2

ASC-TR-93-5012

Systems Analysis Quality Metrics Algorithms

William R. Williamson
Williamson Consulting
4215 Pennywood Drive
Beavercreek, OH 45430-1843

June 1993

Final Report for Period 29 September 1992 - 31 May 1993

DCS, Development Planning
Aeronautical Systems Center
Air Force Materiel Command
Wright-Patterson AFB, OH 45433-7126

NOTICE

This technical report has been reviewed and is approved for publication.

Ronald S. Vokits
RONALD S. VOKITS
CHIEF, DESIGN,
STUDIES AND ANALYSIS
DEVELOPMENT PLANNING DIRECTORATE

[illegible]
WILLIAM H. DUNGEY
CHIEF, AVIONICS SECTION
DESIGN, STUDIES AND ANALYSIS
DEVELOPMENT PLANNING DIRECTORATE

[illegible]
TONY C. KIM
ELECTRONICS ENGINEER
AVIONICS SECTION
DESIGN, STUDIES AND ANALYSIS
DEVELOPMENT PLANNING DIRECTORATE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1 AGENCY USE ONLY *(Leave blank)*	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	June 1993	Final Report: Sep 92 - May 93

4 TITLE AND SUBTITLE

Systems Analysis Quality Metrics Algorithms

5 FUNDING NUMBERS

6 AUTHOR(S)

William R. Williamson

7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Williamson Consulting
4215 Pennywood Drive
Beavercreek, OH 45430-1843

8 PERFORMING ORGANIZATION REPORT NUMBER

None

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Tony C. Kim (513-255-9696)
ASC/XREDA
Development Planning Directorate
Aeronautical Systems Center, AFMC
Wright-Patterson AFB, OH 45433-7126

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

ASC-TR-93-5012

11 SUPPLEMENTARY NOTES

This is a Small Business Innovation Research (SBIR) Program, Phase I

12A DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12B DISTRIBUTION CODE

13 ABSTRACT *(Maximum 200 words)*

The objective of the Systems Analysis Quality Metrics Algorithms program is to formalize IDEF$_0$ standards of good practice and to develop algorithms for the integration of metric primitives into whole diagram and whole model metrics. Formalizations are recommended for activation definition, data content of arrows, arrow labeling, interpretation of unlabeled arrows, generic interfaces, tunneled arrows, and mechanisms. Metric scoring algorithms are described for syntax, semantics, coupling structure, the "Every box must have a control" rule, cohesion, five balance concepts, and activation disclosure.

14 SUBJECT TERMS

IDEF$_0$	Quality	Algorithms
SADT	Metrics	
Function Modeling	Formalizations	

15 NUMBER OF PAGES

33

16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

This is the Final Report on Contract F33657-92-C-2155, Systems Analysis Quality Metrics Algorithms, a Small Business Innovative Research (SBIR) Phase I contract. The contract was awarded to Williamson Consulting, 4215 Pennywood Drive, Beavercreek, Ohio, 45430, on 29 September 1992. The Principal Investigator was William R. Williamson. The work was performed at the Williamson Consulting facility at the above Beavercreek, OH address.

The sponsor of the research was the USAF Aeronautical Systems Division (ASD), Deputy for Development Planning (XR), Mission Area Planning (XRS) and Advanced Systems Analysis (XRM) Directorates. The XR project manager was Robert Bachert, who also contributed technical expertise to the project.

The research was completed on 29 May, 1993. An SBIR Phase II proposal is under consideration.

The research reported herein establishes feasibility of substantive, machine implementable, IDEF0 quality metrics. Feedback from the IDEF0 community is solicited. Respond to Bill Williamson at the above Williamson Consulting address.

1.0 EXECUTIVE SUMMARY

Contract F33657-92-C-2155, Systems Analysis Quality Metrics Algorithms, was awarded on 29 September, 1992 to Williamson Consulting. The research effort, reported herein, was completed on 29 May, 1993. This was a Phase I Small Business Innovation Research (SBIR) contract.

1.1 Research Objectives and Results

The Phase I Systems Analysis Quality Metrics Algorithms research had three technical objectives.

- Formalize identified IDEF$_0$ standards of good practice.
- Develop algorithms for the integration of metric primitives into whole diagram and whole model metrics.
- Formalize developed training materials into an advanced IDEF$_0$ training course.

The accomplishment of these objectives in the Phase I effort is discussed in the following sections.

1.1.1 IDEF$_0$ Formalizations

The Systems Analysis Quality Metrics[1] research identified seven specific areas in which formalization was required: (1) *activation* definition, (2) data content of arrows (based on the "all of the tokens" concept), (3) arrow labeling requirements, (4) interpretation of unlabeled arrows, (5) generic interfaces, (6) tunneled arrow usage, and (7) mechanism usage.

All of these formalization issues were addressed by the Systems Analysis Quality Metrics Algorithms research. In each case the approach was the same. The author prepared briefing materials covering the formalization issues. These were presented at IDEF Users Group Workshops (October '91, May '92, October '92). The results are a consensus of those who attended the workshops. These are presented in Section 2.0 herein.

In the same time frame, although not a part of the Systems Analysis Quality Metrics Algorithms research, work was in process, by the National Institute of Standards and Technology (NIST), developing an IDEF$_0$ Federal Information Processing Standard (FIPS). A primary purpose of the (any) FIPS is formalization of syntax and semantics. As a part of our Section 2.0 material we freely comment on the version of the developmental FIPS which was available to us.[2]

---

[1] Williamson, W.R., "Systems Analysis Quality Metrics", ASD-TR-92-5005, Adroit Systems, Inc. for DCS, Development Planning, Aeronautical Systems Division, Air Force Systems Command, Wright-Patterson AFB, OH 45433, Dec 1991

[2] "IDEF Users Group, Standards and Specifications Committee, IDEF$_0$ Working Group, IDEF$_0$ FIPS Comment Document", IDEF Users Group, 1900 Founders Drive, Kettering, OH 45420, 5 February, 1993

1.1.2 Algorithm Development

The Systems Analysis Quality Metrics research recommended algorithm development in six areas: (1) Syntax, (2) Semantics, (3) Coupling and Cohesion, (4) Balance, (5) Activation Disclosure, and (6) Total Model Quality.

Scoring algorithms are presented in Sections 3.1 and 3.2 for Syntax and Semantics as defined in the FIPS. The algorithms for the scoring of individual syntax and semantics items are quite trivial. The challenge is to combine the individual scores into meaningful total scores. To date, neither the FIPS nor the original Users Manuals[3,4] support the development of total syntax or semantics metrics.

Coupling and Cohesion algorithms are described in Section 3.3. We developed separate algorithms for Coupling Structure, for the scoring of the "Every Box Must Have a Control" rule, and for Cohesion.

In Section 3.4 we present algorithms for Balance, the best measures of model development consistency. Five balance concepts are discussed.

Section 3.5, Amplification, is the discussion of an Activation Disclosure algorithm.

The Systems Analysis Quality Metrics Algorithms research did not address a Total Model Quality algorithm. This is discussed in Section 5.0, Recommended Future Research.

1.2 References

The key reference material used in the research reported herein are the three documents already noted as references 1 through 4 (References 3 and 4 are identically the same document with slightly different publication formats.). We refer to these documents throughout this report, without further reference notes, as "Systems Analysis Quality Metrics", the "FIPS", and the "Users Manual" (or just "UM").

1.3 Organization of Final Report

The remainder of this report is organized into four sections. Section 2 presents the Formalization research. Section 3 presents the Algorithm Development results. The Training Materials are described in Section 4. Recommended Future Research is discussed in Section 5.

[3] "Integrated Computer-Aided Manufacturing (ICAM) Function Modeling Manual (IDEF0)", UM110231100, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson AFB, OH 45433, June 1981

[4] "Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part II, Volume IV-Function Modeling Manual (IDEF0)," AFWAL-TR-81-4023, Volume IV, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson AFB, OH 45433, June 1981

2.0 FORMALISMS

2.1 Activation Theory

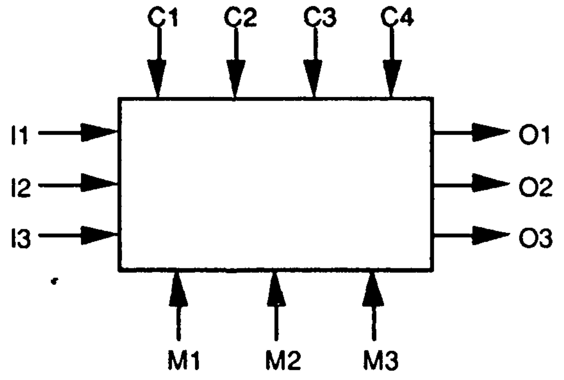
2.1.1 General Philosophy

The UM describes activations. **"A box may perform various parts of its function under different circumstances, using different combinations of its inputs and controls and producing different outputs. These are called the different activations of the box."** Essentially these same words appear in the FIPS. In both instances, the description occurs, in passing, in the context of another discussion.

