7 Quantifying the Contribution of Task Complexity Factors

In this chapter, practical guidelines to quantifying the contribution of each complexity factor will be explained. In this regard, Table 7.1 that shows an overall quantification scheme will be helpful**.**

Table 7.1 Eight phases to quantify the contribution of each complexity factor

Phase	Description
1	Extracting the task structure of a procedure
2	Identifying the required actions with the sequence of actions
3	Identifying distinctive actions
4	Identifying necessary information about each distinctive action
5	Assigning the level of domain knowledge to each distinctive action
6	Assigning the level of engineering decision to each distinctive action
7	Constructing four kinds of graphs
8	Quantifying the contribution of each complexity factor

7.1 Extracting a Task Structure

As shown in Table 7.1, the first phase is to identify all the proceduralized tasks as well as the associated procedural steps prescribed in a procedure (i.e., a task structure). In other words, as shown in Fig. 1.1, since a procedure consists of a series of proceduralized tasks containing one or more procedural steps, identifying all the procedural tasks included in the procedure is the first phase in quantifying the complexity of proceduralized tasks. A typical example is Fig. 5.5, which clarifies a part of the task structure of the SGTR procedure of KSNPs (Park et al. 2005).

7.2 Identifying Required Actions with Their Sequence

If the task structure of a procedure being considered is identified, we have to identify the required actions with their sequence. For example, let us look at the following action descriptions that are prescribed in the *Instructions* of the fourth procedural step depicted in Fig. 5.5.

- **IF** pressurizer pressure is less than 123.9 kg/cm^2, **THEN** verify SIAS and CIAS are automatically actuated
- **IF** SIAS and CIAS are **NOT** automatically actuated, **THEN** manually actuate SIAS and CIAS

From the above action descriptions, it seems that the former contains two kinds of required actions, such as *pressurizer pressure is less than 123.9 kg/cm^2* and *verify SIAS and CIAS are automatically actuated*. Similarly, it appears that the latter also consists of two kinds of required actions, such as *SIAS and CIAS are NOT automatically actuated* and *manually actuate SIAS and CIAS*. In addition, since these action descriptions have conditional statements (e.g., a clause followed by IF, THEN, WHEN, WHILE, etc.), it is possible to understand an action sequence to be followed by qualified operators. However, two problems still remain.

The first problem is that some action descriptions do not satisfy the basic requirement of an action description – *each action should consist of one ACTION VERB, an OBJECT, and action specifications*. Figure 7.1 illustrates this problem more clearly.

Original description	Object	Action verb	Action specifications	Remark
Pressurizer pressure is less than 123.9 kg/cm^2	Pressurizer pressure		Less than 123.9 kg/cm^2	• Omitted ACTION VERB
SIAS and CIAS are NOT automatically actuated		SIAS CIAS	NOT automatically actuated	• Omitted ACTION VERB • OBJECT contains two kinds of components having different functions
Manually actuate SIAS and CIAS	SIAS CIAS	Actuate	Manually	• OBJECT contains two kinds of components having different functions
Verify SIAS and CIAS are automatically actuated	SIAS CIAS	Verify	Automatically actuated	

Fig. 7.1 Comparing the basic requirements of an action description

As shown in Fig. 7.1, the description of *verify SIAS and CIAS are automatically actuated* action does not satisfy the basic requirement because it mentions multiple *OBJECT*s, such as SIAS and CIAS, at the same time. In addition, the description of *pressurizer pressure is less than 123.9 kg/cm*^{2} action do not fulfill the basic requirement because there is no *ACTION VERB*.

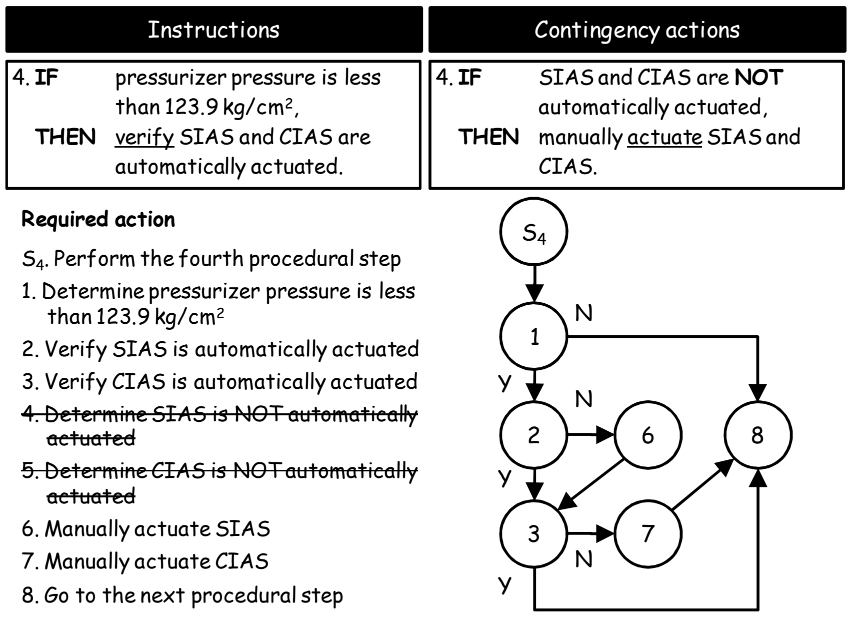
In order to resolve the problem of having multiple *OBJECT*s in an action description, therefore, we have to subdivide this action into two separate action descriptions that contain a single *OBJECT*. Moreover, the omission of an *ACTION VERB* can be corrected by adopting a hypothetical *ACTION VERB* when the description of an action includes any conditional statement. In other words, since qualified operators have to decide whether a conditional statement is satisfied or not, it is expected that the decision of a conditional statement can be substantiated using appropriate *ACTION VERB*s, such as *determine* or *verify,* for example. Consequently, Table 7.2 summarizes the required actions identified in the fourth procedural step depicted in Fig. 5.5.

Table 7.2 Identifying required actions

Original description	Subdivided action
IF pressurizer pressure is less than 123.9 kg/cm^2, THEN verify SIAS and CIAS are automatically actuated	Determine pressurizer pressure is less than 123.9 kg/cm^2
	Verify SIAS is automatically actuated
	Verify CIAS is automatically actuated
IF SIAS and CIAS are NOT automatically actuated, THEN manually actuate SIAS and CIAS	Determine SIAS is NOT automatically actuated
	Determine CIAS is NOT automatically actuated
	Manually actuate SIAS
	Manually actuate CIAS

The second problem is that some action descriptions seem to be less meaningful because they just represent (or emphasize) the opposite situation of an action. Let us look at two kinds of required actions, such as *verify SIAS is automatically actuated* and *determine SIAS is NOT automatically actuated*. In this case, the latter is unnecessary (or *vice versa*) because the former already encompasses two possible cases – whether SIAS has been automatically actuated or not. In other words, since *verify* forces qualified operators to make a decision, of which the result is either *YES* or *NO*, the whole sequence of required actions can be understood without the latter.

