CAMELEON Project

R&D Project IST-2000-30104

Title of Document:	The CAMELEON Glossary
Author(s):	G. Calvary, J. Coutaz, D. Thevenin
	L. Bouillon, M. Florins, Q. Limbourg, N. Souchon, J. Vanderdonckt
	L.Marucci, F. Paternò, C. Santoro
Affiliation(s):	UJF, UCL, CNUCE
Date of Document:	August 13th 2002
CAMELEON Document:	D1.1 Companion
Distribution:	Reviewers
Keyword List:	Glossary, Cameleon project
Version:	V1.1

CAMELEON Partners:

CNUCE Institute, Pisa, Italy

Université catholique de Louvain, Belgium

University of Grenoble, France

MTCI, Motorola, Turin, Italy

IS3-GICE, Paris, France

Title: The CAMELEON Glossary	**Id Number: D1.1 Companion**

Abstract

The CAMELEON Glossary gathers definitions for the vocabulary used within the CAMELEON project. The purpose of this glossary is to serve as a companion for reading documents produced by the consortium. It also seeks to provide the project with a reference vocabulary for discussing issues addressed in the project. This document also contains the list of mathematical definitions of concepts that are relevant and a list of used mathematical symbols.

Title: The CAMELEON Glossary	**Id Number: D1.1 Companion**

Table of Contents

1. Introduction

The CAMELEON Glossary gathers definitions for the vocabulary used within the CAMELEON project. The purpose of this glossary is to serve as a companion for reading documents produced by the consortium such as [D1.1 02]. It also seeks to provide the project with a reference vocabulary for discussing issues addressed in the project.

Because Human Computer Interaction is a multi-disciplinary domain, the definition of a term may differ from domain to domain. In the CAMELEON project, we primarily address the system perspective of HCI. As a result, terms will be defined consistently with the focus of the project.

Every partner and SIG member of the CAMELEON project is welcome to suggest revisions to the glossary.

1.2. Instructions

Entries in the glossary may be words or expressions (e.g., "task model"). They are listed in alphabetical order. The glossary is intended to be written in the "dictionary" style.

1.3. Flags

Special flags are used to express exceptions. They include:

- [comment]: any useful comment.
- [dangling]: a definition that has not received a common agreement yet in the consortium.
- [obsolete]: an obsolete definition. It must be followed by a new definition.
- [ref]: the definition draws upon or replicates widely-accepted meanings. In this case, the source reference is denoted as [ref] and is listed in the "References" section of the document.
- [see also]: a reference to a related concept.
- [syn]: the term is a synonym. It is followed with the term it is synonym of.
- < to be expanded as needed>

1.4 Abbreviations

A list of abbreviations can be found at the end of the document in Section 3.

2. Terms

Abstract Interaction Object (AIO). (a) An abstraction of a set of CIO's with respect of a set of properties. (b) Within the Arch software architecture reference model, it belongs to the Logical Presentation functional component. [syn] Abstract interactor, logical interactor.

Abstract interactor. [syn] Abstract Interaction Object.

Abstract User Interface (Abstract UI). A canonical expression of the renderings and manipulation of the domain concepts and functions in a way that is independent from the concrete interactors available on the targets. It is expressed in terms of workspaces (as in ARTStudio [Thevenin 01]), or in terms of Presentation Units (as in SEGUIA [Vanderdonckt 93, 99] – See http://www.isys.ucl.ac.be/bchi/research/seguia.htm), or in terms of Presentations (as in TERESA [Paternò 99]).

Abstract task. In ConcurTaskTrees [Paternò 99], a task that has subtasks belonging to different task categories (e.g., subtasks that are allocated to the user and to the system).

Abstraction. (a) Process that transforms description into description whose semantic content and scope are richer/higher than the content and scope of the initial information. (b) Result of the process of abstracting. In the context of reverse engineering, elicitation of descriptions that are more abstract than the descriptions that serve as input to this process. [comment] Opposite of reification. [Amodeus 95]

Adaptability. [dangling] (a) Capacity of a UI to adapt its behaviour according to a set of predefined options. (b) Capacity of a UI to adapt its behaviour from an explicit human intervention.

Adaptable User Interface (Adaptable UI). A UI that supports adaptability.

Adaptation. (a) Process of adapting. (b) Result of the process of adapting.

