated elements.

The intent is to create programs which are easy to modify over their lifetime. Jackson's major insight was that requirement changes are usually minor tweaks to the existing structures.

For a program constructed using JSP, the inputs, the outputs, and the internal structures of the program all match, so small changes to the inputs and outputs should translate into small changes to the program.

JSP structures programs in terms of four component types:

- fundamental operations
- sequences
- iterations
- selections

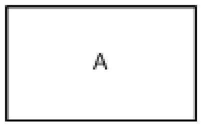
The method begins by describing a program's inputs in terms of the four fundamental component types. It then goes on to describe the program's outputs in the same way. Each input and output are modelled as separate Data Structure Diagram (DSD).

The input and output structures are then unified or merged into a final program structure, known as a Program Structure Diagram (PSD). This step may involve the addition of a small amount of high-level control structure to marry up the inputs and outputs. Some programs process all the input before doing any output, whilst others read in one record, write one record and iterate. Such approaches have to be captured in the PSD.

The PSD, which is language neutral, is then implemented in a programming language. JSP is geared towards programming at the level of control structures, so the implemented designs use just primitive operations, sequences, iterations and selections. JSP is not used to structure programs at the level of classes and objects, although it can helpfully structure control flow within a class's methods.

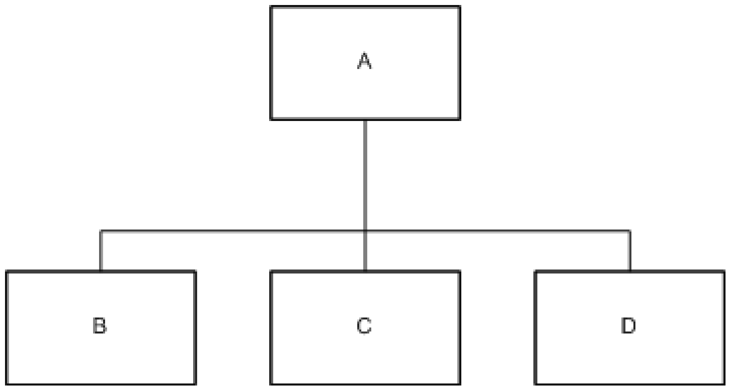
JSP uses a diagramming notation to describe the structure of inputs, outputs and programs, with diagram elements for each of the fundamental component types.

- A simple operation is drawn as a box.

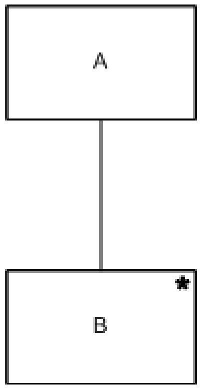
  An operation

- A sequence of operations is represented by boxes connected with lines. In the example below, operation A consists of the sequence of operations B, C and D.

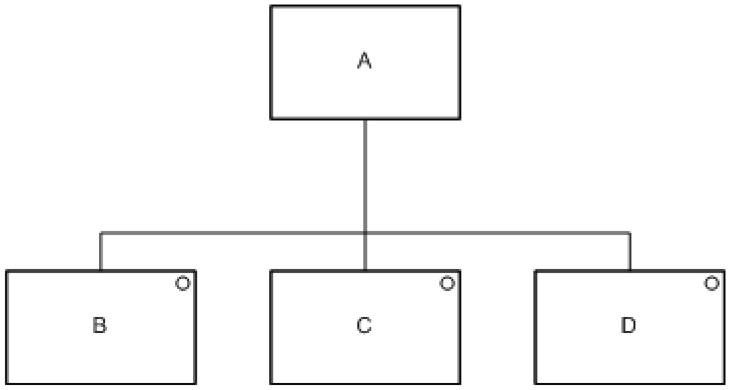
  A sequence

- An iteration is again represented with joined boxes. In addition, the iterated operation has a star in the top right corner of its box. In the example below, operation A consists of an iteration of zero or more invocations of operation B.



An iteration

- Selection is similar to a sequence, but with a circle drawn in the top right-hand corner of each optional operation. In the example, operation A consists of one and only one of operations B, C or D.



A selection

Basic JSP Design Method

JSP consists of the following steps:

1. Draw system diagram. (This step may be omitted when obvious.)
2. Draw data structures for program input(s) and output(s).
3. Form program structure based on the data structures from the previous step.
4. List and allocate operations to the program structure.
5. Create the elaborated program structure with operations and conditions added to the basic program structure
6. Translate the structure diagram into structure text or program code.

The result of applying JSP is a program that reflects problem structure as expressed in a model of its inputs and outputs. If changes to the program are required that only affect local components, the changes can be easily made to corresponding program components. A program's structural integrity – its correspondence with the problem's structure – is the primary way that we can reduce errors and costs in software maintenance.

