IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections

IEEE Computer Society

Sponsored by the
Software and Systems Engineering Standards Committee

1175.2™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

15 January 2007

IEEE Std 1175.2™-2006

Recognized as an
American National Standard (ANSI)

IEEE Std 1175.2™-2006

IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections

Sponsor

Software and Systems Engineering Standards Committee
of the
IEEE Computer Society

Approved 29 December 2006

American National Standards Institute

Approved 15 September 2006

IEEE-SA Standards Board

Abstract: Interconnections that need to be understood and evaluated when buying, building, testing, or using Computer-Aided Software Engineering (CASE) tools are described in this recommended practice. This recommended practice is intended to help people interconnect tools by identifying and characterizing various contexts for tool interconnection. Each context serves to define a group of interconnections pertinent to various functional perspectives. Each group contains interconnections that have a common kind of endpoint in the environment. This recommended practice considers four contexts: an organizational context for a tool, the individual user context for a tool, the platform context for a tool, and a peer context for a tool. Within a context, subsets of interconnections are characterized by a collection of common features applicable to a given functional perspective. The purpose of this recommended practice is to establish sets of interconnection features with which each perspective on a CASE tool's interconnections can be characterized.

Keywords: Computer-Aided Software Engineering (CASE) tools, tool communications, tool interconnections

---

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-5233-1 SH95576
PDF: ISBN 0-7381-5234-X SS95576

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Introduction

This introduction is not part of IEEE Std 1175.2-2006, IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections.

The 1175™ family of standards

NOTE—References to "P1175.X" in this recommended practice refer to members of the 1175 family of standards that were not yet approved at the time that this recommended practice was published.

This recommended practice is a member of the 1175 family of IEEE standards. The members of this family include the following:

Standard number	Title
IEEE Std 1175.1™-2002 *, †	IEEE Guide for CASE Tool Interconnections—Classification and Description
IEEE Std 1175.2™-2006	IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections
IEEE Std 1175.3™-2004	IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying Software Behavior
IEEE P1175.4 ‡	Draft Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior
IEEE P1175.5 ‡	Draft Standard for CASE Tool Interconnections—Syntax for Transferring Behavior Specifications

* The IEEE standards or products referred to in this table are trademarks of the Institute of Electrical and Electronics Engineers, Incorporated.

† IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

‡ This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

This family of standards replaces IEEE Std 1175™-1991.[a] IEEE Std 1175-1991 was advanced to a full-use standard in 1994. It covered a number of closely related subjects, and the scope of material contained was able to serve a number of divergent interests.

This family of standards restructures and substantially augments the material in IEEE Std 1175-1991. It has been divided into several individually useful documents in order to facilitate its use by different communities of interest. These guides, recommended practices, and standards generally address issues involved in characterizing the kinds of interconnections that exist between a computing system tool and its

[a] Although approved in 1991, IEEE Std 1175-1991 was actually published in 1992, and is sometimes found referenced as IEEE Std 1175-1992. It appears in the standards listing on the IEEE Xplore Web site (http://ieeexplore.ieee.org/) as "IEEE Std 1175, 17 Aug. 1992," with the title "IEEE trial-use standard reference model for computing system tool interconnections." In 1994, the term "trial-use" was removed from the title when the standard was approved for full-use status. The 1994 version, which was identical to the 1992 publication except for the title and minor editorial corrections, is not available on the IEEE Web site.

environment. Although particularly intended to address the implementation and use of computer-aided software engineering (CASE) tools, the discussion of interconnections in this family actually has wider applicability to computing system tools in general, beyond only CASE tools.

Four kinds of interconnections with a computing system tool are addressed: interconnections with organizations, users, platforms, and other computing system tools. Consideration of interconnections is important to understanding, selecting, implementing, and using computing system tools. Also, although many computing system tools do not need to communicate behavior descriptions of subject systems, their creators need to develop such behavior descriptions for the tools themselves.

A brief summary overview of each of the members of this family of standards is given in the following paragraphs. A more complete overview is available in IEEE Std 1175.1-2002, which provides an integrated overview of the members of the 1175 family of standards, and it describes the fundamental concepts that provide a basis for organizing the material.

IEEE Std 1175.1-2002, IEEE Guide for CASE Tool Interconnections—Classification and Description

IEEE Std 1175.1-2002 is a guide to the IEEE 1175 family of standards. It describes how these standards are intended to be used to accomplish the effective integration of computing system tools into a productive engineering environment and sets forth the fundamental concepts on which these standards are based. These concepts establish the integrating framework for the other members of this family of standards. IEEE Std 1175.1-2002 describes the scope of application of each member standard, the various issues addressed in each standard, and the interrelationships among the members of the 1175 family of standards.

IEEE Std 1175.2-2006, IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections

This recommended practice presents four contexts for a computing system tool's interconnections that offer insight into the operational problems of interconnecting computing system tools with their environment. This recommended practice establishes recommended collections of standard contextual attributes describing relationships between a computing system tool and its organizational deployment, its human user, its executable platform, and its peer tools, as illustrated in Figure a. These contextual attributes are of the "news-story" form that includes: who, what, when, where, and why. The values of these contextual attributes are references to organizational, industrial, and professional standards. By assisting users to reach a clear understanding of the context of operation for a computing system tool, this recommended practice contributes to the effective implementation and application of computing system tools.

IEEE Std 1175.3-2004, IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying Software Behavior

IEEE Std 1175.3-2004 is an expansion of Part 3 of IEEE Std 1175-1991. It focuses specifically on a common set of modeling concepts found in commercial CASE tools for describing the operational behavior of a software product, and it provides a formal, logical model for describing this behavior. IEEE Std 1175.3-2004 also defines a Semantic Transfer Language (STL) for communicating software behavior descriptions from one tool to another. A notable feature of the STL is its design for human readability, which makes STL text files suitable for use in software design reviews by users unfamiliar with computing system tool diagramming notations. In addition, the design of the STL syntax readily permits analysts to prepare and edit STL descriptions using a text editor or word processor.

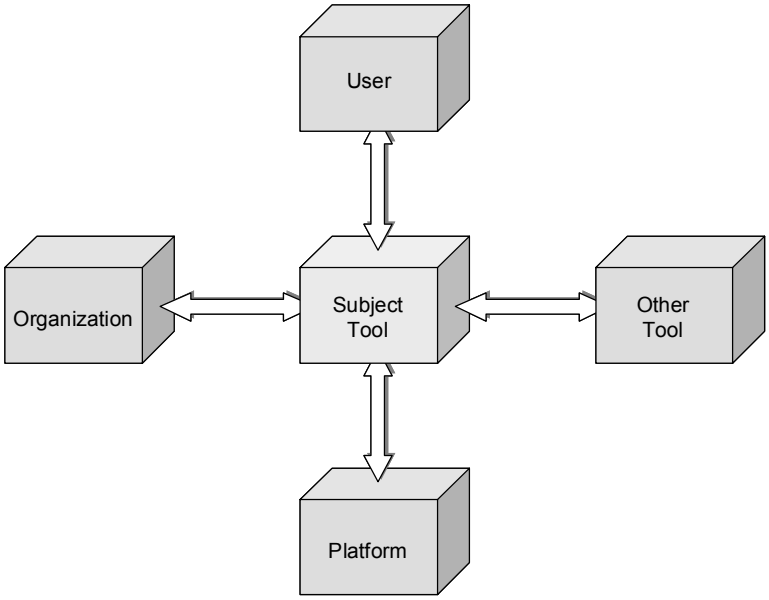

Figure a

To permit backward compatibility with Part 3 of IEEE Std 1175-1991, IEEE Std 1175.3-2004 makes no changes to the STL syntax or to the rules for conformance to this syntax as originally defined in Part 3 of IEEE Std 1175-1991. However, some aspects of the 1991 syntax that were previously left as user-defined have now been specified in order to increase the consistency and reliability with which the STL may be used for exchanging software specification information. In addition, improvements have been made in how the STL syntax is defined and explained. Finally, the STL Interconnection Profile has been replaced with more straightforward, "user-friendly" tabular and comma-separated-value formats to define a Tool Interconnection Profile that can serve the same purpose as the original form of the profile.

IEEE P1175.4, Draft Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior

IEEE P1175.4 encompasses the description of other computing systems supported by IEEE Std 1175.3-2004, but it goes further, providing a basis for representing a wider variety of computing systems. Specifically, IEEE P1175.4 provides the necessary semantic elements for describing general hardware/software systems, including hardware-only, software-only, or mixed system components, and it allows these different types of components to be treated in a consistent manner.

IEEE P1175.5, Draft Standard for CASE Tool Interconnections—Syntax for Transferring Behavior Specifications

IEEE P1175.5 provides a structured syntax and a specific set of data elements that can be exchanged as a text stream or text file with users or other tools. This syntax permits describing a system's operational behavior as an instance of the reference metamodel in IEEE P1175.4. Figure b illustrates the relationship between the IEEE P1175.4 metamodel and the IEEE P1175.5 XML syntax in the context of the Tool–Tool exchange of a computing system description. IEEE P1175.5 also provides a logically equivalent human-readable syntax.

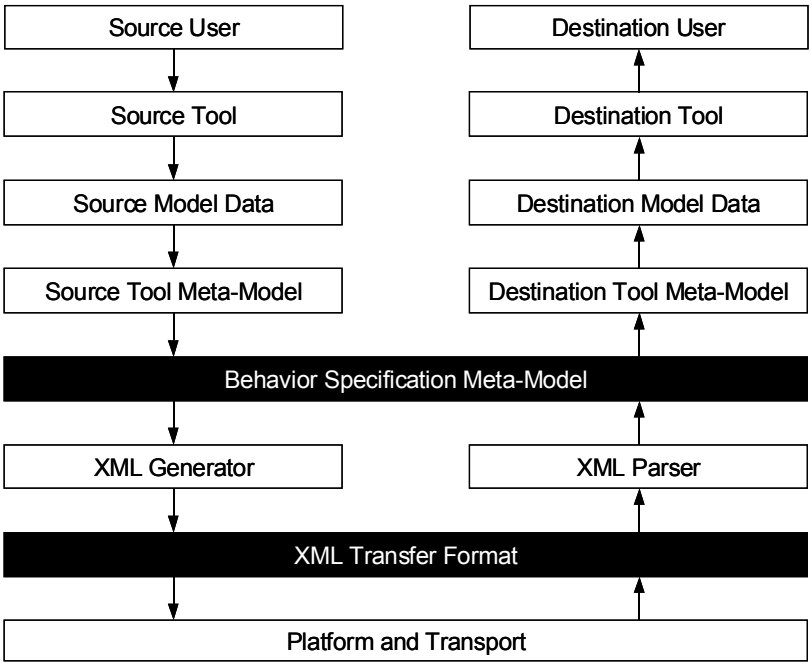

Figure b

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this recommended practice may require use of subject matter covered by patent rights. By publication of this recommended practice, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Participants

At the time this recommended practice was completed, the 1175 Working Group had the following membership:

Carl A. Singer, *Chair*

Jimi E. Arvidsson
Bakul Banerjee
Abby Beifeld
Peter L. Eirich
Subramanya R. Jois
Paul C. Jorgensen
Sohel M. Khan
Dwayne L. Knirk
Mohamed Ashraf Kottilungal
Horace H. Lawrence
Álvaro F. C. Medeiros
Lou F. Pinto
Robert M. Poston
Subrato Sensharma
Robert M. Wessely

The following members of the balloting committee voted on this recommended practice. Balloters may have voted for approval, disapproval, or abstention.

Ali Al Awazi
Charles L. Barest
Juris Borzovs
Lawrence W. Catchpole
Danila Chernetsov
Theo Clarke
Raul Colcher
Tommy P. Cooper
Geoffrey Darnton
Terry L. Dietz
Thomas J. Dineen
Scott P. Duncan
Marc Emmelmann
Yaacov Fenster
Andrew C. Fieldsend
Allan M. Gillard
Randall C. Groves
John Harauz
Werner Hoelzl
Dennis Horwitz
Jeffrey A. Kautzer
Mark J. Knight
Dwayne L. Knirk
Thomas M. Kurihara
Susan K. Land
Dewitt T. Latimer, IV
David J. Leciston
Solomon Lee
Michael W. Malia
Richard A. Martin
Richard A. McBride
Johnathon Meichtry
Gary L. Michel
James W. Moore
Michael J. Munroe
Rajesh K. Murthy
Michael S. Newman
Charles Kamithi Ngethe
Donald M. Parker
Alexander J. Polack
Robert A. Robinson
Fernando Lucas Rodriguez
Bartien Sayogo
Stephen C. Schwarm
Subrato Sensharma
Friedrich Stallinger
Thomas E. Starai
K. S. Subrahmanyam
Vincent J. Tume
Oren Yuen

When the IEEE-SA Standards Board approved this recommended practice on 15 September 2006, it had the following membership:

Steve M. Mills, *Chair*
Richard H. Hulett, *Vice Chair*
Don Wright, *Past Chair*
Judith Gorman, *Secretary*

Mark D. Bowman
Dennis B. Brophy
William R. Goldbach
Arnold M. Greenspan
Robert M. Grow
Joanna N. Guenin
Julian Forster*
Mark S. Halpin
Kenneth S. Hanus
William B. Hopf
Joseph L. Koepfinger*
David J. Law
Daleep C. Mohla
T. W. Olsen
Glenn Parsons
Ronald C. Petersen
Tom A. Prevost
Greg Ratta
Robby Robson
Anne-Marie Sahazizian
Virginia Sulzberger
Malcolm V. Thaden
Richard L. Townsend
Walter Weigel
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan H. Cookson, *NIST Representative*

Jennie Steinhagen
IEEE Standards Program Manager, Document Development

Angela Ortiz
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections

1. Overview

1.1 Scope

This recommended practice presents four contexts for a computing system tool's interconnections that offer insight into the operational problems of interconnecting computing system tools with their environment. This recommended practice establishes recommended collections of standard contextual attributes describing relationships between a computing system tool and its organizational deployment, its human user, its executable platform, and its peer tools. The values of these contextual attributes are references to organizational, industrial, and professional standards.