Adaptive User Interface (Adaptive UI). A UI that supports adaptivity.

Adaptivity. Capacity of a UI to adapt without any explicit human intervention.

Application task. In ConcurTaskTrees [Paternò 99], a task executed by the system. For example, the compilation of C code is an application task. [syn] System task.

Archetypal (domain/context of use/adaptation) model. A declarative (domain/context of use/adaptation) model that serves as input to the design of a particular interactive system.

Backward recoverability. Capacity of the system to provide the user with an undo facility to return to a previous state [Amodeus 95].

Basic task. A task that can no longer be decomposed, i.e., if it were decomposed further, it would be expressed in terms of physical actions. [syn] Elementary task.

Basic window. Logical window from which it is possible to navigate to the other windows belonging in a same Presentation Unit [Vanderdonckt 99].

Browsability. Capacity of the system to provide the user with means to make perceivable different portions of the system functional state. (By modifying the presentation state, the user may access different portions of the state of the functional core.) [Amodeus 95].

Central domain concept. A domain concept needed by the user to accomplish a task.

Cluster (of platforms). A composition of elementary platforms and/or of clusters. It may be homogeneous or heterogeneous.

Co-domain. Range of a function. Set of possible mappings ('solutions') for a particular function. For example, the co-domain of the reification function from the abstract user interface (the domain in this case) is the set of possible concrete user interfaces. [see also] Function, Mapping.

Component. Part of a whole. It can be instantiated as a software module, a subsystem, an agent, an interactor, an abstraction, a device, etc. [Amodeus 95].

Component model.

Composite Abstract Interaction Object. An AIO is said to be composite if it can be decomposed into smaller AIO units.

Composition of functions. The composition of the function $g:A \rightarrow B$ and $f: B \rightarrow C$, written as $f \circ g$, is a function such that any element of the co-domain of g corresponds to an element of the domain of f. In this definition, f and g can be composed because co-*domain(g)=domain(f)*. The reverse composition $g \circ f$ is not necessarily valid since co-*domain(f) can be different from domain(g)*.

Compound workspace. In ArtStudio [Thevenin 01], a workspace composed of multiple workspaces. [syn] Presentation Unit (PU).

Consistency. A criteria applied frequently to increase the predictability of a UI. Consistency allows the user to generalize from specific situations to similar situations. But it is difficult at times to determine, at design time, which situations a user will consider similar or dissimilar [Gram 96].

Concrete Interaction Object (CIO). An entity of the UI world that users can perceive (e.g., text, image, animation) or manipulate (e.g., a push button, a list box, a check box). A widget provided by a toolkit. [syn] Physical interactor, physical interaction object.

Concrete User Interface (Concrete UI). (a) A concrete physical interactor-dependent expression of the UI. (b) In ARTStudio [Thevenin 01], a simulation of the final UI than runs only within a multi-target development environment.

Configuration. [dangling] A particular reification of presentation and dialog models for a specific UI for a given context of use.

Context. An all-embracing term for which there is no consensual definition. To be operational, context can only be defined in relation to a purpose, i.e., a finality. For the purpose of the CAMELEON project, context is a short cut for "context of use". [syn] Context of Use.

Context aware UI : a User Interface that can detect changes in the context of use.

Context change. A short cut for "change in the context of use" as well as for "Target change".

Context dependent (entity). An entity, e.g., a description, whose nature is specific to a particular context of use.

Context dependent task model. A task model specific to a particular context of use.

Context independent (entity). An entity, e.g., a description, whose nature is shared by all of the contexts of use considered for the entity.

Context independent task model. A task model that integrates tasks and transitions that are common to all of the contexts of use envisioned for the system.

Context of interaction. [syn] Target.

Context of use. [syn] Target.

Context semi-dependency (of an entity). Property of an entity, e.g., a description, whose nature accommodates more than one context of use, but not all of the contexts of use considered for that entity.

Context sensitive UI. A context aware UI, which, in addition, can react to changes of the context of use. A UI that is adaptable and/or adaptive to multiple contexts of use.

Context sensitivity (of an entity). Capacity of an entity to change the values of its attributes depending on context change.

Core configuration (of resources). An immutable set of hardware and software resources (e.g., a laptop, a PDA, the Intel Personal Server [Want 01]).

Core resources. Software and/or hardware resources packaged as a core configuration.