Based on the above explanations, the required actions listed in Table 7.2 can be reduced to the preliminary action sequence shown in Fig. 7.2. It is to be noted that a hypothetical action (i.e., *go to the next procedural step*) is added to Fig. 7.2 because, in most cases, qualified operators have to conduct proceduralized tasks that consist of two or more procedural steps. In addition, the action *perform the fourth procedural step* is added, because each action sequence should have a unique start point.



Instructions	Contingency actions
4. **IF** pressurizer pressure is less than 123.9 kg/cm^2, **THEN** verify SIAS and CIAS are automatically actuated.	4. **IF** SIAS and CIAS are **NOT** automatically actuated, **THEN** manually actuate SIAS and CIAS.

Required action

S_4. Perform the fourth procedural step
1. Determine pressurizer pressure is less than 123.9 kg/cm^2
2. Verify SIAS is automatically actuated
3. Verify CIAS is automatically actuated
4. ~~Determine SIAS is NOT automatically actuated~~
5. ~~Determine CIAS is NOT automatically actuated~~
6. Manually actuate SIAS
7. Manually actuate CIAS
8. Go to the next procedural step

Fig. 7.2 Identifying required actions with their sequence

7.3 Identifying Distinctive Actions

The preliminary sequence of actions presented in Fig. 7.2 is very important for constructing an ACG that can quantify the contribution of two kinds of complexity factors – the number of actions to be conducted by qualified operators and the logical entanglement to be followed by qualified operators. This implies that a set of distinctive actions (DAs) should be carefully identified before constructing an ACG. For example, let us assume a hypothetical ACG with two different procedural steps as depicted in Fig. 7.3.

In Fig. 7.3, each procedural step consists of six actions with the same sequence of actions. This means that the contributions of two kinds of complexity factors about these procedural steps are also identical because they share the same number of actions with the associated sequence. Unfortunately, this result seems to be unrealistic. For example, Kleinsorge et al. (2002) and Mayr and Keele (2000) experimentally showed that the response times of shifting from *Task B* to *Task A* are relatively higher when unqualified operators performed a nonrepeated task set (i.e., *Task C* → *Task B* → *Task A*) instead of a repeated task set (i.e., *Task A* → *Task B* → *Task A*). This strongly supports the notion that the repetition of identical actions will reduce the overall complexity of proceduralized tasks. In this regard, the contribution of complexity factors related to an ACG should be larger when qualified operators conducted procedural step S_1 because there are no repeated actions in it.

For this reason, it is necessary to identify all DAs that are included in a procedure through analyzing the specifications as well as the peculiarity of all the required actions.

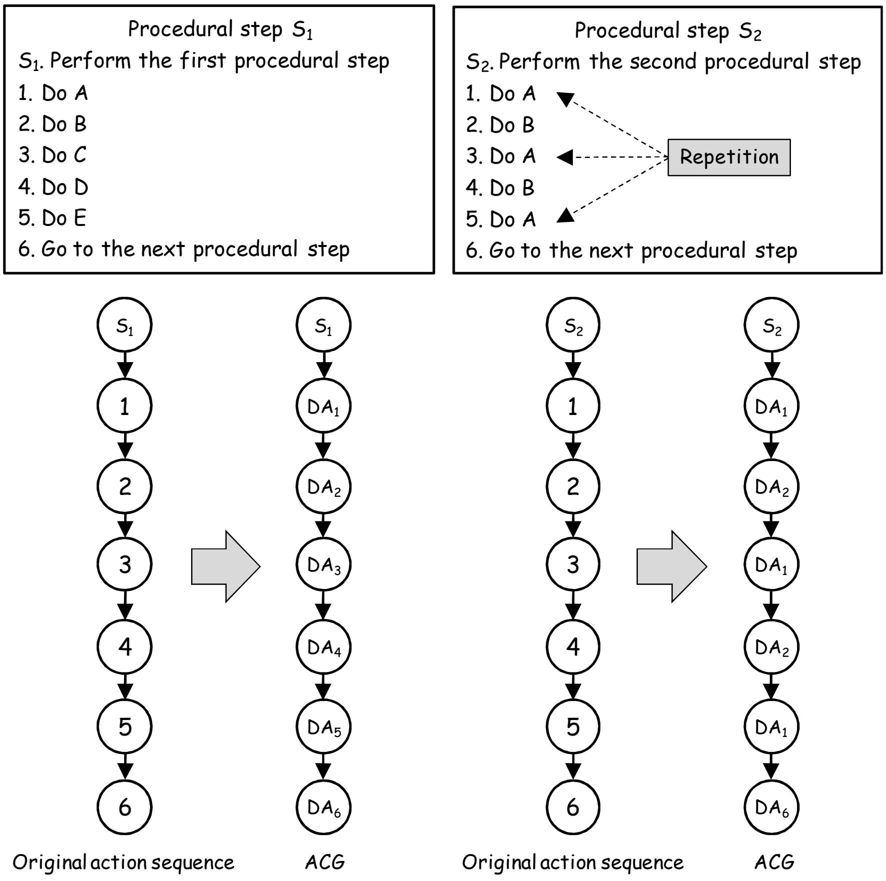


Fig. 7.3 Hypothetical ACGs with two different procedural steps

To this end, Table 7.3 gives an example of a typical usage of an action analysis form, in which distinctive actions can be easily distinguished. For instance, although the original descriptions of two required actions (the second and third actions) are different, they are regarded as the same action (i.e., DA_2), because they share the same action specifications (i.e., the same *OBJECT*, *OBJ*, *INH*, and *NL*) with no *peculiarity*. In contrast, although original descriptions of the first and ninth actions are identical, they should be distinguished as different actions (i.e., DA_1 and DA_8, respectively) because of the *peculiarity* of the ninth action.

Table 7.3 Example of the usage of an action analysis form

	ID	Required action	ACTION VERB	OBJECT	Peculiarity	ACCEPTANCE CRITERION	MEANS	CONSTRAINT
DA_1	1	Stop HPSI pumps	Stop	HPSI pumps	–	OBJ	INH	NL
DA_2	2	Verify pressurizer pressure is less than 123.9 kg/cm^2	Verify	Pressurizer pressure	–	OBJ	INH	NL
	3	Verify pressurizer pressure is between 135.0 and 165.0 kg/cm^2	Verify	Pressurizer pressure	–	OBJ	INH	NL
DA_3	4	Verify pressurizer pressure is abnormally decreasing	Verify	Pressurizer pressure	–	SUB	INH	NL
DA_4	5	Open SBCS valve #1 to 100%, until RCS temperature is less than 260°C	Open	SBCS valve #1	–	OBJ	INH	OBJ_C
DA_5	6	Open SBCS valve #1 to 100%, until RCS temperature is stable	Open	SBCS valve #1	–	OBJ	INH	SUB_C
DA_6	7	Stabilize RCS temperature	Stabilize	RCS temperature	CC	NC	NM	NL
DA_7	8	Depressurize pressurizer pressure using a pressurizer spray valve	Depressurize	Pressurizer pressure	CC	NC	DEG	NL
		IF necessary, perform ANY of the following						
DA_8	9	Stop HPSI pumps.	Stop	HPSI pumps	SEL	OBJ	INH	NL
		…						

*DA is short for distinctive action.