In fact, activation theory is fundamental to IDEF$_0$ and to its predecessor, SADT. Marca[5] devotes most of Chapter 19 to the subject. But IDEF$_0$ writers have failed to identify activation theory as such, even when they are writing about it. The word "activation" does not appear in the UM Glossary nor in either the FIPS Normative or Informative Definitions. However, consider the following discussion (from Section 4 of the UM and Annex B of the FIPS):

The basic interpretation of the box shown below is: In order to produce any subset of the outputs [01, 02, 03], any subset of the entries [I1, I2, I3, C1, C2, C3, C4, M1, M2, M3] may be required. In the absence of further detailing it cannot be assumed that:

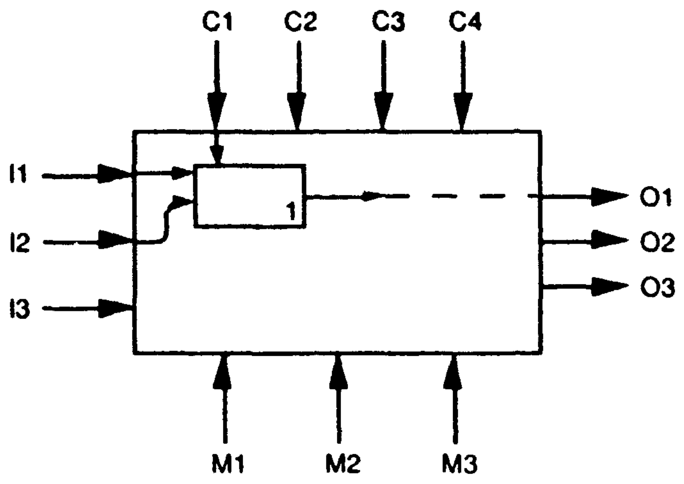
a. any output can be produced without all entries present, or
b. any output requires all entries for its production.



The partial detailing of the box shown below as it might appear in an FEO diagram indicates that I3, C2, C3, and C4 are not required for producing O1. This illustrates the points that:

a. some form of further detailing will specify the exact relationship of inputs and controls to outputs;
b. until that detailing is provided, limiting assumptions about relationships "inside" each box should not be made;
c. reading of a diagram should concentrate on the arrows, which are explicit, rather than on box contents, which are only implicit.

[5] Marca, David A. and McGowan, Clement L., "SADT, Structured Analysis and Design Technique", McGraw-Hill Book Company, New York, 1988



It would be difficult to find a better example of the significance of activation theory. We have a fundamental notion of IDEF0, **"Only that which is explicitly stated is necessarily implied."**, and a corollary, **"Any further detailing not explicitly prohibited is implicitly allowed."** In other words, all activations are possible until eliminated by further explicit detailing.

2.1.2 *Possible Activations*

The number of possible activations of an IDEF0 box can be formulated as,

$$\text{No. of possible activations} = 2^{I} \,.\, (2^{C} - 1) \,.\, (2^{O} - 1) \,.\, (M^*)$$

where the exponents are: I = no. of input arrows, C - no. control arrows, O (alphabetic) = no. of output arrows. Since portrayal of mechanisms is optional in IDEF0, we have.

$$M^* = 1, \text{ if no mechanism arrows are depicted}$$
$$M^* = 2^{M}\text{-}1, \text{ if mechanism arrows are depicted}$$

The exponent M is the number of mechanism arrows. The formulation incorporates a requirement for the presence of an entity in at least one control arrow, at least one output arrow, and at least one mechanism arrow.

2.1.3 *Activation Language*

In IDEF0, further detailing is most commonly provided by decomposition. Although text and/or FEOs could also be used. In SADT another option was available --- an Activation Rules notation.

The notation is quite simple. A box number (node number) followed by an asterisk (*) followed by an Activation Rule number uniquely identifies the rule. The rule is a logical expression of ICOM codes defining a precondition, followed by an arrow, followed by a logical expression of ICOM codes defining a postcondition. The identification expression and the rule are separated by a colon. The logic operators AND, OR, and NOT, along with parentheses, provide the means for expressing arbitrarily complex rules.

A23*1 : I1 and I2 and C1 ⇒ 01 says "Box A23, activation rule number 1 : Input number 1 and Input number 2 and Control number 1 produce Output number 1".

A31 * 2 : I1 and C1 ⇒ 02 and 03 says, "Box A31, activation rule number 2 : Input number 1 and Control number 1 produce Output number 2 and Output number 3."

The Activation Language is most useful at the atomic level of a model, i.e., associated with boxes which are not going to be decomposed. If a box is going to be decomposed, the decomposition will provide insight into the activations. The activation rules would be, at least partially, redundant with the decomposition.

2.1.4 *Applicability to Quality Metrics*

We stated in Systems Analysis Quality Metrics, and reiterate here, that activation theory offers a lot to model quality evaluation. Activations are a measure of complexity. More precisely, activation ambiguity is a measure of complexity. The more (undisclosed) possible activations which exist on a diagram the greater the complexity of the diagram. Indeed, one can make a case that the motivation behind decomposition is to eliminate activation ambiguity.

A model which explicitly reveals all activations is of higher quality than a model which leaves some, or al , activations ambiguous. One way to reveal all activations without excessive further decomposition is to use the Activation Language from SADT.

2.2 Arrow Labeling

2.2.1 *All of the Tokens Concept*

An arrow is assumed to contain everything that is specified by its label. What that might mean in any particular instance is, or course, subject to the interpretation of the label. Ultimately, the content of arrows should be disclosed by their decomposition. This is the underlying concept of Removing the Limitations of Natural Language.[6]

Whatever the content of an arrow, when that arrow constrains a box the box is assumed to require all of the content of the arrow for that constraint. This is the "all of the tokens" concept. This concept provides the rationale for our coupling structure metric.

However, the "all of the tokens" concept is not currently an IDEF$_0$ semantics rule. The concept was the basis for the environmental and record coupling metrics in the UM. So, it existed as a good practice rule. We feel more strongly about the issue. We believe this should be an IDEF$_0$ semantics rule.

A statement of the rule sounds like a labeling requirement. "The label on any arrow constraining a box should be indicative of the content of the arrow which is actually used by the constraint. A more generic label indicative of an arrow content greater than what the box actually uses is inappropriate." Coming up with such a label is seldom easy. But coming up with wonderful labels is not as important as the mind-set

[6] Ross, D. T., "Removing the Limitations of Natural Language", Summary of a lecture presented at the Software Engineering Workshop, Schenectady, NY, 31 May 1979, SofTech internal document No. 9061-25, SofTech, Inc. 460 Totten Pond Rd, Waltham, MA, 02154, July 1979

imposed by the rule. If authors merely think about arrows as all of the tokens required by an activity, then coupling structure problems will be significantly reduced.

2.2.2 *Labeling Requirements*

We noted in Systems Analysis Quality Metrics that defining arrow segments and then simply stating that labels are associated with segments is not generally interpreted by the community as a requirement for a label on every arrow segment. In addition, we didn't care for the attempt to define arrow segments in the first place. So, we recommend two formalization concepts here: a different way of describing arrows, or parts of arrows, which appear on diagrams, and a requirement for labeling arrows, or parts of arrows. The labeling rules embody the arrow description rules.

- Every direct interface arrow, i.e., an arrow from the output side of a box to the input, control or mechanism side of a box on the same diagram, requires a label. (Note: The FIPS defines this as an "Internal Arrow".)

- Every boundary arrow, i.e., an arrow arriving at the boundary of a diagram as an input, control, mechanism (ICOM coded or tunneled) arrow and going to the input control or mechanism side of a box on the diagram, or an arrow from the output side of a box on the diagram and leaving the diagram boundary as an output (ICOM coded or tunneled) arrow, requires a label.

- Every arrow entering a join and the bundled arrow leaving the join require labels.

- The bundled arrow entering a branch and the arrows emanating from the branch require labels.

We recommend making the distinction, described in the following section, between fan-outs and other renderings of arrow branches and joins in order to clearly indicate that there is no requirement for a label on the "apparent segment" between consecutive branches and joins in a fan-out.

It should be noted that the FIPS has adopted the arrow segment definition approach to arrow labeling requirements. The FIPS defines an Arrow Segment as, "**A directed connection between any two of the following: IDEF0 box side, branch (fork or join) or unconnected end (source or use).**" That's close. Technically speaking, "any two", is wrong. An arrow (segment) cannot go from one unconnected end to another unconnected end. Although, we still believe that the attempt to define arrow segments is not the best approach (Its been tried twice now --- once in the UM and again in the FIPS --- and hasn't been precisely correct, yet.), we applaud the FIPS attempt to do it in 25 words, or less (22 actually). The true test of a good method of defining something (anything) is conciseness.

2.2.3 *Interpretation of Unlabeled Arrows*

We observed in Systems Analysis Quality Metrics that conventions for the interpretation of unlabeled arrows emanating from branches and entering joins had been omitted from the UM. The conventions are common knowledge in the IDEF0 community and exist in training materials. The conventions are depicted in Figure 1.

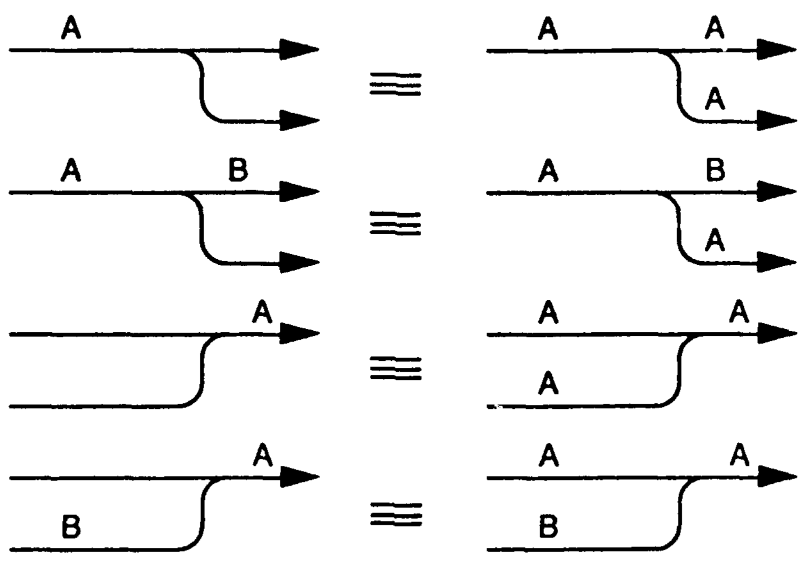


Figure 1. Conventions for Interpretation of Arrow Labels at Branches and Joins

In words, "An unlabeled arrow leaving a branch is assumed to be everything that was part of the arrow entering the branch.", and, "An unlabeled arrow entering a join is assumed to be everything that is part of the arrow leaving the join.". These conventions are pretty much common knowledge in the IDEF0/SADT community and are usually a part of training materials. But, they're not in the UM.