Corrective decoration. A decoration used by designers to override standard options of the multi-target UI development environment. Corrective decorations are reactive.

Cross-platform UI. [syn] Multi-platform UI.

Customizability. Capacity of the user interface to support adaptability and/or adaptivity.

Decoration. Kind of information attached to a description element. A way to modify the interpretation of the description element without modifying the element per se.

Description. Any representation of a real or imagined system or entity for a particular purpose. [syn] Model.

Design recovery. [dangling] Process of recovering UI design options and models from existing source code (for example, by code static analysis, by using dynamic analysis, by behavioral analysis, by program understanding), documentation analysis, trace examination, code instrumentation,...Effective design recovery implies a thorough knowledge of the domain of discourse, information external to the UI source code, and deductions from it.

Device. Physical artefact used by a system to acquire (input device) or deliver (output device) information. Examples include keyboard, loudspeaker, pen, mouse. Short cut for Physical device.

Device assignment. Relation between a device over a state and a non empty subset of expressions of an interaction language. A device d is assigned in state s to a set E of expressions of a language l, if it does not exist any device equivalent to d over s and E. Assignment is permanent if the relation holds for any state. Assignment is total if the relation holds for E equals to the set of expressions that define l. For example, a mouse is permanently assigned to the expression of window resizing in the direct manipulation interaction language. [Amodeus 95]

Device equivalence. Relation between a non empty set of devices over a state and a non empty set of expressions in an interaction language. Devices in a set D are equivalent over a state s and a non empty set E of expressions in an interaction language L, if all of the expressions of E can be elaborated using either one of the devices in D. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holds for E equals to the set of expressions that define L. For example, keyboard and microphone can be totally and permanently equivalent over natural language. [Amodeus 95]

Device redundancy. Relation between a set of devices over a state and an expression of an interaction language. Devices of a set D are used redundantly in some state s for an expression e of a language l, if these devices are equivalent over s and e, and if they are used simultaneously to express e. For example, the user can spell a character using the microphone and type in the same character. [Amodeus 95].

Device complementarity. Relation between a set of devices over a state and a non empty subset of expressions of an interaction language. Devices of a set D are complementary over a state s and a non empty set E of expressions of a language l, if E can be partitioned such that for each partition Ep of E, it exists a device d of D assigned over s and Ep. Complementarity is permanent if the relation holds for any state. Complementarity is total if the relation holds for E equals to the set of expressions of l. Language complementarity is best illustrated by spoken natural

languages where concept names must be typed in. For example, in Munix, a multimodal user interface for Unix, commands that involve a file name such as remove, can be expressed using the microphone for the command name and options while file names must be elaborated with the keyboard. [Amodeus 95].

Directive decoration. Decoration used when it corresponds to rules that cannot be easily expressed in terms of general-purpose inference rules. For example, suppose the multi-target development environment includes the following generation rule: "any domain concept of type Integer must be represented as a Label in the Concrete UI". If the designer wants the temperature domain concept to be represented as a gauge, a directive decoration can be attached to that particular concept. Directive decorations are pro-active.

Dialogue Controller (DC). In the Arch software architecture reference model [Arch 92], denotes the software component that is in charge of task actions sequencing.

Dialogue (of the User Interface). An ordered set of physical actions between the user and the interactive system.

Dialogue model. [dangling] An abstract description of the actions, and their possible temporal relationships, that users and systems can perform at the user interface level during an interactive session [Paternò 99].

Distribution (of the User Interface). The allocation of the user interface across the devices of a cluster of platforms. The granularity for distribution is one of (in decreasing order): application level (e.g., full replication of the UI on target platforms), workspace level (e.g., windows and panels), domain concept level, pixel level.

Distributed User Interface (distributed UI). A user interface whose components are allocated (statically or dynamically) to different devices of a cluster of platforms.

Domain. See Mapping.

Domain concept. (a) A concept relevant to users to accomplish tasks in a particular domain. (b) An element of the domain ontology.

Domain model. A description of the domain concepts and their relationships.

Dynamic multi-target User Interface (Dynamic MUI). A multi-target UI adaptable and/or adaptive at run time, e.g., during a session. [syn] Context-sensitive UI.

Elasticity. [dangling] Capacity of a UI to support different contexts of use witloof any reconfiguration. Plasticity should be distinguished from elasticity with respect to the predefined usability properties.