This recommended practice describes interconnections that need to be understood and evaluated when buying, building, testing, or using Computer-Aided Software Engineering (CASE) tools. CASE tools are developed for use in creating computing systems. By assisting users to reach a clear understanding of the context of operation for a computing system tool, this recommended practice contributes to the effective implementation and application of computing system tools.

This recommended practice does not describe the processes of evaluating, acquiring, or adopting CASE tools. This recommended practice is limited to the technical aspects of CASE tools. It does not include issues in the management, marketing, or training domains.

1.2 Interconnections

In this family of standards, the word *interconnection* has an abstract connotation—it includes all ways in which a CASE tool's successful operation depends on its environment.[1] Thus, an interconnection is an association between a CASE tool and something in the environment. This association affects both endpoints of the association, though not necessarily in the same way. There are the following two categories of interconnections:

[1] IEEE Std 1175.1-2002 [B12] provides an introduction to the IEEE 1175 family of standards.

a) A *passive* interconnection is an interoperability agreement. It describes a common interpretation of one or more phenomena shared between the tool and something in the environment. A passive interconnection describes a semantic issue at the boundary between a CASE tool and its environment.

b) An *active* interconnection is an interaction mechanism. It allows the action of one thing to cause a change or to stimulate an action in another thing. An active interconnection describes a physical interaction between the CASE tool and its environment.

NOTE—An example of a passive interconnection is a statement that Tool A executes on microprocessor M. The shared phenomenon is a CPU instruction set, and the interpretation is that the software executable has been compiled into instructions for the cited CPU. An example of an active interconnection is the invocation of a server-side function through the Common Gateway Interface (CGI) by an HTTP GET request over an Ethernet connection.[2]

Interconnections could be organized and described on the basis of their structural and operational characteristics, but a much more powerful and flexible approach is to segregate and characterize them based on purpose or significance in the system life cycle (see 4.3).

Within this recommended practice, CASE tool interconnections are partitioned into contexts and interconnection perspectives. Each context serves to define a group of interconnections pertinent to various functional perspectives. A group contains interconnections that have a common kind of endpoint in the environment. Within a group, subsets of interconnections are characterized by a collection of common features. Such subsets represent different perspectives in the interconnection group. Perspectives are associated with a viewpoint from which various persons regard CASE tools. Because perspectives are based on human needs and interests, they provide a more suitable organization of interconnection descriptions than would the basic structural or operational aspects of the interconnections themselves.

An interconnection feature is a property by which members of an interconnection perspective are characterized. This recommended practice recommends that interconnection features be characterized by reference to specific documentation. These documents may be project or organization documents, vendor documents, training material, books, articles published in the archival literature, professional or industry standards, or international standards. What is important is that feature characterizations are written down and are readily available as references to members of the organization.

1.3 Purpose

The purpose of this recommended practice is to establish sets of interconnection features with which each perspective on a CASE tool's interconnections can be characterized. Passive interconnections require nothing more than an interpretation agreement. Active interconnections require characterization of actual interactions between the tool and its context. Such characterization may include the possible sequences of interactions (protocol), the form of interaction contents (syntax), and the interpretation of interaction contents (semantics).

A unique feature of CASE tools is that the contents of their primary interactions with users and with other tools are descriptions of other computing systems. IEEE Std 1175.3™-2004 [B13] and IEEE P1175.4[3] focus specifically on the specification of computing system component behaviors. Those standards are semantic references for characterizing Tool–User interactions and Tool–Tool interactions at CASE tool interfaces.

Recognizing the importance of context for achieving meaningful information exchange, this recommended practice is intended to help people working to interconnect tools identify the context for tool

[2] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.
[3] This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

communications. Each context serves to define a group of interconnections pertinent to various perspectives. This recommended practice considers four contexts: Clause 4 identifies an organizational context for a tool; Clause 5 identifies the individual user context for a tool; Clause 6 identifies the platform context for tool use; and Clause 7 identifies a transfer context for a tool. Each context establishes one endpoint for the interconnections included in that group. Figure 1 illustrates these context relationships.

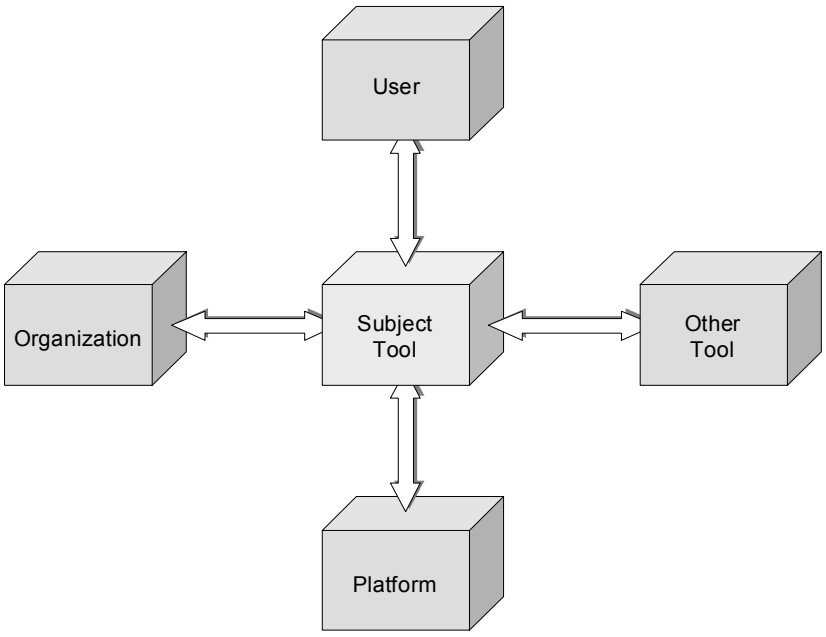

Figure 1—Context for tool usage and information exchange

When a tool is integrated into (or transferred into) an organization, some things usually change. Tool users may need to learn new methods or skills. Company policies or standards may need to be updated. Measurements of work products may need to be reconsidered. The changes that an organization makes to adopt a new tool often cost more than the tool itself. Knowing what changes an organization needs to make to use a tool should make integrating the tool easier. To help integrate tools into organizations, this standard provides recommended practices for characterizing **Tool–Organization interconnections** (see Clause 4).

When a tool is first learned by a new user, is re-learned after a period of not being used, or is used intensively over a period of time, the manner in which the tool interacts with the user becomes a critical factor for the ultimate success of the tool. Tools that are too difficult to learn or use will not survive over the long run, as users will either keep complaining until management replaces the tool (in an organizational setting) or will find ways to do their jobs without using the tool. To help the selection of tools that will work well, this standard provides recommended practices for characterizing **Tool–User interconnections** (see Clause 5).

When a tool is integrated into a hardware-software platform, it needs to operate with the components of the platform. The tool will need a physical context—the mechanisms that enable information transfer. The physical context or environment is provided by a platform. A platform consists of computer hardware (e.g., CPUs, disk drives, networks) and computer software (e.g., operating systems, database management systems, human interface systems). To help integrate tools into platforms, this standard provides recommended practices for characterizing **Tool–Platform interconnections** (see Clause 6).

When tools are integrated into a family of tools, they need to exchange information. There are many different methods or mechanisms for implementing information exchanges, processes for transferring information, and kinds of information to be transferred. To help tools exchange information, this standard provides recommended practices for characterizing **Tool–Tool interconnections** (see Clause 7).

Prior to establishing a Tool–Tool information transfer, it is essential to clarify the scope of the transfer and whether both tools involved have sufficient overlap in their content to support a meaningful exchange. The other standards in this family provide reference models to help identify the scope for a transfer of descriptive information concerning software components and hardware/software systems.

When information is exchanged among tools, all the tools need the format or syntax of the information and the meaning or semantics of the information. To help the integration of tools into productive environments, and to make the transfer of semantic information among tools easier, other standards in this family define a Semantic Transfer Language (STL). STL provides the necessary semantic elements for describing both general hardware/software systems and software components. Those standards also define a syntactic representation of STL descriptions in the eXtensible Markup Language (XML).

1.4 Audience

This recommended practice defines an organization of CASE tool characteristics into four contexts: interconnection to organizations, interconnection to users, interconnection to platforms, and interconnection to other tools. A base set of common features is defined within each perspective. These features are to be used for characterizing CASE tools. These common features are intended to help buyers, builders, testers, and users of professional tools to communicate more effectively.

1.5 Organization of this recommended practice

This document contains seven clauses and one annex. The following annotations provide a roadmap to the contents:

Clause 1	*Overview*—This clause describes the purpose and scope of this recommended practice, and identifies the context for employing tools in organizations, for users, on different computing platforms, and with respect to the information contained in other tools. This clause also describes how this recommended practice is to be applied and provides conformance criteria.
Clause 2	*References*—This clause lists the references necessary for implementing this recommended practice.
Clause 3	*Definitions*—This clause provides definitions of terms.
Clause 4	*Recommended practices for characterizing Tool–Organization interconnections*—This clause defines common features of the organizational context in which a tool works. Questions such as "Will a tool work with the organization's methodology?" are addressed in this clause.
Clause 5	*Recommended practices for characterizing Tool–User interconnections*—This clause defines common features of the user interaction context in which a tool works. Questions such as "Will a tool work well with the anticipated user community?" are addressed in this clause.
Clause 6	*Recommended practices for characterizing Tool–Platform interconnections*—This clause defines common features of the hardware and software context in which a tool works. Questions such as "Will a tool work with the organization's equipment and software?" are addressed in this clause.
Clause 7	*Recommended practices for characterizing Tool–Tool interconnections*—This clause defines common features of the transfer context in which a tool works. Questions such as "How is information exchanged among tools?" are addressed in this clause.
Annex A (informative)	*Bibliography*—This annex identifies other standards and informative references that are cited within this recommended practice.

The relationships between the interconnection contexts described in Clause 4 through Clause 7 are illustrated in Figure 2. The important features include the many-to-many associations, the vertical layering and horizontal partitioning, and the characterization of interconnections as active or passive.

Viewing downwards from the top, an organization or enterprise structures its work into a number of processes. To be either mutually supporting or just non-interfering, the organization's policies and procedures specify various passive interconnections between these processes. Being work structures, processes do not have active interactions between themselves, but their implementation agents do. Two people may or may not be interconnected with active interactions, but if they are agents in the same process or in interconnecting processes, they will usually communicate through active interconnections.

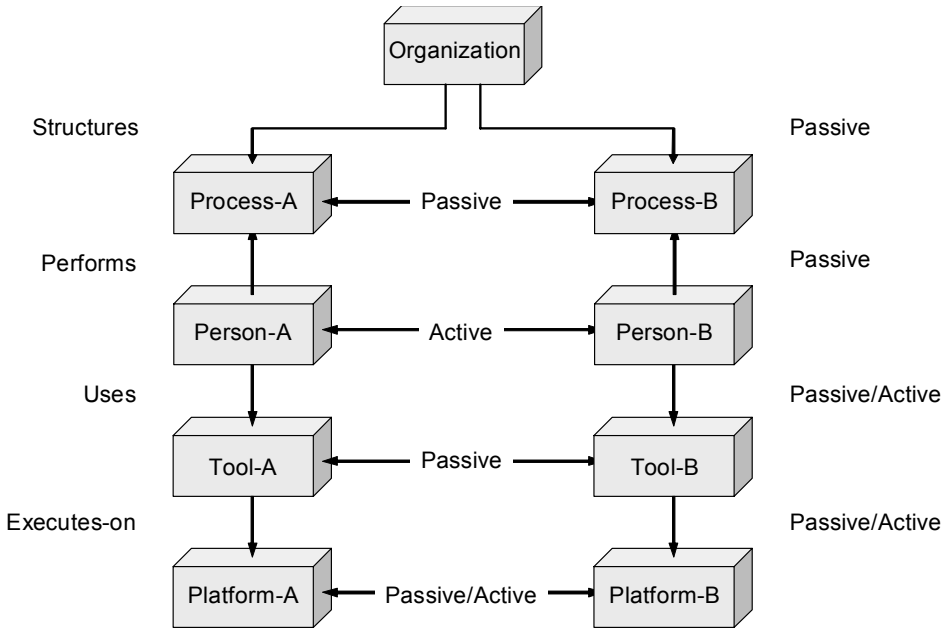

Figure 2—Relationships between interconnection contexts

When computing systems are part of the implementation of a process, then active communication between people can be augmented with interconnection between tools used by the people. When people are communicating through the two tools, any "decision" recorded using the first tool can be replicated in the second tool if (but only if) the tools share an effective semantic transfer mechanism. This tool-mediated person-to-person communication occurs through three active interconnections. As shown in Figure 2, Tool-A interacts with Platform-A, Platform-A interacts with Platform-B, and Platform-B interacts with Tool-B. Most frequently, a description of these active interactions addresses the syntax of communication structures (passive interconnection). It is a separate issue to describe the message interpretation that Tool-A and Tool-B share for information to be passed through those communication structures. This message interpretation is passive interconnection, even though it is frequently discussed as the "communication" between the tools. The similarity with the layered OSI communications model (described in 6.3) is intended.