Jackson System Development (JSD)

JSD is a method of system development that covers the software life cycle either directly or, by providing a framework into which more specialized techniques can fit. Jackson System Development can start from the stage in a project when there is only a general statement of requirements. However, many projects that have used Jackson System Development actually started slightly later in the life cycle, doing the first steps largely from existing documents rather than directly with the users. The later steps of JSD produce the code of the final system. Jackson's first method, Jackson Structured Programming (JSP), is used to produce the final code. The output of the earlier steps of JSD are a set of program design problems, the design of which is the subject matter of JSP. Maintenance is also addressed by reworking whichever of the earlier steps are appropriate.

From the technical point of view there are three major stages in Jackson System Development, each divided into steps and sub-steps. From a manager's point of view there are a number of ways of organizing this technical work. In this overview we first describe the three major technical stages and then discuss JSD project planning, the variation between plans, and the reasons for choosing one rather than another.

JSD: The Modeling Stage

In the modeling stage the developers make a description of the aspects of the business or organization that the system will be concerned with. To make this a description they must analyze their business, choosing what is relevant and ignoring what is not. They have to consider the organization as it will be, not as it is now.

The model description is written very precisely. This precision forces the developer to ask detailed questions. It encourages good communication and understanding between developers, users, and everyone else involved with the new system.

The model description consists of actions, entities and related information.
An action is an event, usually in the external reality, that is relevant to the system and whose occurrence the system must record.
In implementation terms, actions might cause database updates.
We start Jackson System Development by making a list of actions with definitions and associated attributes. Diagrams describe ordering relationships between actions. The diagrams describe the entities, people or, things that the system is concerned with.
The data that is to be stored for each entity is then defined. In effect we are choosing what is to be remembered by each entity about the actions that affect it. The full definition of this data includes an elaboration of the entity diagram to show in detail the update rules.
The result of the modeling stage is a set of tables, definitions and diagrams that describe:

- *in user terms exactly what happens in the organization and what has to be recorded about what happens, and*
- *in implementation terms, the contents of the database, the integrity constraints and the update rules.*

JSD: The Network Stage
In the network stage we build up a precise description of what the system has to do, including the outputs that are to be produced and the way the system is to appear to the user. This description is in terms of a network of programs. More precisely, it is a network of Communicating Sequential Processes (CSP), a concept developed by Tony Haoare. We start this network by making one program for each of the entities that was defined during the modeling stage. The network is then built up incrementally by adding new programs and connecting them up to the existing network. New programs are added for the following reasons:

- To collect inputs for actions, check them for errors, and pass them to the entity programs. In this way entity programs are kept up-to-date with what's happening outside;
- To generate inputs for actions that do not correspond to external events. Such actions are substitutes for real world events, perhaps because those events cannot be detected;
- To calculate and produce outputs.

There are two means of connecting programs in the network. These are by data streams (represented on our network diagram of circles) and by state vector inspection (represented on our network diagrams by diamonds). Whatever kind of connection is appropriate, the entity programs play a pivotal role in the construction of the network. Most of the new programs can be connected directly to the entity programs.

We draw a whole set of network diagrams to describe the system. Different networks usually only have entity programs in common. The complete system is represented by the overlay of all the diagrams.

The diagrams are supported by textual information describing the contents of the data streams and state vector connections. The new programs that are added to the network are defined using the same diagrammatic notation used to describe the ordering of actions. These new programs are designed using the JSP (Jackson Structured Programming) method, which is now a subset of JSD.

JSD: The Implementation Stage

The result of the implementation stage is the final system. *This stage is the only one directly concerned with the machine and the associated software on which the system is to run. Therefore, as well as producing and testing code, the implementation stage covers physical design issues.* In particular it covers:

- physical data design, and
- reconfiguring the network by combining programs.

Physical data design is about the design of files or databases. The details of database design depend on the particular DBMS being used. However, the necessary information about the application is all available from the network stage. The most important is the data defined for each entity and the high volume accessing of that data as defined by the frequently used state vector connections.

The result of the network stage is a highly distributed network of programs. Often, for convenience or efficiency, we convert programs into subroutines, in effect combining several programs into one, so that a fragment of the network is implemented as a single program. The network is reconfigured from a form appropriate for specification into a form appropriate for implementation.

JSD: Projects and Plans

We have presented the three stages of JSD as a simple linear progression. On a project, however, the stages overlap to a greater or lesser degree, and not just because people make mistakes that have to be corrected later. The stages and sub stages are nevertheless important because they classify and organize the technical work, they clarify the choices open to a project manager, and illuminate the risks when a decision has to be taken out of order.

The following are some examples of the overlap of the stages:

- We can start adding programs to the network before the model is complete.
- The detail designed of many of the simpler programs in the network can be done at the same time they are implemented.
- The physical data designed can be started before the low frequency programs have been added to the network.
- We may do a little each of model, network and implementation as the basis of a feasibility study.
- On a large project the model-network-implementation of one release may overlap with that of the next.

None of these overlapping is compulsory. A set of circumstances exists that makes each sensible. A project plan is made based on the technical framework of JSD and on the political and organizational circumstances of the project.