Elastic UI. [dangling] UI that supports elasticity.

Elementary Abstract Interaction Object. An AIO that cannot be decomposed any further. [syn] Simple AIO.

Elementary platform. A platform from the resources of which it is not possible to compose two platforms.

Elementary task. [syn] Basic task.

Elementary workspace. In Artstudio [Thevenin 01], a workspace that cannot be decomposed further.

Enabled task Set (ETS). The set of tasks that are enabled over the same period of time according to the constraints indicated in the task model [Paternò 02].

Entry point. Within the multi-target reference framework [Calvary 02], any level of abstraction in the reification process from which the development of a multi-target UI is initiated.

Environment. A short cut for Physical Environment. [syn] Physical environment.

Epilogue. (a) A closing functional portion of the execution of a reaction to context change. It includes the restoration of the execution context (e.g., resuming suspended task). (b) In ARTStudio [Calvary 01, Thevenin 01], a reference to a function of the Functional Core after the execution of a task.

Equivalence. A relation that is reflexive, symmetric and transitive. [Amodeus 95].

Evolution model. A model that specifies the actions that should be undertaken at run-time by the multi-target user interface, when entering and leaving a particular context of use.

Executable User Interface. A UI ready for execution. It is expressed as compiled or interpreted code.

Extension resource. A hardware or software resource that can be added to (or removed from) a core configuration or a platform (e.g., external keyboard, mouse).

Factorization. An operation, which from a set of target-specific descriptions of class *X*, produces a new description of class *X* composed of a context-independent part (i.e., shared by all the targets) and of context-dependent parts specific to each target.

Factorization decoration. A decoration that expresses exceptions to the nominal case used as the reference in the process of multi-targeting.

Final User Interface. The UI produced at the very last step of the reification process supported by a multi-target development environment. It is expressed as source code.

Final description. A description produced by a multi-target development environment that is not reified any further.

Final model. [syn] Final dscription.

Fission. (a) A computation of a process abstracting/reifying an information type into a collection of different information types to be transferred to a set of processes. (b) Decomposition of an information type at some level of abstraction into multiple information types of the same level of abstraction [Amodeus 95].

Flexibility of the User Interface. A capacity of the UI to provide users with multiple ways of achieving tasks. Is refined into a number of properties.

Forward engineering. The process of developing a software product. The opposite of reverse engineering.

Framework. A structure intended to guide and support some human process.

Function. If A and B are two non-empty sets, then a function f in A x B exists if it associates each element in A with one and only one element in B.

Functional Core (FC). In the Arch software architecture reference model, implements the domain-dependent concepts and functions of an interactive system. [Arch 92]

Functional Core Adaptor. In the Arch software architecture reference model, accommodates various forms of mismatch between the Functional Core and the user interface per se of an interactive system. [Arch 92]

Fusion. (a) Computation of a process abstracting/reifying a collection of information types received from distinct processes into a different information type to be transferred to another process. (b) Composition of multiple information types at some level of abstraction into a single information type of the same level of abstraction [Amodeus 95].

Goal. A desired modification of the current state or an inquiry to obtain information on the current state.

Grouping (of interactors). Relationships among interactors indicating that they are logically connected.

Heterogeneous cluster. A cluster of elementary platforms, whose classes are different (e.g., a cluster composed of a PC and a PDA).

Hierarchy (of interactors). Relationship among interactors indicating that they have different level of importance for the user.

High-level task. A task that can be decomposed into a set of sub tasks.

Homogeneous cluster. A cluster of elementary platforms, whose classes are identical (e.g., a cluster composed of a PC's).

Honesty. A property that the presentation of the system renders its functional state appropriately (e.g., it does not distort the functional state) and in a way that is understood correctly by the user [Amodeus 95].

Hybrid multi-target User Interface (Hybrid MUI). A multi-target UI composed of pre-computed components and dynamic components.

Idempotence. Property of a function to reproduce its co-domain when applied multiple times. Some functions have the special property that applying them more than once to the same co-domain produces no further change after the first application. For instance, f(f(x)=f(x). To ensure a true bidirectional engineering of UI at any level of abstraction, the composition of all the functions involved and their corresponding inverse function should be idempotent [Bouillon 02c]. If we define *rei(fi)* as the reification function from the concrete interface to the final interface, *abs(ci)* as the reverse engineering process(abstraction function) from the final interface to the concrete UI and f as the composition of these two functions, then applying once or several times the function *f* to a final interface will not change the co-domain of the function *f*.