There was an abbreviated attempt to incorporate the conventions into the FIPS. The material is presented in Figure 2.

All or part of the content of an arrow may follow a fork or join. All contents are provided through all branches unless otherwise indicated by a special label on each arrow segment. These conventions are illustrated in Figure 10.

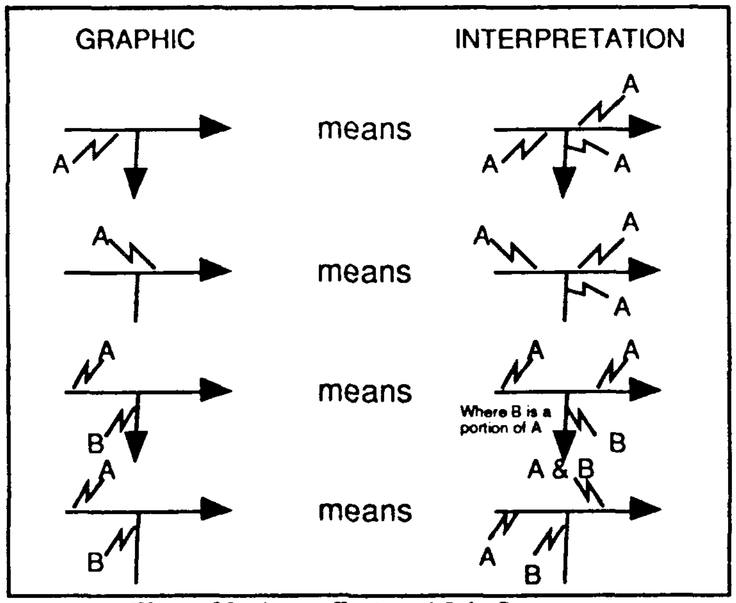


Figure 10. Arrow Fork and Join Structures

Figure 2. Arrow Fork and Join Structures from FIPS

The top item in FIPS Fig. 10 is identical to the top item in Fig. 1. The second item in FIPS Fig. 10 is identical to the third item in Fig. 1. The third item in FIPS Fig. 10 is identical to the second item in Fig. 1. But the fourth item in FIPS Fig. 10 does not correspond to the other item in Fig. 1 nor, in fact, to any SADT writings.

The problem is that the fourth item in FIPS Fig. 10 is not a convention. The fourth item in FIPS Fig. 10 is a definition --- any time two (or more) arrows join, the resulting bundle is the combination (union --- if you're into set theory terminology) of the joined arrows. The issue, insofar as conventions are concerned, is not the interpretation of the bundled arrow. The bundled arrow is always the combination of the joined arrows, by definition. The issue is whether, or not, the label of the bundle is known, "by convention". The answer to that question is "No!". It is important that a FIPS distinguish between definition and convention. In this particular instance, the distinction was missed.

We find the convention for branches useful. It's not at all uncommon for an arrow to branch into carbon copies of itself and go to several places. It's convenient to not have to explicitly label the branches. We consider the convention for joins to be patently absurd. Why would you want to add something to something else which already includes what you're adding before you add it? Why would there be two different sources of "all of the tokens" of the same data? The convention appears to exist to handle things that shouldn't happen in the first place. We suspect this is a Guru rule. We've never seen rationale for it.

So, we have no solid rule requiring labels for all branches. And, we have no conventions (in the UM) for the interpretation of unlabeled arrows, We have the conventions in the FIPS. But one was added which isn't really a convention. And, probably more important, is the omission of the convention which was always there, in SADT writings. As if that weren't enough of a problem, it turns out that the conventions have a flaw.

The convention (we'll limit the discussion to branching, for convenience) works fine for an arrow which splits into exactly two parts. But when the arrow splits into three, or more, parts we have the problem depicted in Figure 3. The small segment (per UM segment definitions) between two adjacent branches is an unlabeled segment.

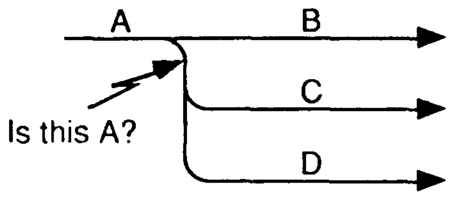


Figure 3. Arrow Segment Ambiguity

Thus, by convention, it is A. That means C and D, in combination, are all of A. That, in turn, means B is part of C and D, since B is part of A and the combination of C and D is A. The problem is that this almost never what the author meant to say. He was simply splitting A into three things; B, C, and D. If he intended two consecutive splits into two parts he would have (presumably) drawn in like Figure 4. Is Figure 4 identical, in

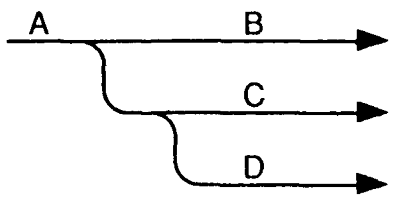


Figure 4 For Comparison with Figure 3

meaning to Figure 3? It's probably debatable. What if it had been drawn like Figure 5? (This works for three branches, but not for four, or more, branches.) Figure 5 has no ambiguity.

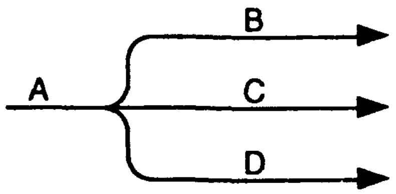


Figure 5. For Comparison with Figures 3 and 4

We have two choices:

- We can dictate that "segments" between consecutive branches do not exist. Figure 3, 4, and 5 mean the same thing --- a "three-branch".
- We can interpret "fan-outs", branches which are aligned perpendicular to the direction of flow of arrows out of the branch (Figure 6), differently from other renderings. Fan-outs are "n-branches" --- the ambiguous segments don't exist. Other renderings are subject to the application of the conventions of Figure 1.

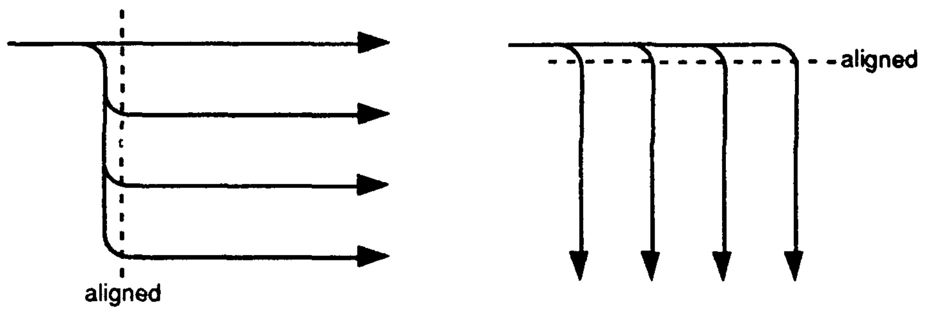


Figure 6. Fan-Outs

So we have a requirement for formalization. We need to: 1) Define unambiguous conventions for the interpretation of unlabeled arrows, 2) Incorporate the conventions into the FIPS, and 3) Make an unambiguous statement in the FIPS regarding the requirements for labeling arrows.

The metric currently treats fan-outs as n-branches and other renderings as subject to the application of the conventions.

2.3 Generic Constraints

Generic constraints occur when an arrow branches into carbon copies of itself and goes to several boxes. This is generally undesirable. The All of the Tokens discussion in Section 2.2.1 addresses the generic interface problem indirectly. The problem can also be addressed by specific syntax rules.

The general form of a syntax rule limiting generic constraints would be something like, "An arrow cannot branch into carbon copies of itself and go to more than N boxes." Picking N is not trivial. And, it should be based on some substantive rationale rather than be propagated as a Guru rule. We believe N is probably different for inputs, controls, and mechanisms. We've already expressed our opinion relative to controls ---

every box should have at least one control for which N=1. We believe N may be different depending upon where a branch occurs in the hierarchy. N should refer to the ultimate number of boxes to which the arrow goes in the hierarchical decomposition, not a "per diagram" number.

Our preference is to try the "all the tokens" concept and see if it solves the generic interfaces problem. Then, if a need still exists, consider a syntax rule.

2.4 Tunneled Arrow Usage

Tunneled arrows can be used for several different purposes. There are no published rules for the correct application of tunneled arrows.

The term "tunneled arrow" is applied to two unrelated concepts. The only relationship is that both are depicted syntactically by parentheses. Arrows which are parenthesized at the edge of a diagram arrive from, or depart to, the outside of the model without having appeared on higher level diagrams. This fits one's intuitive concept of arriving or departing "through a tunnel". Arrows which are parenthesized where they touch a box are simply excluded from further consideration in the model. That's not a tunneling concept. That's more like a "black hole" concept.

The use of parentheses fits the tunneling concept well, but is simply a handy notation for the black hole concept. In text, a parenthesized remark is something not important enough to feature in the main discussion but considered worth mentioning "in passing". That's what happens with a tunneled arrow --- it wasn't important enough to bring down through the hierarchical decomposition but is worth mentioning on a particular diagram. One could make a strong case for parenthesizing both ends of an arrow to indicate that this in the only diagram upon which it appears. The other application is just the opposite. The arrow has been featured until now, but we have no interest in discussing it further. That's not a parenthetical remark concept.

