SLOVENSKI STANDARD

SIST-TP CLC/R217-014:2004

september 2004

Okvir EDA - Informacijski model za naloge in seje

EDA framework - Task and session information model

ICS 35.020

Referenčna številka
SIST-TP CLC/R217-014:2004(en)

CENELEC

R217-014

REPORT

November 1996

English version

EDA framework - Task and session information model

This CENELEC Report has been prepared by the Technical Committee CENELEC TC 217, Electronic Design Automation (EDA). It was approved by TC 217 on 1996-01-30 and endorsed by the CENELEC Technical Board on 1996-07-02.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Ref. No. R217-014:1996 E

Foreword

This report has been prepared by the Technical Committee CENELEC TC 217, Electronic Design Automation (EDA).

It was approved by TC 217 on 1996-01-30 and approved for publication by the CENELEC Technical Board on 1996-07-02.

Contents

Introduction

Within this document[1] a design environment is seen as the collection of application and framework tools and framework services that the end-user works within. A "session" is a fixed period of time during which the end-user is connected to the design system's tools, data and resources. During a session, a "task" may be created to execute one or a sequence of process steps. The execution of a "task" may continue over multiple "sessions".

The presentation and management of tools, data and processes in a framework is highly dependent on the design environment, the design problem at hand and the design methodology used to address the design problem. For example, some design problems lend themselves to a solution that involves stepping through a series of pre-specified procedures, which may favor a flow-based methodology (e.g. the qualification phase leading up to the release of a design to an ASIC vendor). Other design problems are best solved by the repeated execution of a well-defined set of analysis tools on a very small number of data objects, which favors a tool-centered methodology (e.g. analog design). Finally, there are a class of design problems that favor a data-centered methodology, where a single tool is applied repeatedly across a set of data objects (e.g. physical layout). Support for these different methodologies is required if concurrent engineering is to be effectively applied across an enterprise, each type of user interacting with the design environment in a way that meets their specific needs, but in all cases insuring that the overall design remains correct and consistent when viewed at the project level.

In spite of the various methodologies used to address specific design problems, there are opportunities for standardization. However, this standardization is only possible if there is a clear separation (i.e. interface) between the services used to provide data management and consistency and the services used to provide flow status and control. The underlying data management and consistency services that support the tool-, data- and flow-centered user views can be common among the various methodologies, as can the interface provided to allow design tools to take advantage of these services (via a process of adapting tools into a framework called *encapsulation*). Support for the various methodologies then becomes a matter of presenting the correct tool execution view to the user, depending on the specific design problem, via framework tools such as a graphical user interface.

Building tool-, data- and flow-centered execution views upon a common set of underlying data management services provides the unification of data and flow management that allows a complete, yet flexible, framework-based Session Model solution to be developed.

---

1. There are three versions of it:
Version 0.1: first draft outline based on CFI Document "Task and Session Information Model" Version 0.92, May 2, 1994
Version 0.2: result of the Cenelec TC 117 / WP 4 internal review of version 0.1, June 24, 1994
Version 0.3: formal modifications, June 20, 1995

1 Scope

This document will focus on delivering a specification for a set of services that can be built to provide a consistent and homogenous system environment for the end-user. Included as part of the specification will be an Information Model that defines this execution environment in terms of the following basic objects[1]:

- User
- Group
- Workspace
- Role
- Session
- Computing Resource
- Task
- Task Template
- Application Kind
- Application Object
- Execution Object
- Flow Template
- Task Template Proxy

The behavior of these objects will also be defined in the Information Model. The result will be a set of standard objects, and a standard interface where the function of these objects is accessed. Note that these functions are intended to be used by CAD framework implementors to design framework tools to perform any necessary work within the design system environment. It is not the intent to define the underlying implementation based on the Information Model, or to provide an implementation of framework tools based on the model.

1. These objects will be defined in more detail later in this document.

2 Information Model

The Session Model is the focal point for all information about the user, the collection of objects and resources available to the user, and the state of the connection between the user and framework over a finite period of time. When the user disconnects from the framework the Session ends with some of the information kept persistent and available for the next framework connection.