Initial description. Within a multi-target development environment, a description provided by a human designer/developer.

Initial model. [syn] Initial description.

Interaction device. [syn] Interaction resource, physical device.

Interaction language. Language used by the user or the system to exchange information. A language defines the set of all possible well-formed expressions, i.e., the conventional assembly of symbols, that convey meaning [Amodeus 95].

Interaction language assignment. Relation between an interaction language over a state and a non empty subset of conceptual units of a system. An interaction language l is assigned in state s to a set of conceptual units C, if it does not exist any interaction language equivalent to l over s and c. Assignment is permanent if the relation holds for any state. Assignment is total if the relation holds for C equals to the set of conceptual units of the system [Amodeus 95].

Interaction language complementarity. Relation between a set of interaction languages over a state and a non-empty subset of conceptual units. Interaction languages of a set *L* are complementary over a state *s* and a non empty set *C* of conceptual units of the system, if *C* can be partitioned such that for each partition *Cp* of *C*, it exists a language *l* of *L* assigned over *s* and *Cp*. Complementarity is permanent if the relation holds for any state. Complementarity is total if the relation holds for *C* equals to the set of conceptual units of the system. Language complementarity is best illustrated by coreferential expressions. For example, natural language and direct manipulation are complementary over the conceptual unit “city” and any state where the specification of a city name is possible: “flights from this city” and selection of a city name through direct manipulation [Amodeus 95].

Interaction language equivalence. Relation between a set of interaction languages over a state and a non empty subset of conceptual units of a system. Interaction languages of a set *L* are equivalent over a state s and a non empty set *C*

of conceptual units of the system, if all of the conceptual units in *C* can be represented using either one of the language in *L*. Equivalence is permanent if the relation holds for any state. Equivalence is total if the relation holds for *C* equals to the set of conceptual units of the system [Amodeus 95].

Interaction language redundancy. Relation between a set of interaction languages over a state and a conceptual unit of a system. Interaction languages of a set are used redundantly in some state s for a conceptual unit *c*, if these languages are equivalent over *s* and *c*, and if they are used simultaneously to represent c. For example, a wall is represented redundantly by the system via a red line (graphics interaction language) and the message "mind the red wall!" (natural language) [Amodeus 95].

Interaction object [syn] Interactor.

Interaction resource. An input or output device used by the user to manipulate and/or observe the state of an interactive system. Examples include screens, keyboard, mouse, fingers, real world objects (such as phicons).

Interaction space. A collection of interactors that support the execution of a set of logically/semantically connected tasks. In graphical user interfaces, an interaction space can be mapped onto a window, a set of panels. [syn] Workspace, presentation unit.

Interaction task. In ConcurTaskTrees, a task performed by the user for modifying or observing the state of the interactive system

Interaction capacity (of an interactor). The general purpose interaction tasks that the interactor is able to support (e.g., selection, deletion, navigation).

Interactive system. A computational system that supports a set of tasks with the participation of one or more humans.

Interactor. [dangling] (a) An abstraction of a software component that allows users to manipulate and/or observe domain concepts and functions. (b) A computational abstraction that allows the rendering and manipulation of entities (domain concepts and/or tasks) that requires input and output resources.

Interactor Model. A description that makes explicit the properties of an interactor for a specific purpose. For example, for the purpose of multi-targeting, this description includes the representational capacity, the interaction capacity and the usage cost of the interactor.

Interface model. An interface model represents all the relevant aspects of a user interface in some type of interface modeling language. Objects typically included in a comprehensive interface model are user tasks, domain elements, users, presentation items, and dialog structures. The elements of an interface model are grouped into model components [Puerta 99]. [see also] Model component.

Introspection (of an entity). The capacity of the entity (e.g., an interactor, a software component) to export its properties and behaviour to other entities at their requests.

Inverse functions. Function defined by inversing domain and co-domain of a previously existing function. Forward and reverse engineering can be seen as two inverse functions since the four reification steps (used for the UI production) can be recovered by their corresponding abstraction processes [Bouillon 02c]. The inverse function of a function f is denoted f^{-1}. In this case, f is said to be *inversible*.