Since there are no rules for the application of tunneled arrows, there is no basis for metrics. We offer some opinions.

- Inputs should not be tunneled into a model.
- Controls may be tunneled into a model, but cannot be utilized as the sole control on a box.
- Byproducts of an activity may be tunneled out of a model, but cannot be the sole output of the activity.
- Mechanisms may be tunneled into a model but should not be the sole mechanism on a box (even though other mechanisms may be implicit rather than explicit due to current methodology treatment of mechanisms).
- The black hole concept should not be used except when the arrow goes to all lower level boxes. The motivation is clutter reduction.
- Tunneling should never be used to transfer data between two boxes in the same model. This is pathological coupling.

2.5 Mechanism Usage

A mechanism shows how an activity is performed. Every activity requires a mechanism. The activity can't be performed without it. But, we don't have to show mechanisms on our diagram unless it suits us to do so --- unless it's "part of our story".

Is the same argument valid for controls? Have you ever wanted to decompose a system from the "how" viewpoint, and wished you could omit controls as "not part of your story"? We're not advocating that. Just wanted to "run it up the flagpole ...". It's not absurd in an enterprise model where we frequently encounter controls such as "understanding", "experience", "training" and where we're trying to depict "who's responsible" or whether a function should be "manual" or "automated", i.e., mechanism issues.

In any event, we need some formalization on mechanism usage. Mechanism usage, like branching and joining arrows, is "allowed". But, we have no rules for when it's ok, or not ok, or recommended, or required. We don't know whether mechanism cohesion is stronger, weaker, or about the same, as control or input cohesion.

We offer no specific recommendations. Our cohesion metric makes no distinction between control, input and mechanism direct interfaces.

3.0 ALGORITHM DEVELOPMENT

3.1 Syntax Rules

The IDEF0 syntax rules, as published in the FIPS (Section 3.2.1.3, pg. 12) are:

Boxes

1. **Boxes must be sufficient in size to insert box name.**
2. **Boxes must be rectangular in shape.**
3. **Boxes must have square corners.**
4. **Boxes must be drawn with solid lines.**

Arrows

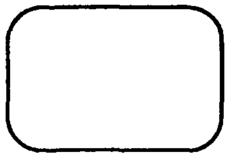
1. **Arrows that bend must be curved at 90 degree arcs.**
2. **Arrows must be drawn in solid line segments**
3. **Arrows must be drawn vertically or horizontally but never diagonally**
4. **Arrows must touch the outer perimeter of the function box and must not cross into the box.**
5. **Arrows must attach at box sides, never at corners.**

We'll address each item individually.

In addition to being large enough to contain the box name, i.e., verb phrase, we believe it should be further stated that all of the boxes on a given diagram should be the same size. In other words, all of the boxes on a diagram are large enough to contain the largest box name on the diagram. In any event, we have a trivial scoring scheme. Every box on the diagram which is the "right" size scores a point. The diagram score is the number of points divided by the number of boxes on the diagram. If all of the boxes are the right size, the score is 100%.

The rectangular shape requirement for a box is also trivially scored. Boxes with the required rectangular shape score a point. The diagram score is the number of points divided by the number of boxes on the diagram. If all of the boxes are rectangular the score is 100%.

Within reasonable limits we are soft on the issue of square corners. We've seen many diagrams with boxes drawn as:



We don't find that objectionable. The important point is that the boxes still qualify as rectangular and that they're all drawn the same. A serious violation would occur only if the author was using different shapes to depict different messages. The scoring algorithm would be the same as for box size and for rectangular shape. Boxes drawn "right" get a point. The score is points divided by boxes.

Drawing boxes with solid lines again leads to the same trivial scoring scheme. Boxes drawn with solid lines get a point. The score is points divided by boxes.

It is clear that when we're talking about syntax relative to boxes, scoring of individual syntax items is trivial. We're simply dividing the number of times the syntactic item was done correctly by the number of opportunities to do it. The number of opportunities to correctly apply a box syntax item is equal to the number of boxes on the diagram. The same concept can be applied to arrows. But, the number of opportunities can vary.

For 90 degree arrow bends we score a point for each 90 degree bend and divide by the total number of bends on the diagram.

Arrows drawn with solid line segments are divided by the number of arrows on the diagram.

Arrows drawn vertically and horizontally are divided by the number of arrows on the diagram.

Arrow ends which touch the sides of boxes correctly without crossing into the box are divided by the number of arrow ends which are supposed to touch boxes on the diagram. Note that we're not counting whole arrows here. We're counting ends of arrows (sources and sinks, or arrowheads and tails).

Arrow ends which attach at sides of boxes, rather than at corners, are divided by arrow ends which attach to boxes.

We see that scoring individual syntax items is trivial. The real issue is that of combining the individual scores into whole diagram or whole model scores. Averaging, or weighted averaging ---since we may not consider all syntactic items to be equally important, would be the logical thing to do. Although, it is far beyond the scope of this research effort to specify the "right" weights. A much greater issue is deciding what qualifies as "all" of the syntax rules. You can't do a weighted average until you identify all of the members of the set to be weighted. The nine items discussed above fall far short of being all of the IDEF$_0$ syntax rules. We will elaborate further after discussing semantics.

3.2 Semantics Rules

The IDEF$_0$ semantic rules, as published in the FIPS (Section 3.2.2.3, p. 15) are:

<u>Boxes and Arrows</u>

1. A box must be named with a verb or verb phrase
2. Each side of a function box must have a standard box/arrow relationship
   a) Input arrows must interface with the left side of a box.
   b) Control arrows must interface with the top side of a box.
   c) Output arrows must interface with the right side of the box.
   d) Mechanism arrows must point upward and must connect to the bottom side of the box.
   e) Call arrows must point downward, must connect to the bottom side of the box and must be labeled with the reference expression for the box which details the subject box.
3. Arrows, except for call arrows, must be labeled with a noun or noun phrase
4. A "squiggle" must be used to link an arrow with its associated label, unless the arrow label relationship is obvious.
5. Arrow labels must not use the following terms: function, input, control, output, mechanism, or call.
6. An arrow may be composed of multiple arrow segments.

We're not sure what to make of the <u>Boxes and Arrows</u> title. We already have the title "Semantic Rules". There is no other subtitle. We'll discuss each rule separately.

We consider the naming of a box with a verb phrase to be syntax, not semantics. In any event, the metric is the same as all of the metrics we've introduced so far. If the box is named with a verb phrase, it gets a point. The diagram score is the number of points divided by the number of boxes on the diagram.

The arrow and side-of-the-box relationships are semantics, as advertised. However, we have a classical "chicken and egg" paradox relative to the application of a metric. Notwithstanding the possibility that an expert reviewer might disagree with the opinion of the author regarding whether an arrow should be an Input, Control, or Mechanism (Note that there is no basis for disagreement relative to Output or Call arrows.) the simple fact of the matter is that arrow roles are defined by the side of the box with which they interface. We have no way of knowing whether the author placed an arrow on the wrong side of a box (other than our disagreement with the arrow role --- a matter of opinion) unless the text material specifically says (for example) that an arrow on the top or bottom of a box is an "input" arrow. There are some purely syntactical items. Arrows touching the top or left side of a box must be incoming arrows. Arrows touching the right side of a box must be outgoing arrows. Arrows touching the bottom of a box may be incoming or outgoing: If they interface with another box or have an ICOM code or tunneled source, then they must be incoming (mechanisms); If they just have a model/node number label, then they must be outgoing (calls). These "rules" are purely syntax and could be scored the same way we've scored everything else --- correct renditions divided by opportunities.

Arrow labeling metrics are far more complicated than the simple issue of whether the label is a noun, or noun phrase. There is the issue of the labeling of every arrow segment (as distinct from every arrow). There is the issue of whether or not unlabeled segments can be interpreted, by convention. There is the issue of inadequate proximity of labels, or failure to use a "squiggle", rendering it difficult, or impossible, to tell which label goes with which segment. Basically, we want to ask two questions about every arrow segment: 1. Is the content of the arrow segment interpretable?, and 2. Is the label which provides that interpretation a noun phrase? If the answer to both questions is yes, then the segment gets a point. As usual, the diagram score is the number of points divided by the number of arrow segments. (See, also, Section 2.2 for more discussion of arrow labeling issues.)

Why isn't there a rule in the FIPS for labeling call arrows?

We've already addressed the problem of inadequate proximity of labels to arrow segments or failure to use "squiggles". A machine implementable algorithm for assessing proximity of labels to arrow segments is straightforward.

Determination of the presence of an unallowable word in a label is straightforward and easily scored.

The sixth item, "An arrow may be composed of multiple arrow segments." is true. So what? What's the rule? How do you break it? What do we want to measure? Its a simple statement of fact --- an observation ---- not a rule.

We noted after our syntax discussion that scoring items individually was trivial. But, combining individual scores into whole diagram and whole model scores requires a concept of what qualifies as "all" of the rules. We noted that the nine items listed as Syntax Rules didn't qualify as all of the rules. Depending upon how you count, we added at least four more from the list of Semantic Rules. If you continue in the FIPS (Section 3.3.4, pp. 31-33) you find another list titled "Diagram Layout: Rules" with the subtitle "Diagram/Box/Arrow Syntax Rules":

Diagram/Box/Arrow Syntax Rules

1. **Context diagrams are diagrams that have node numbers A-n, where n is greater than or equal to zero. Non-context diagrams are diagrams that have other node numbers.**

2. The lowest level context diagram contains only one box. (The highest level context diagram is the diagram with node number A-n (A minus n), with the largest n.)

3. The box number of the single box on the lowest level context diagram is 0.

4. A non-context diagram must have at least three boxes and no more than six boxes.

5. Each box on a non-context diagram must be numbered in its lower right hand inner corner, in order (from upper left to lower right) from "1" to at most "6".

