The CORE methodology

J S Briggs

Department of Computer Science
University of York

1. Introduction

1.1. What is CORE?

CORE stands for COntrolled Requirements Expression, a methodology for the elicitation of requirements.

It is a method for producing a requirements specification that employs diagrammatic notation supported by English prose.

1.2. The stages of the CORE methodology

The methodology is carried out in 7 main stages, of which the first 5 are the most important and the final 2 are sometimes omitted. The stages (and the abbreviations I shall used for them) are:

- Problem definition (PD)
- Viewpoint structuring (VS)
- Tabular collection (TC)
- Data structuring (DS)
- Single viewpoint modelling (SVM)
- Combined viewpoint modelling (CVM)
- Constraints analysis (CA)

The stages are normally carried out in this order, in particular the first two must be carried out before any of the others can be attempted. Once we have done VS, we may choose to do TC, DS and SVM on each viewpoint in turn rather than doing all the TCs, then all the DSs, then all the SVMs. CVM and CA require information from all the earlier stages, so come last.

1.3. Some jargon

To avoid confusion, before we start let's define the following terms:

Target System	the system we're trying to build; not to be confused with the host system (the one we are doing the developing on) or any system that we are using in the development.
Analyst	the person (or group of people) who are given the task of establishing the requirements specification.
Users	the people who will be using the target system once it is completed

	and installed – often the main source of requirements concerning the functions of the system.
Customer	the person who is paying for the target system, and is responsible for ensuring that the system is delivered, satisfies the users, etc.

1.4. Some overall comments

Requirements analysis is inherently an iterative process. As ideas and objectives are refined, the proposed features of the new system also change. These changes must be recorded in a revised requirements document to ensure that it always reflects the current goals and objectives of the system.

Some form of version control for successive editions of the requirements documents is recommended.

2. Problem definition

In the early stages of the project, there is normally no clear statement of its objectives. This is not to be unexpected, since there will not be a clear idea at this point what exactly the system should do. One of the purposes of the requirements analysis phase of a project is to refine the objectives, and the PD stage of CORE is intended to elicit and document these.

When a problem area has been identified, typically several different solutions to it will be proposed. These may be discussed at different levels of the organisation but quite often what happens is that the person or group requested to produce a specification for the new system will not be aware of the original business objectives which motivated the desire for it. The analyst will get a slightly different opinion from each person they talk to as to what features the new system should have. Some of these ideas will be motivated (and quite unintentionally) by the individual user's immediate needs, and may conflict with the overall goals of the organisation.

The aim of the PD stage is to overcome that by defining the overall objectives, and by identifying the person or persons with the final say in those. The PD statement should be very concise – not more than 3 pages – and have six sections corresponding to the following steps in its determination.

2.1. Identify the customer authority

Decisions about the features, cost and timescales of a new system usually involve people at different levels within the organisation. However, only one person (or at most a small group) can make the ultimate decisions that have major implications on cost or timescale. This person is, by definition, the Customer Authority. The final decision rests with them. This person must assist with and approve the PD document as well as any major changes to system features which arise later.

The best way of ensuring that the Customer Authority carefully reviews and approves a document is to request their signature on the front of it. Once signed this is visible proof of approval for its contents. This is a useful step in getting them involved in the analysis and specification process.

2.2. Define the business objectives

This section should begin with a paragraph on the historical background that provided the motivation for a new system. It should identify the organisation providing the funding and list the key business objectives of the organisation. What game is the organisation in, and how does it play the game?

Sources of information for this step include discussions with senior management and from things such as corporate plans and strategy documents.

2.3. Identify the problems with the current system

The new system is probably a replacement for an existing system (the existing system might however be manual rather than automated). There must be something wrong with the current system, else why replace it? By looking at current system documentation, existing organisational procedures and by talking to users it should be possible to come up with a list of the current system's inadequacies.

2.4. Define the principal features of the new system

What are the goals and objectives to be satisfied by the new system? These might be the reverse of the problems with the existing system but often there will be totally new requirements that are wanted. This section should begin by discussing how the new system will further the previously stated business objectives, and go on to discuss in general terms the functional objectives of the system and finally to list its principal features. Where features are optional (e.g. desirable but not to be implemented if too expensive) then this should be stated. Again the analyst can determine the contents of this section by talking to system users and management.

2.5. Discuss possible future expansions

Often the desire for a new system is the first step in the development of further new systems with which it will interact. Where future expansion is envisaged (to whatever vague an extent) it should be stated explicitly here. The designers can then take it into account and possibly save the customer much grief in the future.

2.6. Establish costs and timescales

There will almost certainly be cost and timescale restrictions placed on the design and implementation of the new system. These should be stated here. Their relationship to the business objectives may also be stated (e.g. "the company will go bankrupt if the system is not ready in 18 months", or "if the cost is more than double the estimate, the system will not be worth having"). This section should also state the expected development cost and schedule.

Guidelines

Before completing this stage, the analyst should have found the answers to the following questions:

a) Which person in the organisation has the authority to approve major changes in the cost or functionality of the system?

b) What are the primary business objectives of the organisation?

c) What are the major problems of the current system?

d) Why is a new system needed?

e) What critical business functions depend on the system?

f) How will the new system help to achieve the primary business objectives of the organisation?

g) What are the probable future expansions to the new system?

h) Which units of the organisation will be affected by the new system?

i) Which units of the organisation are likely to affected by any extensions being considered?

j) What cost range does the organisation realistically expect to pay for the new system?

k) What features could be eliminated or delayed to reduce costs?

l) How long can the system installation be delayed before it seriously affects the business objectives for the system?