1.6 Application of this recommended practice

Application of this recommended practice consists of organizing the significant information about a tool within the contexts of organization, user interaction, platform, and related tools, as described in this recommended practice. Conformance to the reference models described in this recommended practice consists of completing the Tool Interconnection Profiles provided at the end of Clause 4 through Clause 7. The profiles serve to identify the context for tool interconnections with organizations, users, platforms, and other tools.

Although they are not covered in this recommended practice, other important aspects of tool selection and usage, such as security aspects and ongoing vendor support, should also be considered in addition to the information summarized within these profiles. The tool context profiles in this recommended practice would provide useful inputs to those processes.

2. Normative references

The following referenced document is indispensable for the application of this document. The latest edition of the referenced document (including any amendments or corrigenda) applies.

The following glossary standard shall be used, when applicable, for tool or software terms not defined in this recommended practice.

IEEE Std 610.12™, IEEE Standard Glossary of Software Engineering Terminology.[4, 5]

3. Definitions

For the purposes of this recommended practice, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B2][6] should be referenced for terms not defined in this clause or in IEEE Std 610.12.

The following definitions describe specific terms as used within the context of this recommended practice. Additional relevant terms defined in Clause 2 of IEEE Std 610.12 include: CASE, interface, semantics, software tool, and syntax.

3.1 active interconnection: A physical interaction mechanism allowing the action of one thing to cause a change or to stimulate an action in another thing.

3.2 behavior: Observable activity of a computing system, measurable in terms of quantifiable effects on the environment whether arising from internal or external stimulus; also, the peculiar reaction of a thing under given circumstances.

3.3 CASE tool: A software tool used for Computer-Aided Software Engineering (CASE).

3.4 environment: Anything affecting a subject system or affected by a subject system through interactions with it, or anything sharing an interpretation of interactions with a subject system.

3.5 interconnection: An association between a CASE tool and something in the environment that affects both endpoints, though not necessarily in the same way.

3.6 interconnection feature: A property by which members of an interconnection perspective are characterized.

3.7 interconnection group: A collection of interconnections to a CASE tool that have a common kind of endpoint in the environment.

3.8 interconnection perspective: A subset of interconnections that share common features in an interconnection group.

3.9 organization: People and processes assembled to produce a specific output (product or service).

[4] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).
[5] The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.
[6] The numbers in brackets correspond to those of the bibliography in Annex A.

3.10 passive interconnection: An interoperability agreement describing a common interpretation of one or more phenomena shared between two interacting things.

3.11 platform: A collection of hardware and software components that are needed for a CASE tool to operate.

3.12 semantic agreement: A passive interconnection in which two things agree on a common interpretation of statements (symbol arrangements) by reference to a shared thing or phenomenon. *Contrast:* **syntactic agreement.**

3.13 subject system: A computing system (existing or to be created) about which descriptive information is being developed in a CASE tool.

3.14 subject tool: A particular CASE tool that is the focus for a description of interconnections and tool content.

3.15 syntactic agreement: A passive interconnection in which two things agree on a set of symbols and symbol arrangements (statements) by which they will communicate. *Contrast:* **semantic agreement.**

3.16 tool: A device that performs or assists in the performance of user or organization process tasks that support, directly or indirectly, the achievement of production goals. (In the context of this recommended practice, tool is used as a short form for software tool, and more specifically for CASE tool.)

NOTE—The reason for employing this broader definition for tool is that the provisions of this recommended practice will also apply to more general computer applications in organizations; many of the provisions will apply to other types of tools as well.

3.17 user: The person who derives engineering value through interaction with a CASE tool.

4. Recommended practices for characterizing Tool–Organization interconnections

4.1 Organization context for tools

To the extent a tool's use is compatible with and supportive to the context of an organization where the tool is used, it will benefit that organization. If the extent of compatibility and support is low, then one of the following situations will likely occur: 1) the tool is not used; 2) the tool will be used ineffectively; 3) the tool will be modified; or 4) the organization will be modified. If a change to either the tool or the organization is required, the amount of time and effort needed to make the change represents a cost that should be evaluated before the change is made. The potential costs of such organizational changes should be considered as part of the costs of acquiring a tool.

How well a tool interconnects with an organization may be considered from the following three perspectives:

a) *Job function:* Who uses the tool? What do people use the tool for?

b) *Life cycle:* When is the tool used?

c) *Process support:* What does it take to make a tool successful in an organization?

These interconnection perspectives are illustrated in Figure 3.

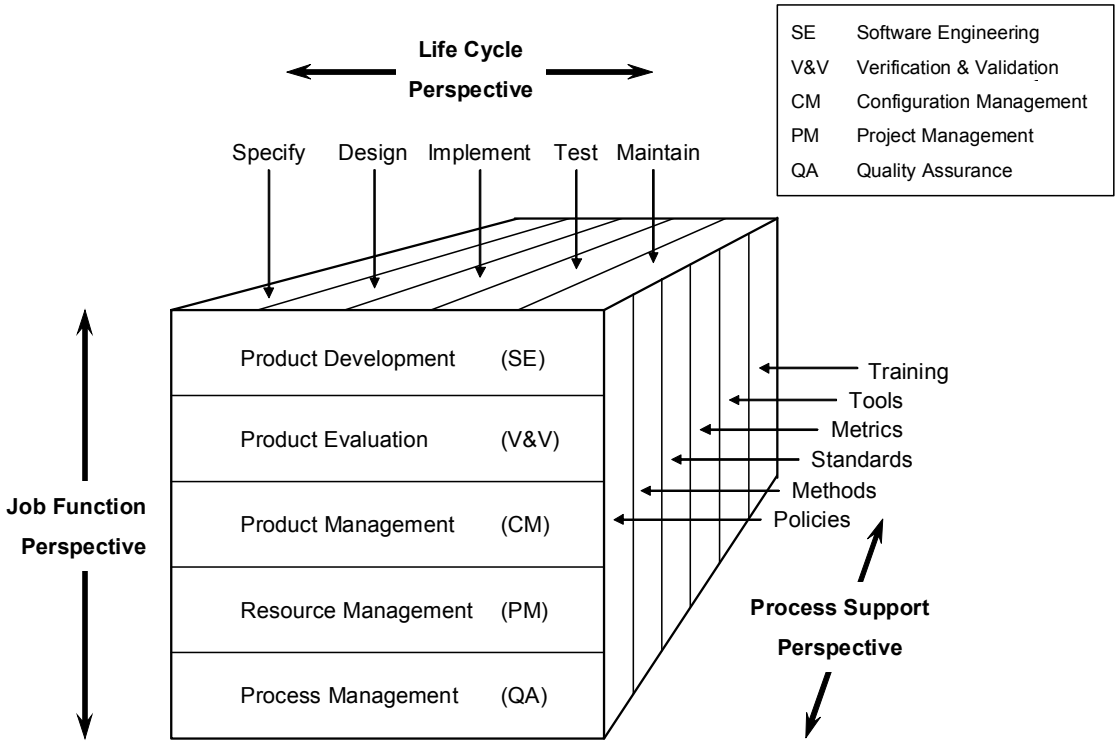

Figure 3—Tool–Organization interconnections

4.2 Job function perspective of a tool

The most direct interconnection between a tool and an organization is between the tool and members of the organization who use the tool to accomplish various software engineering tasks. Although these tasks are performed by individuals, it is more appropriate to identify those individuals by the roles they play in the various software engineering processes. Roles may or may not be related to job titles or organization titles. In this recommended practice, roles are associated with job functions on a project.

A job function is a collection of activities fulfilling a specific purpose for the benefit of the project. On a one-person project, a single individual may spend some percentage of available work time performing each of the following job functions:

a) Product development job functions
   1) Specifying a product to meet requirements
   2) Designing the software architecture and components
   3) Implementing data structures and algorithms
   4) Modifying specifications, designs, and implementations

b) Product evaluation job functions
   1) Inspecting product development artifacts
   2) Verifying and validating product development artifacts
   3) Testing deliverable software artifacts
   4) Modifying verification and validation artifacts

c) Product management job functions
   1) Protecting intellectual property
   2) Controlling life-cycle artifacts
   3) Managing functional content of project artifacts
d) Resource management job functions
   1) Organizing the work and assigning personnel
   2) Controlling schedules and costs
   3) Managing risks
e) Process management job functions
   1) Establishing the work environment
   2) Supplying the process support elements
   3) Implementing process improvement

The previous list contains examples of job functions. A more complete set of software job functions is available in IEEE Std 1074™-2006 [B11] and in the specific practices of the SEI CMMI℠ [B24].[7]

Sometimes, a tool may be used for several different job functions. In such cases, the organization using the tool will probably want to identify different groups or classes of users, such as the following:

— *Initial user class:* People who are responsible for creating new data for entry into the tool. Usually there is only one initial user class. Examples of initial users are requirements analysts and software designers.

— *Intermediate user class:* People who reference and modify data in a tool. Usually, there are many intermediate user classes. Examples of intermediate management users are configuration control and quality control specialists. Examples of intermediate technical users are designers or programmers who use the output of a requirements analysis tool. Examples of intermediate checking users are auditors, reviewers, inspectors, and testers.

— *Final user class:* People who remove or retire project data from a tool. An example of a final user is a quality assurance specialist who extracts software metrics data from a tool, records the metrics in a corporate database, and stores project data in an off-site location.

The specific job functions that are defined for an organization are not important to this recommended practice for the purpose of characterizing the Tool–Organization interconnections. What is important is that job functions are defined by the organization and that the tool users performing those job functions are clearly identified.

4.3 Life-cycle perspective of a tool

A tool is interconnected with an organization when it is actually being used on a project. The period of time when a tool is being used may be identified by naming the dates or by naming generic periods of time, such as life-cycle phases or stages. Life-cycle phases in a project may be distinct or overlapping. They are marked by the development and production of particular work products. These work products are outputs of tasks that capture the essential result of the task and make it available for delivery and for later reference. When tools are integrated into the performance of tasks, work products may be databases of information

[7] This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.

that capture information analysis and creation. Deliverables may be produced as reports of these results in document form.

Examples of some life-cycle phases are defined in the following list:

a) *Specify phase:* That period of time in a product's life cycle when the primary technical activities are concerned with analyzing the application's requirements and defining the product's specifications. Typical work products may include a concept of operations, requirements, and interface specifications; project plans for development, verification and validation, quality assurance, and configuration management; and project standards, metrics, and tools.

b) *Design phase:* That period of time in a product's life cycle when the primary technical activities are concerned with designing the product's architecture and components. Typical work products may include architecture specifications, custom and COTS components specifications, and interface specifications and agreements; an artifact repository, issue tracking and control procedures, software construction and evaluation procedures; inspection reports and change requests; and integration and system test plans.

c) *Implement phase:* That period of time in a product's life cycle when the primary technical activities are concerned with generating, acquiring, and integrating the components. Typical work products may include custom code, commercial code, scripts, compilation and build instructions; unit test scaffolding, scripts, and outputs; manuals and on-line help; system test support; and inspection reports and resolution.

d) *Test phase:* That period of time in a product's life cycle when the primary technical activities are concerned with testing the components and the total software product. Typical work products may include test results; failure reports; code updates; and unit test updates.

e) *Maintain phase:* That period of time in a product's life cycle when the primary technical activities are concerned with understanding, correcting, enhancing, and adapting the software. Typical work products may include modification requests and requirements; revised code and configuration management system entries for the revisions; and regression test results.

The following IEEE, EIA, and international ISO/IEC standards provide useful information for characterizing the life cycle:

— Annex B of ISO/IEC 15288:2002 [B23], also found in IEEE Std 15288™-2004 [B16], provides a set of exemplar life-cycle stages.

— ISO/IEC 12207:1995 [B21] does not address phases or stages of the software life cycle, but it does address mapping life-cycle processes into a "life-cycle model." Such mappings often take the form of phases or stages. These may also be found in IEEE/EIA 12207.0-2004 [B17].

— Annex C of IEEE Std 1074™-2006 [B11] discusses how its activities may be mapped into phased software project life-cycle models.

— Clause 5 of IEEE Std 1220™-2005 [B12] describes "typical system life-cycle stages of development and operations."

— ANSI/EIA-632-1999 [B1] provides life-cycle process definitions.

Tools that are used in more than one life-cycle phase may be grouped according to the following three main periods of use:

— *Phase of initial usage:* A tool is first used when information is initially entered into it. Initial use can occur in only one product development life-cycle phase.

— *Phases of intermediate usage:* Additional uses of a tool occur when information captured by the tool is referenced or modified. These uses occur only after the initial use and may continue into a number of later life-cycle phases.

— *Phase of final usage:* Final use occurs when information is extracted from the tool for the last time on a computing system development project. Final use occurs in only one life-cycle phase, usually the maintenance phase.

