

How Can CASE Help?

A Look at the Feasibility of Supporting Structured Analysis with CASE

David Jankowski
California State University,
San Marcos

Abstract

One important role of a CASE tool is to serve as a methodology companion — to assist an analyst in the creation of documentation passed to succeeding phases of the life cycle and to guide the analyst through a particular systems development methodology. While many vendors claim their CASE products supports a particular methodology, the actual level of support varies greatly from one CASE tool to another and, for a particular tool, the level of support varies among the rules of the methodology.

The feasibility of using CASE to provide automated checking for the rules of structured analysis is presented within the context of a framework for examining CASE methodology support. Two popular CASE tools are compared against a feasibility benchmark by examining system specifications created using the tools. The results indicate that methodologically consistent specifications are more likely to be achieved when the methodology support provided by the CASE tool is as rigorous as possible.

ACM Categories: D.2.2

Keywords: computer-aided systems engineering tools, systems development methodologies

Introduction

One important role of a CASE tool is to serve as a methodology companion — to assist the user in the creation of documentation passed to succeeding phases of the life cycle and to guide the user through a particular systems development methodology (McClure, 1989). The level of methodology assistance provided by a CASE tool varies from product to product and may include graphics support for various diagramming techniques; a data dictionary for storing entities associated with a systems project; and automated checks, which serve to enforce a particular methodology and help ensure the completeness and consistency of the resulting specifications. Unfortunately, while many vendors claim their CASE product supports a particular information systems development methodology, the actual level of methodology support varies greatly from one CASE tool to another and, for a particular tool, the level of support varies among the rules of the

methodology. Further, the level of methodology support provided by the tools is often cited as being insufficient (Crosslin, Bergin, & Stott, 1993; Henderson & Cooperider, 1990, p. 251; Loy, 1993, p. 31; Sumner, 1993).

In this paper, a framework for classifying CASE methodology support is proposed and applied to structured analysis methodology rules. The framework is then used to determine the feasibility of automating the structured analysis methodology. Next, the structured analysis methodology support provided by two popular commercial CASE products is displayed using the framework.¹ Finally, some preliminary research into the influence different levels of CASE methodology support has upon the functional specification is presented.

The goal of this discussion is to advance the literature concerning CASE tools and their influence on system specification quality and to provide some insight into how CASE can improve productivity by eliminating much of the labor — intensive task of completeness and consistency checking of specifications. By providing designers, programmers, and/or code generators with methodologically correct specifications, system quality can be improved.

From the proposed framework, hypotheses may be generated and tested in an attempt to determine the “optimal” level of methodology support provided by CASE tools. Further, CASE-tool selection criteria, of which support for a particular methodology is an important criteria (Everest & Alanis, 1992), will be refined by identifying differences in the level of methodology feedback provided by the CASE products.

A Framework for Classifying Case Methodology Feedback

The first substantial work aimed at classifying CASE tools based upon their level of methodology support was presented in Vessey, Jarvenpaa, and Tractinsky (1992). Using

¹ Structured analysis was chosen to illustrate an application of the framework due to its popularity in the professional world. However, any technique, methodology (including a custom methodology), or subset thereof can be used. Similarly, the framework can be applied to any number of CASE tools.

terminology taken from the decision support system literature, the authors described CASE methodology support as being restrictive, guided, or flexible. A restrictive CASE tool is described as being “designed to encourage the user to use it in a normative manner” (p. 92). A guided CASE tool is described as being “designed to encourage, but not to enforce, the user to use it in a normative way” (p. 92). A flexible CASE tool is described as being “designed to allow the user complete freedom in using it” (p. 92). After defining their framework, Vessey et al. (1992) apply it to twelve PC-based CASE tools. Each CASE tool was examined and results were reported with respect to a set of seventeen methodology rules for data flow diagrams identified from systems analysis and design texts. CASE tools were labeled as “restrictive,” “guided,” or “flexible” based upon the number of rules implemented in a restrictive fashion.