6. Each box that has further decomposition is required to have the node number (or C-number or page number) of the child diagram which details it written beneath the lower right hand corner of the box.

7. Each box must have a minimum of one control arrow and one output arrow.

8. A box may have zero or more input arrows.

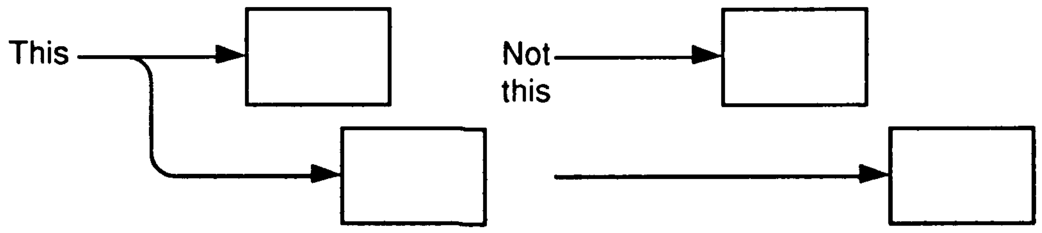
9. A box may have zero or more mechanism arrows.

10. A box may have zero or one call arrow. (Most boxes do not have a call arrow.)

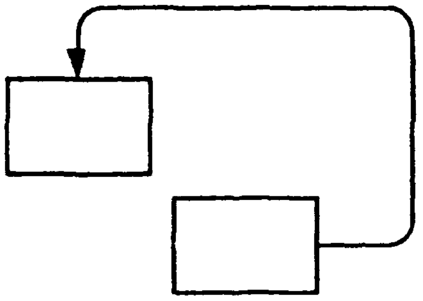
11. Arrows must be drawn as horizontal and vertical straight line segments, diagonal line segments are not used.

12. The unconnected end of a boundary arrow shall have the proper ICOM code specifying its connection to the parent box or shall be tunneled.

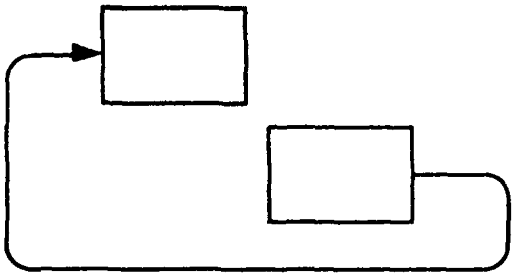
13. Open-ended boundary arrows must be connected to show all the places affected, unless this results in an unreadable diagram. A boundary arrow shall fork to any multiple uses. Any multiple sources shall join to form an output boundary arrow.



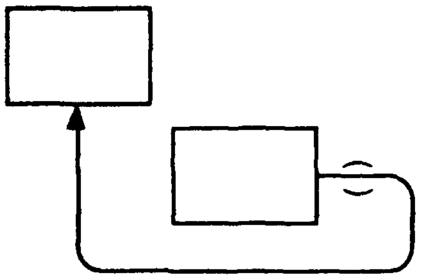
14. **Control feedbacks shall be shown as "up and over".**



Input feedbacks shall be shown as "down and under".



Mechanism feedbacks shall be shown as "down and under".



15. **An output arrow that connects to a mechanism shall be shown tunneled at its source, to indicate the entire box inside is offered as support through the resulting mechanism support arrows.**

16. **Box names and arrow labels shall not use the following words: function, activity, process, input, output, control, or mechanism.**

We already had Syntax Rules and Semantic Rules. So, what are these? A mixture of several kinds of rules and guidelines, some of which are merely recommended good practice, but including some more very important syntax rules (like the requirement for every box to have a control). We have chosen not to discuss these individually. All, except number 15, were discussed in Systems Analysis Quality Metrics. Rule 15 is the invention of the FIPS authors. It was never a part of IDEF0 nor SADT. It should be deleted from the FIPS pending IDEF community approval (which it would probably fail to get).

We felt that the UM did a poor job of stating the IDEF0 rules. We feel that the FIPS has taken a step backwards by stating the rules less well than the UM. However, the UM sat dormant for over twelve years without modification. The FIPS should not have that problem. It should eventually evolve into a quality document because the mechanism for updates exists. The UM had no such mechanism.

3.3 Coupling and Cohesion

Algorithms are required for the assessment of coupling structure, cohesion, and the rule that "Every box must have a control.". In addition to separate algorithms for these factors, a combined algorithm is required to capture intimate relationships between the factors. For example, some coupling and cohesion measures involve trade-offs between good coupling and good cohesion.

3.3.1 Coupling Structure

In Systems Analysis Quality Metrics we discussed coupling from three perspectives: pathological or content coupling; control coupling; and coupling structure. We considered these to be distinct viewpoints on coupling leading to distinct approaches for evaluation or measurement. Only the coupling structure viewpoint leads naturally to a quantitative metric.

The distinction between various coupling structure characterizations: environmental or common coupling, external coupling, and record or stamp coupling; involves interpretation of the vagueness, or globalness, of arrow labels. However, we believe a valid metric exists which ignores those distinctions. Our premise is that how global a data item is correlates, highly, with how many boxes it goes to.

A coupling structure problem exists, potentially, whenever an arrow splits into carbon copies of itself and goes to several boxes. The occasional occurrence of arrows splitting into a couple of branches, and going to a couple of boxes, is not a severe problem. In fact, many practitioners would argue that that's not a problem at all. Frequent occurrences is likely a problem. A large number of branches, per occurrence, is undoubtedly a problem.

We propose a metric algorithm which counts the number of occurrences and the number of branches per occurrence of arrows which split into carbon copies of themselves. We simply refer to this as a Coupling Structure Algorithm, without alluding to characterizations such as environmental, record, or external.

A fairly simple algorithm would count the number of arrow sinks on a diagram. Then the algorithm would trace back each arrow to see if the arrow came from a branch (on the same diagram). If the arrow came from a branch it would be counted as an arrow with a coupling structure problem. Only unlabeled arrows are considered to be part of the branch insofar as this metric is concerned. An arrow which splits into four arrows, two of which have unique labels, is only considered to be a split into two arrows, the two unlabeled arrows, for the purpose of this metric. In addition, for each occurrence of a branch, one arrow is considered to be legitimate. In other words, an arrow which splits into two carbon copies of itself produces only one "problem" arrow (no need to specify which one), an arrow which splits into three carbon copies of itself produces two "problem" arrows, and so on. This caters to the occurrences vs branches per occurrence consideration. Two occurrences of a split into two arrows (producing four arrows) produces two "problem" arrows. One occurrence of a split into four arrows (again, producing four arrows) produces three "problem" arrows. The "score" for the diagram would be: the number of sinks minus the number of problems, divided by the number of sinks. The idea being a score of 100% if there are no problems, 90% if ten percent of the arrows have problems, and so on.

Note that the fan-out (and fan-in) interpretation at branches and joins (Section 2.2.3) must be applied to get a realistic count of sinks for the purpose of this algorithm.

Note also that the metric has a tolerance since some branches are not problems at all. But, we're not interested in whether the author got a score of 93% vs 97%. We're interested in whether the score is 30%, or 60% --- failing scores, or marginal scores.

3.3.2 Every Box Must Have a Control

The measurement of whether or not every box in a model has a control is a syntactic correctness item. We've chosen to discuss it here because of the intimate relationship to coupling structure problems.

Splitting a control arrow into carbon copies of itself and sending it to the top of several boxes is not the way to provide a control on every box.

Our recommended metric is that each box must have at least one unique control arrow. We say "at least one" because the existence of one satisfies the syntax, even if the box has half-a-dozen other controls which possess coupling structure problems. As with the coupling structure metric, we have the "minus one" consideration. An arrow which branches and goes to several boxes provides a legitimate control to one of those boxes (unspecified).

This algorithm is complicated by the fact that applying it to each diagram can give very misleading results. The algorithm should be applied to the atomic level of the whole model.

First count the number of atomic level boxes. That's the maximum number of points that can be awarded. The score is going to be the number of points awarded divided by the number of atomic level boxes. So, as usual, the perfect score is 100%.

Give the model a point for every atomic level box with at least one unique control, i.e., a control arrow not shared with another box.

Deduct a point for every atomic level box with no control at all.

Now direct your attention to the remaining atomic level boxes, i.e., those boxes with only shared controls. Count them. The additional points to be awarded cannot exceed this number. Award a point for each different control arrow shared by these boxes with only shared controls. But, don't award more points than boxes. If there are eight controls shared by twelve boxes, eight points are awarded. If there are twelve controls shared by twelve boxes, twelve points are awarded. If there are fifteen controls shared by twelve boxes, the award is still twelve points --- no more than the number of boxes.

The total points are the number of boxes with at least one unique control (regardless of whether or not there are other controls, unique or shared), minus the number of boxes with no controls, plus the number of shared controls between those boxes with <u>only</u> shared controls. The score is the point total divided by the number of atomic level boxes.

There are two additional considerations. An arriving tunneled control from outside of the model cannot be the unique control on a box (per Section 2.4 discussion). Feedback controls cannot be the unique control on a box.

3.3.3 Cohesion

We discussed Cohesion at length in Systems Analysis Quality Metrics. We noted the seven levels of granularity (eight levels, when you include Mechanism Cohesion) is overkill. We noted that the simple addition of Cohesion primitives to get a whole diagram score was wrong.

As the result of a great deal of discussion at Quality Metrology workshops the past couple of years we are convinced that a very simple algorithm for scoring Cohesion is appropriate. If every box on a diagram has a direct interface with some other box on the diagram, then the Cohesion is 100%. It makes no difference whether the interface is a control, input, or mechanism. Give the diagram a point for every box on the diagram which has a direct interface to some other box on the diagram. You only award one point per box. Either it has a direct interface or it doesn't. A box doesn't get more than one point for having more than one direct interface. The diagram Cohesion score is simply the number of points, i.e., the number of boxes with direct interfaces to other boxes, divided by the number of boxes on the diagram.