7.4 Identifying Necessary Information

If a set of DAs has been extracted, then the next phase is the identification of necessary information. In other words, all the information to be processed by qualified operators should be identified in this phase. To this end, three kinds of information pertaining to an action specification (i.e., *MEANS*, *ACCEPTANCE CRITERION*, and *CONSTRAINT*) are necessary for performing the required actions. On the basis of these clarifications, Table 7.4 exemplifies the usage of an information analysis form that can identify necessary information.

Table 7.4 Part of an information analysis form

	MEANS[1]	Type[2]	CONSTRAINT	Type	ACCEPTANCE CRITERION	Type
DA_1	HPSI pumps	AAB	–	–	HPSI pumps	AAB
DA_2	Pressurizer pressure	F	–	–	Pressurizer pressure	F
DA_4	SBCS valve #1	AF (jog control)	RCS temperature	F	SBCS valve #1	AF (jog control)

[1]Refer to action descriptions in Table 7.3

[2]Type denotes the basic type of information summarized in Table 6.9

For example, to accomplish DA_1, qualified operators need to manage control-related information (i.e., *MEANS*), which can be determined by the number of HPSI pumps as well as the number of available operating modes. At the same time, qualified operators need the status of HPSI pumps to clarify the *ACCEPTANCE CRITERION* of DA_1. In this regard, since qualified operators are able to directly identify the operating status of HPSI pumps from HPSI pump controllers, *AAB* (*Array of Array of Boolean*) should be commonly regarded as information about *MEANS* as well as about *ACCEPTANCE CRITERION* (Fig. 6.3). Similarly, *AF* (*Array of Float*) is commonly assigned to DA_4, because the source of information about the *MEANS* and the *ACCEPTANCE CRITERION* is a jog controller by which qualified operators are able to not only continuously adjust the open position of the SBCS valve #1 but also identify its open position.

7.5 Assigning the Level of Domain Knowledge

As mentioned in Sect. 3.3.3, qualified operators may feel a cognitive burden if they have to perform an action that requires a high level of domain knowledge. In contrast, qualified operators can probably perform the required action very easily if they are able to accomplish it with a low level of domain knowledge. Accordingly, four levels of domain knowledge are defined in Sect. 6.3.2 based on the Rasmussen's AH framework. Several rules that facilitate the assignment of the levels of domain knowledge are summarized in Table 7.5.

Table 7.5 Several rules for assigning levels of domain knowledge

ID	Rule description
1	The basic level of domain knowledge should be assigned by a knowledge-mapping table.
2	If the objects of the required actions contain the specific property of an entity, then the level of domain knowledge should be determined based on its entity. Typical examples are process parameters or conditions, such as pressurizer pressure, RCS temperature, etc.
3	If the required action does not include any *MEANS* (i.e., *NM*), then the next higher level of domain knowledge compared to the basic level determined from the knowledge-mapping table should be assigned to it.
4	If the *ACCEPTANCE CRITERION* of the required action is *NC* or *SUB*, then the next higher level of domain knowledge compared to the basic level determined from the knowledge-mapping table should be assigned to it.
5	If two or more required actions are grouped by *SEL*, then (1) the next higher level of domain knowledge compared to the basic level of the knowledge-mapping table should be assigned to each action, (2) the highest level of domain knowledge among all the grouped actions should be determined, and (3) the highest level of domain knowledge should be assigned to all the required actions being grouped.
6	AF should be assigned to all the local operations (i.e., *LO*)

The intention of the first rule is to minimize an inconsistency as much as possible, which might be observed during the assignment of the levels of domain knowledge. Table 7.6 shows a typical knowledge-mapping table that could be used for PWRs. For example, if the *OBJECT* of the required action is a kind of pump, then qualified operators should just need domain knowledge pertaining to the function of a component itself (e.g., *CF*). In contrast, if qualified operators have to consider a boundary that consists of two or more components with distinctive functions or purposes, then it is reasonable to anticipate that they will need system level knowledge (e.g., *SF*).

The second rule is related to the assignment of the level of domain knowledge if the *OBJECT* of the required action represents any attribute of it. For example, let us recall DA_2 in Table 7.3, where the *OBJECT* is *pressurizer pressure*. In this case, since pressure is one of the typical attributes of the pressurizer, it is reasonable to assign the level of domain knowledge based on that of the pressurizer. This means that *SF* should be assigned to DA_2 according to the knowledge-mapping table. Similarly, the level of domain knowledge about DA_6 is *PF* because the RCS encompasses several distinctive systems, such as the reactor vessel, RCPs, SGs, etc. In addition, since each system generally has two or more distinctive functions, the concurrent consideration of identical systems should be regarded as *PF*. For example, if the *OBJECT* of an arbitrary action is RCPs (e.g., *stop all RCPs*) or SGs (e.g., *verify all levels of SGs are greater than 23.5%*), we have to assign *PF* to it in order to represent the level of domain knowledge.

Table 7.6 A knowledge-mapping table that could be used for PWRs

Level of domain knowledge	Corresponding object
Component function (CF)	• All kinds of valves, heaters, reservoirs (tanks), batteries, pipes, etc. • All kinds of pumps except RCPs • All kinds of heat exchangers except SGs and condensers • Anything else that can be regarded as a distinguishable functional unit according to a tacit consensus among qualified operators working in PWRs
System function (SF)	• A building such as a containment or turbine building • Reactor vessel • Pressurizer • SGs • RCPs • Diesel generators • Turbines • Condensers • Any boundary that contains two or more distinctive components that have different functions or purposes
Process function (PF)	Any boundary that contains two or more system functions. A typical example is the simultaneous consideration of system functions such as RCPs or SGs
Abstract function (AF)	Any boundary that contains two or more process functions

The third rule implies the enlargement of domain knowledge due to the absence of a proper *MEANS*. Let us look at Fig. 7.4, which compares the changes in an expected problem space of an arbitrary system containing four valves and a reservoir.