The framework described in Vessey et al. (1992) was a positive step toward classifying CASE methodology support; however, there are portions of their framework that can be refined. First, the three levels of support defined by the authors leave out several possible means of providing methodology support. For example, many CASE tools allow the user to perform methodology checks on request while creating a product (e.g., a data flow diagram). The Vessey et al. framework does not consider this frequently encountered means of providing methodology support. Second, their differentiation between checks made after exiting a “technique” versus exiting a “phase” (in either case the user no longer has direct access to the product) and displaying a violation as an “error” versus a “warning” (in either case the user is left with the decision to accept or ignore the support of the CASE tool) is primarily semantic.

The application of the Vessey et al. framework can also be refined. First, their classification of CASE tools is based upon counting data (specifically, the number of rules enforced in a restrictive fashion); i.e., Tool A is considered to be “more restrictive” than Tool B simply because it has more restrictive methodology rules in its rule base. This assumption fails to take into account the fact that there may be a particular rule, or subset of rules, that contribute more to the consistency and quality of the specifications

than any other rule(s). While Tool B may have fewer rules in its rule base than Tool A, Tool B may have rules in its rule base that are not found in the rule base of Tool A. The authors acknowledge this potential shortcoming by indicating that their three categories may not accurately describe CASE methodology support.

Second, labeling a CASE tool as “restrictive” may cause a prospective user to lose sight of the fact that CASE tools do not implement every rule in their rule base in the same fashion. For a particular CASE tool, some of the rules may be enforced in a “restrictive” manner, some may be enforced in a “guided” manner, some may be enforced in a “flexible” manner, and some may not be enforced at all. For example, Vessey et al. label Visible Analyst Workbench 1.8 as “restrictive” even though only 11 of the 17 identified rules are enforced by the tool. Of these 11 rules, only three are identified as being enforced in a restrictive fashion. As will be discussed in the next section, it is not feasible to enforce all rules in a “restrictive” manner; therefore, it may not be prudent to label a CASE tool as being “restrictive,” “guided,” or “flexible.”

In order to refine the Vessey et al. (1992) framework, components of feedback have been identified in the computer-aided instruction (CAI) literature and applied to CASE tools. Feedback is defined by Kowitz and Smith (1985) to be “a message . . . which is evaluative and intended to improve the functioning of a system” (p. 4).

The CAI literature identifies four components, which, taken together, define computer feedback for a process or product: 1) immediacy, which indicates *when* the feedback is provided (Steinberg, 1991); 2) solicitation, which indicates *how* the feedback is received (Bereiter & Scardamalia, 1987); 3) content, which indicates *why* the feedback is being provided (Wager & Wager, 1985); and 4) response, which indicates *what* the required reaction to the feedback must be (Sassenrath, 1975). In the following subsection, these components of feedback are examined within the context of CASE methodology support.

Restrictive Feedback

Restrictive methodology feedback can be defined as any feedback that requires the user to adhere to the rules of a chosen methodology.

The *immediacy* of the feedback refers to when a rule violation is presented. Methodology feedback that is restrictive will be presented as soon as is feasible to do so in order to keep any methodology violation from propagating through the system specification. This implies that rule violations must be detectable while in the process of performing a specific task, such as data flow diagramming (level 1 restriction). Other rule violations may not be detectable until a task is finished (e.g., while saving a DFD and/or exiting the diagramming tool). This second level of restrictive feedback (level 2 restriction) is necessary in order to prevent the user from being interrupted by mistaken violations that are actually attributable to work in progress.

The *solicitation* of a rule refers to the mechanism by which the rule violation is presented. Feedback that is implemented in a restrictive fashion by a CASE tool will automatically present itself as soon as the CASE tool detects a violation. With the goal of restrictive feedback being to force the user to conform to the rules of a particular methodology, it is important that the *content* of the feedback be context sensitive.

The *response* refers to the set of options available once the CASE tool has presented the feedback. Restrictive feedback will require the user to address the feedback and correct the violation before proceeding.

In summary, rule feedback will be considered to be implemented within the CASE tool in a restrictive fashion if the user is automatically presented with context sensitive feedback while using an operator, or while terminating use of an operator, and is forced to address the feedback before proceeding.