3. Viewpoint structuring

A fundamental part of any analysis is to define the boundaries of the problem. This allows the analyst to establish which aspects of the environment must be considered during the analysis and which can be ignored. The new system and all other systems that interface with it will fall within the problem boundary.

CORE tends to lead the analyst to consider more aspects of the environment than they might otherwise do. This is because we extend the definition of the word "system" to include any entity that processes information. By this definition, human beings, organisations and even phases of the software lifecycle can be considered as systems. To avoid confusion, we call the new system the "target system" since it is target of the analysis.

In CORE terminology, all these systems are called *viewpoints*. The purpose of VS is to provide a framework for considering the requirements of the target system from the viewpoint of the other systems that are sources or destinations of information to and from it.

Creating the viewpoint structure is nothing more than partitioning the environment of the target system into subparts: an application of the "divide and conquer" principle. The secret of deriving a good or useful viewpoint structure is to choose a method of subdivision that results in a structure that assists the subsequent analysis of the target system.

Since the viewpoint structure is merely a representation of the subdivisions of the system, it can be used for any type of system or method of division. There are only three rules:

1. Each viewpoint must perform some processing of information (i.e. none should do nothing).
2. Viewpoints must not overlap. If a function is performed by one viewpoint then it cannot be performed by any other viewpoint.
3. The source or destination of all information used within the system must be one of the viewpoints.

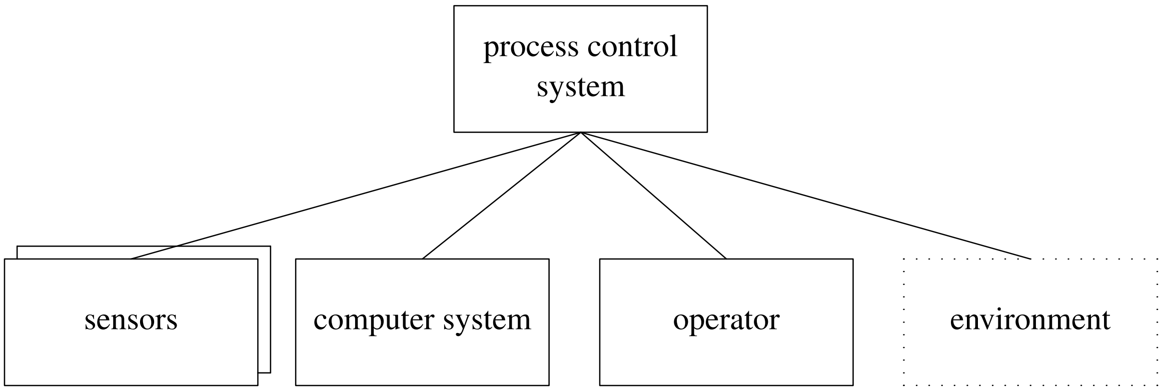
The most common way of analysing any target system is to identify "real-world" entities that are either sources or destinations of information used by that target system. These entities might be human beings, sensing devices, organisations or other computer systems. Each of these entities is potentially a viewpoint.

You must be careful only to choose as viewpoints entities that process information. A computer terminal is not a good choice since it changes only the format of information, not its content. For the same reason, a database is not a good viewpoint because it merely holds information for later retrieval, whereas a database management system would be a valid viewpoint since significant processing may be necessary to perform a query or update request.

In CORE there are two types of viewpoint: *direct* and *indirect*. We are interested in indirect viewpoints solely as sources and destinations of information. How they do this is irrelevant to our analysis. For direct viewpoints, however, we also wish to analyse how the processing occurs within the viewpoint. You will see a third type of viewpoint called a *parallel* viewpoint. Sometimes a target system sends or receives information from many systems that are similar in function. If this is the case, the similar systems can be treated as a group of parallel viewpoints and analysed only once. Conventionally, only direct viewpoints are parallel since we would not want to analyse the processing performed by an indirect viewpoint.

A typical viewpoint structure (of a process control system) is shown below:

Viewpoint structure diagram for *process control system*



In this example, *computer system* and *operator* are direct viewpoints, *sensors* is a parallel viewpoint, and *environment* is an indirect viewpoint.

Once the viewpoints have been established, a textual description of each should be provided. This should enhance the short viewpoint name in a manner that defies misinterpretation.

The final stage is to identify a *Viewpoint Authority* for each viewpoint. This is the person responsible for providing relevant information on a particular viewpoint. The Customer Authority will probably want to designate the Viewpoint Authorities.

Once the VS has been established it is easier to see the extent of the analysis task, since it defines what must be analysed (i.e. the direct viewpoints). A work plan could be produced at this point.

So how exactly do we go about creating a viewpoint structure?

3.1. Identify possible viewpoints of interest

List all the real-world entities that might exchange information with the target system. The list should include any users who operate or view information produced by the target system. In some cases it may be pertinent to include entities that exchange information indirectly with the target system (e.g. via another entity). This information can be sought from users of the existing system and its documentation.

3.2. Draw a Venn diagram

Draw a large circle on a blank sheet of paper. This represents the problem boundary. All entries drawn within the circle are viewpoints that must be analysed as part of the system. Now draw a smaller circle in the centre of the larger circle and label it with the name of the target system. Going through the list of viewpoints generated by the previous step, draw additional small circles containing those entities that have a major information exchange with the target system. Next add circles for those entities that have no direct exchange with the target system but which have significant exchange with other entities within the system boundary. All the remaining entities on the list are probably outside the problem boundary and, if so, need not be considered further.