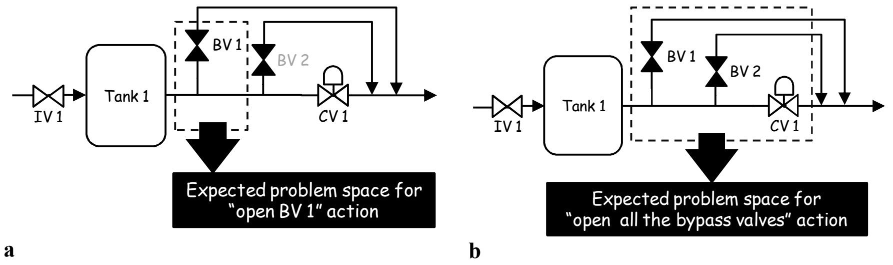

BV 1
BV 2
Tank 1
IV 1
CV 1
Expected problem space for "open BV 1" action
a
BV 1
BV 2
Tank 1
IV 1
CV 1
Expected problem space for "open all the bypass valves" action
b

Fig. 7.4 Two examples of the changes in an expected problem space

Above all, as depicted in Fig. 7.4a, it seems to be obvious that qualified operators focus on a narrow problem space to perform *open BV 1* action (refer to an area enclosed by dotted lines) because the *OBJECT* to be acted on is a single component. In contrast, qualified operators probably enlarge their problem space to perform *open all the bypass valves* action because a higher level of domain knowledge will be necessary to answer several questions, such as *which valves are bypass valves*? or *how many bypass valves are linked to Tank 1*?, etc. In other words, as illustrated in Fig. 7.4b, it is anticipated that this action will compel qualified operators to search a certain problem space that consists of several valves surrounding Tank 1. Accordingly, it is reasonable to assume that the next higher level of domain knowledge compared to the basic level determined from the knowledge-mapping table should be assigned to the required action without having detailed specifications about a *MEANS* (i.e., *NM*). This implies that *SF* should be assigned to *open all the bypass valves* action, because the *OBJECT* of this action includes a couple of bypass valves that share the same function (i.e., *CF*).

The fourth rule closely resembles the third rule because the omission of detailed specifications about an *ACCEPTANCE CRITERION* (i.e., *NC*) probably requires additional cognitive resources to process a higher level of domain knowledge. Let us look at Fig. 7.5, which shows a hypothetical trend about the water level of Tank 1.

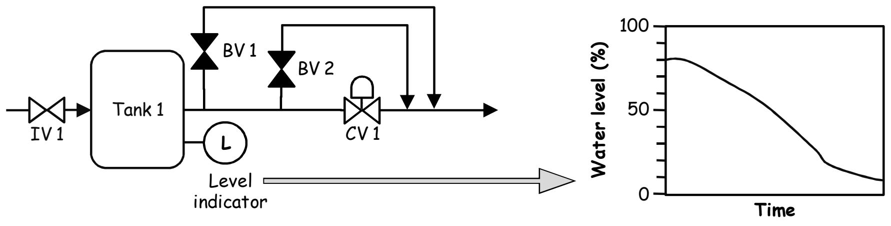


Fig. 7.5 Hypothetical trend in water level of Tank 1

From Fig. 7.5 it is evident that qualified operators can easily perform *verify the water level of Tank 1 is decreasing* action. However, qualified operators are likely to get frustrated when they are faced with *verify the water level of Tank 1 is abnormally decreasing* action because the *ACCEPTANCE CRITERION* of this action varies with respect to the status of surrounding components. That is, if there is no good reason to explain the decrease in the water level of Tank 1, then qualified operators will suspect an abnormal decrease due to other factors, such as a break in a pipe. To this end, qualified operators will carefully observe the status of components that might cause a decrease in the water level of Tank 1, such as the status of BV 1 as well as BV 2 or the position of CV 1 and IV 1, etc. This strongly implies that the fourth rule is meaningful because qualified operators need a higher level of domain knowledge that is indispensable to identifying the associated components to be considered.

The fifth rule is applied when several actions are grouped by *SEL*. For example, let us assume the following equally acceptable actions.

IF necessary, perform **ANY** of the following:
- Stop pump A
- Maintain the water level of pressurizer within 30~50%

In this case, qualified operators have to select the most appropriate action. To do this, as explained in Sect. 6.3.3, qualified operators probably evaluate both actions from many standpoints, such as the suitability of an action for a given situation. From this concern it is natural to assume that qualified operators may need a higher level of domain knowledge compared to an original level assigned by the knowledge-mapping table. Actually, this rule is very similar to both the third and fourth rules because qualified operators need to possess a higher level of domain knowledge to make a decision. However, it is also assumed that the extension of domain knowledge to clarify an effective *MEANS* as well as an ambiguous *ACCEPTANCE CRITERION* (e.g., *SUB* or *NC*) should be different from the selection of the most proper action, because the selection would encompass the evaluation of candidate actions. In other words, since qualified operators have to evaluate not a single action but two or more equally acceptable actions, the total amount of domain knowledge necessary for the selection of the most proper action should be larger than that of a single action with *NM*, *SUB,* and *NC*. Therefore, the sixth rule is considered in order to compensate for this concern. Fig. 7.6 illustrates detailed steps to explain why the *PF* level is commonly assigned to the above two actions.

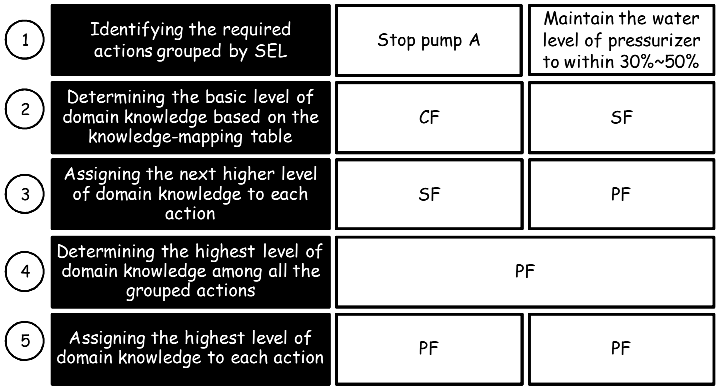

1
Identifying the required actions grouped by SEL
Stop pump A
Maintain the water level of pressurizer to within 30%~50%
2
Determining the basic level of domain knowledge based on the knowledge-mapping table
CF
SF
3
Assigning the next higher level of domain knowledge to each action
SF
PF
4
Determining the highest level of domain knowledge among all the grouped actions
PF
5
Assigning the highest level of domain knowledge to each action
PF
PF

Fig. 7.6 Example illustrating assignment of the levels of domain knowledge when two kinds of required actions are grouped by *SEL*

The last rule concerns actions that require *LO*. As stated in Sect. 6.2.1, it is very difficult to elucidate necessary *MEANS* that would actually be used by field operators. Similarly, it is also difficult to extract an expected problem space to be considered by field operators. However, it seems to be irrational to assign a low level of domain knowledge to this action because higher-level cognitive activities, such as communicating intention between board operators and field operators, are essential for the accomplishment of the required action. Accordingly, for the sake of conservativeness, *AF* is uniformly assumed for actions that require *LO*.