Inversible function. See inverse functions.

Level of abstraction. (a) A layer within a system whose information types are characterized by a given semantic content and scope. The lowest level of abstraction corresponds to the poorest information type with regard to scope and content. The highest level of abstraction corresponds to the richest information type with regard to scope and content. These levels as well as any level in-between depend on the perspective or the objective of the modeler and/or the modeling technique. (b) Different perspectives in the design process of a system [Amodeus 95].

Logical interaction object. [syn] Logical interactor, abstract interaction object.

Logical interactor. [syn] Logical interaction object, abstract interaction object.

Logical Presentation Component (LPC). In the Arch software architecture reference model, insulates the rendering of domain objects from the actual interaction toolkit of the target platform. It is expressed in terms of logical interactors. [Arch 92]

Logical window (LW). A composite AIO or a physical window, or a dialog box, or a panel. [syn] Elementary workspace.

Mapping. A *mapping* $M:A \rightarrow B$ is represented as a set of couples (x,y) where x belongs to A, called the domain of M, written *domain(M)* and B is called the *range of M*, written *range(M)*. A *m-to-1* mapping M associates one or more in one element in *domain(M)* with each element in *range(M)*. An *onto* mapping M associates elements of A with all elements of B.

Migration (of a User Interface). The transfer of all, or parts, of the user interface between different platforms. May occur at run time or between sessions.

Migrability (of a User Interface). The capacity of a UI to support migration.

Migrable (UI). A UI capable of migration.

Nomadic application. An interactive system that supports mobile users.

Modality (in multi-modal interaction). (a) The association of a representational system with an interaction resource. For example, the association "pseudo-natural language – microphone". "pseudo-natural language – keyboard" and "pseudo-

natural language – pen" are three different input modalities for specifying commands in natural language with distinct input devices [Nigay95]. (b) One of the human perceptual senses.

Model. [dangling] a) Any representation of a real or imagined system or entity for a particular purpose. [syn] Description. b) A simplified description of a complex entity or process.

Model component. Any element of an interface model. The basic components of an interface model are the task model, the user model, the domain model, the presentation model, and the dialog model. Interface models are referred to as partial models if they include just some of the basic components and as comprehensive models if they include all of the basic components [Puerta 99].

Multi-environment targeting. The process of supporting multiple classes of environments.

Multi-environment UI. A Multi-target UI sensitive to environments variations: It is adaptable and/or adaptive to multiple classes of environments. The users and platform classes, either are modeled as archetypes, or are implicitly represented in the system.

Multi-lingual UI: A UI able to accommodate variation of the natural language according to what is needed by the user. For example, the user can switch from one language to an-other by selecting it from a UI menu or the system can automatically set it according to a preference stated in a profile.

Multi-modal User Interface (Multi-modal UI). A UI that supports multi-modality.

Multi-modality (of a User Interface). Capacity of a system to support multi-modal interaction, i.e., the user is provided with more than one modality (simultaneously or not) to observe the system state and/or can use more than one modality (simultaneously or not) to communicate information to the system.

Multi-platform targeting. The process of supporting multiple classes of platforms.

Multi-platform UI. A Multi-target UI sensitive to platforms variations. It is adaptable and/or adaptive to multiple classes of platforms. The environment and user classes, either are modeled as archetypes, or are implicitly represented in the system.

Multi-target development environment. A set of tools that supports the development of multi-target UIs.

Multi-target Reference Framework. A conceptual framework that structures the development process of multi-target UIs.

Multi-target User Interface (Multi-target UI). A user interface that supports multiple targets (i.e., multiple types of users, platforms and environments).

Multi-targeting. The process of supporting multiple targets.

Multi-user targeting. The process of adapting to multiple archetypes of users.

Multi-user UI. A Multi-target UI sensitive to users variations. It is adaptable and/or adaptive to multiple archetypes (i.e., classes) of users. The environment and the platform, either are modeled as archetypes, or are implicitly represented in the system.

Observability. The capacity of the UI to make perceivable all of the domain concepts that are central/relevant to the task at hand so that the user is able to determine the state of the system [Amodeus 95].

Observed (domain/context of use/adaptation) model. An executable (domain/ context of use/adaptation) model that supports the adaptation process at run-time.**Ontological model**. An abstract model of the concepts (and their relationships) involved in multi-targeting. Ontological models are instantiated into archetypal and/or observed models.