In the second or subsequent iterations of this step, entries may be designated by the Customer Authority as not being subject to analysis. These entries are moved out of the circle to indicate that they have been considered, but need not be considered further.

As before, users of the existing system and its documents will be the primary sources of information in carrying out this process.

3.3. Refine the Venn diagram

This process helps to clarify the problem. The aim is to refine the contents of the diagram, using it as a tool to improve communications with the customer. At this stage you should also decide whether each remaining viewpoint is direct or indirect, and (if direct) whether it is parallel. The purpose of classifying the viewpoints as direct, indirect or parallel is to further understand their role and relation to the analysis of the target system.

One indirect viewpoint that should always be included is that representing the environment of the target system. This may be considered as being the area of the Venn diagram that is not within any of the inner circles. The "environment" viewpoint can then be used as the source or destination of all information that is not generated or received by any of the other viewpoints being considered. This viewpoint, therefore, enables the problem to be bounded and to meet the requirement of CORE that all data have a source and destination.

3.4. Combine viewpoints, if necessary

Count the viewpoints: if there are seven or fewer, the system can be analysed at one level only, so no combination (and subsequent decomposition) of viewpoints is necessary. However, if there are more than seven viewpoints, any two or more related viewpoints should be combined into a more general viewpoint with an appropriate title. (Draw a circle enclosing the related viewpoints.) If necessary, combined viewpoints can be further subdivided and analysed separately at the next level of refinement.

3.5. Draw the viewpoint structure

This step is simple and mechanical. The Venn diagram and the viewpoint structure diagram are two representations of the same information. The root of the VS tree is the outermost circle (if it lacks a name call it something general like “problem viewpoint structure”) and each other node corresponds to an inner circle. Circles within circles are of course subtrees of their parent node.

3.6. Describe the viewpoints

Just as important as drawing the diagrams is to support them with explanatory text. The names of the viewpoints are just shorthand labels for a collection of concepts. This is OK for those involved in formulating the structure, but the VS diagram is going to be read by many other people and these names may be ambiguous or misleading. It is important therefore to define precisely what is and what is not included in the definition of the viewpoint names.

3.7. Designate the viewpoint authorities

In the later stages of CORE, the more detailed analysis will be conducted on a viewpoint by viewpoint basis. For each viewpoint, the Customer Authority must decide who is the person with the authority to answer questions relating to it. This should be the relevant “expert” on each viewpoint. The same person could conceivably be the authority for more than one viewpoint (or indeed be the Customer Authority) but each viewpoint should have only one authority. This has the side-effect of excluding from the analysis of a particular viewpoint others who are eager to discuss requirements in areas where they have no responsibility.

3.8. Produce a work plan (optional)

In larger projects (e.g. of more than 3 man-months of analysis effort), this is a good point to decide on the schedule for the detailed analysis. Are the viewpoints to be analysed in series or in parallel? Is the analysis task going to be broken down among several people, and if so who is going to be responsible for which viewpoints? How long (estimate) will the analysis of each viewpoint take?

Guidelines

Viewpoint names should be 1 to 5 words. They should identify operational components that perform some processing or transformation of data.

Analysis of the viewpoint structure should proceed top down.

Each direct viewpoint should be decomposed into no more than seven sub-viewpoints. Use more levels if necessary.

A direct viewpoint should be decomposed only after the scope of the viewpoint has been defined, and all other viewpoints at that level have been defined.

When a viewpoint is decomposed, the collected functions of its sub-viewpoints must be equivalent to those of the parent viewpoint.

There should always be an indirect viewpoint called “environment” or some equivalent name.

Before completing this stage, the analyst should have answered the following questions:

a) Who are the major users of the target system?

b) With what other systems does the target system interact?

c) Is there a group of related systems that provide the same or similar functions (i.e. a parallel viewpoint)?

d) If there are more than six direct viewpoints, which viewpoints are most tightly connected to the target system and could therefore form a more general viewpoint?

e) If a viewpoint name could be interpreted in more than one way, does the relevant viewpoint description precisely define what is and what is not included in the viewpoint?

f) Which person will provide information and resolve queries about a specific viewpoint?

4. Tabular collection

The TC stage provides the main mechanism for gathering and analysing information on the flow of data and its transformation. It is possible to check the completeness of a viewpoint and also consistency between viewpoints.

By this stage of the CORE methodology, we have broken the system down into a set of viewpoints which have been named, described and approved by the Customer Authority. The aim of this stage is to gather information about each (direct or parallel) viewpoint. The analyst must identify the data items used or produced by each viewpoint and where (which viewpoint) that item comes from or goes to. Each viewpoint will have a number of actions that use the input data and produce the output data.

The main source of information about what a viewpoint does will be the Viewpoint Authority. They may be able to provide documentation or descriptions of the existing system from which much of the necessary information can be extracted. Ambiguities and conflicts can then be resolved by the VA.

The main basis for this stage is the Tabular Collection Form (TCF) which will be constructed for each viewpoint. The TCF will be the centre of the consultation process with the VA; successive versions of the TCF may be produced before a final version that embodies all the actions and data items of the viewpoint is reached.

The objectives of this stage are four-fold:

(i) to summarise the functions of the viewpoint in 6-12 actions;

(ii) to summarise the data items into 10-30 input data items and 10-30 output data items;

(iii) to identify the source viewpoint of every input data item used by an action of this viewpoint;

(iv) to identify the destination viewpoint of every output data item derived by an action of this viewpoint; and

(v) to ensure that the data items and actions are completely connected.