We don't consider concepts such as Procedural, Temporal, and Logical Cohesion to be particularly useful.

Communicational Cohesion may be a useful concept, i.e., boxes placed on the same diagram because they use, or make, the same information. However, without a clear definition of what qualifies as "same" information we have chosen not to give Cohesion points for this variety of Cohesion. We would be remiss if we didn't point out that, based on frequency of occurrence, Communicational Cohesion may have to receive a score of some sort at some time in the future. We see a lot of Communicational Cohesion diagrams in models developed by very competent IDEF0 practitioners.

3.4 Balance Algorithms

In Systems Analysis Quality Metrics we indicated that the best measure(s) of consistency in a model were a variety of balance considerations. We identified four balance concepts in that document.

- **Balance between activity detail and arrow detail on a diagram - "A diagram with very general and abstractly-labeled function boxes, *but with minutely detailed* labels on the data arrows is <u>unbalanced</u> semantically".** (From UM discussion of semantic completeness on p. 111).

Doug Ross says, "The boxes, which are supposedly the focal point of the whole diagrammatic exercise, don't need names at all. The reason is that every box is going to be detailed further on the next lower diagram. What it means is what is shown there. Only the lowest level of boxes, the atomic level, needs names. At the higher levels the names are a convenience and an aide memoir, but are logically redundant."[7] If you agree with this Doug Ross statement, then this semantic unbalance in not a problem.

However, the converse situation in which the boxes are well-labeled but the arrows have vague, generic, labels is a problem. Notwithstanding Mr. Ross's comments, we would like to have good labels on both boxes and arrows. Arrows are supposed to be decomposed in the hierarchy, too. We don't just decompose boxes. We have a right to expect the decomposition of a vaguely labeled arrow to disclose its meaning. In fact, this is the embodiment of environmental coupling. That's exactly what environmental coupling is --- a balance problem --- the arrow level of detail is not in balance with the activity level of detail. The boxes have been decomposed. But, the arrows haven't been decomposed!

[7] "Douglas Ross Talks About Structured Analysis", (Interview), Computer Magazine, July 1985, pp. 80-88

- Balance between level of detail on diagrams at the same level of a model - Unbalance occurs, typically, in structured modeling endeavors in which different modelers are developing different parts of the model. One author does a better job of gradually introducing detail than another author. This unbalance is quantifiable, by comparing the number of boxes per diagram and number of arrows per box on diagrams at the same level of decomposition in different parts of the model.

- Balance between levels of decomposition in different parts of a model - Only relatively extreme occurrences of level-of-decomposition unbalance should be considered to be quality shortcomings. After all, one part of the system being modeled may be more complex than another part. But, extreme cases usually reflect a purpose and context problem. The author really only wanted to model part of the system. Level-of-decomposition unbalance is easily quantified by node analysis.

- Balance between arrow detail of joins and branches of an arrow bundle - The level of detail of arrows joining to create an arrow bundle and the level of detail of arrows branching from the same bundle should be similar. Ideally, the arrows forming a bundle and branching out of the bundle should be identically the same data items. This is measurable via interface analysis techniques, two-way tracing of the bundle to its sources and sinks and comparison of the data items at the two ends.

Since publishing Systems Analysis Quality Metrics we have identified a fifth balance consideration. In the page-pair publication format specified for IDEF0 we always have both text and diagrams. These can, of course, be unbalanced. One might have a very complicated diagram and only a few, inadequate, lines of text. However, the problem we see most frequently is the converse --- a fairly simple, often generic or superficial, diagram accompanied by extensive text. The information is in the text instead of in the model!

We will address each of the five balance concepts, and possible algorithms for measuring them, in the following sections.

3.4.1 Balance Between Activity Detail and Arrow Detail in a Diagram

Although this form of semantic unbalance may be an appropriate human review checklist item, we believe the coupling structure metric will always detect the problem as a failure to decompose the arrow. In other words, an automated system may fail to detect the problem on the diagram where it first occurs, but will catch it later as a decomposition glitch. So, we propose no specific metric for this balance problem.

3.4.2 Balanced Diagram Level of Detail

One would expect diagrams at the same level in a hierarchical decomposition to depict similar levels of detail. We have a somewhat unique consideration here. We have an unbalance condition which is very easy to measure, to quantify. All we have to do is compare the number of boxes per diagram and the number of arrows per box on diagrams at the same level in a decomposition. But, how much do we care? How important is this balance, or unbalance, issue?

So our problem isn't measurability. Our problem is how we rank the problem, once measured, in the overall scheme of things. Intuitively, small variations are of little, or no, concern. Large variations probably indicate a decomposition strategy problem. In a structured team-modeling environment, large variations might indicate a difference in the skill levels of the authors. This would be a useful management metric--alerting the team leader to skill differences of team authors.

We will simply measure the complexity of diagrams in different parts of the model using the possible activations formula presented in Section 2.1.2. We will compute means and standard deviations as a measure of balance. High standard deviations will indicate an unbalance. Low standard deviations will indicate good balance. What qualifies as "high" and "low" cannot be predicted at this point in time. We will have to learn, as we gain experience with the metric, what qualifies as "high" or "low", or "good" or "bad".

3.4.3 Balanced Level of Decomposition

The concern here is when some boxes on the A0 diagram are decomposed to six levels of detail and other boxes on the A0 diagram receive little or no decomposition. We are concerned with wide variations in level of decomposition., We are not concerned with small variations. One part of a system may just be more complex than another part. So, the model reflects that difference in complexity.

However, we can't lose sight of the fact that, in the model, our decomposition strategy has defined the "parts". Variations in complexity may be the result of our decomposition strategy rather than a characteristic of the system being analyzed.

In extreme cases, this unbalance suggests a redefinition of context. If only one box on the A0 diagram has been decomposed, then the A0 diagram should become an A-1 diagram, and the decomposed box should become the "model", i.e., the A-0 box.

As was the case with Balanced Diagram Level of Detail, we have something which is easy to measure, and the metric is easy to automate. The problem is ranking the measurement in the overall scheme of things.

Again, we find means and standard deviations to be a good measure. We simply count the number of atomic level boxes generated by each A0 diagram box. The standard deviation of the number of atomic level boxes under each A0 box is a good measure of balanced level of decomposition. This occurs because of the leverage factor which occurs when a box is decomposed. If the only difference between levels of detail in different branches occurs because of the 3 to 6 boxes per diagram rule, then the standard deviations will be on the order of 4 or 5. But, if more boxes are decomposed in one branch than another, then the standard deviations will be much higher because of the much larger number of atomic level boxes.

Of course, only time and experience will tell us, "how high is high", and, "how low is low".

3.4.4 Arrow Bundle Join and Branch Detail

The level of detail of arrows joining to form a bundle and the level of detail of arrows branching out of a bundle should be balanced, i.e., similar, or the same.

The balance problem occurs only with bundled arrows which are both created, by joins, and used, by branching in the model. If a bundled arrow enters the model, from outside, and is decomposed by branching, that's an ideal state of affairs. No consistency problem exists. If a bundled arrow is created, by joins, and leaves the model as a bundled arrow, that too is an ideal state of affairs. No consistency problem exists. This is simply the decomposition, in the model, of an output arrow.

The balance problem occurs when an arrow, created by joins, goes to other boxes in the model without being decomposed. Often, the coupling structure problem exists when this happens. The converse situation occurs when an arrow is created as the output of an activity as a bundled arrow, without disclosure of component parts via a join, and is later decomposed, by branching, into component parts and routed to several activities.

The significance of this balance problem is related to cohesion. A component arrow on diagram "A", which is joined into a bundle, and the same component arrow which branches out to a box on diagram "B", is a direct interface arrow from the source box on diagram "A" to the sink box on diagram "B". The two boxes have strong cohesion, but are on different diagrams. Maybe they belong on the same diagram (via a different decomposition) and maybe they don't. But, if the unbalance problem exists, the cohesion in not disclosed at all. We believe that a lot of clues to an improved decomposition are clouded, if not buried, by unbalanced arrow bundling. The author, using unbalanced arrow bundling, has hidden, from himself, possible alternative decompositions.

This metric, automatable by algorithms for arrow tracing, is considered to be a significant model quality measure. In particular, the metric is a measure of the extent to which an author has disclosed, to himself, the available decomposition options.

3.4.5 Balance Between Diagram and Text

This is more of a problem when the text is excessive than when the diagram is over-complex. We have a diagram complexity measure. When the diagram is over-complex, more text is not the answer. So, our concern is when the diagram is relatively simple, but the text seems to go on, and on, and on ... . This is a difficult metric. Extremes are easy to spot. If there's a lot of text accompanying a simple diagram, we have a problem. But what qualifies as "a lot" of text and what qualifies as "a simple" diagram? Figure 7 is an example from, "The DoD Enterprise Model".[8] The diagram is very simple: A mutual control interface between box 1 and box 2; and an Input-Control mutual interface between box 1 and box 3. However, the text runs on for a full page, addressing all sorts of issues not portrayed in the diagram.