The specific life-cycle phases that are defined for an organization are not important for the purpose of characterizing the Tool–Organization interconnections. What is important is that life-cycle phases are defined and the periods of tool use are clearly identified.

4.4 Process support perspective of a tool

Tools are mechanizations that aid or replace human effort in developing work products. To be successful in an organization, a tool needs to perform its job in the same environment as its user, subject to the same expectations and constraints as its user. The environment, expectations, and constraints are established by a medley of process support elements.

Collectively, these process support elements form a context in which the tool is used. The more these process support elements form an integral context with the tool, the more successful will the tool be. In some application domains, more capable organizations are more likely to demonstrate a greater degree of mutual support among these elements. In other application domains, it may be more effective to have few process support elements and a loose application of those that do exist. In either case, conflicts between a tool's usage model and the actual context in which it is applied reduce the likelihood of successful use.

Process support elements may be grouped into six types as shown in Figure 3. Every member of the organization needs each of the following elements in order to be effective at work:

a) *Policies:* Policies are written descriptions of who performs what activities in which life-cycle phases. Policies may also be called directives or instructions.

b) *Methods:* Methods are written descriptions of how to perform an activity. Methods may also be called methodologies, techniques, or procedures.

c) *Standards:* Work product standards are written descriptions of the items (documents, code, or data) that are produced in an activity. Work product standards may also be called documentation format standards. See the following standards as examples:

   — IEEE Std 730™-2002 [B3] discusses software quality assurance plans.

   — IEEE Std 828™-1998 [B4] discusses software configuration management plans.

   — IEEE Std 829™-1998 [B5] discusses software test documentation.

   — IEEE Std 830™-1998 [B6] discusses software requirements specifications.

   — IEEE Std 1012™-2004 [B8] discusses software verification and validation.

   — IEEE Std 1016™-1998 [B9] discusses software design descriptions.

   — IEEE Std 1058™-1995 [B10] discusses software project management plans.

   — IEEE Std 1074™-2006 [B11] discusses the development of software life-cycle processes.

   — IEEE Std 1233™-1998 [B15] discusses the development of system requirements specifications.

d) *Metrics:* Metrics are written descriptions of how to evaluate work products and work processes quantitatively. These evaluations result in measurements. Standards provide a basis for acquiring comparable measurements. Such measurements allow tracking of on-going work and estimation of future work. For example, see IEEE Std 982.1™-1988 [B7].

e) *Tools:* Tools are the devices that enable effective use of the preceding process support elements. They may be passive (manual or automated) or active (automated, as the CASE tools that are the subject of the 1175 family of standards).

f) *Training:* Training is a fast track to experience in the application of the process support elements. Job-based training explains what the organization's process support elements are and how tools are to be used in the work context they provide.

Support elements aid a person who is performing a job or using a tool by providing the answers to the following essential questions.

Question	Process support element in which answers are found
What am I supposed to do? When do I do it?	Policies
How am I supposed to do it?	Methods
What am I supposed to produce?	Standards
How will I know it is a good work product?	Metrics
What is the easiest way to do the right thing?	Tools
Where do I get the answer to these questions?	Training
NOTE—Tool administration issues such as licensing, backup/recovery, and data integrity protection are tool-specific. They are not included in these process support features.	

The specific support elements that are defined for an organization are not in themselves important for the purpose of characterizing the Tool–Organization interconnections. What is important is that support elements are clearly identified, defined, and available to tool users.

4.5 Tool–Organization Interconnection Profile

The context for adopting and integrating a CASE tool into a project or organization environment is established in part by the structure of that environment in terms of roles to which people are assigned, in part by the work products to be created and their interdependencies, and in part by the process support elements that are in place. Identifying the job functions of users, the contribution made to life-cycle work products, and the intent and support of process activities provides the following benefits:

— The most appropriate process roles for applying the tool may be identified.

— The most suitable time for using the tool may be determined.

— The compatibility of the tool with the project's processes may be evaluated.

The Tool–Organization Interconnection Profile comprises the information identified in 4.5.1, 4.5.2, 4.5.3, and 4.5.4. The information required in this profile is expected to be available from process definition documents and project management documents. Completion of this profile needs only to cite the appropriate information in these sources. The purpose of this profile is to collect into one place, which may be a tool information repository, all references to appropriate documents that characterize each of the tool

interconnection features in this context for a particular CASE tool. What is important is that feature characterizations are written down or otherwise recorded and are readily available as references to members of the organization.

4.5.1 Tool–Organization Interconnection Profile metadata

The Tool–Organization Interconnection Profile is identified by the following metadata:

a) *Tool identification:* Provide an unambiguous reference to the tool being profiled. This may include the manufacturer's or supplier's name, the common name of the tool, and a release label.

b) *Organization:* Provide a reference to the organization or project that sets the context for this profile.

c) *Date of profile:* Provide the "truth" date (i.e., true-as-of date) for this profile.

4.5.2 Job function perspective

The job function perspective considers the user roles in the project or organization and characterizes the interconnections they have with the tool to accomplish their job. This characterization is organized by the create/retrieve/update/delete information life cycle. The job function perspective is characterized by the following:

a) *Purpose:* Provide a description of how the project or organization uses (or would use) this tool. Indicate how its use supports the mission of the project or organization. Describe the benefits it provides to the project or organization.

b) *Initial user:* Provide a reference to the user role in the project or organization that is the primary originator of information input to the tool.

c) *Intermediate users:* Provide a reference to the user role(s) in the project or organization that retrieve information from the tool in order to perform their jobs. Describe which user roles update information in the tool.

d) *Final user:* Provide a reference to the user role in the project or organization that removes information from the tool. Indicate whether the information is to be kept elsewhere.

4.5.3 Life-cycle perspective

The life-cycle perspective considers the software life-cycle process activities and characterizes the tool's use within those activities. This characterization is organized by the time phasing of life-cycle activities. The life-cycle perspective is characterized by the following:

a) *Purpose:* Provide a description of how this tool's use fits (or would fit) into the development life cycle of the project or organization. Identify the life-cycle phases in which the tool is used. Identify the benefits that derive from using the tool across different life-cycle phases.

b) *Initial phase:* Provide a reference to the phase in which the initial user enters information into the tool.

c) *Intermediate phase:* Provide a reference to the phase in which the intermediate users retrieve information from the tool. Indicate how often the information is updated.

d) *Final phase:* Provide a reference to the phase in which the final user removes information from the tool. Indicate how long the information is to be kept.

4.5.4 Process support perspective

The process support perspective considers the tool's interconnections with the other process support elements. The process support perspective is characterized by how the tool complements other process support elements or requires complementing by them:

a) *Purpose:* Provide a description of how this tool's use is (or would be) compatible with other elements of the development process used by this organization or project. Describe why this tool is selected for the intended use. Document which of the process elements it supports.

b) *Policy:* Reference each project or organization guidance document that identifies to staff members what they are expected to do with this tool in the various roles they may be expected to play.

c) *Methods:* Reference each project or organization guidance document that describes to staff members how to organize and perform their work in order to gain the most productivity in fulfilling their roles with this tool.

d) *Standards:* Reference each project or organization guidance document that identifies to staff members the criteria for acceptable work products prepared with this tool.

e) *Metrics:* Reference each project or organization guidance document that identifies the measurements used to show the health, productivity, and magnitude of tool application in the project or organization.

f) *Tools:* Reference each project or organization guidance document that identifies other tools that are to interact with the subject tool. Describe the relationship between the subject tool and those other tools. Indicate whether they provide inputs to the subject tool, use outputs from the subject tool, or have some other relationship.

g) *Training:* Reference each project or organization guidance document that describes to staff members which training they can best use to fulfill their role in the project or organization by using the tool in an expected way.

5. Recommended practice for characterizing Tool–User interconnections

5.1 User context for tools

The 1175 family of standards distinguishes between a *subject tool* and a *subject system*. In this recommended practice, a CASE tool whose interconnections are being profiled is a *subject tool*. A computing system about which information is being shared among users and a subject tool is a *subject system*. (That computing system itself could be a CASE tool.) In this subclause, "user" refers to the principal tool users who derive engineering value from the tool; it does not include other types of users, such as installers, administrators, or operations personnel.

There are two distinct Tool–User interconnection perspectives for a subject tool: *system modeling* and *user operation*. The *system modeling* perspective is concerned with what can be said about a subject system and how those things can be said. This is the content and meaning of a specific subject system's description to be conveyed to a subject tool. The *user operation* perspective is concerned with how to use the capabilities of a subject tool and how to get results from that tool. This has nothing to do with any subject system. These two perspectives are illustrated in Figure 4 and described in the following paragraphs.

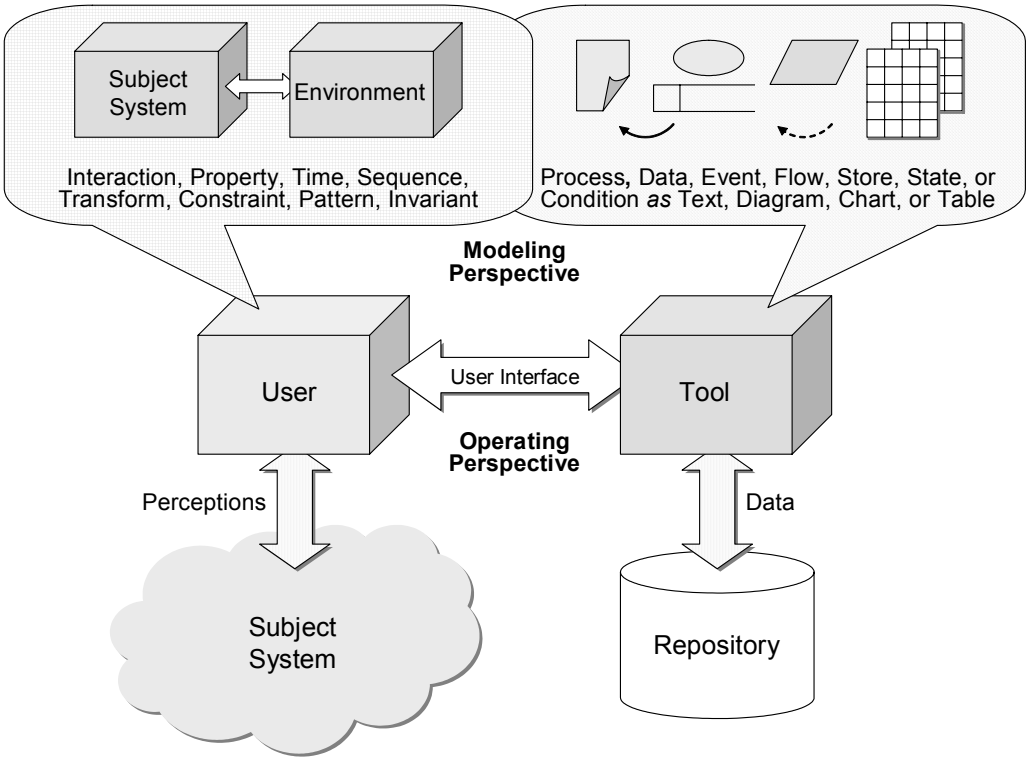

Figure 4—Tool–User interconnections

The *system modeling* perspective of the Tool–User interconnection focuses on the information about a *subject system* communicated between the user and the subject tool. When specifying a subject system, a modeler uses various viewpoints and concepts to structure and quantify different characteristics of that system. Examples of common characteristics include a list of actions the subject system performs and the assembly of mechanisms enabling the subject system to do those things.

A CASE tool user is concerned about encoding conceptions and perceptions of a subject system into data that can be manipulated by the subject tool. Figure 4 illustrates the modeling perspective by showing, on one side, some of the concepts and percepts by which a user describes a subject system and, on the other side, some of the data constructs and presentation forms that CASE tools may provide for capturing facts about the subject system. A faithful mapping between user description and tool description of the system model is necessary. Both tool and user need to share an interpretive scheme so that inferences about the subject system made by the tool will correspond to inferences about the subject system made by a user. The central issue in the modeling perspective is the degree of *conformity* in system model representation and interpretation between user and tool.

The *user operation* perspective of Tool–User interconnection focuses on execution of the subject tool by people performing their jobs. The user working with a subject tool has a mental model of how to use the tool to accomplish various tasks. This mental model is a combination of strategic, tactical, and procedural knowledge for using such tools. Users are concerned about using the subject tool's capabilities to manipulate the description data they provide to that subject system. The central issue in the operating perspective is how to use the subject tool's interface and capabilities to assist the software engineering of the subject system.

The system modeling and user operation perspectives of Tool–User interconnection have little overlap in content. The modeling perspective addresses the ways that an *analyst, designer, programmer,* or *tester*

characterizes a subject system. The operation perspective addresses the ways that a *user* characterizes the subject tool use.

The manufacturer of a CASE tool sees that tool as its *subject system* and creates descriptions of the tool from the modeling perspective. The user of a CASE tool employs primarily models of the *subject tool* from the operation perspective.

5.2 System modeling perspective of a tool

The system modeling perspective focuses on the *content* of interconnection, that is, the representation and the meaning of the information about the subject system passed between tool and user. While a description of the characteristics of the subject system is independent of any subject tool, the actual expression of various facts in that description depends on the subject tool's metamodel and user interface.