The numbers given above (6-12 and 10-30) are intended as guidelines to reflect the optimum size of a TCF. Of course they are not rigid rules and may be exceeded where justified. However, a TCF with fewer than six actions may be at too low a level of decomposition, while one with more than 12 should suggest aggregation or exceptionally a return to the VS stage and the possible decomposition of the viewpoint into two or more sub-viewpoints. Too many actions and data items in a TCF also make it difficult to draw.

The basic structure of a TCF is as follows:

Source	Input	Action	Output	Destination

The process of this stage is to fill in the columns (normally working from the middle outwards) and then to join entries in neighbouring columns together. This approach has the following advantages:

a) it is possible to ensure that all information in the documentation is extracted and included in the TCF;

b) incomplete information is easily spotted due to the graphical representation;

c) interviews with Viewpoint Authorities can be directed towards filling in gaps and resolving inconsistencies;

d) since input and output data are connected to other viewpoints by the source and destination columns, the analyst can easily establish consistency between viewpoints.

The procedure goes as follows:

4.1. Select a viewpoint

The process must be repeated for each viewpoint in turn. It is usually a good idea not to start with the target system, but instead to begin with another viewpoint that is closely connected to it (i.e. by information flows).

It is worth emphasising here the importance of strict version control. Different versions of the same TCF will look very similar so it is important to annotate them with such information as

(i) name of the viewpoint;

(ii) author (name of the analyst compiling the form);

(iii) date;

(iv) version number.

4.2. Extract information from available documentation

A simple and effective way of extracting relevant information from documentation is to go through it highlighting or underlining text. If the document contains information on more than one viewpoint, different colours can be used for each viewpoint. Whenever some text is encountered that is relevant, highlight it and copy it (verbatim) to the TCF.

If the text refers to an action or function (usually verb clauses in the text) it should be placed in the action column. If the text refers to data (noun clauses) it should be placed in either the input column or the output column (if unsure as to which, mark this for future reference to the Viewpoint Authority). It does not matter at this stage how messy the TCF looks: the aim is to be sure that every possible piece of relevant information has been extracted and placed in the form.

It is important that this step be carried out in a fairly "dumb" fashion. Any modification of the information involves an assumption by the analyst which may be at odds with the Viewpoint Authority. Resolution of any oddities can be done later.

If no documents are available, it is best to start with the outputs of the viewpoint since these are usually easier to identify.

4.3. Connect the column entries

The inputs used by each action and the outputs produced by each action should be specified by drawing a line from the name of the action to the name of the data item. At this stage don't worry about crossed-lines or the density of ink: later re-arrangement should reduce that.

For each input data item, it should be possible to identify the viewpoint (or exceptionally viewpoints) that is its source. The name of each such viewpoint should be entered into the source column and a line drawn from it to the name of the input. Similarly for outputs, the destination viewpoint(s) should be identified, entered into the rightmost column and linking lines drawn.

4.4. Refine the TCF

Once all the obvious connections have been made, the analyst will inevitably have encountered some problems. Now is the time to go back to the Viewpoint Authority and to try and resolve these. If the analyst tries to resolve them themself then they risk making invalid assumptions. It will often be the case that the problem is caused by a missing action or data item – probably because mention of it was absent from the original documentation – these should be added. It may also be the case that an action or data item is redundant and can be deleted.

The most commonly encountered problems (and suggested means of solving them) are:

(a) Data items not connected to any action – if the data item is genuinely part of the viewpoint, a name must be agreed for the action that uses or derives this data.

(b) Data items that are internal to the viewpoint (i.e. this viewpoint is both the source and destination of the data item) – this information may be premature at this point; consider deleting the data item.

(c) Actions that have no inputs and/or no outputs – either the data items have not been recorded, in which case the Viewpoint Authority must be consulted, or the action is really part of a more general action that does have inputs and outputs.

(d) Sources and destinations that are not existing viewpoints – either the definition of an existing viewpoint must be expanded or alternatively a new viewpoint must be defined and the VS modified accordingly.

4.5. Produce the next version of the TCF

A TCF with many crossed lines can be cleaned up by re-ordering the columns. Actions that use the same data items could be moved next to each other; similarly data items to/from the same source/destination viewpoint could be adjoined. An alternative would be to associate some meaning to the order of actions: for example to list them in the sequence of their occurrence.

Great care should be taken to ensure that no information is lost in this process. Every item on the old TCF should be ticked off as it is transcribed to the new version. Version control information should be updated appropriately. Old versions of the TCF should be retained in case, for example, a deletion decision is reconsidered.

A TCF is complete when it has been reviewed and approved by the Viewpoint Authority.

4.6. Verify data flows between viewpoints

Once the final version of a TCF has been produced, it is important to ensure that it is consistent with the TCFs for other viewpoints. What this means is that the outputs of each viewpoint should correspond with the inputs of the destination viewpoint. Conversely the inputs of each viewpoint should correspond with the outputs of the source viewpoint. This validation process will highlight any naming inconsistencies that have to be resolved, and inputs and outputs that have been identified in one viewpoint but overlooked in another.

Naming inconsistencies are usually the result of different viewpoints calling the same thing by different names. The relevant Viewpoint Authorities should be persuaded to agree to a single name. As a last resort, the Customer Authority may be needed to arbitrate.

4.7. Describe data items and actions

This step involves the creation of a data dictionary, since by this step the names of all data items and actions should have been fixed. A suggested format is

For data items:

Name of the data item
List of attributes that apply to it
Volumetric information (size, frequency, transmission media, etc.)
List of viewpoints in which the item occurs

For actions:

Name of the action
List of attributes that apply to it
Performance information (frequency of execution, expected or desired speed, etc.)
Narrative description of what it does
List of viewpoints in which the action occurs

Guidelines

Data flow analysis is undertaken for each viewpoint separately. Only direct and parallel viewpoints are analysed.