Guided Feedback

The second type of methodology feedback available from a CASE tool is guided feedback. Guided feedback can be defined as any feedback that guides the user in executing a systems development methodology by assisting the user in using its methods. A CASE tool's feedback may guide the execution of the systems development process by providing the user with suggestions and information regarding the procedures of a particular systems development activity as well as the resultant product of that activity.

Two types of guided feedback can be made available by a CASE tool: active guidance and passive guidance. Active guidance is informative and suggestive advice that is unsolicited, i.e., the CASE tool delivers the feedback when it detects a need for guidance. Active guidance can be provided by the CASE tool while the user is performing a particular task (level 1 active guidance) or it may be provided by the CASE tool when a task is complete (e.g., while saving and/or exiting) (level 2 active guidance). The feedback may be pre-sented in the form of an error notification and/or suggestion for correcting the violation. It is then left to the discretion of the user to determine whether or not to correct the violation.

The second type of guided feedback provided by a CASE tool is passive guidance. Passive guidance is informative and suggestive advice that is solicited from the CASE tool by the user. Passive guidance may be requested while performing a task (level 1 passive guidance) or it may be implemented as a separate function outside of the task (level 2 passive guidance). As with active guidance, feedback may be presented in the form of error notifications and/or suggestions for correcting the violations.

Flexible Feedback

Finally, an alternative to embedding restrictive and guided feedback within a CASE tool is

flexible feedback or the complete lack of support for a methodology or a particular methodology rule. The methodology feedback available from a CASE tool is summarized in Table 1.

Feasibility of Case Support for Structured Analysis

By adopting a particular systems development methodology and mandating its use, an organization is expecting the products of the systems development activities to conform to the chosen methodology. To achieve the objective of methodology prescription, an organization, in theory, should be able to purchase a CASE tool that supports the chosen methodology. However, the concept of methodology support can be handled differently from one CASE tool to another. Further complicating the issue is the fact that some methodology rules cannot, for practical reasons, be implemented in a restrictive fashion.

When discussing the enforcement of methodology rules via restrictive feedback, the conflict between an actual methodology violation and work in progress must be addressed. The CASE tool should not interrupt the work to report a violation if the suspected violation may be a symptom of unfinished work. To account for this, a rule violation that may be the result of unfinished work should be handled in one of two ways: 1) the violation can be automatically

	Immediacy	Solicitation	Content	Response
Level 1 Restriction	Creation	Automatic	Context Sensitive	Mandatory
Level 2 Restriction	Exit/Save	Automatic	Context Sensitive	Mandatory
Level 1 Active Guidance	Creation	Automatic	Context Sensitive or Simple Notification	Override
Level 2 Active Guidance	Exit/Save	Automatic	Context Sensitive or Simple Notification	Override
Level 1 Passive Guidance	Creation	Request	Context Sensitive or Simple Notification	Override
Level 2 Passive Guidance	Post-Method	Request	Context Sensitive or Simple Notification	Override

Table 1. CASE Methodology Support Framework

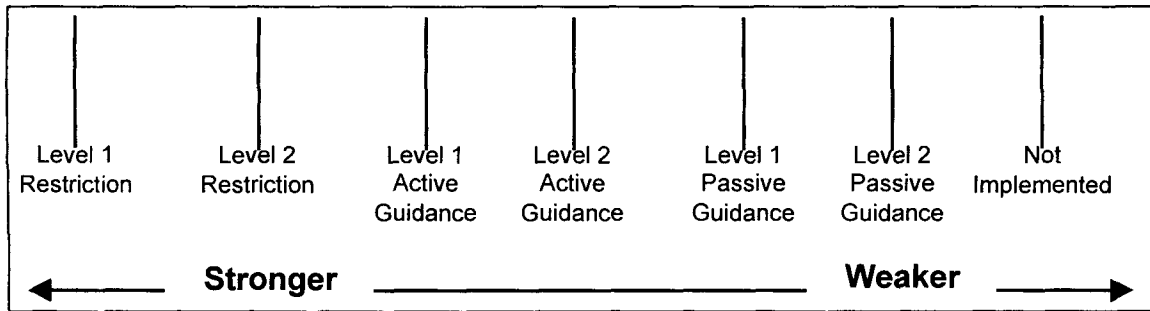


Figure 1. Spectrum of CASE Methodology Enforcement

presented when saving a diagram or exiting the diagramming tool (level 2 active guidance), or 2) the violation is presented upon the request of the user (level 1 or level 2 passive guidance). In either of the above methods, the user is not bothered with an erroneous interruption while drawing the diagram. Notification of violations via active guidance would have the probable effect of identifying possible inconsistencies earlier than would passive guidance; however, active guidance might also prove to be a nuisance by forcing the user, every time a diagram is saved, to read through a list of violations that can be attributed to unfinished work.