7.6 Assigning the Level of Engineering Decision

After the level of domain knowledge has been assigned, the level of the engineering decision should be assigned. Table 7.7 summarizes several practical rules related to determining the level of the engineering decision**.**

Table 7.7 Practical rules related to assigning levels of engineering decisions

ID	Rule description
1	The lowest level of the engineering decision (i.e., ED-1) is assigned to an action whose *ACCEPTANCE CRITERION* is *OBJ*, unless its property is not *Trend*
2	ED-1 is assigned to an action whose *CONSTRAINT* is specified by *OBJ_C*
3	The second level of the engineering decision (i.e., ED-2) is assigned to an action if the property of an *ACCEPTANCE CRITERION* is *Trend*
4	The second level of the engineering decision (i.e., ED-2) is assigned to an action if the property of a *CONSTRAINT* is *Trend*
5	ED-2 is assigned to an action whose *ACCEPTANCE CRITERION* is *RI*
6	ED-2 is assigned to an action whose *CONSTRAINT* is *RI_C*
7	The third level of the engineering decision (i.e., ED-3) is assigned to an action if its peculiarity is *CC*
8	ED-3 is assigned to an action whose *ACCEPTANCE CRITERION* is either *SUB* or *NC*
9	ED-3 is assigned to an action whose *CONSTRAINT* is *SUB_C*
10	ED-3 is assigned to an action if there is no specification about *MEANS* (i.e., *NM*)
11	The fourth level of the engineering decision (i.e., ED-4) is assigned to an action if its peculiarity is *SEL*
12	ED-4 is assigned to an action that requires *LO*

For example, let us consider *verify the water level of Tank 1 is less than 30%* action whose *ACCEPTANCE CRITERION* is specified in the form of a discrete value. In this case, qualified operators should be able to easily determine whether the *ACCEPTANCE CRITERION* is satisfied or not. Therefore, this action belongs to the first level of the engineering decision (i.e., ED-1) because a simple decision will be made based on a clear decision criterion.

In addition, the second level of the engineering decision (i.e., ED-2) should be assigned to *verify the water level of Tank 1 is decreasing* action, if we recall that the meaning of ED-2 is an action that forces qualified operators to integrate lower-level information to create higher-level information (Table 6.12). In other words, determining the trend of the water level belongs to ED-2 because qualified operators need to identify the status of the water level by integrating a data series.

Moreover, several rules pertaining to the assignment of the third level (i.e., ED-3) as well as the fourth level of the engineering decision (i.e., ED-4) can be understood in connection with their definitions. For example, let us consider *maintain the water level of Tank 1 within the range 30% to 50% by using CV 1* action in Fig. 7.5. In order to accomplish this action, qualified operators have to answer supplementary questions, such as *how suitable the open position of CV 1 is in this situation*? That is, if the water level is very close to 50%, then qualified operators will be apt to completely close CV 1. In addition, if the change in the water level is not too drastic, then qualified operators will adjust the open position of CV 1 along with the trend of the water level. Obviously, since qualified operators have to establish a proper decision criterion by themselves based on the nature of an ongoing situation, it is meaningful to assign ED-3 to this action. Similarly, if qualified operators have to conduct an action in which there is no specification about *MEANS*, they will probably establish a decision criterion by themselves in order to come up with the proper method for coping with an ongoing situation. Accordingly, it is reasonable to assign ED-3 to this kind of action.

However, the last rule is worthy of special note, because it is assumed valid for the same reason as the assignment of the level of domain knowledge. That is, since it is very difficult to elucidate how field operators can actually perform the required action in a local place, the highest level (i.e., *ED-4*) is assigned for the sake of conservativeness.

7.7 Constructing Four Kinds of Graphs

When all the aforementioned phases are finished, it is possible to construct four kinds of essential graphs through which the contribution of each complexity factor can be quantified by the concept of graph entropies. Let us consider an arbitrary task structure that consists of two procedural steps, *Step*$_1$ and *Step*$_2$, as depicted in Fig. 7.7.

		Instructions	Contingency actions
Task (T)	**Step$_1$**	**IF** pressurizer pressure is less than 123.9kg/cm^2, **THEN** verify SIAS and CIAS are automatically actuated.	**IF** SIAS and CIAS are **NOT** automatically actuated, **THEN** manually actuate SIAS and CIAS.
	Step$_2$	**IF** pressurizer pressure is less than 121.0kg/cm^2 **AND** SIAS is actuated, **THEN** perform BOTH of the following: a. Stop **ONE** RCP in each loop . b. **IF** RCS subcooling margin is less than 15°C, **THEN** stop **ALL** RCPs	

Fig. 7.7 An arbitrary task comprises two procedural steps

First, based on the task structure shown in Fig. 7.7, all the required actions could be identified as listed in Table 7.8. In addition, a set of DAs can be extracted as listed in Table 7.9.

Table 7.8 Required actions included in each procedural step

Procedural step	ID	Required action
$Step_1$	1	Perform $Step_1$
	2	Determine pressurizer pressure is less than 123.9 kg/cm^2
	3	Verify SIAS is automatically actuated
	4	Verify CIAS is automatically actuated
	5	Manually actuate SIAS
	6	Manually actuate CIAS
	7	Go to the next procedural step
$Step_2$	8	Perform $Step_2$
	9	Determine pressurizer pressure less than 121 kg/cm^2
	10	Determine SIAS is actuated
	11	Stop one RCP in each loop
	12	Determine RCS subcooling margin is less than 15^{o}C
	13	Stop all RCPs
	14	Go to the next procedural step

Table 7.9 Action analysis form for the required actions included in *$Step_1$* and *$Step_2$*

DA	ID	ACTION VERB	OBJECT	MEANS	ACCEPTANCE CRITERION	CONSTRA-INT	Pecu-liarity
S_1	1	Perform	$Step_1$	INH	OBJ	NL	–
DA_1	2	Determine	Pressurizer pressure	INH	OBJ	NL	–
	9	Determine	Pressurizer pressure	INH	OBJ	NL	–
DA_2	3	Verify	SIAS	INH	OBJ	NL	–
DA_3	4	Verify	CIAS	INH	OBJ	NL	–
DA_4	5	Actuate	SIAS	INH	OBJ	NL	–
DA_5	6	Actuate	CIAS	INH	OBJ	NL	–
DA_6	7	Go to	Next procedural step	INH	OBJ	NL	–
	14	Go to	Next procedural step	INH	OBJ	NL	–
S_2	8	Perform	$Step_2$	INH	OBJ	NL	–
DA_7	10	Determine	SIAS	INH	OBJ	NL	–
DA_8	11	Stop	(One) RCP	INH	OBJ	RI_C*	–
DA_9	12	Determine	RCS subcooling margin	INH	OBJ	NL	–
DA_{10}	13	Stop	RCPs	INH	OBJ	NL	–

*The specification, such as “in each loop,” corresponds to the static configuration (Table 6.5)

Consequently, Fig. 7.8 shows two ACGs for *Step*$_1$ and *Step*$_2$ that are constructed based on DAs summarized in Table 7.9.