In practice, a Viewpoint Authority or document may provide at the same time details that belong to different viewpoints. These should be recorded in the relevant TCFs.

If during TCF analysis, connections for certain data items or actions cannot be found, don't remove the information from the TCF. Later analysis may resolve it. However if it becomes clear that an entry is misplaced (e.g. it belongs in another TCF) then it should be removed.

Before completing this stage, the following questions should be answered:

a) What functions are performed by the viewpoint?

b) What information is produced by the viewpoint, and for whom?

c) Which viewpoints contain the events that trigger off the actions in the viewpoint under consideration?

d) Is it possible for an action to have any inputs in addition to those currently shown on the TCF?

e) Can an action produce outputs additional to those shown on the current TCF?

The following rules should not be broken without justification:

1. The viewpoint under analysis should not be the source or destination of any of its own inputs and outputs.
2. Every action should have at least one input and one output. An action should ideally have two or three inputs and as many outputs.
3. A TCF should contain between 6 and 12 actions. If there are too many or too few this can be resolved by aggregating related actions into composite actions, or by decomposing actions into two or three more detailed actions. Exceptionally the viewpoint structure may be reconsidered.

The final version of the TCF for each viewpoint is then used as the starting point for the next stages of the analysis.

5. Data structuring

Having defined the major functions and data flows of the target system and its environment, this stage allows us to analyse the data further.

Each data item is analysed in the context of the viewpoint that generates it – all the outputs of a viewpoint are therefore analysed as a group. Since TC ensures that any input is an output from another viewpoint, it is usually not necessary to analyse inputs separately. (An exception might be inputs from an indirect viewpoint.)

Again analysis proceeds viewpoint by viewpoint. The viewpoints can be considered in any convenient order.

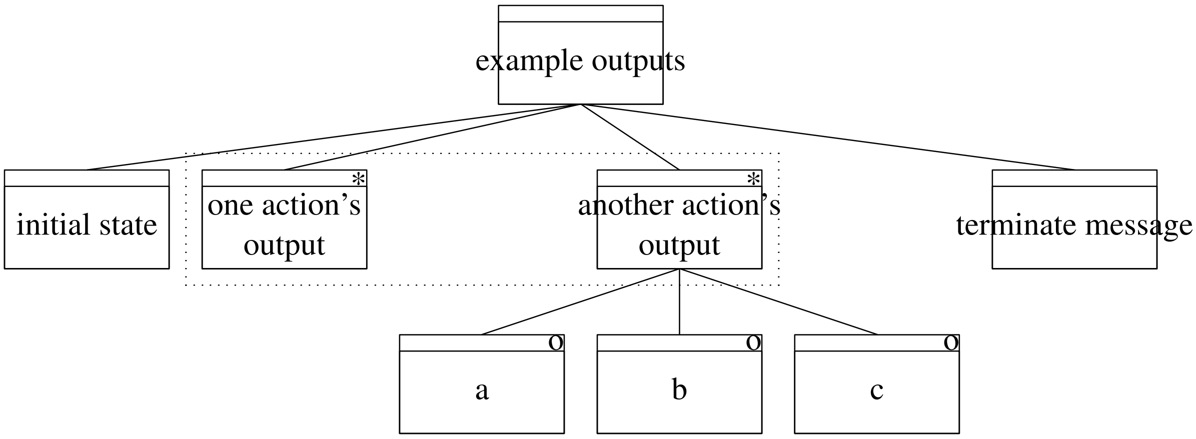
The first step is to determine the order of derivation in the group of output data items. This will determine the order in which the actions are processed. Normally the first output will be the result of some initialisation activity, then there is often a group of items that can be processed in any order, and finally the results of some termination activity.

When the order has been established, each of the data items must be annotated to show whether it occurs only once, is iterated, or occurs only under specific conditions. The diagrammatic conventions for this are similar to Jackson data diagrams.

The final step is to decompose each data item into its parts. If the target system is large, the data item names will usually symbolise large composite data structures. It is not the aim to decompose these structures completely since this might entail going into too much detail. However, we do want to show the general contents of each data item. Typically this will result in each data item being decomposed into three to six subparts. Sometimes it is appropriate to perform further decomposition on these subparts but that it is usually left to the design phase.

The product of this stage is (for each viewpoint) a diagram of this form:

Data structure diagram for the *example outputs* viewpoint



The process of Data Structuring is as follows:

5.1. Select a viewpoint

The root of the DS diagram for a particular viewpoint is a node labelled ‘‘<viewpoint name> outputs’’. The names of the nodes at the next level correspond with those of the outputs of the viewpoint.

5.2. Determine order of derivation

The outputs of the viewpoint will be arranged left to right in the order in which they are derived. If several items can be derived in any order, a dotted line is drawn around the group. If two or more data items are mutually exclusive (i.e. if one data item is produced then the others will not be), one box is drawn at this level to represent the group and then at the next level the box is decomposed into the mutually exclusive items.

5.3. Annotate data item boxes

Each data item box must now be annotated to indicate whether it occurs once, more than once, or may be selected from a group of mutually exclusive items. If a data item occurs once, there is no special annotation. If a data item may occur zero or more times, the box is annotated with an asterisk. If an item is one of a mutually exclusive group, all members of the group are annotated with a circle.