The remainder of this section examines the feasibility and practicality of automating 28 rules of structured analysis. The rules (presented in Appendix A) were compiled from several popular systems analysis and design textbooks, one being, Edward Yourdon's *Modern Structured Analysis* (1989). The feasibility of automating these 28 structured analysis methodology rules (presented with the rules in Appendix A) is reported with respect to the spectrum shown in Figure 1. The spectrum displays methodology

enforcement is possible for all of the 28 rules investigated in this study.

Application of the Framework

Vessey et al.'s (1992) evaluation included two popular CASE tools, Intersolv's *Excelerator* (Version 1.8) and Visible Systems's *Visible Analyst Workbench (VAW)* (Version 3.0). The authors categorized *Excelerator* as one of the two most flexible CASE tools in their study and *VAW* as one of the two most restrictive CASE tools in their study. Utilizing more recent versions of these CASE tools — *Excelerator* 1.9 and *Visible Analyst Workbench* 3.1 — and the 28 rules presented in the Appendix A (Vessey et al. report their findings based on 17 methodology rules), the framework presented in the previous section has been applied to the two CASE tools and is summarized in Table 3.² Methodology enforcement for the tools was initially determined through the vendor-supplied documentation. The author, through personal use of the tools, then verified the documentation.

From Table 3, it can be seen that the two CASE tools are quite different in their approach to enforcing the structured analysis methodology

Rule #	During Creation			During Exit/Save		Post-Method	Not Feasible
	Automatic		On Request	Automatic		On Request	
	Mandatory	Override		Mandatory	Override		
1							
2							
3							
4							
5*							
6							
7							
8							
9							
10							
11							
12*							
13							
14							
15							
16							
17*							
18							
19							
20*							
21							
22							
23							
24*							
25*							
26*							
27							
28							
	Level 1 Restriction	Level 1 Active Guidance	Level 1 Passive Guidance	Level 2 Restriction	Level 2 Active Guidance	Level 2 Passive Guidance	Not Feasible

* See Section 3, paragraph 2, for a discussion of rules with multiple levels of enforcement feasibility.

Table 2. CASE Tool Methodology Enforcement Feasibility

Rule #	During Creation			During Exit/Save		Post-Method	Not Implemented
	Automatic		On Request	Automatic		On Request	
	Mandatory	Override		Mandatory	Override		
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12*							
13							
14							
15							
16							
17*							
18							
19							
20							
21							
22							
23							
24**							
25							
26							
27							
28							

* Exceclator 1.9 may not detect special cases of this violation. **Exceclator 1.9 does not support slit data flows




 VAW 3.1
 Exceclator 1.9
 Both CASE Tools

Table 3. Visible Analyst Workbench 3.1 and Exceclator 1.9 Methodology Enforcement

It can also be noted from Table 3 that it is inappropriate to apply a label such as "restrictive" or "guided" to either of these CASE tools. VAW implements only 11% (3 of 28) of the rules in a restrictive fashion, 57% (16 of 28) are implemented in a guided fashion (level 1 active and level 1 passive guidance), while the remaining 32% (9 of 28) of the rules are not implemented. Excelerator implements 64% (18 of 28) of the rules in a guided fashion (level 2 passive guidance), while the remaining 36% (10 of 28) of the rules are not implemented. While both CASE tools claim to support Yourdon structured analysis it is clear that their respective implementations of the methodology are quite different.