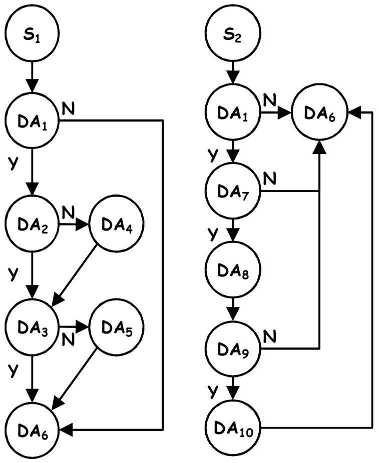


Fig. 7.8 Two ACGs about *Step*$_1$ and *Step*$_2$

Second, necessary information to be processed by qualified operators can be identified from DAs. Table 7.10 shows the source of necessary information when qualified operators working in a conventional MCR, have to perform several DAs.

Table 7.10 Information analysis form for *Step*$_1$ and *Step*$_2$

ID	MEANS	Type	CONSTRAINT	Type	ACCEPTANCE CRITERION	Type
DA_1	Pressurizer pressure indicator	F	–	–	Pressurizer pressure indicator	F
DA_2 DA_7	SIAS status indicator	B	–	–	SIAS status indicator	B
DA_3	CIAS status indicator	B	–	–	CIAS status indicator	B
DA_4	SIAS actuator	B	–	–	SIAS status indicator	B
DA_5	CIAS actuator	B	–	–	CIAS status indicator	B
DA_8	RCP controller	AB	–	–	RCP controller	AB
DA_9	RCS subcooling margin indicator	F	–	–	RCS subcooling margin indicator	F
DA_{10}	RCP controllers	AAB	–	–	RCP controllers	AAB

Here, there are some points to be noted.

- Necessary information related to S_1, S_2, and DA_6 is not identified because these actions are assumed at our discretion.

- Although the original descriptions of DA_2 and DA_7 are different, the sources of necessary information are the same.
- As SIAS and CIAS can be actuated by a kind of binary controller, their status indicators are necessary to confirm the *ACCEPTANCE CRITERION* (Fig. 6.2a).
- The *CONSTRAINT* of DA_8 is not considered because qualified operators perhaps recall a kind of domain knowledge to perform this action. That is, since information related to identifying *one RCP in each loop* could be extracted from domain knowledge of qualified operators, it is impossible to designate the type of basic information, such as *F* (Float) or *B* (Boolean), etc. Similarly, there are times when it is difficult to identify the types of necessary information if the *ACCEPTANCE CRITERION* or the *CONSTRAINT* of an action has the property such as *equation*, *formula,* or *dynamic configuration*. Therefore, in order to compensate for this problem, two rules are predefined in Table 7.7. In other words, since the recall of domain knowledge to determine *RI* or *RI_C* could be regarded as the creation of higher level information by integrating lower level information, ED-2 is assigned to an action that contains either *RI* or *RI*_C.

Based on the necessary information summarized in Table 7.10 with the aforementioned notes, we can extract a set of distinctive information (DI) as listed in Table 7.11. This means that qualified operators are supposed to manage at least this kind of information to perform *Step$_1$* and *Step$_2$*. It is to be noted that *RCP controllers* are only considered as DI_6 because the source of information about DA_{10} includes that of DA_8. Accordingly, it is possible to construct two ISGs for *Step$_1$* and *Step$_2$*, as depicted in Fig. 7.9, in which the representation of necessary information will be illustrated by all nodes that are linked to the root nodes, S_1 or S_2.

Table 7.11 Distinctive information identified from *Step$_1$* and *Step$_2$*

		Meaning	Type
$Step_1$	*DI_1	Pressurizer pressure indication	F
	DI_2	SIAS status indication	B
	DI_3	CIAS status indication	B
	DI_4	SIAS actuator	B
	DI_5	CIAS actuator	B
$Step_2$	DI_1	Pressurizer pressure indication	F
	DI_2	SIAS status indication	B
	DI_6	RCP controllers	AAB
	DI_7	RCS subcooling margin indicator	F

*DI: distinctive information

Third, we are able to construct two AHGs for *Step$_1$* and *Step$_2$* using the list of

DAs and the associated rules to assign the level of domain knowledge. Table 7.12 summarizes the level of domain knowledge assigned to each DA. For example, according to the second rule in Table 7.5, the level of domain knowledge about DA_1 should be SF because pressure is the typical property of a pressurizer.

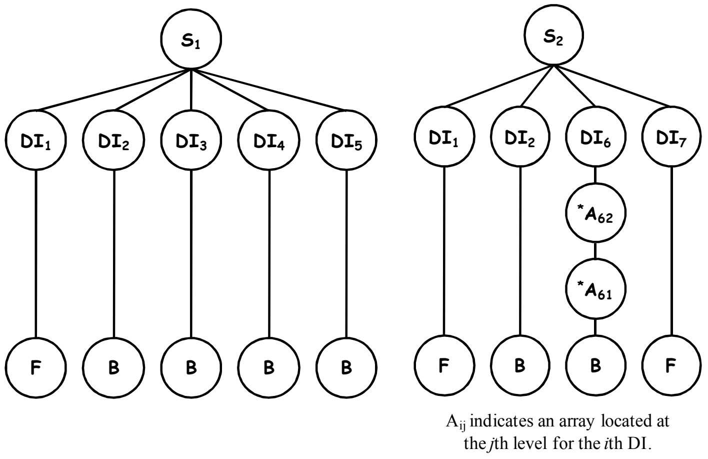
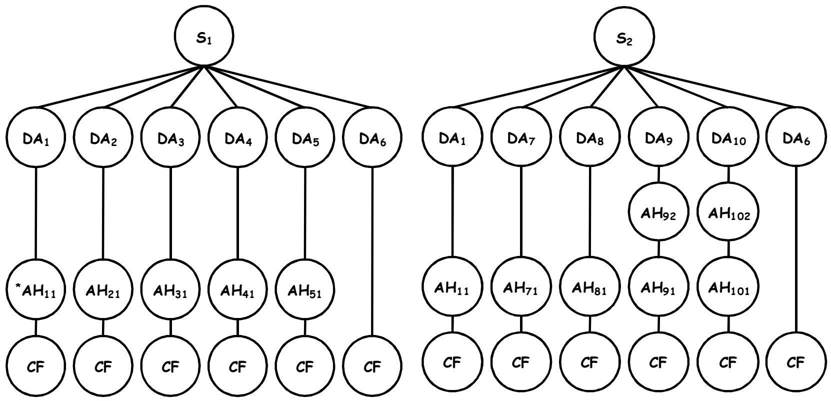