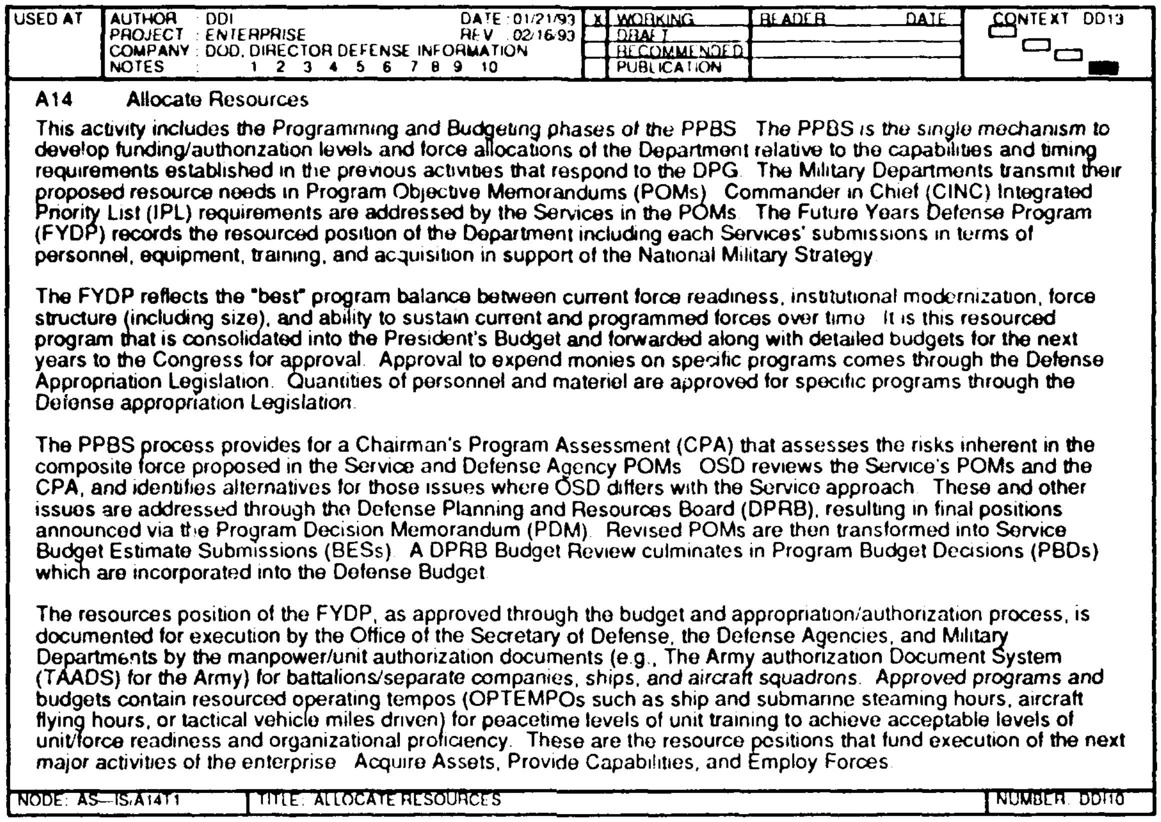
So, how do we measure this balance item? We can count the number of words of text and measure the complexity of the diagram using the possible activations formulation of Section 2.1.2. Over a period of time a correlation will emerge which will be an indication of the appropriate balance between text and diagrams.

3.5 Amplification

We would like to measure the degree of decomposition (amplification) present in the child diagram of a box. We would like to measure the degree of decomposition detailed in the child diagram of each of the arrows associated with a box. In short, we are decomposing both boxes and arrows and would like to measure how well we've done that. However, there is a tradeoff between amplification and complexity. We would like to measure a "just right" amount of amplification. Not too much, not too little --- the gradual introduction of detail.

The minimum amplification present in any decomposition is the set of child activities introduced. That's the degree of amplification which would have been present on a hierarchy chart. We expect more from an IDEF$_0$ diagram. When the activity has been decomposed but the corresponding arrows have not been decomposed, then we have the balance problem discussed in section 3.4.1. Our Coupling Structure metric of Section 3.3.1 should capture the problem. Identifying this as an amplification problem would amount to double jeopardy --- two demerits for the same problem.

[8] "Working Draft, The DoD Enterprise Model, A White Paper", Project ENTERPRISE, Office of the Director of Defense Information, Office of the Secretary of Defense, February 1993



USED AT	AUTHOR: DDI DATE: 01/21/93	X WORKING	READER DATE	CONTEXT DD13
	PROJECT: ENTERPRISE REV: 02/16/93	DRAFT		
	COMPANY: DOD, DIRECTOR DEFENSE INFORMATION	RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10	PUBLICATION		

A14 Allocate Resources

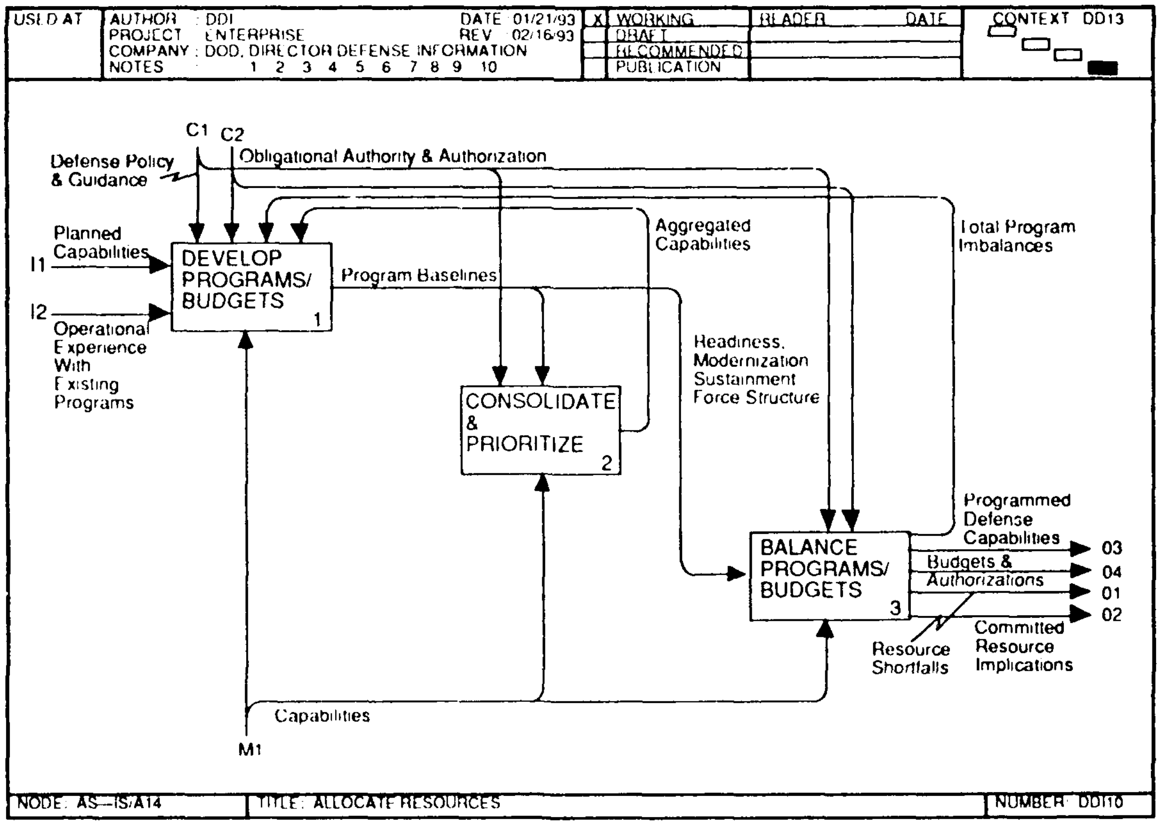
This activity includes the Programming and Budgeting phases of the PPBS. The PPBS is the single mechanism to develop funding/authorization levels and force allocations of the Department relative to the capabilities and timing requirements established in the previous activities that respond to the DPG. The Military Departments transmit their proposed resource needs in Program Objective Memorandums (POMs). Commander in Chief (CINC) Integrated Priority List (IPL) requirements are addressed by the Services in the POMs. The Future Years Defense Program (FYDP) records the resourced position of the Department including each Services' submissions in terms of personnel, equipment, training, and acquisition in support of the National Military Strategy.

The FYDP reflects the "best" program balance between current force readiness, institutional modernization, force structure (including size), and ability to sustain current and programmed forces over time. It is this resourced program that is consolidated into the President's Budget and forwarded along with detailed budgets for the next years to the Congress for approval. Approval to expend monies on specific programs comes through the Defense Appropriation Legislation. Quantities of personnel and materiel are approved for specific programs through the Defense appropriation Legislation.

The PPBS process provides for a Chairman's Program Assessment (CPA) that assesses the risks inherent in the composite force proposed in the Service and Defense Agency POMs. OSD reviews the Service's POMs and the CPA, and identifies alternatives for those issues where OSD differs with the Service approach. These and other issues are addressed through the Defense Planning and Resources Board (DPRB), resulting in final positions announced via the Program Decision Memorandum (PDM). Revised POMs are then transformed into Service Budget Estimate Submissions (BESs). A DPRB Budget Review culminates in Program Budget Decisions (PBDs) which are incorporated into the Defense Budget.

The resources position of the FYDP, as approved through the budget and appropriation/authorization process, is documented for execution by the Office of the Secretary of Defense, the Defense Agencies, and Military Departments by the manpower/unit authorization documents (e.g., The Army authorization Document System (TAADS) for the Army) for battalions/separate companies, ships, and aircraft squadrons. Approved programs and budgets contain resourced operating tempos (OPTEMPOs such as ship and submarine steaming hours, aircraft flying hours, or tactical vehicle miles driven) for peacetime levels of unit training to achieve acceptable levels of unit/force readiness and organizational proficiency. These are the resource positions that fund execution of the next major activities of the enterprise: Acquire Assets, Provide Capabilities, and Employ Forces.