Research Results

Since the advent of CASE, the idea that systems analysts will be more productive and systems development deliverables will be of better quality has been debated. Several case studies have been published that investigate such factors as the acquisition of CASE, the acceptance of CASE, and the influence CASE may have on maintaining systems. However, there has been virtually no published work, especially experimental, that investigates the influence of CASE methodology support on the output of systems development activities. The author uses the framework presented in the previous section to examine the influence of CASE methodology feedback on the output of structured analysis (1994).

Sixteen project teams used Excelerator 1.9 and VAW 3.1 (eight project teams per tool) to create a functional specification for a hotel information system. The subjects were undergraduate (seniors) management information systems majors. Prior to beginning the study, the subjects received classroom training in structured analysis and performed several exercises requiring them to use structured analysis. The subjects next received classroom and laboratory instruction for their assigned CASE tool. The subjects, using their assigned CASE tool, replicated structured analysis exercises that had originally been performed with pencil and paper.

After having received methodology and CASE training, the subjects received a requirements specification for a hotel reservation system. The

specification was taken from JAD training materials used by a major information systems consulting organization. The subjects were required to use their assigned CASE tool in support of structured analysis to create a functional specification, consisting of data flow diagrams, a data dictionary, and primitive process specifications, for the reservation system. Subjects were motivated to perform the task by the fact that the functional specification was the major component of a two-month term project for their systems development course.

At the conclusion of the project, the number of violations of each of the 28 methodology rules was determined from the specifications. It was hypothesized that the number of violations of a particular rule would be dependent upon the type of feedback provided for that rule, i.e., a rule with restrictive feedback would be violated less frequently than a rule with guided feedback. Results of the ANOVA tests for the 28 methodology rules are presented in Table 4. The principal research results are:

- a) Regardless of the type of feedback provided, rules pertaining to the internal consistency (rules 1-18) of a data flow diagram are rarely violated (for 13 of the 18 rules there were no violations). This result is consistent with previous studies (e.g., Yellen, 1990) which are characterized by small projects that have little chance to examine hierarchical relationships, comparing the use of CASE to draw DFDs versus drawing DFDs by hand. These studies have found no difference between treatments and, in some cases, diagrams drawn by hand (implying no automated methodology support) were "better" than those drawn with the aid of CASE.
- b) Adherence to the hierarchical consistency rules (i.e., the relationship between diagrams or between a diagram and the dictionary; rules 19-28) is more dependent than adherence to the internal consistency rules upon the feedback provided by the CASE tool. Unlike the internal consistency rules, which are applied to only one diagram at a time, the hierarchical consistency rules may not be easily verified by a visual inspection of a single data flow diagram. As the system becomes more complex, the number of "links" between the diagram levels grows

Rule #	Predicted Number of Violations	Actual (mean) Number of Violations		Significance*
		VAW	Excelerator	
1	VAW = Excel.	0.00	0.00	
2	VAW > Excel.	0.00	0.00	
3	VAW = Excel.	0.00	0.13	.3343
4	VAW < Excel.	0.00	0.00	
5	VAW < Excel.	0.00	0.00	
6	VAW = Excel.	0.00	0.00	
7	VAW < Excel.	0.00	0.75	.0004
8	VAW < Excel.	0.13	0.63	.1080
9	VAW < Excel.	0.00	0.00	
10	VAW < Excel.	0.00	0.00	
11	VAW < Excel.	0.00	0.00	
12	VAW < Excel.	0.00	0.00	
13	VAW < Excel.	0.00	0.00	
14	VAW < Excel.	0.00	0.00	
15	VAW > Excel.	5.50	29.75	.0023
16	VAW < Excel.	0.00	0.00	
17	VAW < Excel.	1.50	0.38	.2620
18	VAW < Excel.	0.00	0.00	
19	VAW < Excel.	0.00	1.13	.0230
20	VAW > Excel.	0.75	0.13	.3396
21	VAW < Excel.	0.00	0.50	.3343
22	VAW < Excel.	0.75	10.38	.0181
23	VAW < Excel.	3.13	12.25	.1251
24	VAW < Excel.	0.00	0.00	
25	VAW > Excel.	10.75	20.38	.2850
26	VAW > Excel.	0.00	0.00	
27	VAW > Excel.	16.50	11.75	.1633
28	VAW < Excel.	0.00	7.88	.0905

*Entries in bold indicate a significance at an α of 0.05.