The engineering of a useful computing system is based on the development of various technical descriptions of that subject system. These descriptions characterize various aspects of the computing system as seen from an engineering viewpoint. The set of descriptive elements associated with a particular aspect is usually organized into a model of that aspect of the subject system. Such models are particular instances of various description metamodels. Computing system developers and testers have evolved these metamodels for understanding and describing significant features of such systems. A metamodel provides a vocabulary and a collection of concepts, relationships, and measures with which particular models of subject systems may be expressed. Underlying metamodels can be largely independent of the application domain, but it is usual to associate domain-specific language elements with the specific modeling constructs.

Numerous common aspects and associated elements are used for describing computing systems from the technical perspective. Among these are the following:

- The need for the subject system (e.g., problem requirements)
- Ways that users can apply the system to benefit from its capabilities (e.g., use cases)
- What the subject system does (e.g., behavior specifications)
- How the subject system works (e.g., architecture and design, mechanisms)
- The software instructions (e.g., code) that objectify the subject system by controlling the computing system capabilities
- The representative behavior instances that demonstrate the subject system conforms to its models and satisfies the problem requirements and specifications (e.g., tests)

There is a wide range of interpretations that CASE tools and their utilities may apply to a user's descriptive information. A graphical tool may see data as drawing elements with attributes, connections, and simple constraints. A textual tool may see data as various records with keys and referential constraints. Such data manipulation tools are useful, but they infer nothing about the subject system to which the data applies. They are purely mechanical data manipulation aids to the user.

Modern CASE tools provide extensive functionality in support of computing system development, based on detailed metamodels, primarily for the design and code aspects. Many of these design and code metamodels have been refined and reduced to definition through books, courses, and tools. A small number of concepts and relationships underlie these descriptions, but they appear in a large variety of ways in modern computing systems. The semantic interpretations of these models find unambiguous expression ultimately through processor control by generated code.

Among CASE tool vendors and users, descriptive metamodels at the more abstract levels—problem domain requirements, function and behavior, and architectural allocation and traceability—are only now

advancing as far. These descriptive metamodels also are based on a small number of concepts and relationships, but they are more abstract and not as readily traced into computing system operation as are those for design and code. Often there is less rigor in setting out formal definitions of these description elements, lessening the likelihood that useful inferences or deduction can be made from these descriptions. Rather than evolving more comprehensive description metamodels, developments in this area are more often focused on refining only a few of these concepts and relationships, ignoring the broader context. Such subsets of these concepts and relationships still provide partial metamodels for certain kinds of computing systems. These partial metamodels emphasize a few features of such systems in detail, but they ignore other "less important" features entirely.

According to the foregoing considerations, the system modeling perspective of the Tool–User interconnection is characterized along the following dimensions:

a) The subject system aspects included in the model

b) The features of description for those aspects

c) The concepts, relationships, and measures providing the technical metamodel framework

d) The terms and semantics with which those concepts and relationships are expressed and interpreted

Other members of the 1175 family of standards address these issues for specifying the operational behavior of computing systems. These are intended to complement existing analysis, design, code, and test models.

5.3 User operation perspective of a tool

The user operation perspective focuses on the *dynamics* of interconnection—the mechanism, protocols, and syntax of interactions with the subject tool. It is largely independent of any subject system, depending rather on the kinds of information capture and production mechanisms and the functions acting on that information.

The CASE tool user fills some role in a project or organization with respect to the development of some particular subject system. The user applies the tool to assist in performing part of a job dealing with some information about that subject system. The user interacts with the tool through an interface that provides mechanisms for directing tool operation and for transferring system modeling data between the user and the tool.

The operation perspective comprises two kinds of features: task accomplishment and tool interface. The user applies the tool as part of his or her work to accomplish a particular task. This description of *task accomplishment* with the tool is a dynamic characterization. The significant features of this description are what things the user can accomplish by appropriately selecting and sequencing particular actions from the subject tool's repertoire.

Individual tool capabilities are offered to the user through a *tool interface*. This tool interface is the means by which the user engages the tool for communication and control. It is characterized essentially as a collection of input/output mechanisms through which the value of the tool may be realized. Tool interface information describes static characteristics—what is available to the user at a particular time. Task accomplishment information describes dynamic characteristics—what sequence of user activities leads to the performance of the user's work.

According to the considerations in 5.3.1 and 5.3.2, the user operation perspective of the Tool–User interconnection is characterized along the following dimensions:

a) The tool function with respect to the technical perspective elements (e.g., information collection, data storage, analysis, reporting, transformation, generation of successor information)

b) The interface syntax by which the subject system information is represented by the tool for input and output

c) The interface semantics by which tool input and output information is interpreted for the subject system

d) Data integrity guarantees provided by the tool for protection against intentional and inadvertent corrupting actions

5.3.1 Task accomplishment—How the user understands the tool

The manner in which a tool is used depends largely on the user's knowledge of the tool and its operations, within the application context(s) in which the tool is employed.

An understanding of human interactions with computing systems is captured in a user's *mental model*. A mental model is a representation of a subject tool with some plausible cascade of causal associations connecting the inputs of the system to its outputs. This mechanistic model serves to guide both learning and problem-solving behaviors. Mental models may be considered in the following three categories:

— *Procedural sequences (task oriented):* Knowledge of the steps to be taken to achieve a desired result, including conditions or situations in which the sequence might need to be altered. Examples include step-by-step instructions for completing a data-entry transaction or how-to-do-it instructions from "Help" for a CASE tool.

— *Methods (job oriented):* Knowledge of the goals and objectives for a job situation, and the methods that may be used, enabling the user to choose appropriate methods for a task, determine the proper procedural sequences for the system, and identify any additional capabilities needed to complete a desired task.

— *Concepts (theory of how the system works):* Knowledge of the intended or actual behavior of a tool. Aspects may include a stimulus-response view or inherent causality and integrity relationships.

With the exception of tools that may be used solely on a straightforward procedural basis, most users of CASE tools will also maintain a mental model of how a *subject tool* operates, and how to use the tool to build system modeling results consistent with their mental models for the subject system. Some users will also maintain a mental model of how the tool operates functionally, and perhaps how the tool works internally to collect, store, retrieve, and organize data; to compute results; and to make inferences. Users will rely on these mental models to guide their actions using the tool and to accomplish the tasks necessary for describing the subject system. These mental models will guide tool use for answering questions about the system as part of the work product review and use processes.

5.3.2 Tool interface—How the user applies the tool

The nature of the interface through which a user interacts with a tool has a major impact on the degree of user acceptance for the tool and on the effectiveness of its use. The predominant interface used in many software engineering tools is the familiar "point-and-click" interface, often with drag-and-drop operations available for various types of icons. There are many standardized features of this interface that make it familiar to CASE tool users.

The use of such well-understood interfaces can offer significant advantages for both user training and productivity. While users may envision push-button terminals requiring no typing, barcode-based data-entry devices, speech recognition, and virtual-reality displays, these interface features add value only when

they enhance the user's ability to manipulate a subject system's description and to accomplish the user's tasks.

The type of application interface used can vary widely according to the nature of the tool, the characteristics of the users, and the environments in which those users interact with the application. It is not necessarily appropriate for the tool's interfaces for all classes of users to be the same. The user operation perspective addresses each user's tasks and the suitability of one or more of the tool's interfaces for accomplishing those tasks.

5.4 Tool–User Interconnection Profile

The context for using a CASE tool is established in part by the kind of information to be transferred between the user and the tool, in part by the nature of the user interface provided by the tool, and in part by the nature of the tasks for which the user employs the tool. Identifying the syntax and semantics of interactions with the tool, and the means and methods of tool use to accomplish user tasks, can provide the following benefits:

- The appropriateness of the tool for supporting development of computing systems in the application problem domain may be evaluated.
- The conformance of tool use with user's problem-solving approach may be evaluated.
- The detailed needs for user training may be determined.
- The cost of tool use and the benefits realized from tool use may be compared.

The Tool–User Interconnection Profile is comprised of the information identified in 5.4.1, 5.4.2, and 5.4.3. Most information is expected to be provided by reference to project or organization documents, tool vendor documents, training material, books, articles published in the archival literature, professional or industry standards, or international standards. This information pertains to system analysis and modeling methods (information models and diagramming standards) and human factors interface guides. Completion of this profile needs only to cite the appropriate information in these sources. The purpose of this profile is to collect into one place, which may be a tool information repository, all references to appropriate documents that characterize each of the tool interconnection features in this context for a particular CASE tool. What is important is that feature characterizations are written down or otherwise recorded and are readily available as references to members of the organization.

For a tool serving multiple different user roles, it may be useful to make multiple instances of the profile elements in 5.4.2 and 5.4.3.

5.4.1 Tool–User Interconnection Profile metadata

The Tool–User Interconnection Profile is identified by the following metadata:

a) *Tool identification:* Provide an unambiguous reference to the tool being profiled. This may include the manufacturer's or supplier's name, the common name of the tool, and a release label.

b) *Organization:* Provide a reference to the organization or project that sets the context for this profile.

c) *Date of profile:* Provide the "truth" date (i.e., true-as-of date) for this profile.

5.4.2 System modeling perspective

The system modeling perspective considers the models by which the tool records, relates, represents, and manages the specification of various characteristics of the user's subject system. The following perspective characterizes how the description of a subject system is constructed and how statements about the subject system in that description depend on the tool's descriptive capability. The system modeling perspective is characterized by the following:

a) *Purpose:* Provide a description of how the user uses (or would use) this tool. Describe how its use supports the user's tasks. Indicate the predominant function(s) of the tool with respect to the user's information. Consider such functions as analyze, convert, store/retrieve, organize, and format.

b) *Technical perspectives:* Identify which system modeling aspect(s) of a subject system, if any, comprise the subject data used with the tool. (Common aspects are listed in 5.2.)

c) *Features:* Identify the characteristics of the subject system that are included in the engineering aspects of the modeling perspective.

d) *Concepts, relationships:* Describe the fundamental concepts whose instances are observables of the subject computing system. Describe the various relationships by which they are interpreted. Identify which characteristics of system modeling are quantifiable (e.g., rate of stimulus occurrence, response latency bounds, domain boundary of an input property, average retention volume of relationship instances).

e) *Tool vocabulary, syntax, semantics:* Identify the terms with which facts in the system model are communicated. Describe the kinds of transformations of the user's information that are provided by the tool. Describe the kinds of inferences from the user's information that the tool can provide. Indicate how the tool presents the user's information.

5.4.3 User operation perspective

The user operation perspective characterizes the way that users of the tool interact with the tool. It identifies the information and other support the user needs in order to complete the tasks for which the tool is used. The means of data entry, data viewing, and tool operation depend on a tool's interface mechanisms. The user operation perspective is characterized by the following:

a) *Use cases:* Provide references to documented scenarios in which the tool is used to assist or implement the user's software engineering tasks.

b) *Task performance:* Provide an indication of the appropriate user model for task accomplishment in this user role: procedural sequences, methods, or concepts.

c) *Training:* Provide references to training available to the user. Include training in the user's software engineering tasks, as well as training in the proper use of the tool interface.

d) *Tool function:* Provide a description of the primary engineering function(s) provided by the tool. Consider such functions as information collection, data storage, analysis, reporting, transformation, and generation of successor information.

e) *Control interactions:* Provide an enumeration of the predominant mechanism(s) by which the user controls the tool's operations. Consider how the user solicits tool functions, configures options, and interrogates tool status. Include, for example, graphical user interface screens, keyboards and pointing devices, command line options, configuration files, and batch or scripting files.

f) *Data entry:* Provide an enumeration of the predominant mechanism(s) by which data are entered into the tool. Include, for example, graphical user interface screens, keyboards and pointing devices, data file import, and tool controlled object import.

g) *Data manipulation:* Provide an enumeration of the predominant mechanism(s) by which the user modifies or analyzes data. Include, for example, predefined transformation and organization functions, direct selection and manipulation, and scripting.

h) *Data presentation:* Provide an enumeration of the predominant mechanism(s) by which data are presented to the user. Include, for example, on-screen display, printed output, file export, audio headset, or virtual-reality headset.

i) *Presentation elements:* Provide an enumeration of the predominant formats in which the data is portrayed. Include, for example, text, tabular data, graphs, 3D visualization, spatial data maps, diagrams (such as those used for data or process models), pictures, and animations.

j) *Customization control:* Provide an enumeration of the predominant tool functions for which the user has the ability to modify the tool's functionality. Indicate which interaction controls are used for making these changes. Include, for example, time and date display, priority sequence for formula calculation (e.g., a spreadsheet), language used (e.g., English, French, etc.), and display symbols.

6. Recommended practices for characterizing Tool–Platform interconnections

6.1 Hardware-software platform context for tools

Just as tools are interconnected to the organizations that use them, tools are interconnected to the platforms on which they operate. A platform is a collection of hardware and software components (possibly distributed) that are needed for a tool to operate. The hardware components include items such as computers, workstations, personal computers, terminals, networks, disk drives, printers, and displays. The software components include items such as operating systems, database management systems, communications and network management systems, human interface systems (keyboard, mouse, graphics, haptics, oral, or aural), and programming systems. A tool platform may be a development environment or an operating environment.