Ordering (of interactors). Relationship among interactors indicating that some ordering (e.g., temporal ordering) exists among them.

Peripheral domain concept (for a task). A domain concept that is not central for the task but that may have an impact on it.

Physical action. Action performed either by the user or by the system on a physical device.

Physical device. [syn] Input device/resource, output device/resource.

Physical environment. The physical setting where the interaction takes place. It can be modeled as the set of objects, persons and events that are peripheral to the current activity but that may have an impact on the system and/or users behaviour.

Physical interaction object. [syn] Physical interactor, concrete interaction object.

Physical interactor. [syn] Physical interaction object, concrete interaction object.

Physical Presentation Component (PPC). In the Arch software architecture reference model, renders the domain concepts and functions in terms of physical interactors [Arch 92].

Plastic User Interface (Plastic UI). A multi-target user interface that preserves usability across the targets, that is, the properties elicited at the design stage are kept within a predefined range of values as adaptation occurs to different targets.

Plasticity. The capacity of a multi-target UI to preserve usability across the targets.

Plasticity domain (of a multi-target UI). Set of contexts of use that the multi-target UI covers while preserving usability.

Plasticity threshold (of a multi-target UI). The boundary of a plasticity domain.

Platform. Set of physical and software resources that function together to form a working computational unit whose state can be observed and/or modified by a human user. It may be an elementary platform or a cluster of platforms. A short cut for "Target Platform".

Portability. The capacity of a system to run on different target platforms. Covers three situations: changes in the hardware resources, changes in the software resources, and moving the user to a different platform using the 'same' system.

Pre-computed multi-target User Interface (pre-computed MUI). Results from adaptation performed during the design, implementation or installation phases of the development process of the UI: given a functional core, a specific user interface is generated for every known target.

Predicate. A boolean-valued function of the state, behaviour, or trace of a system. A predicate may represent a property [Amodeus 95].

Presentation. (a) The information provided by the user interface at a given time. (b) The process of rendering information.

Presentation Abstract Interaction Object. An AIO whose role is to present information without allowing any user interaction.

Presentation Unit (PU). [syn]. (a) Interaction space, workspace. (b) In TRIDENT [Vanderdonckt 93], a presentation environment required for carrying out an interactive task. It includes one or more Logical Windows and a basic window that gives access to these Logical Windows. For instance, a tabbed dialog box is here mapped onto a PU, which is itself decomposed into LWs corresponding to the dialog box appearances depending on the active tab; conversely, a web form can be mapped onto a composite AIO in a particular LW of a given PU.

Probe (for detecting context changes). Software mechanism that monitors and detects context changes.

Prologue. (a) Opening functional portion of the execution of a reaction to context change. It prepares the reaction: the current task is completed, suspended, or aborted; the execution context is saved; if not ready for use, the new version of the user interface is produced on the fly (e.g., a new presentation, a new dialogue sequence, etc.). (b) In ARTStudio, reference to a function of the Functional Core before the execution of a task.

Property. An observable characteristic of a system that can be described by a predicate and measurable [Amodeus 95].

Reaction (to context change). A three-step process that permits adaptation to context changes: situation recognition, reaction computation, and reaction execution.

Reaction computation. Identification of candidate reactions to context change, then selection of one of them that best fits the situation.

Reaction execution. A three-step process that consists of a prologue, the commutation to the new UI, and an epilogue.

Reachability. Property that allows that some state or set of states can be reached from a given state through user's physical actions on the system [Amodeus 95].

Recoding (of a UI). [dangling] Any functionally equivalent transformation of the source code of a final UI. Reformatting and Refactoring are particular cases of recoding.

Reconfigurability. [dangling] Capacity of a UI to support multiple targets simultaneously by offering multiple UI configurations although not preserving usability necessarily.

Recovery. The performance of actions that take a system from some `unsafe' or undesired state to one satisfying some safety property [Amodeus 95].

Recoverability. Property that the system provides the user with means to undo the effect of some action. [Amodeus 95].

Redesigning (a UI). Changes to design characteristics. Possible changes include restructuring design architecture, altering the domain model, etc. Such changes may derive from a change of context of use.