Table 4. Summary of Comparison of Rule Violations Between CASE Tools

exponentially. As the number of levels in a data flow diagram set increase the difference between the number of processes (and data flows into and out of the processes) on level

n and level $n+1$ increases in an amount proportional to x^{n+1} where x is the number of processes on level n . As the size of the domain to be verified increases, the

opportunity for a methodology rule violation to occur also increases, thus necessitating a greater level of methodology assistance from the CASE tool.

- c) There is virtually no difference between providing level 2 passive guidance and no feedback at all (rules 2-15, 20-25, 26, 27)

Because the structure charts are derived directly from the data flow diagrams, it is crucial that the data flows be properly displayed on the data flow diagrams. In particular, when a process is decomposed, its input and output data flows must be propagated to the next diagram level. A failure to propagate the data flows correctly

"detailed structure" may be necessary. However, he cautions that there are also situations when detailed structure may be "inhibiting and frustrating" (p. 12).

Adelson and Soloway (1985) state that the level of support provided for an analyst by a tool should reflect the experience of the analyst with the problem domain and the design technique. Nunamaker, Dennis, Valacich, Vogel, and

mechanisms to ensure methodological consistency of specifications.

References

Adelson, B., and Soloway, E. (1985). "The Role of Domain Experience in Software Design," *IEEE Transactions on Software Engineering*, SE-11, pp. 1351-1360.
Bass, M. P., and Galletta, M. (1987). The

- "Group Support Systems Research: Experience from the Lab and Field," in L. M. Jessup & J. S. Valacich (Eds.), *Group Support Systems*. New York: MacMillan, pp. 125-145.
- Page-Jones, M. (1992). "The CASE Manifesto," *CASE Outlook*. 6(1), pp. 33-42.
- Sassenrath, J. M. (1975). "Theory and Results of Feedback and Retention," *Journal of Educational Psychology*. 67, pp. 894-899.
- Steinhilber, E. B. (1994). "Computer Assisted Instruction," *A Synthesis of Theory, Practice, and Technology*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Sumner, M. (1993). "Factors Influencing the Adoption of CASE," in T. J. Bergin (Ed.), *Computer-Aided Software Engineering: Issues and Trends for the 1990s and Beyond*. Harrisburg, PA: Idea Group, pp. 130-155.
- Vessey, I., Jarvenpaa, S. L., and Tractinsky, N. (1992). "Evaluation of Vendor Products: CASE Tools as Methodology Companions," *Communications of the ACM*. Vol. 35, No. 4, pp. 90-105.
- Wager, W., and Wager, S. (1985). "Presenting Questions, Processing Responses, and Providing Feedback in CAI," *Journal of Instructional Development*. Vol. 8, No. 4, pp. 2-8.
- Yellen, R. E. (1990). "Systems Analysts Performance Using CASE Versus Manual Methods," in J. F. Nunamaker, Jr. (Ed.), *Proceedings of the 23rd Hawaii International Conference on System Sciences*. Los Alamitos, CA: IEEE Computer Society Press, pp. 497-500.
- Yourdon, E. (1989). *Modern Structured Analysis*. Englewood Cliffs, NJ: Yourdon Press.

- Instruction," *A Synthesis of Theory, Practice, and Technology*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Sumner, M. (1993). "Factors Influencing the Adoption of CASE," in T. J. Bergin (Ed.), *Computer-Aided Software Engineering: Issues and Trends for the 1990s and Beyond*. Harrisburg, PA: Idea Group, pp. 130-155.
- Vessey, I., Jarvenpaa, S. L., and Tractinsky, N. (1992). "Evaluation of Vendor Products: CASE Tools as Methodology Companions," *Communications of the ACM*. Vol. 35, No. 4, pp. 90-105.
- Wager, W., and Wager, S. (1985). "Presenting Questions, Processing Responses, and Providing Feedback in CAI," *Journal of Instructional Development*. Vol. 8, No. 4, pp. 2-8.
- David Jankowski** is an assistant professor of management information systems at California State University, San Marcos. He received his Ph.D. from the University of Arizona. Prior to his graduate studies, he was a software engineer for a Southern California defense contractor. His research interests include the use of CASE technology in developing information systems, and information systems education. His research has been published in *Journal of Computer Information Systems*, *Journal of Systems Management*, and *Journal of Information Systems Education*. E-mail: doctorj@csusm.edu