NODE: AS—IS/A14T1	TITLE: ALLOCATE RESOURCES	NUMBER: DDI10



USED AT
AUTHOR: DDI DATE: 01/21/93
PROJECT: ENTERPRISE REV: 02/16/93
COMPANY: DOD, DIRECTOR DEFENSE INFORMATION
NOTES: 1 2 3 4 5 6 7 8 9 10
X WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER DATE
CONTEXT DD13
C1 C2
Defense Policy & Guidance
Obligational Authority & Authorization
Planned Capabilities
I1
I2
Operational Experience With Existing Programs
DEVELOP PROGRAMS/ BUDGETS 1
Program Baselines
Aggregated Capabilities
Total Program Imbalances
CONSOLIDATE & PRIORITIZE 2
Readiness, Modernization Sustainment Force Structure
BALANCE PROGRAMS/ BUDGETS 3
Programmed Defense Capabilities
O3
Budgets & Authorizations
O4
O1
O2
Resource Shortfalls
Committed Resource Implications
Capabilities
M1
NODE: AS—IS/A14
TITLE: ALLOCATE RESOURCES
NUMBER: DDI10

Figure 7. Example of Text and Diagram Unbalance

We believe the best measure of amplification is activation disclosure. This is an attempt to kill two bids with one stone. Activation disclosure is amplification. Activation disclosure is also complexity reduction. So, we combine amplification with simplification when we use activation disclosure as a metric. That's the best of both worlds.

Activation disclosure is the elaboration of possible activations of a box in the decomposition of the box. This was discussed, with a truth table and an example in Systems Analysis Quality Metrics. The FIPS material in section 2.1.1 is an example -- although not a very good example, due to the use of an FEO for disclosure rather than an actual decomposition diagram. But the point is, in the decomposition of a box, certain inputs, controls and mechanisms are seen not to be related to certain outputs. That eliminates some activation rules from the truth table, thus reducing the number of possible activations relative to the a priori number from the formulation of Section 2.1.2.

It is also possible, with the use of the Activation Language discussed in section 2.1.3, to have positive disclosure. That is, disclosure of activations, rather than just elimination of non-activations. If we assume the use of the Activation Language to occur only at the atomic level of the model, then the results have to be "rolled up" through the hierarchy to the diagram of interest.

In any case, the recommended algorithm is as follows. Begin with the parent box of the diagram of interest. Compute the possible activations using the formulation in Section 2.1.2. Do not include tunneled arrows which will not appear on the (child) diagram of interest. Although tunneled arrows are perfectly legitimate activators of a box, if they're not going to be shown on the diagram of interest then their contribution to activations is not going to be disclosed. So, we don't include them in our activation disclosure algorithm.

We now set about reducing the number of possible activations by those which are disclosed in the diagram. If, for example, we note that input number (ICOM code) 3 (of four inputs) cannot produce output number 2 (of three outputs), then the number of undisclosed activations is reduced by $(2^{I-1})(2^{O-1}) = 2^3 \bullet 2^2 = 8 \bullet 4 = 32$. This is the number of truth table entries containing both I3 and O2. I is the number of input arrows and O is the number of output arrows.

If we note that inputs number 2 and 3 (of three inputs) cannot produce outputs number 3 nor 4 (of four outputs), then the number of undisclosed activations is reduced by $(2^I - 2^i)(2^O - 2^o) = (2^3 - 2^1)(2^4 - 2^2) = (8 - 2)(16-4) = 6 \bullet 12 = 72$. This is the number of truth table entries containing (I2 or I3) and (O3 or O4). Lower case i is the number of input arrows which <u>can</u> produce the noted outputs. Lower case o is the number of outputs which <u>can</u> be produced by the noted inputs.

Although we used a simpler formulation in our first example, in which we were only concerned with a single input and a single output, the second formulation is, technically, the correct one. In our first example we would have had $(2^4 - 2^3)(2^3 - 2^2) = (16 - 8)(8-4) = 8 \bullet 4 - 32$, using the latter formulation.

After we have accounted for all disclosed activations, the activation disclosure score is the a priori possible activations, minus the undisclosed activations, divided by the possible activations. As usual, a perfect score is 100%. This would occur if all activations were disclosed. We would never expect to see a score of 100%, except at, perhaps the lowest level of a model, because we are only scoring adjacent levels in the hierarchy with this algorithm. It may take several levels to disclose all activations of a particular activity.

A full-blown algorithm would start at the bottom level of the model and "roll-up" scores, so that the score for any given box would reflect disclosure at all subordinate levels rather than just the next lower level.

4.0 TRAINING MATERIALS

The Systems Analysis Quality Metrics research significantly advanced the state-of-the-art of IDEF0 modeling. Mere insight into IDEF0 quality factors improves one's expertise as a modeler.

The tutorial materials developed during the Phase I effort were presented at IDEF Users Group Workshops in Albuquerque[9] and Ft. Worth[10] and in the form of two-hour briefings to the IPO[11], WRDC MANTECH[12] and AFHRL/LRL[13]. Valuable feedback occurred on all occasions.

In 1992 two additional workshops were presented[14,15]. This resulted in more valuable feedback.

The advanced training materials have been organized into three modules: Quality Metrics Baseline, Frequently Occurring Problems, and Selected Topics. These three modules comprise a one-half day seminar, about three-and-one-half hours. They have been incorporated into the Author's IDEF0 training materials and, as such, have been offered, formally, in a training environment, as distinct from a workshop environment.

The Quality Metrics Baseline module addresses Syntax, Semantics, Coupling, and Cohesion. The Syntax and Semantics topics are a critique of the UM material and closely follow the material in Section 2.1 of Systems Analysis Quality Metrics. The Coupling and Cohesion topics are tutorial material and closely follow the material in Section 2.2 of Systems Analysis Quality Metrics. This Coupling and Cohesion tutorial was presented at the Washington (Dulles) Users Group and was well received.

The Frequently Occurring Problems module list of topics includes: Generic Interfaces, Generic Activities, Mutual Constraints, Sequence Diagrams, Precedence vs Cohesion, Detection Aids vs Metrics, Feedback, and Organizational Functions.

The Selected Topics module takes an in-depth look at Arrow Labeling, Environmental Coupling, A Control on Every Box, Mechanisms, and Tunneled Arrows.

---

[9] Williamson, W.R., "IDEF Quality Metrology", Workshop presented at 1st 1991 IDEF Users Group, Albuquerque, NM, May, 1991

[10] Williamson, W.R., "IDEF Quality Metrology", Workshop presented at IDEF Users Group Mini-Symposium, Ft. Worth, TX, October, 1991

[11] Williamson, R.W., "IDEF0 Quality Metrics", presented to Dictionary Methodology committee and Application Validation Methodology Committee at the IGES (Initial Graphics Exchange Specification)/PDES (Product Data Exchange using STEP) Organization (IPO) Quarterly Meeting, Pittsburgh, PA, July, 1991

[12] Williamson, W.R., "Systems Analysis Quality Metrics", presented to the Aeronautical Systems Division (ASD), Wright Research and Development Center (WRDC), Manufacturing Technology Directorate (MANTECH), Wright-Patterson AFB, OH August, 1991

[13] Williamson, W.R., "Systems Analysis Quality Metrics," presented to Air Force Human Resources Laboratory (AFHRL), Logistics and Human Factors Division (LRL), Wright-Patterson AFB, OH, November, 1991

[14] Williamson, W.R., "IDEF Quality Metrology", Workshop presented at IDEF Users Group Symposium, Daytona Beach, FL, May, 1992

[15] Williamson, W.R., "IDEF Quality Metrology" Workshop presented at IDEF Users Group Symposium, Washington (Dulles), Herndon, VA, October, 1992

The primary material covered at the Fort Worth, Daytona Beach and Washington (Dulles) Users Group workshops was the Selected Topics material. The purpose of these workshops was to get a feeling for consensus of opinion on formalization issues. The results were presented in Section 2 of this report and were incorporated into the training materials.

5.0 RECOMMENDED FUTURE RESEARCH

The Systems Analysis Quality Metrics final report included recommendations for future research in four areas: Formalizations, Algorithm Development, Training Materials Development, and Software Specifications. Those recommendations were, at that time, intended for a Phase II SBIR contract. A Phase II SBIR contract is, typically, five to ten times the size of a Phase I SBIR contract. We picked a subset of those recommendations as the basis for the Systems Analysis Quality Metrics Algorithms Phase I research.

We addressed all of the recommended formalization issues. Although, we see a need for a great deal more formalization than currently exists, the work in progress to develop an IDEF$_0$ FIPS and an IEEE Standard should, in time, satisfy the need. Thus, we recommend no further formalization research within the context of the SBIR programs.

We addressed all but one of the recommended algorithm development items. In addition to the algorithms discussed herein, we need an algorithm to combine the results into a total model quality score. The algorithm is more complicated than, for example, simple weighted averaging. The metrics are interrelated. There are, for example, relationships between coupling and cohesion. Another example is the fact that cohesion and activation disclosure are both measures of amplification. Coupling structure and possible activations are both measures of complexity. Many of the syntax rules for clutter reduction are complexity measures. This is not a trivial algorithm. It remains to be addressed by future research.

The training materials development was completed. Continued improvement is always recommended. But, additional evolution of advanced training materials needn't occur under the heading of "research" within the SBIR program. Further work would be enhancement, rather than new development.

No software specification work was proposed under this Phase I effort. Thus, Software Specifications remains a recommendation for future research. In Systems Analysis Quality Metrics we frequently made reference to measures which were machine detectable, machine implementable, machine computable. In most cases we assumed this to be intuitively apparent. In some cases we alluded to how we would do it. But, there is a wide gap between recognizing that something can be automated and writing a rigorous, unambiguous, explicit, specification of how to do it. This is a significant development endeavor.

U.S. GOVERNMENT PRINTING OFFICE: 1993 - 750-113/82095