The Session is owned by one, and only one, active user. Information about the user is contained in the User object that owns the Session. Multi-user environments are a collection of Sessions. Communication may be within and between Sessions.

A single Session may have access to and use computing resources from any mixture of hardware platforms, peripheral devices, operating systems, database management systems, and other system enablers. A Session may also include any number of frameworks each with their own data management and tool management facilities. In all possible configurations of these environments there is one, and only one, Session owned by one, and only one, user. Such environments of computing resources associated with a single user and session may exist in parallel with the other type of environment - design environments composed of multiple users and sessions.

The Session Model represents user objects and resources. User objects are workspaces, tools, data, and tasks. The collection of tools available to a user include framework tools, application tools, system tools, and hardware devices such as printers. All tools within the Session's collection of computing resources, that the user owning the Session has access authorization to, are available to the Session subject to context constraints.

Data Management Objects are objects the user is authorized to access from resources such as data and communication servers and include things such as libraries, repositories, and mail. Some of these Application Objects may be in "accessed" or "checked-out" state. This state may persist between Sessions. Information about data is contained in the Application Object.

Task objects represent the binding of Application Objects and possibly other Session Model objects to a Task Template on an abstract level. This resolves to either a Flow or a Tool Template and specifies how to process the objects with one or more tools. Task objects may persist over Sessions. Information about a task is contained in the Task object.

Information about computing resources is contained in the Computing Resources object. This information defines system characteristics associated with computing resources such as state, location, and variable settings (e.g., environment, system, X-defaults, language, etc.).

A Workspace is provided for a team or a user for gathering together all the data, tasks, and resources used in a particular project. All Session Model objects, except the Session itself, may be reserved in such workspaces for exclusive treatment.

2.1 General Concepts

2.1.1 Access

Authorization to access objects in the Session Model is determined by:

- Role - specifies access to methods of an object type
- Group - specifies access to instances of objects

There are two ways in which access may be defined:

- Positively by allowing access by anyone to any object until access is explicitly restricted by Group
- Negatively by explicitly specifying Group access to allow any access beyond that allowed to the creating User.

In the Session Model each object type (or entity) may have its own default access method, negative or positive (unrestrict or restrict). The default range of access by object type may then be from a single user to all users with the appropriate Role.

2.1.2 Version

All Session Model objects, except Session, Workspace, Task, and Execution Object, may have versions. All relationships and attributes are kept with the versions.

2.1.3 Transaction

The Session Model may optionally support transactions composed of operations on Session Model objects. The levels of support can range from:

- single to multiple Sessions
- single to multi-level nesting
- serial and parallel transactions

2.1.4 Persistence

Every TSM entity is persistent.

2.2 Illustration

The relationships have names that express a direction. This is true only for a conceptual level. Actually they are all bidirectional and can be used the other way round, too. That is why some relationships seem to be missing, e.g., Group contains User.

The cardinality is given by the following syntax:

(min source .. max source : min target .. max target)

Sometimes the range min .. max may be exactly 1.