Appendix A

Structured Analysis Methodology Rules and Automation Feasibility

Process Rule

1. **A parent process must be specified before a child process.** This is the only methodology rule that enforces the process of top-down design. This rule may be restrictively (Level 1) enforced by a CASE tool by not allowing processes to be linked posthoc; i.e., the only way new processes (with the exception of the context diagram process) may be created is through the decomposition and subsequent refinement of a parent process.

Product Rules — Context Diagram (Internal Consistency)

4. **A context diagram must exist.** This rule can be restrictively (Level 1) enforced by a CASE tool by defining the first data flow diagram to be the context diagram and applying all other context diagram rules to this diagram.
5. **The context diagram must contain only one process.** This rule can be restrictively (Level 1) enforced by a CASE tool by not allowing the user to access a new process symbol if a process already exists on the diagram. However, the requirement that the context diagram must have a process cannot be restrictively enforced in order to account for unfinished work. Active guidance (Level 2) can be provided while saving the diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic internal consistency checks on work in progress.
6. **The context diagram must contain at least one input from an external entity and one output to an external entity.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished work. For example, an unfinished context diagram might contain input from an external entity but no output to an external entity. Active guidance (Level 2) can be provided when saving the diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic internal consistency checks on work in progress.
7. **The context diagram process must be numbered zero (0).** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically numbering the context process for the user when the process is created, and not allowing the user to change the numbering.

Product Rules — Process (Internal Consistency)

8. **A process must have at least one input data flow and one output data flow.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished work. For example, an unfinished diagram might contain a process with an output data flow but no input data flow. Active guidance (Level 2) can be provided when saving the diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic internal consistency checks on work in progress.
9. **A process must be connected to at least one of the following: data store, process, external entity.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished work. For example, immediately after a process is drawn it is freestanding. Active guidance (Level 2) can be provided when the diagram is saved or the diagramming tool is exited. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic internal consistency checks on work in progress.
10. **A process must be labeled.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically prompting the user to enter a label when the process is created and requiring the user to enter a label at the prompt.

Product Rules — External Entity (Internal Consistency)

11. **An external entity must appear for the first time on the context diagram.** This rule can be restrictively (Level 1) enforced by a CASE tool by verifying the name of any external entity placed below the context diagram with a list of names of those external entities appearing on the context diagram. If the external entity is appearing for the first time in the diagram set (but not on the context diagram) the CASE tool can disallow the placement of the external entity.
12. **An external entity must be connected to a process.** This rule has two possible scenarios, each of which requires a different enforcement mechanism. In the first scenario the external entity is free

13. **An external entity must be labeled.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically prompting the user to enter a label when the external entity is created and requiring the user to enter a label at the prompt.

Product Rules — Data Flow (Internal Consistency)

14. **A data flow must be an interface between a process and either a second process, a data store, or an external entity.** This rule may be enforced in a restrictive (Level 1) manner by not allowing a data flow to be drawn as a free standing object. Instead, a data flow should only be created by indicating the two existing objects that the flow is connecting. If one of the objects is not a process the CASE tool can prevent the data flow from being created.
15. **A data flow into a data store must have a composition that is a subset of the data store's composition.** ~~This rule cannot be enforced in a restrictive fashion because doing so will not take into~~

account unfinished work. For example, the user can choose not to explicitly define, via the data dictionary, the composition of the data stores and/or data flows until after the diagram has been drawn. In this case, active guidance (Level 2) or passive guidance (Level 1 or Level 2) can be used to indicate any potential inconsistencies or the existence of an undefined data flow/store. Even if the CASE tool required the user to immediately enter the data dictionary after creating a data store or a data flow, the user must still be allowed to leave the composition definition unfinished.