Fig. 7.9 Two ISGs of *$Step_1$* and *$Step_2$*

Table 7.12 Level of domain knowledge of each DA

	DA	Original description	OBJECT	Level of domain knowledge
$Step_1$	DA_1	Determine pressurizer pressure is less than 123.9 kg/cm^2	Pressurizer pressure	SF (pressure is the typical property of a pressurizer)
	DA_2	Verify SIAS is automatically actuated	SIAS	SF (SIAS the typical property of a HPSI system)
	DA_3	Verify CIAS is automatically actuated	CIAS	SF (CIAS is the typical property of a containment)
	DA_4	Manually actuate SIAS	SIAS	SF
	DA_5	Manually actuate CIAS	CIAS	SF
	DA_6	Go to the next procedural step	Next procedural step	CF
$Step_2$	DA_1	Determine pressurizer pressure is less than 121.0kg/cm^2	Pressurizer pressure	SF
	DA_7	Determine SIAS is actuated	SIAS	SF
	DA_8	Stop one RCP in each loop	RCP	SF
	DA_9	Determine RCS subcooling margin is less than 15^{o}C	RCS subcooling margin	PF (subcooling margin is the typical property of a RCS)
	DA_{10}	Stop all RCPs	RCPs	PF
	DA_6	Go to the next procedural step	Next procedural step	CF

Here, it is to be noted that the level of domain knowledge about DA_6 is assumed to be CF. That is, since this action is introduced at our discretion, it is meaningless to consider the level of domain knowledge about DA_6. For this reason, the lowest level of domain knowledge is assigned to DA_6. Figure 7.10 depicts two AHGs for *Step₁* and *Step₂* based on the levels of domain knowledge summarized in Table 7.12.



AH_{ij} indicates an array located at the *j*th level for the *i*th DA.

Fig. 7.10 Two AHGs of *Step₁* and *Step₂*

As for the last graph, two EDGs of *Step₁* and *Step₂* can be constructed based on DAs as well as the associated rules to assign the level of the engineering decision. Table 7.13 summarizes the level of engineering decision assigned to each DA.

Table 7.13 Level of engineering decision about each DA

	ID	MEANS	ACCEPTANCE CRITERION	CONSTRAINT	Peculiarity	Assigned level
Step₁	DA_1	INH	OBJ	NL	–	ED-1
	DA_2	INH	OBJ	NL	–	ED-1
	DA_3	INH	OBJ	NL	–	ED-1
	DA_4	INH	OBJ	NL	–	ED-1
	DA_5	INH	OBJ	NL	–	ED-1
	DA_6	INH	OBJ	NL	–	ED-1
Step₂	DA_1	INH	OBJ	NL	–	ED-1
	DA_7	INH	OBJ	NL	–	ED-1
	DA_8	INH	OBJ	RI_C	–	ED-2
	DA_9	INH	OBJ	NL	–	ED-1
	DA_{10}	INH	OBJ	NL	–	ED-1
	DA_6	INH	OBJ	NL	–	ED-1

For example, according to the first rule given in Table 7.7, the level of engineering decision for DA_1 should be ED-1 because the *ACCEPTANCE CRITERION* of this action is *OBJ*. In addition, the fifth rule in Table 7.7 indicates that the level of the engineering decision for DA_8 should be ED-2 because the *CONSTRAINT* of this action is *RI_C*. In this way, the levels of all the distinctive actions can be systematically determined. As a result, Fig. 7.11 depicts two EDGs for *Step₁* and *Step₂*.

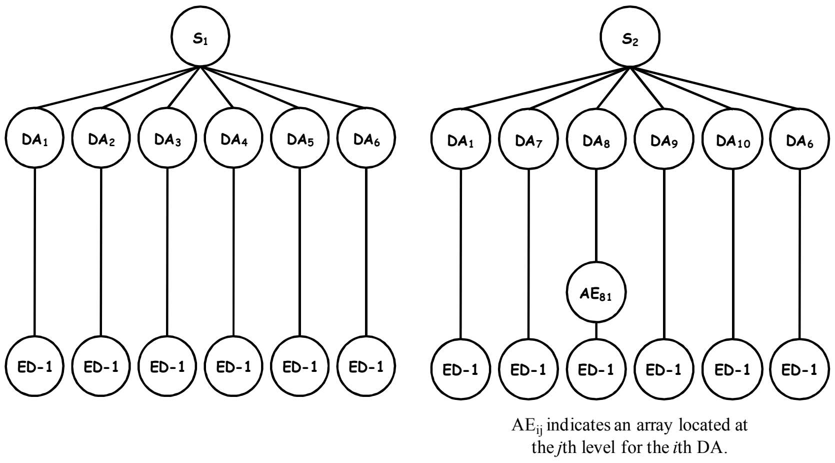


Fig. 7.11 Two EDGs of *$Step_1$* and *$Step_2$*

7.8 Quantifying Five Kinds of Complexity Factors

When all four graphs are constructed, it is possible to quantify the contributions of five kinds of complexity factors based on the associated graph entropies, as clarified in Table 7.14.

Table 7.14 Graph entropies to quantify the associated complexity factors

Complexity factor	Graph entropy
Number of actions	Second-order entropy of an ACG
Logical entanglement	First-order entropy of an ACG
Amount of information	Second-order entropy of an ISG
Amount of domain knowledge	Second-order entropy of an AHG
Level of engineering decision	Second-order entropy of an EDG

For example, let us quantify the contribution of the number of actions in a task depicted in Fig. 7.7. To this end, we need to quantify the second-order entropy of the two ACGs shown in Fig. 7.8. This means that it is essential to introduce the sum of graphs that belong to one of the graph operations.

The sum of two graphs X and Y is mathematically defined as follows (Mowshowitz 1968a): “The sum of X and Y is the graph $X \cup Y$ given by $V(X \cup Y) = V(X) + V(Y)$ and $E(X \cup Y) = E(X) + E(Y)$ where V(X) and E(X) denote the set of vertices (i.e., nodes) and the set of edges (i.e., arcs) included in a graph X, respectively.”

Mathematically, the sum of graphs means the simple union of all the nodes as well as the arcs included in all the graphs under consideration. Here, it should be emphasized that there are two rationales supporting the notion that the sum of graphs is meaningful in quantifying the complexity of proceduralized tasks.

First, this concept makes it possible to quantify the contribution of each complexity factor by considering all the necessary graphs of the associated procedural steps without any modification. For example, Fig. 7.12 summarizes the result of node classifications with respect to the sum of two ACGs shown in Fig. 7.8.