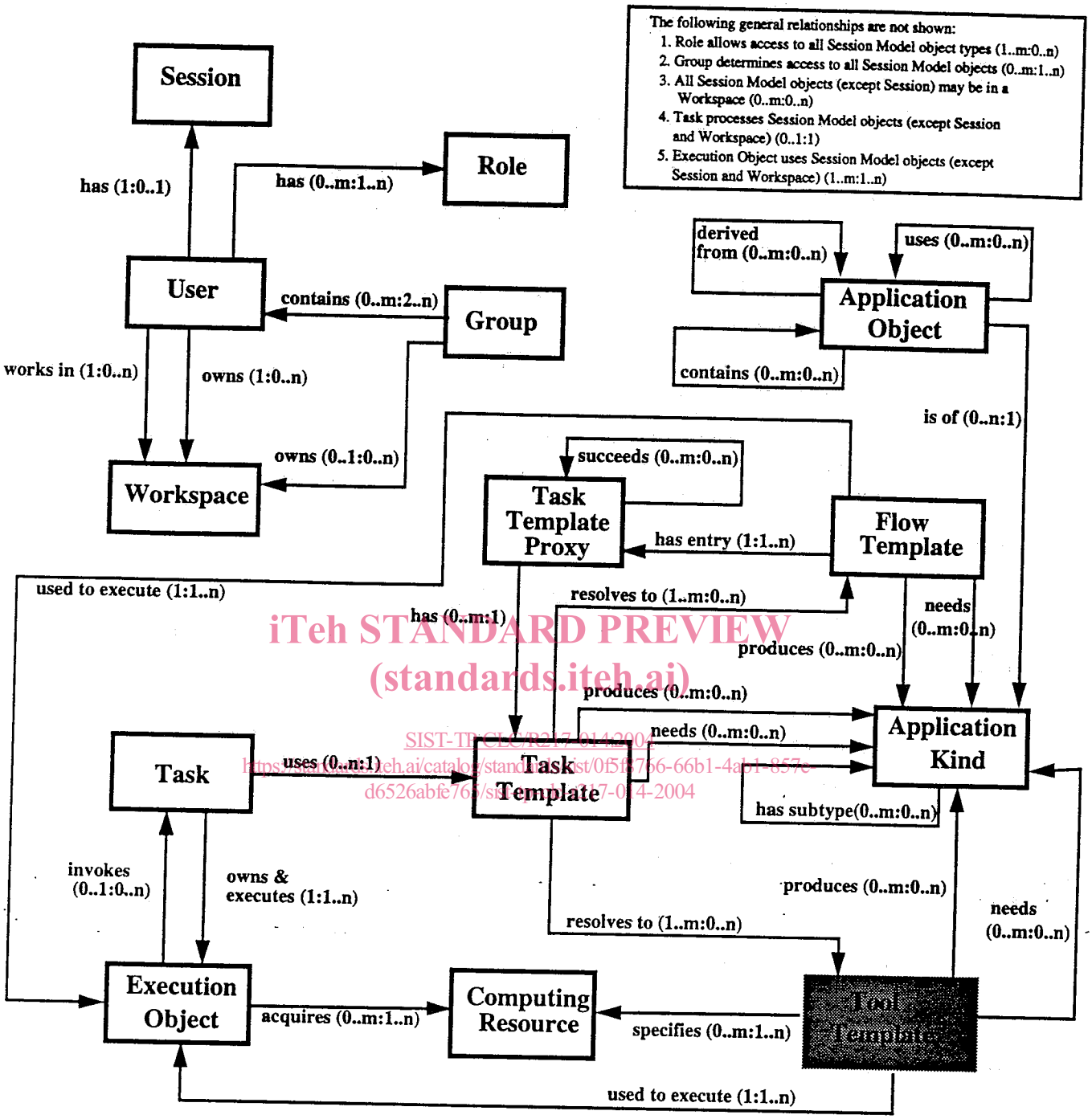

Figure 1: Task and Session Model

2.3 Description

All entities in the Session Model have the following attributes and methods:

Attributes: Name, Properties, Access State

Methods: Create, Destroy, Get Attribute, Set Attribute, Query Relationship, Create Relationship, Destroy Relationship

In the entity descriptions below only unique attributes and methods are listed.

2.3.1 Session

The Session object represents an active user and contains state and history information. The Session state information indicates whether the Session is connected, suspended, locked in initialization, termination, or update processing.

A subset of Session information which users expect to be retained between connections is persistent. The currency of this information must be sufficient to provide persistency over synchronous and asynchronous (including abrupt system failure) terminations.

Attributes

Availability: enabled/disabled

Execution State: activated, inactive, locked

Mode: foreground/background

Event Log (optional): a pointer to session history information

Relationships

User has Session (1:0..1)

Methods

2.3.2 User

Information is accessed when the user connects to the framework, with a name which uniquely identifies the persistent user context, to create the Session. This name could be composed of a userid and a unique qualifier.

The information in this object includes user preferences. User preferences model how the user expects the framework to behave and communicate. This includes the communication style which defines the preferred use of input devices, such as keyboards and mice, as well as presentation preferences such as window arrangement, color scheme, and national language.