16. **A data flow must be labeled.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically prompting the user to enter a label when the data flow is created and requiring the user to enter a label at the prompt.

Product Rules — Data Store (Internal Consistency)

17. **A data store can only exist as an interface between two processes.** This rule has two possible scenarios, each of which requires a different enforcement mechanism. In the first scenario the data store is free standing or connected to only one process (and is not connected to the parent process) and is, therefore, in violation of the methodology rule. However, this may be attributed to work in progress rather than an error. Active (Level 2) or passive (Level 1 or Level 2) guidance can be used to detect this situation. In the second scenario, the user attempts to connect a data store to anything but a process. The CASE tool may prohibit this type of connection from being made (Level 1 Restriction).
18. **A data store must be labeled.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically prompting the user to enter a label when the data store is created and requiring the user to enter a label at the prompt.

Product Rules — Data Flow Diagram (Hierarchical Consistency)

19. **A parent process must exist unless it is a context diagram.** This rule can be restrictively (Level 1) enforced by a CASE tool by only allowing a new diagram to be created (except for the context diagram) from a process decomposition resulting in a new (child) diagram level.

Product Rules — Process (Hierarchical Consistency)

20. **A process must decompose to either another data flow diagram or a primitive process specification.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished leveling. Level 1 Active Guidance can be provided by giving the user the option of creating a primitive process specification after creating the process. Level 2 Active Guidance regarding the completeness of the set of data flow diagrams can be provided when saving a diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic completeness checks on work in progress.
21. **A process must be numbered with respect to its parent.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically numbering all processes when they are created and not allowing the user to change the numbering.

Product Rules — Data Flow (Hierarchical Consistency)

22. **An input (output) data flow to a parent process must appear as net input (output) on the child data flow diagram decomposed from the process.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically carrying down all input and output data flows from a parent process to a child diagram when moving between diagram levels and not allowing the net input and net output data flows to be deleted from the child diagram.
23. **A net input (output) data flow on a child data flow diagram must appear on the parent process as input (output).** This rule can be restrictively (Level 1) enforced by a CASE tool by not allowing insertions of net input and net output data flows on a child diagram.
24. **A set of input data flows on a child data flow diagram that were split from a data flow connected to the parent process must match the parent data flow's composition.** This rule cannot be enforced in a restrictive fashion unless the parent data flow has been decomposed to a record or element definition in the data dictionary (see rule 25). Level 1 Active Guidance can be provided by giving the user the option to enter the definition in the data dictionary after splitting the data flow. Level 2 Active Guidance regarding the completeness of the set of data flow diagrams can be provided when saving a diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic completeness checks on work in progress.
25. **A data flow must decompose to either a record definition or an element definition.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished work. Level 1 Active Guidance can be provided by giving the user the option to enter the definition in the data dictionary after creating the data flow. Level 2 Active Guidance regarding the completeness of the set of data flow diagrams can be provided when saving a diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic completeness checks on work in progress.

Product Rules — Data Store (Hierarchical Consistency)

26. **A data store must decompose to either a file definition or a record definition.** This rule cannot be enforced in a restrictive fashion because doing so will not take into account unfinished work. Level 1 Active Guidance can be provided by giving the user the option to enter the definition in the data dictionary after creating the data store. Level 2 Active Guidance regarding the completeness of the set of data flow diagrams can be provided when saving a diagram or exiting the diagramming tool. Passive guidance (Level 1 or Level 2) can be provided if the user does not wish to be distracted with automatic completeness checks on work in progress.

Product Rules — Primitive Process Specifications (Hierarchical Consistency)

27. **All inputs and outputs of a primitive process specification must match those of the corresponding parent process on the data flow diagram.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically carrying down input and output data flows from the parent process to the primitive process specification upon creation of the primitive process specification. Further, the CASE tool should not allow any of the inputs or outputs to be deleted from the primitive process specification nor may any inputs or outputs be added to the primitive process specification.
28. **A primitive process specification must be labeled with the same identifier as the corresponding primitive process on the data flow diagram.** This rule can be restrictively (Level 1) enforced by a CASE tool by automatically labeling the primitive process specification with the corresponding process label and not allowing the user to change the label.