There are two distinct Tool–Platform interconnection perspectives for a subject tool: *platform obligations* and *coordination*. Platform obligations include tool-specific interconnections with the operational platform for incorporating common platform capabilities into the tool's operation. There is a minimal set of platform obligations for a tool to provide expected capabilities. Coordination includes either unidirectional or bidirectional interconnections with other tools made through mutual interactions with the operational platform. These perspectives are illustrated in Figure 5.

An organization may use a homogeneous platform, and a tool will be operational on that platform. However, some organizations may use a variety of platforms and platform configurations, such as stand-alone work centers (workstations, personal computers, or both), and a tool may be operational on several platforms. To be usable on any platform, tool execution imposes obligations on that platform. These obligations usually include expected CPU family, expected peripherals, operating system versions, associated code libraries (e.g., DLLs), and possibly other services (e.g., database management systems, human interface management systems, communication and networking systems, programming tools, utilities). For every tool an organization chooses to use, that organization needs to provide a platform satisfying a minimal set of *platform obligations* for that tool to be used with expected capabilities. These obligations bound the portability of the tool.

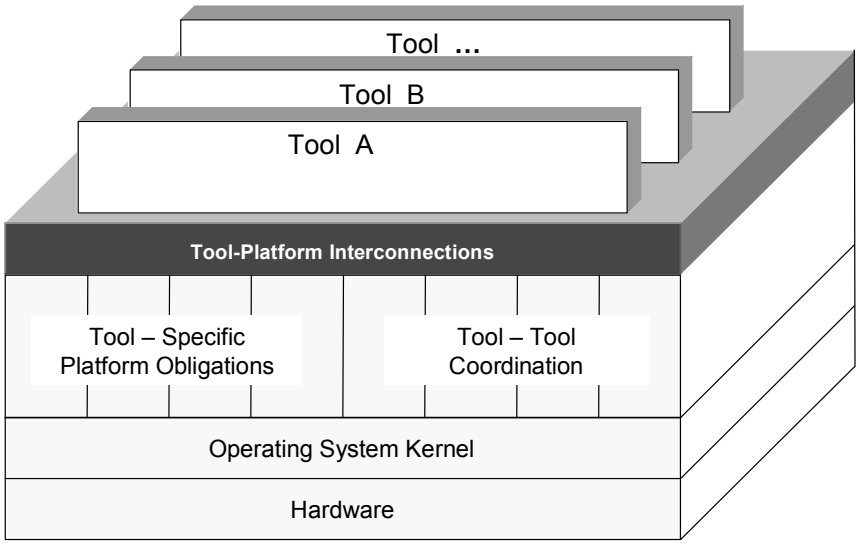

Figure 5—Tool–Platform interconnections

When multiple tools (or multiple copies of an individual tool) are to be used together in an integrated fashion, there is another set of platform issues to be considered. Specifically, the platform needs to provide capabilities for implementing all collaborations that may be required between the tools. Tool–Tool coordination may be unidirectional or bidirectional, and may be required at various levels. Tool coordination may be simple. For example, data connection may be made through sequenced, asynchronous non-concurrent sharing of a file (source tool writes the file at one time and destination tool reads the file later), and tool activation and operation requests may be user-directed. Tool coordination may be more complex. For example, two tools may share a common data-access space, each tool may signal the other when it changes relevant information in that space, and the signaled tool may execute autonomously to update other tool-specific data about the shared data. What is of concern here is not the protocols of tool collaboration, but rather the capabilities of the platform required to implement such protocols.

6.2 Platform obligation perspective of a tool

How well a tool interconnects with a specific platform depends on which platform services and controls are required for successful operation of the tool. These platform services and controls may include the following:

a) *Hardware:* CPU families in platform computing systems, peripheral equipment

b) *Operating system kernel:* Accesses, manipulates, and controls hardware capabilities for application programs on a platform

c) *Operating system services:* Basic I/O capabilities, file systems, swap files, user access control, security, performance management, window management, etc.

d) *Database management services:* Store and retrieve code, data, text, and graphics information for the tool

e) *Communications and networking services:* Send, receive, encode, decode, and route information and service requests to and from individual sources and destinations

f) *Human interface management services:* Accept, display, and manipulate data in textual and graphical forms on a platform for the human user of an application program (tool)

g) *Programming services:* Other software that may be needed for the successful operation of the subject tool (e.g., compilers, interpreters, DLLs, libraries)

Many of these elements are frequently bundled by vendors into standard host platform systems, identifiable by name and version.

6.3 Coordination perspective of a tool

The coordination perspective of a tool characterizes the platform mechanisms enabling coordination between a subject tool and other tools. In general, coordination between tools includes communication of information from one tool to another and appropriate temporal sequencing of execution of the tools. Such coordination may be accomplished through a combination of automated and manual activities. The ability of tools to communicate easily with one another over links between their host platforms enhances their value. Such communication may use automated data communication and control synchronization, or it may require intervention by users to coordinate operation and information between tools.

Communication with other tools may mean a syntactic transfer or sharing of data with a semantic equivalence between generator and recipient. To achieve a semantic transfer of information requires a collection of syntactic agreements across the layers of this interface, as well as between the peers. A basic model for this collection of agreements is provided in ISO/IEC 7498-1-1994 [B18]. The model is illustrated in Figure 6. The dashed arrows show the actual communication interactions that accomplish the Tool-A to Tool-B communication.

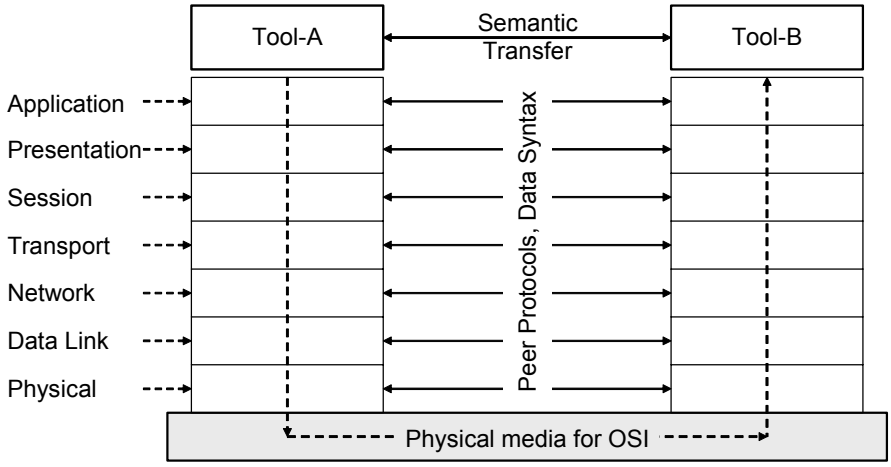

Figure 6—OSI seven-layer reference model and peer protocols

Attributes of a well-defined interface specification for any Tool–Platform interconnection are described in the POSIX® system architecture; see ISO/IEC TR 14252:1996 [B22].

Tools may use a variety of standards to represent data to be communicated. Using such standards is crucial to achieving syntactically valid communication between tools as follows:

a) *Data exchange:* Either use standard file formats or convert from proprietary formats to standard file formats

b) *Document exchange:* Standard formats for transfer of complete document types

c) *Data description exchange:* File formats for describing symbols and tables

d) *Programming language:* Language-specific data representations for direct exchange of data between executing programs

The achievement of semantically valid communication between tools does not have the same extent of standards for its support. Semantically valid communication is described in more detail in Clause 7 of this recommended practice.

Temporal sequencing of execution means a synchronization of behaviors between the subject tool and the other tool. Such coordination has often been left to the tool users to accomplish, but greater integration and "seamlessness" of interfaces requires a consideration of trigger-type coordination, inhibition-type coordination, and chaining-type coordination. Furthermore, such coordination may be required in both directions. For example, Tool-A operation may inhibit Tool-B operation, but then chain to Tool-B to update its view of information changed by Tool-A, and Tool-B operation may chain back to Tool-A to review changed information generated by Tool-B in response. Such coordination issues can be characterized as stimulus relationships (Tool-A simply passes control to Tool-B), or as obligation relationships (Tool-A uses Tool-B to provide needed services upon "request" and awaits Tool-B's reply before continuing).

More complex protocols may be used for interleaving passing and sharing of data and control between tools, but this perspective on Tool–Platform interconnection is focused on the standards and mechanisms of the platform that are necessary for executing these coordination protocols.

The user's productivity and the tool's effectiveness can be reduced if significant amounts of work are involved in moving data from a tool on one platform to a tool on a different platform and coordinating tool operations in accord with data dependency requirements. If tools working on different platforms use special or proprietary services of the platforms, it may be especially difficult to get these tools to work together. Standard interaction protocols and interaction content formats used to enable transfers to peer tools constitute Tool–Platform interoperability agreements. Many platform standards need to be considered in evaluating a tool's ability to interconnect through platforms. Conversely, a proposed platform needs to be evaluated to determine which standards the platform supports. The ability to freely interconnect tools or relocate tools will generally be limited by the use of proprietary formats, tools, or platforms.

6.4 Tool–Platform Interconnection Profile

The context for installing and executing a tool on a hardware and software infrastructure is established in part by the operating services provided by that infrastructure, in part by the connectivity available to other tools and other platforms, and in part by the interoperability agreements among the platform elements.

Identifying the platform services required for operation of the tool, the peer collaborations available to the tool for integration, and the interoperability agreements supported by the platform provides the following benefits:

- The adequacy of the platform to support effective operation of the tool may be evaluated.
- The potential for leveraging the tool to support additional processes may be determined.

The Tool–Platform Interconnection Profile is comprised of the information identified in 6.4.1, 6.4.2, and 6.4.3. Most information is expected to be provided by reference to commercial platform specifications, tool vendor documents, professional or industry standards, or international standards. This information pertains to application deployment environments such as operating systems, component-oriented languages, object request brokers, networks, and middleware systems. Completion of this profile needs only to cite the appropriate information in these sources. The purpose of this profile is to collect into one place, which may be a tool information repository, all references to appropriate documents that characterize each of the tool interconnection features in this context for a particular CASE tool. What is important is that feature characterizations are written down or otherwise recorded and are readily available as references to members of the organization.

6.4.1 Tool–Platform Interconnection Profile metadata

The Tool–Platform Interconnection Profile is identified by the following metadata:

a) *Tool identification:* Provide an unambiguous reference to the tool being profiled. This may include the manufacturer's or supplier's name, the common name of the tool, and a release label.

b) *Organization:* Provide a reference to the organization or project that sets the context for this profile.

c) *Date of profile:* Provide the "truth" date (i.e., true-as-of date) for this profile.

6.4.2 Platform obligation perspective

The platform obligation perspective identifies those hardware, software, and service obligations on the host computing platform that enable the tool to provide expected capabilities. The platform obligation perspective is characterized by the following:

a) *Hardware:* Identify and characterize the computing processor and other peripheral equipment that provide the processing cycles for the tool's operation.

b) *Operating system kernel:* Identify and characterize the operating system environments under which the tool can operate.

c) *Operating system services:* Identify and characterize the required presence and configuration of application and user-level platform services that are required for proper tool operation. Include, for example, paths for locating required code and data, file system configuration, user privileges, and administrative tools.

d) *Database management services:* Identify and characterize any database management system application components and data structures that have been incorporated into the tool and any other file management structures used by the tool to store and manage its data content.

e) *Communications and network services:* Identify and characterize the services or protocols used by the tool to send, receive, encode, decode, and route information and service requests between tools running on different platforms.

f) *Human interface management services:* Identify and characterize the elements of the platform and operating system used by the tool to accept and display data, text, and graphical information for the human user of the tool.

g) *Programming services:* Identify and characterize any compilers or interpreters that are required to support tool operation. Include, for example, the programming language(s) in which tool scripts have been written, any language-related software on which the tool depends for its operation, and any languages in which the tool generates code.

6.4.3 Coordination perspective

The coordination perspective identifies those service and control obligations on the host computing platform that enable the tool to communicate and coordinate with other tools. The coordination perspective is characterized by the following:

a) *Data exchange:* Identify and characterize the file or message encoding and formatting used by the tool to store and retrieve data items within its platform environment or to exchange data items with other tools. Include, for example, text files, binary date files, images, database files, as well as communication messages and brokered objects.

b) *Document exchange:* Identify and characterize the specialized formats used by the tool to prepare documents that may be accessed by use of word processors, spreadsheets, presentations, or any other applications typically found in office application suites.

c) *Data description exchange:* Identify and characterize any specialized data organization formats related to the nature of the data maintained within the tool and used for data storage and retrieval (e.g., XML or comma-separated-value formats).

d) *Programming language:* Identify and characterize any direct program-to-program data transfers between the subject tool and other tools. Include the required data elements and data structure formats that are to be exchanged.

e) *Control exchange:* Identify and characterize the platform mechanisms needed to coordinate or synchronize execution of the subject tool with other tools. Consider standard operating system mechanisms and programming environment capabilities to start, stop, suspend, resume, spawn, and rendezvous execution threads. Consider synchronization of concurrent operation between tools. Operating system capabilities are frequently packaged into message-based protocols through various middleware components and may be identified there.

f) *Coordination:* Identify expected data and control exchange protocols to be used by interconnecting tools. When data is moved and not replicated, consider how transfers of custody are controlled.