5.4. Decompose data items

Each data item may now be decomposed into its subparts. This may be somewhat arbitrary and it is left to the analyst to determine the most informative breakdown and to what level of decomposition to go. Remember that 3-6 subparts is the recommended number for each decomposition.

If the viewpoint structure diagram has more than one level of decomposition, we would expect to see these data item subparts appearing as data items that are passed between viewpoints at the next level of refinement.

The order of subparts is usually somewhat arbitrary, so it makes sense to arrange them in the order in which one might expect to see them in a report or display.

5.5. Annotate decomposed data items

As for step 3.

Guidelines

Check that every output of a viewpoint can be found on the data structure diagram.

Every data item output by the viewpoint should be at the first level of decomposition unless it mutually excludes another data item, in which case it will appear at the next level.

Normally, all data items that belong to an indeterminate-order group will also be iterated.

If a data item has a mutual exclusive annotation, all other data items with the same parent must also be annotated as mutually exclusive.

It is recommended that each data item of the viewpoint should be decomposed into three to six subparts.

The subparts of data items should also be annotated if they are iterated or mutually exclusive.

6. Single viewpoint modelling

Once we get to this stage of the CORE methodology, we have gathered most of the information about the target system. This and the next stage simply serve to present that information in alternative ways, but in so doing to reveal more detail about the functionality of the system.

SVM diagrams are simply an alternative representation of the information contained in TCFs. The actions, inputs and outputs of the TCF all appear in the SVM diagram. However, extra details of internal data flows, the control of actions and the time ordering of actions are added at this stage.

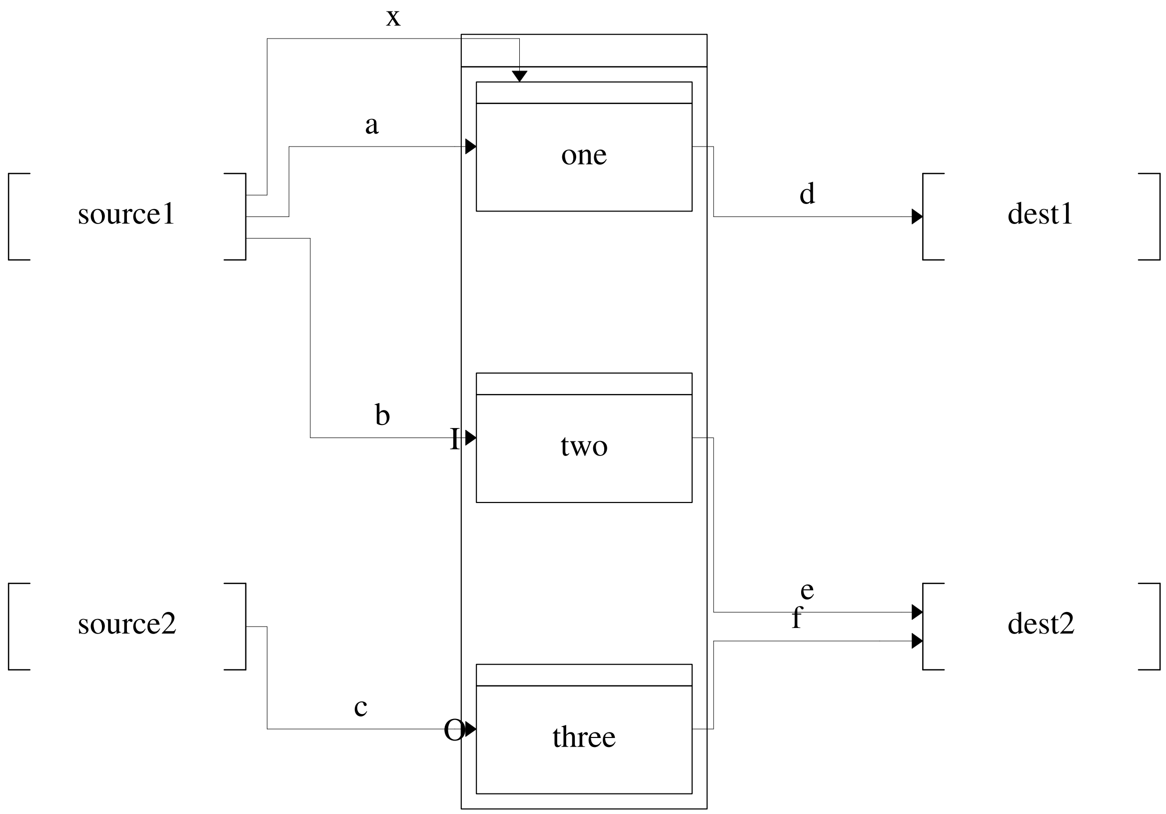
Clarity is important, so it is essential for the analyst to be prepared to spend time rearranging the boxes and lines of the SVM to minimise line crossing and clutter.

With two exceptions, new actions and data items should not be added at this stage. The exceptions are firstly that if it is realised that a data item or action was not detected during the TCF stage, it should be added, and consistency and completeness checks remade. The second (and more normal) exception is when a need for a data flow internal to the viewpoint is identified. This is not unusual since there is frequently a need for information transfer between two actions of a viewpoint. The name of the new data item should be added to the data dictionary.

An important aspect of SVM diagrams is that they provide a visual sense of time flow. This is accomplished by positioning the action boxes in chronological order from left to right (or from top to bottom). However, if the clarity of the diagram would suffer badly, this rule may be relaxed and other means of indicating time (e.g. in the description of the viewpoint) used.