The result of node classifications of Step$_1$

Class	Identical node	Neighbor node
I	$\{S_1\}$	$\{DA_1\}$
II	$\{DA_1\}$	$\{S_1, DA_2, DA_6\}$
III	$\{DA_2\}$	$\{DA_1, DA_3, DA_4\}$
IV	$\{DA_3\}$	$\{DA_2, DA_4, DA_5, DA_6\}$
V	$\{DA_4\}$	$\{DA_2, DA_3\}$
VI	$\{DA_5\}$	$\{DA_3, DA_6\}$
VII	$\{DA_6\}$	$\{DA_1, DA_3, DA_5\}$

The result of node classifications of Step$_2$

Class	Identical node	Neighbor node
I	$\{S_2\}$	$\{DA_1\}$
II	$\{DA_1\}$	$\{S_2, DA_6, DA_7\}$
III	$\{DA_6\}$	$\{DA_1, DA_7, DA_9, DA_{10}\}$
IV	$\{DA_7\}$	$\{DA_1, DA_6, DA_8\}$
V	$\{DA_8\}$	$\{DA_7, DA_9\}$
VI	$\{DA_9\}$	$\{DA_6, DA_8, DA_{10}\}$
VII	$\{DA_{10}\}$	$\{DA_6, DA_9\}$

The result of node classifications of the sum of two graphs (Step$_1$ and Step$_2$)

Class	Identical node	Neighbor node
I	$\{S_1, S_2\}$	$\{DA_1\}$
II	$\{^{*}DA_1\}$	$\{S_1, DA_2, DA_4\}$
III	$\{^{*}DA_2\}$	$\{DA_1, DA_3\}$
IV	$\{^{*}DA_3\}$	$\{DA_2, DA_4\}$
V	$\{^{*}DA_4\}$	$\{DA_1, DA_3\}$
VI	$\{^{*}DA_5\}$	$\{DA_3, DA_6\}$
VII	$\{^{*}DA_6\}$	$\{DA_1, DA_3, DA_5\}$
VIII	$\{^{**}DA_1\}$	$\{S_2, DA_6, DA_7\}$
IX	$\{^{**}DA_6\}$	$\{DA_1, DA_7, DA_9, DA_{10}\}$
X	$\{^{**}DA_7\}$	$\{DA_1, DA_6, DA_8\}$
XI	$\{^{**}DA_8\}$	$\{DA_7, DA_9\}$
XII	$\{^{**}DA_9\}$	$\{DA_6, DA_8, DA_{10}\}$
XIII	$\{^{**}DA_{10}\}$	$\{DA_6, DA_9\}$

*A node that belongs to the ACG of Step$_1$.
**A node that belongs to the ACG of Step$_2$.

Fig. 7.12 Distinctive classes to quantify the second-order entropy on the sum of two graphs

As can be seen from Fig. 7.12, two nodes (S_1 and S_2) should be considered identical, because they share the same neighbor node, DA_1. In contrast, it is evident that other nodes do not have the same neighbor node. Accordingly, since the sum of two graphs has a total of 13 distinctive classes, the second-order entropy of ACGs is

$$H_2(Step_1 \cup Step_2) = -\sum_{i=1}^{13} p_i \cdot \log_2 p_i = -\left\{\left(\frac{2}{14}\right)\cdot \log_2\left(\frac{2}{14}\right) + 12 \cdot \left(\frac{1}{14}\right)\cdot \log_2\left(\frac{1}{14}\right)\right\} = 3.665.$$

This implies that the contribution of the number of actions on the complexity of a proceduralized task can be quantified as 3.665. In this way, the contributions of other complexity factors on the complexity of proceduralized tasks can be quantified. For the sake of convenience, henceforth, it would be better to define five kinds of submeasures covering the associated complexity factors. These submeasures are given below.

- Step size complexity (SSC), which indicates the complexity due to the number of the required actions to be performed by qualified operators, can be quantified by the second-order entropy of an ACG.
- Step logic complexity (SLC), which denotes the complexity due to the logical entanglement of the required actions, can be quantified by the first-order entropy of an ACG.
- Step information complexity (SIC), which represents the complexity due to the amount of information to be processed by qualified operators, can be quantified by the second-order entropy of an ISG.
- Abstraction hierarchy complexity (AHC), which implies the complexity due to the amount of domain knowledge needed by qualified operators, can be quantified by the second-order entropy of an AHG.
- Engineering decision complexity (EDC), which denotes the complexity due to the amount of cognitive resources for establishing the decision criteria of the required actions, can be quantified by the second-order entropy of an EDG.

Second, the sum of graphs makes it possible to explicitly depict the reduction of a task complexity that stems from the repetition of similar actions. In order to clarify the nature of this characteristic, let us compare the SSC values of three ACGs. In Fig. 7.13, it is observed that two ACGs (i.e., $\text{Step}_1 \cup \text{Step}_2$) share common graph nodes, DA_1 and DA_6. This means that the value of the SSC about the sum of two ACGs explicitly represents the reduction of a task complexity due to the common graph nodes. According to the theory of graph entropies, the diminution of entropy values due to mutual information (i.e., common graph nodes) is represented by the concept of mutual information (Abramson 1963).

For example, as illustrated in Fig. 7.13, the SSC value about the sum of ACGs is 3.665, while the SSC values of $Step_1$ and $Step_2$ are 2.087 and 2.807, respectively. In theory, the SSC value about the sum of ACGs should be the sum of SSC values of each ACG because the sum of graphs was defined as the simple union of all the nodes as well as the arcs included in all the graphs under consideration. However, since there is mutual information originated from common graph nodes, the actual SSC value of the sum of ACGs is less than the expected value. This implies that the graph entropy value decreases as the number of identical graph nodes increases. Consequently, the complexity of a proceduralized task will decrease in proportion to the number of identical actions to be repeated by qualified operators.

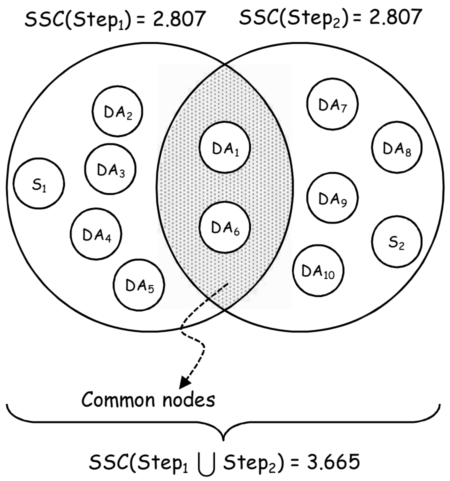


Fig. 7.13 Comparing SSC values of three ACG

References

Abramson N (1963) Information theory and coding. McGraw-Hill, New York

Kleinsorge T, Heuer H, Schmidtke V (2002) Process of task-set reconfiguration: Switching operations and implementation operations. Acta Psychol 111:1–28

Mayr U, Keele SW (2000) Changing internal constraints on action: the role of backward inhibition. J Exp Psychol: Gen 129(1):4–26

Mowshowitz A (1968a) Entropy and the complexity of graphs: I. An index of the relative complexity of a graph. Bull Math Biophys 30:175–204

Park J, Jung W, Kim J, Ha J (2005) Analysis of human performance observed under simulated emergencies of nuclear power plants. KAERI/TR-2895, Daejeon, South Korea