7. Recommended practices for characterizing Tool–Tool interconnections

7.1 Tool collaboration context

Recommended practices for collaboration among tools help to establish the mechanistic context for Tool–Tool information transfers. Collaboration may occur in one direction or in both directions between two tools. To compose the functionality from two tools on a subject system description requires communication of information from one tool to another and appropriately coordinated execution of the tools, as described in 6.3. Collaboration uses capabilities of the host platform for the tools, but it is the characteristics of the collaboration protocols that are the subject of Clause 7.

There are two perspectives from which to characterize Tool–Tool interconnections. The *linkage perspective* addresses the mechanisms and processes by which a subject system's description in one tool is physically transferred to another tool and the use of such transfers for tool collaboration. Each tool on its host platform uses the compatible, reciprocal services described in the platform service interconnections to accomplish this transfer.

The *information perspective* addresses the meaningful transfer of a subject system's description (model content) from one tool by another tool. The central issue in this perspective is the *conformity* of model representation and model interpretation between the two tools. When two tools describe the same characteristics of the subject system, conformity would require a faithful mapping between each tool's description of those characteristics. Both tools need to have a common interpretive scheme so that inferences made by one tool will map to the inferences made by the other tool. The issues are similar to those of Tool–User interconnection described in 5.2.

7.2 Linkage perspective of a tool

7.2.1 Mechanisms for information transfer

Information transfer among tools may be performed by many different transfer mechanisms. A mechanism is some combination of platform hardware and software that perform the transfer of information. Different mechanisms provide different features of the information. Some implementations of these mechanisms are invisible to tool users. Some common forms of information transfer are illustrated in Figure 7.

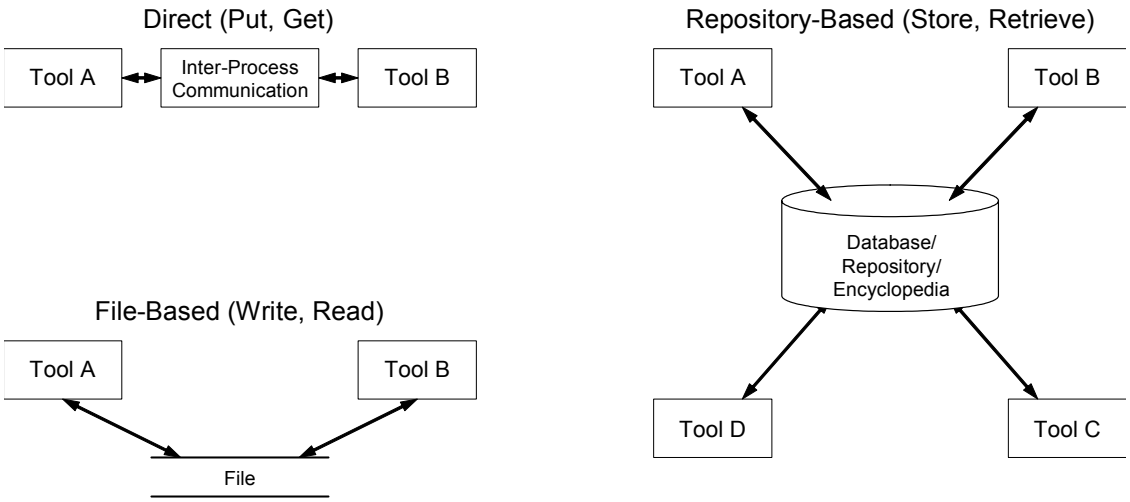

Figure 7—Example of information transfer mechanisms

Each of the mechanisms serves a particular need, and each might be used for particular situations. The direct transfer mechanism is the most efficient from a response time perspective. The file transfer mechanism is best from a simplicity of implementation perspective. The central database or repository mechanism is best from a multi-tool integration perspective. An in-memory shared working area mechanism fits somewhere between direct and repository-based transfer mechanisms. The communicating system mechanism is best from an open systems perspective.

There are many possible mechanism tradeoffs for performance. For example, the speed of information transferred can be maximized if error checking is minimized, and the volume of information transferred can be minimized if data is compressed.

Since every transfer mechanism is best for some situation and optimizes some characteristic, every mechanism will be needed at some time. The model illustrated in Figure 8 represents a process-independent information transfer mechanism.

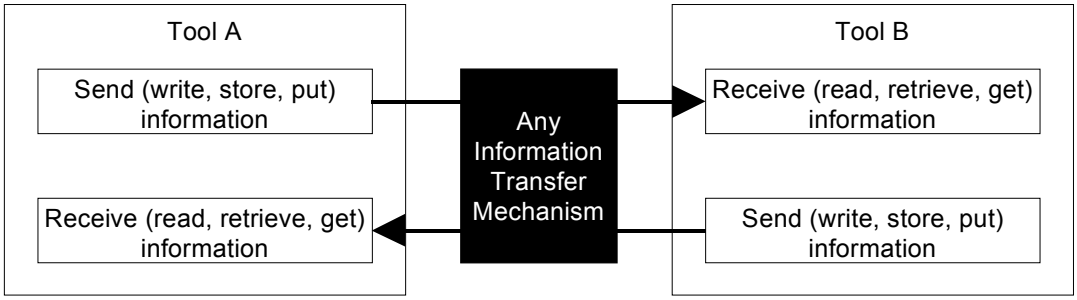

Figure 8—Tool interconnection process model

The send process can be described by defining the services it provides, as follows:

— *Extract information from a tool-specific form:* In this process, the sending tool extracts information from its storage in its stored internal form. The design and operation of this process are affected by the information available in the sending tool and are constrained by the implementation of the sending tool.

— *Encode or format the information into a standard form:* In this process, the sending tool converts the information from its internal form to the standard form. The design and operation of this process are affected by both the information from the sending tool and the standard format for information.

— *Transmit the information in the standard form:* In this process, the sending tool delivers the transmitted information to an interconnection method (mechanism) such as the receiving tool, a file handler, a database management system, or a communication system. The design and operation of this process are affected by both the information to be transmitted in the standard format and the transfer mechanism.

The receive function can be described by defining the services it provides, as follows:

— *Recognize the smallest units of information:* In this process, the receiving tool accepts information in the standard form from the interconnection mechanism and identifies separate units of information. This process is well understood. The software for this function is commonly called a lexical scanner. If the language for information transfer is described in a standard form, available tools can generate the software for scanners.

— *Recognize the structures of information:* In this process, the receiving tool recognizes collections of information. This process is well understood. The software for this function is commonly called a parser. If the language for information transfer is described in a standard form, available tools can generate the software for parsers.

— *Format information from the standard form into the receiving-tool-specific form:* In this process, the receiving tool converts standard form information into its own internal (tool-specific) form.

— *Map semantic information provided by the sending tool to semantic information usable by the receiving tool:* This process is not well understood. It depends directly on the agreement about the semantic interpretation of transferred information between sending and receiving tools.

— *Check the information needed by the receiving tool:* Identify and resolve all conflicts between available information and needed information. Resolve all conflicts of missing, incorrect, and extra information. This resolution process is not well understood and needs to be developed for each specific receiving tool.

It may not be possible for the makers of the sending and receiving tools to establish the details of the process before the tools are created. In order to accommodate this continually changing environment, the following transfer policy is endorsed by this recommended practice:

a) The sending tool provides whatever information it has about the subject system.

b) The receiving tool accepts whatever information is provided, verifies the information, and discards unneeded or questionable information that is of no value to it.

c) At the option of the user, the receiving tool should provide diagnostic reports describing the information discarded and produce a post-transfer review file containing the discarded information.

This “receiver beware” policy is a compromise between efficient transfers and flexible transfers. While the transfer could be more efficient if the quality and quantity of information were known, the process needs to accommodate information of unknown quantity and unknown quality.

7.2.2 Controls for collaboration

Basic control information is the information that allows the communicating tools to start, stop, send, receive, and resend information. Differences in control information and the control of information transfer

are what separates methods of transfer such as direct transfer systems (puts, gets), repository-based systems (writes, reads), file-based systems (writes, reads), communicating systems (sends, receives). Control information is typically defined by platform standards (operating system, database management system, and/or communication system) and, increasingly, with middleware standards.

Collaboration controls are the tactics (or protocols) used by a tool for collaborating with other tools through information transfers. Unidirectional tool collaboration may be as simple as the sequenced, asynchronous non-concurrent sharing of a file (sending tool writes the file at one time and receiving tool reads the file at a later time) and tool activation and operation requests may be user-directed. However, as tool suites become more integrated, tool coordination may become more complex. For example, a subject tool may write a file and invoke execution of a second tool; the second tool uses the transferred subject data to derive some analysis results and writes those into another file; the second tool terminates and the subject tool retrieves the analysis results and uses them for additional user benefit. The interconnections to be characterized in this Tool–Tool linkage perspective are these collaboration protocol patterns, not the underlying platform capabilities used for their implementation.

7.3 Information perspective of a tool

Information transferred between CASE tools can be distinguished in three categories, based on the receiving tool's purpose for the information. The first category is *subject system information*. This is the descriptive information about a subject system that has been aggregated into the source tool and that is to be used by the destination tool to provide engineering value to that tool's user. This is the system modeling perspective information described in 5.2.

The second category of transferred information is *management information*. This is information comprising those attributes of the subject information that allow for its proper management and use.

Information transfer requires values of data provided in a particular syntax and interpreted with a particular semantics. Thus, the third category of transferred information includes *syntax information* and *semantic information*. This is information that describes the syntax and semantics of the data carrying the subject information, by which the receiving tool reconstitutes information about the subject system from the transferred data. It is not required that this information actually be transferred from a sending tool to a receiving tool. Rather, this information may be provided using a passive interconnection, that is, an interoperability agreement standardizing the subject information transfer.

7.3.1 Subject system information

Subject system information may be perceived in three layers as shown in Figure 9.

At the lowest layer are the fundamental concepts by which a subject system is described. Different fundamental concepts are used to describe different aspects of the subject system. In the figure, for example, are listed some concepts that are used to describe operational behaviors of computing systems and computing system elements, and the aggregation of element behaviors into behaviors of the system. A description of subject system behavior is a catalog of many specific instances of these concepts and instances of relationships between these concepts.

Each cataloged instance of a concept is defined with a name and a set of concept attributes. For example, a named element of application information may be defined to be represented by data of a particular type and a domain of specified values. Relationships are an integral part of the semantic interpretation by which concept instances are unambiguously interpreted into inferences about the subject system. Relationship instances are also cataloged in the description. There are many relationship types. For example, a named action may be triggered by the occurrence of a named interaction, the named action may occur only if a named precondition is true, the named action may cause a named response interaction, and the completion of the named action may leave the system in a particular control state.

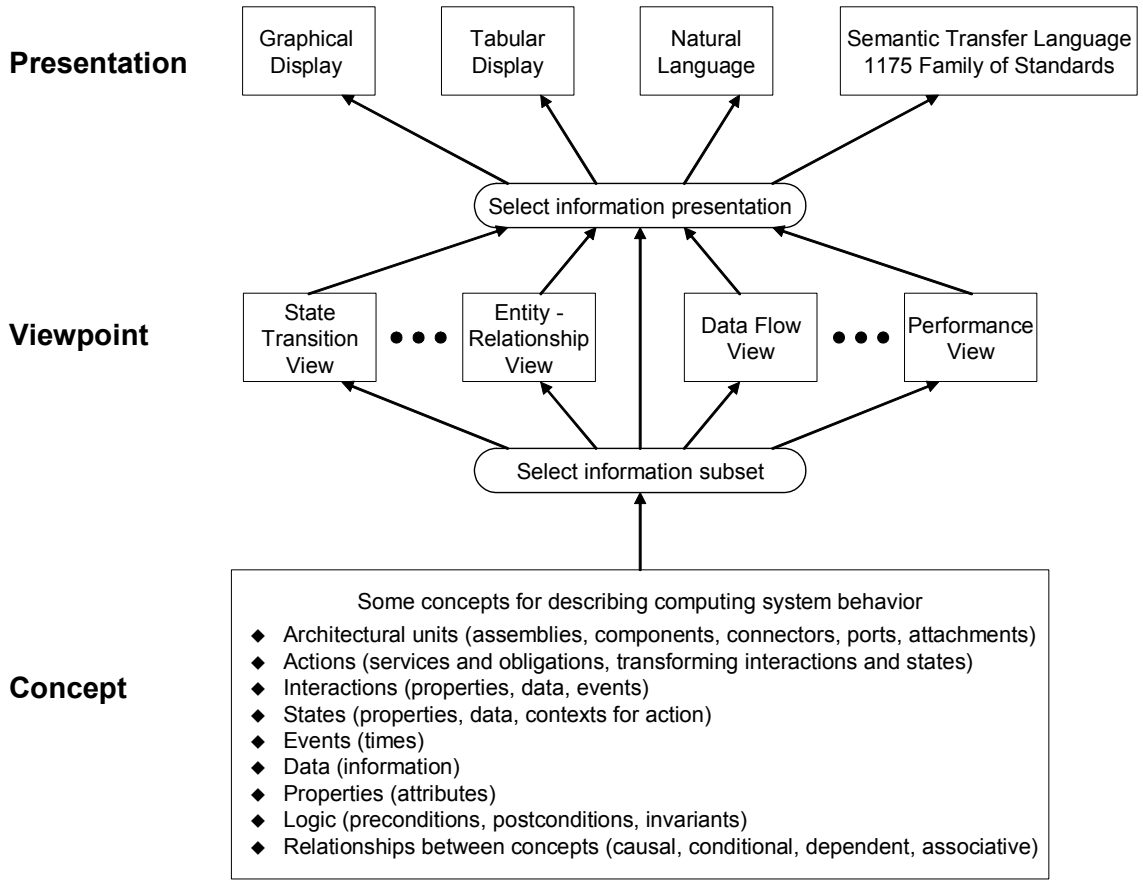

Figure 9—Presentations, viewpoints, and concepts

Historically, subject systems are described with a variety of models, each focusing on one particular set of phenomenological characteristics. Such viewpoint descriptions include only a subset of concept and relationship instances while omitting others. Some familiar viewpoints include data flow models, state transition models, Petri nets, activity thread models, and decision (precondition) tables. Viewpoint information is selected to satisfy a particular set of related questions. Examples of such questions include "liveness" and safety in state transition networks, cardinality and referential integrity in entity relation sets, dependency analyses in data flow networks, and throughput estimation in a resource-constrained environment.