An example of an SVM is:

Single viewpoint model diagram for the *example* viewpoint



As before, the procedure for producing an SVM diagram is applied to each viewpoint in turn, in any convenient order. Make sure that you have the most recent version of the viewpoint's TCF before starting.

6.1. Position the first action

Looking at the actions in the TCF, select the one that occurs first. If in doubt, look for data that is derived first. Draw an action box in the upper-left of the page and enter the action name as it appears in the TCF.

6.2. Connect the external data flows

Add the source and destination viewpoint names: sources go on the left of the page, destinations on the right.

Connect all the external data flows between the action box and the source and destination boxes. Input flows into the left of a box; output flows out of the right-hand side.

6.3. Position the next action

Select the next action that occurs. If the action clearly occurs after the first action, its box should be placed to the right of the first box, otherwise it should be placed below the first box. Add this action's source and destination data flows.

6.4. Add necessary internal data flows

Consider the actions that you have positioned. Is there any need for a transfer of information between them? If so, add this data flow and give it a name.

Internal data flows such as this should only be added if they are major flows in the information processing model of the viewpoint. There may be internal data flows necessary for the implementation of the viewpoint, but these should not be drawn.

Repeat steps 3 and 4 for each action in the TCF.

6.5. Add control information

In this step we add information about the operation or execution of the actions relative to each other.

In a model, some actions only occur once. Others occur several times, either because they must occur at regular intervals of time or because they are triggered by some external event. If the primary reason for the occurrence of an action is the arrival of some input data then no annotation is necessary. If an action occurs at regular time intervals, we represent this with a control flow coming from the environment to represent the fact that a time interval has passed. Control flow lines are directed to the top (left-hand corner) of the action box. Any action that is executed repeatedly is indicated by an asterisk in the top corner of its box.

If the action occurs because of some external event, you must ask yourself whether the event also passes information at the same time, in which case the action occurs because input data has arrived. If the event merely causes the action to occur it should be added as a control flow. The source of the control flow must be either another viewpoint or another action of this viewpoint.

‘‘Selection’’ is another type of control flow between actions. Sometimes only one of a group of actions will occur depending on some external data or control flow. We indicate this by putting a circle in the top corner of each box in the group, then we add a control flow that goes to each of these actions. In essence, this is a kind of ‘‘case-statement’’ construct. The condition on which the selection depends is added in text at the top of each box.

6.6. Add data and control flow information

It is often useful to indicate the type of the data flow or control flow. In this context there are two types: channels and pools. A channel flow means that every data item generated must eventually be processed (in computer systems this might imply a queue or buffer). whereas a pool flow means only that the latest value of the data is of interest (in computer systems, generally implying that the previous value of the data is overwritten). If an action occurs every time a data item is produced, then this is a reliable indication that that flow should be a channel. A channel is indicated by ‘‘I’’ on the flow line, while a pool is indicated by ‘‘O’’. It is normal only to annotate input flows: the type of an output flow depends on where the output is used. If the data or control flows are very abstract it is usually best to leave their type unspecified.

6.7. Describe the viewpoint model

We now have a complete graphical model of the system. Such models are very useful because they form visual conceptual models and make the task of understanding easier for the reader. The information processing of the viewpoint can be seen at a glance. This is partly because the model is two-dimensional, and partly because the information is limited to the primary data flows and operations thus avoiding the problem of not being able to see the wood for the trees.

The disadvantage of the graphical model is that it is still subject to some interpretation, precisely because it does not contain detail. This dichotomy is resolved by supplementing the diagram with explanatory text.

The text should *complement* the diagram, filling in the gaps where too much may be subject to interpretation. The text also allows the opportunity to explain the reasons for particular operations and data flows. The diagram and supporting text should be kept together at all times.

Guidelines

Before completing this stage the analyst should have answered the following questions:

a) Is there one (and only one) action box for every action of the corresponding TCF?

b) Is there a data flow for every input and output of the TCF?

c) Have all the essential information flows between actions (internal data flows) been represented?

d) Are all the source and destination viewpoints represented on the diagram?

e) Are some actions performed several times relative to the other actions, and if so are they marked with an asterisk and a controlling data flow?

f) Does every direct viewpoint have a corresponding SVM diagram?

g) Does every SVM diagram have a corresponding textual description that complements the diagram?

7. Combined viewpoint modelling

An SVM diagram is a graphical representation of the information processing of a single viewpoint. It conveys at a glance all the information received and generated by the viewpoint as well as the internal processing and some of the internal data. While this is useful for a single viewpoint, it is less useful for analysing a sequence of operations or a transaction that involves several viewpoints. For this reason, CVM diagrams may be produced. These are a representation of the information processing that occurs between viewpoints.

It is usually unnecessary to analyse all transactions between viewpoints. This stage is intended mainly to analyse those transactions that involve critical performance or reliability aspects of the proposed system.

The SVM diagrams provide all the information that will be used in the CVM diagrams. It is merely necessary to indicate against each action the name of the viewpoint that performs it. This is done by putting the viewpoint name beneath the action box, with a short arrow pointing up towards the bottom of the box. This is called the action's mechanism.

The steps of combined viewpoint modelling are as follows:

7.1. Select a transaction of interest

What is an interesting transaction? In a real-time system, it might begin with an external event coming from the environment and continue until it manifests itself as some new information presented to the user. In a data processing context, an interesting transaction might be an update to a customer account. Usually several primary transactions of the system and its environment can be readily identified. If in doubt, select a data flow that is initiated in the environment and follow it through the relevant viewpoints.