For a particular viewpoint, there are conventional representation formats for the information. Different representation formats are more or less suitable for exposing certain aspects of the information. For example, graphical formats enable discernment of patterns that would not be obvious in tabular form, tabular formats enable presentation of massive detail while providing easy sorting and lookup, natural language may communicate a lower density of information in an easy-to-comprehend format, and structured text may facilitate understanding within particular contexts.

Individual CASE tools normally deal with subject system information across all three layers in Figure 9. The most extensive collaboration between CASE tools is achieved by transferring concept instance information. A more restricted collaboration is achieved by transferring only those concept instances belonging to a particular viewpoint.

Whether transferring all concept instances or subsets of instances, there is a question about presentation format. To use the CASE tool's native presentation format for information could add immense complexity into the information transfer problem. Some graphical and tabular formats would be complex but manageable. It is recommended that a standard syntax and semantics be used for transferring the data for concept instances between tools.

Various descriptions of software and systems concepts may be found in a number of other standards, books, and technical papers, but these do not have a broad consensus for use. In the 1175 family of standards, IEEE Std 1175.3-2004 provides a set of concepts for describing software behavior, and IEEE P1175.4 provides a set of feature types for describing dynamic computing systems behavior.

7.3.2 Management Information

Management information is information that helps use subject information. Each instance of information about a concept needs to be managed and controlled. If management and control information is transferred via a tool interconnection, the information should be in a standard format.

a) *Ownership:* Information the tool transfers about ownership or custodianship of the subject system's master information.

b) *Configuration and change information:* Information the tool transfers for version and configuration control of the subject system information. An example of this kind of information would be the version number of an instance of a concept or the release number for a software program. See IEEE Std 828-1998 [B4].

c) *Project management information:* Information the tool transfers for tracking the subject system's information. An example of this kind of information would be the date on which an instance of the concept was approved. See IEEE Std 1058-1995 [B10].

d) *Measurement (metrics) information:* Information used to quantify subject system information. An example of this kind of information would be the count of data items included in a data dictionary. See IEEE Std 982.1-1988 [B7] for information relating to metrics.

7.3.3 Semantic information

Semantic information is the information that represents the meaning of transferred data. Communicating tools cannot process information the same way unless they have exactly the same interpretation of the data. Two tools have a shared interpretation of the data if both tools can make similar inferences about the subject system from the data. Similarity between each tool's inferences is the test for demonstrating the shared interpretation.

Communicating tools may transfer data that satisfies syntactic requirements, yet they may interpret the data in totally different ways. For example, many tools hold definitions of a "state" in a state model, but what does "state" mean, and how should that definition be represented? If the "state" is a control state, it is a label associated with a list of operations available for execution at an instant of time. If the "state" is a data state, it is a collection of data elements and their values at a particular instant of time.

Removing such ambiguities requires understanding how different tool producers interpret the different methodologies and also understanding the ontologies by which they describe "stateful" behavior. This is semantic information. Many methodologies and their notations are not mathematically precise, so different interpretations of their meanings are possible. Different interpretations are possible and translations are ambiguous.

For two tools to share an interpretation of transferred data, either one tool needs to process multiple interpretations, or both tools need to process multiple interpretations (and choose the right interpretation for each transfer), or both tools need to be able to transform their information to a common interpretation.

7.3.4 Syntactic information

Syntactic information describes the physical structure or form of the information representation. Syntactic information may or may not be actually transferred among tools, but it will be used in the sending and receiving processes.

If the communicating tools interpret syntax in one and only one way, syntax information does not need to be transferred among tools. If the communicating tools transfer information that describes the syntax, the information is said to have self-describing syntax. ISO/IEC 8824-1:2002 [B19] and ISO/IEC 8825-1:2002 [B20] provide the standards necessary for defining information with self-describing syntax. Together ISO/IEC 8824-1:2002 [B19] and ISO/IEC 8825-1:2002 [B20] describe the following information:

— Atomic data elements (bits) and how those atomic data elements will be grouped for interpretation (coded) into primitive data values such as characters, character strings, and numbers.

— How primitive (unstructured) data items may be grouped together into structured data items such as words or fields.

— The delimiters (characters, symbols, or counts of characters) that separate primitive data items.

— How structured data items may be grouped together to form data items of larger structures such as lists, records, files, tables, and databases.

7.4 Tool–Tool Interconnection Profile

The context for sharing information between CASE tools is established in part by coordination and consistency requirements for shared data management, in part by the mechanisms for sharing the data, and in part by the semantic agreements for converting the data transferred from one tool's viewpoint to another.

Identifying the mechanisms, processes, and syntax for transferring data between tools, and documenting the semantic agreements for interpreting the classes of information contained in that data together provide several benefits as follows:

— The errors in transfer due to a single data name having differing interpretations between tools may be prevented.

— The procedures for tracing data propagation, and for maintaining and assuring consistency, may be automated.

— The requirements for infrastructure glue between tools may be derived.

— The potential for leveraging development information into additional tools and processes may be determined.

The Tool–Tool Interconnection Profile is comprised of the information identified in 7.4.1, 7.4.2, and 7.4.3. The information required in this profile is expected to be available from documents such as project or organization documents, vendor documents, training material, books, articles published in the archival literature, professional or industry standards, or international standards. Completion of this profile needs only to cite the appropriate information in these sources. The purpose of this profile is to collect into one place, which may be a tool information repository, all references to appropriate documents that characterize each of the tool interconnection features in this context for a particular CASE tool. What is important is that feature characterizations are written down or otherwise recorded and are readily available as references to members of the organization.

7.4.1 Tool–Tool Interconnection Profile metadata

The Tool–Tool Interconnection Profile is identified by the following metadata:

a) *Tool identification:* Provide an unambiguous reference to the tool being profiled. This may include the manufacturer's or supplier's name, the common name of the tool, and a release label.

b) *Organization:* Provide a reference to the organization or project that sets the context for this profile.

c) *Date of profile:* Provide the "truth" date (i.e., true-as-of date) for this profile.

7.4.2 Linkage perspective

The linkage perspective identifies mechanisms, flow controls, and collaboration strategies for synchronizing the data content of the tool with other interconnected tools. The linkage perspective is characterized by the following:

a) *Export transfer mechanism:* Describe the mechanism(s) by which the tool exports data.

b) *Export transfer control:* Describe how export data transfers are initiated and controlled.

c) *Export collaboration control:* Describe the collaboration protocols by which the data in other tools are made consistent with the subject tool. Describe the protocols by which the subject tool incorporates the services of other tools (send and receive).

d) *Import transfer mechanism:* Describe the mechanism(s) by which the tool imports data.

e) *Import transfer control:* Describe how import data transfers are initiated and controlled.

f) *Import collaboration control:* Describe the collaboration protocols by which data in the subject tool are made consistent with other tools. Describe the protocols by which other tools incorporate the services of the subject tool (receive and send).

7.4.3 Information perspective

The information perspective identifies the substantive information content that can be exchanged with the tool. The information perspective is characterized by the following:

a) *Export subject information:* Describe the concepts for subject system description, of which instances are exported by the tool. Include all concept names, characteristics, and relationships with other concepts that are instantiated for the subject system and for which data are exported. Identify and describe common viewpoint(s) with which these concepts are associated. Identify and describe any presentation forms that are exported.

b) *Export management information:* Describe the various items of management information that are exported. Indicate how these are related to the subject information, such as measures, attributes, or supplements.

c) *Export semantic information:* Describe the interpretation rules according to which other tools can recover concept instance information about the subject system from the exported data (e.g., slot-structures, entity-relation types). Indicate whether a description of these interpretation rules is sent to receiving tools, or whether they are implemented by a fixed interoperability agreement to be satisfied by other tools. If a rules description is sent, explain how it is encoded for transfer. Indicate whether this transfer is solicited or directed.

d) *Export syntax information:* Describe the codes and structures according to which other tools can scan and parse exported data (e.g., ASN.1, HTML, XML, comma-separated-ASCII-value). Indicate whether a description of these codes and structures is sent to receiving tools, or if they are implemented by a fixed interoperability agreement to be satisfied by other tools. If a description of codes and structures is sent, explain how it is encoded for transfer. Indicate whether this transfer is solicited or directed.

e) *Export other information:* Identify and describe any other information that may be exported from the tool and explain its value to a receiving tool.

f) *Import subject information:* Describe the concepts for subject system description, of which instances are imported by the tool. Include all concept names, characteristics, and relationships with other concepts that are instantiated for the subject system and for which data are imported. Identify and describe common viewpoint(s) with which these concepts are associated. Identify and describe any presentation forms that are imported.

g) *Import management information:* Describe the various items of management information that are imported. Indicate how these are related to the subject information, such as measures, attributes, or supplements.

h) *Import semantic information:* Describe the interpretation rules according to which other tools can encode concept instance information about the subject system in the imported data (e.g., slot-structures, entity-relation types). Indicate whether a description of these interpretation rules is received from sending tools, or whether they are implemented by a fixed interoperability agreement to be satisfied by other tools. If a rules description is received, explain how it is decoded after transfer. Indicate whether this transfer is solicited or expected.

i) *Import syntax information:* Describe the codes and structures according to which the tool scans and parses imported data (e.g., ASN.1, HTML, XML, comma-separated-ASCII-value). Indicate whether a description of these codes and structures is received from sending tools, or if they are implemented by a fixed interoperability agreement to be satisfied by other tools. If a description of codes and structures is received, explain how it is decoded after transfer. Indicate whether this transfer is solicited or expected.

j) *Import other information:* Identify and describe any other information that may be imported from other tools and explain limitations on tool use if that information is not supplied.

Annex A

(informative)

Bibliography

[B1] ANSI/EIA-632-1999, Processes for Engineering a System.[8]

[B2] IEEE 100™, *IEEE Authoritative Dictionary of IEEE Standards Terms,* Seventh Edition.[9, 10]

[B3] IEEE Std 730™-2002, IEEE Standard for Software Quality Assurance Plans.

[B4] IEEE Std 828™-1998, IEEE Standard for Software Configuration Management Plans.

[B5] IEEE Std 829™-1998, IEEE Standard for Software Test Documentation.

[B6] IEEE Std 830™-1998, IEEE Recommended Practice for Software Requirements Specifications.

[B7] IEEE Std 982.1™-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.

[B8] IEEE Std 1012™-2004, IEEE Standard for Software Verification and Validation.

[B9] IEEE Std 1016™-1998, IEEE Recommended Practice for Software Design Descriptions.

[B10] IEEE Std 1058™-1995, IEEE Standard for Software Project Management Plans.

[B11] IEEE Std 1074™-2006, IEEE Standard for Developing Software Life Cycle Processes.

[B12] IEEE Std 1175.1™-2002, IEEE Guide for CASE Tool Interconnections—Classification and Description.

[B13] IEEE Std 1175.3™-2004, IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying Software Behavior.

[B14] IEEE Std 1220™-2005, IEEE Standard for Application and Management of the Systems Engineering Process.

[B15] IEEE Std 1233™-1998, IEEE Guide for Developing System Requirements Specifications.

[B16] IEEE Std 15288™-2004, Adoption of ISO/IEC Std 15288:2002.

[B17] IEEE/EIA 12207.0™-1996, IEEE/EIA Standard: Industry Implementation of International Standard ISO/IEC 12207:1995.

[8] ANSI publications are available from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).
[9] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).
[10] The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

[B18] ISO/IEC 7498-1:1994, Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model.[11]

[B19] ISO/IEC 8824-1:2002, Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation.

[B20] ISO/IEC 8825-1:2002, Information technology—ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

[B21] ISO/IEC 12207:1995, Information technology—Software life cycle processes.

[B22] ISO/IEC TR 14252:1996, Information technology—Guide to the POSIX® Open System Environment (OSE).

[B23] ISO/IEC 15288:2002, Systems engineering—System life cycle processes.

[B24] Technical Report CMU/SEI-2002-TR-011, CMMI℠ for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Continuous Representation, Carnegie Mellon University (CMU), Software Engineering Institute,[12] http://www.sei.cmu.edu/publications/documents/02.reports/02tr011.html.[13]

[11] ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[12] The CMU Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

[13] At the time the referenced document was published, CMMI was a service mark; at the time of the publication of this recommended practice, CMMI is a registered trademark.