7.2. Position the first viewpoint identifier

Once a transaction has been selected, the first viewpoint involved must be identified. Place this on the left-hand side of the diagram, then draw the relevant data flow that it generates. This data will be used by an action of some other viewpoint. From the appropriate SVM diagram identify the action and draw it on the diagram. Add the action's mechanism arrow with the name of its viewpoint.

7.3. Position the next action(s)

The first action may produce several outputs going to several other actions. Typically not all these actions will be an essential part of the transaction we are concerned with. The analyst must decide which actions and flows are to be included, and which left out, of the CVM diagram. When an essential action is found, it should be drawn to the right of the first action and the data flow between them added. Eventually a chain of data flow, action, data flow will be produced from left to right across the diagram. All actions must have a mechanism arrow to identify their viewpoint. This process continues until all actions that are an essential part of the transaction are included.

7.4. Decide where to end the transaction

If you find difficulty in finding the end of a transaction, continue until a data flow goes to an indirect viewpoint. This is a suitable place to stop in the absence of a better reason.

Many transactions start with an event that occurs in the environment viewpoint. The transaction can then be continued until a data flow finally returns to the environment. Another way is to start and/or end the transaction with the user. In theory a transaction can start or end at any point.

7.5. Draw a combined action box

A combined action box is a border for the CVM diagram. The name of the transaction should be entered in the bottom left-hand corner of the box.

7.6. Write a text description

Text is required to supplement a CVM diagram for the same reasons it was for an SVM diagram. The text should complement the diagram rather than simply repeat what is already shown.

Guidelines

Before completing this stage, the analyst should have answered the following questions:

a) What are the primary events that affect the system?

b) What are the critical timing aspects of the system?

c) What are the critical reliability aspects of the system?

d) Have all important transactions been identified?

e) Do all the actions flow in a sequence from left to right?

f) Are all the data items and actions taken from SVM diagrams (i.e. no new information has been produced)?

g) Does the transaction name capture the essence of the operations involved in the transaction?

h) Does every transaction have a textual description that complements the diagram?

8. Constraints analysis

The preceding stages of the CORE methodology have resulted in the production of a logical information-processing model of the new system. At this stage we switch to consideration of the less well-understood aspects of the system, the non-functional requirements. These are things like reliability, security or fault-tolerance which are often expressed in intangible terms like ‘‘the system must be reliable’’. Undefined qualities such as these can never be tested and are worse than useless since they can provide grounds for extended arguments at system acceptance time.

The constraints analysis procedure addresses each class of constraint by relating it to the relevant viewpoint model diagram (single or combined). In any event, a written description of the results of the analysis with any relevant diagrams will be produced. The analyst must take great care to word any descriptions in a way that allows the required features to be tested.

The following is not a list of steps as such. Rather it is a non-exhaustive list of possible constraints. If other constraints are relevant to a particular application they should be analysed in a similar manner.

8.1. Specify performance requirements

An important aspect of performance is the speed at which the system responds to a user operation. Determine and specify the delays that are acceptable in response to each type of user input.

Determine the peak loading on the system and specify the throughput required during this period. A useful technique is to produce a CVM diagram that includes all the actions involved in the processing of the peak load. Then follow the relevant data flows step by step, calculating at each point the probable processing times.

8.2. Specify reliability requirements

The reliability of a system is determined by how often it fails and whether it can recover from a failure. Be cautious about specifying unnecessarily high reliability since this can be very expensive. On the other hand, a simple analysis may reveal that reliability can be substantially improved by the addition of such inexpensive features as an alternative transmission link or a ‘‘retransmit on error’’ capability. Again, a useful technique is to follow the critical transactions on a CVM diagram asking at each point ‘‘what could go wrong?’’.

8.3. Specify design constraints

Generally, more design constraints than strictly necessary are imposed. There is a tendency to give too much weight to statements such as ‘‘the programmers prefer to use assembly language’’, ‘‘computer X must be used because it was used on the last project’’, or ‘‘computer X must be used because we get a big discount if we buy one’’. The organisation should bear in mind the long term ramifications of such constraints.

Extendibility is an important design constraint. At this stage we need to specify how the design must be constrained so that the future expansions identified in the Problem Definition can be incorporated without major redesign.

8.4. Specify environmental constraints

Specify the physical characteristics of the location where the system will be installed: i.e. temperature, dust levels, space, power supplies, etc.

8.5. Specify other constraints

Might include:

- economic constraints;
- timescale constraints;
- development constraints;
- maintenance requirements;
- political constraints (e.g. technology export restrictions).

Guidelines

Before completing this stage, the analyst should be able to answer the following questions:

a) What is the most critical area of system performance, and what circumstances will cause the heaviest load in that area?

b) What are the longest acceptable response times to user inputs, and under what loading conditions?

c) What are the typical failures that will occur, and what is their impact on the system?

d) What mean times between failures can be tolerated for the various components of the system?

e) What are the most likely future extensions to the system, and what impact will they have?

f) What are the routine maintenance functions (hardware and software) that will have to be performed?

g) What size space will be available to accommodate both equipment and staff?

9. A software requirements document

The software requirements document should consist of:

(i) the problem definition document;

(ii) the viewpoint circle diagram, the viewpoint structure diagram, the viewpoint descriptions, and the identities of the viewpoint authorities;

(iii) the final version of the TCF for each viewpoint, reviewed and approved by the viewpoint authority (data structure diagrams can be included in an appendix);

(iv) relevant SVM diagrams and complementary descriptions;

(v) relevant CVM diagrams and complementary descriptions;

(vi) descriptive text and diagrams for each category of constraint.

Acknowledgements

This document has been adapted from the CORE Method Manual developed by System Designers Ltd.