Redocumenting (a UI). From the UI source code, process of deriving another form of UI documentation such as, but not limited to, data structure, data flow diagram, I/O analysis. [see also] Reformatting.

Reengineering (a UI). Examination and the alteration of a subject interactive system to reconstitute it in a new form and the subsequent implementation of the new form. This process encompasses a combination of sub-processes such as reverse engineering, restructuring, redocumentation, forward engineering, and retargeting [STSC]. [Syn] Renovation, reclamation.

Refactoring (of a UI). [dangling] A functionally equivalent transformation of the source code of a final UI to improve its efficiency, its performance. [see also] Recoding.

Reflexivity. Reflexive functions are functions mapping an existing UI representation at a given level of abstraction to another UI representation at the same level of abstraction for the same context of use [Bouillon 02b]. In ArtStudio, reflexive functions are performed manually [Thevenin 01].

Reformatting (a UI). A functionally equivalent transformation of a source code that changes the structure of the code to improve readability.

Regenerating (a UI). The composition of the two reification steps from the abstract UI to the final UI. It is used to complete the process of retargeting. To obtain another UI for any other computing platform, regenerating can be defined similarly by the composition $regen_f = rei_c \; o \; rei_a$ so as to represent the complete process by $regen_f o \; retarg_a$ [Bouillon 02b]. [see also] Retargeting.

Reification. Transformation of a description (or of a set of descriptions) into a description (or a set of descriptions) whose level of abstraction is lower than that of the source one(s). In the multi-target reference framework [Calvary 02], the inference process that covers the inference process from high-level abstract descriptions to run-time code. Opposite of Abstraction.

Relation (between interactors). [dangling] Many to one relationship among interactors indicating that one is related to many (for example can control their disabling).

Representation multiplicity (of a domain concept). The capacity of the system to offer alternative representations for a domain concept [Gram 96].

Representational capacity (of an interactor). Types of domain concepts the interactor is able to represent (e.g., a table, an integer).

Restriction. Function obtained by restraining the domain of the initial function to those elements of the domain that satisfy a given predicate. In VAQUITA [Bouillon 02a,c, Vanderdonckt 01], constraints attached to retargeting/translation functions are defined as restriction of translations and abstractions. See http://www.isys.ucl.ac.be/bchi/research/vaquita.htm. For a given function to be applied in a specific computing platform, there is a need to define a condition to be satisfied when applying this function. For example, a constraint may be imposed when a translation between two computing platforms occurs, such as: "the target computing platform does not allow hierarchy of presentation elements deeper than a certain threshold". The WML language instantiates this constraint to 9 (not more than 9 presentation levels in decks and cards), while certain versions of cHTML instantiates this contraint to 4 (not more than 4 levels of cHTML tags). The restriction of function is therefore required. [syn] Selection. [see also] Retargeting.

Restructuring (a UI). The transformation from one presentation form to another at the same level of abstraction while preserving the subject's system external behavior (functionality and semantics) [IEEE Terminology]. The task, the functions (application model) and the domain models should remain identical. Moreover, the dialog model, which is left unchanged, also represents the external behaviour.

Retargeting (a UI). Process allowing the production of an abstract UI tailored for a particular computing platform from an final UI**.** Retargeting is done at design time. It is the composition of three functions: two successive abstractions followed by a translation for another platform ($retarg_a = trans_a \; o \; abs_a \; o \; abs_c$) [Bouillon 02c] The retargeting / translation function can be subject to restrictions due to constraints imposed by the target platform. Rather, the double abstraction up to the abstract UI level and a translation to a new context of use is independent of any computing platform.

Retasking (a UI). The change of the task model to fit a different context of use.

Revamping (a UI). The change of the user interface without modifying the functional core. Revamping makes possible to considerably modify the look and feel of a user interface. Not only the visual presentation of screens can be changed, but also the phrasing can be redefined, multimedia features can be added, or on-line documentation can be created. However, revamping doesn't imply a change in the requirements, nor re-specification. Revamping is a useful reengineering strategy for organisation wishing to adopt graphical user interfaces (GUIs) by using middleware products, which sit between the legacy system and UI. The new UI is created (manually [Csaba 97] or automatically [Stroulia 00]) at design time. Revamping is frequently performed by the designer to beautify a presentation according to users' needs.

Reverse engineering. (a) the analysis of a software system so that the software is more understandable for maintenance, evolution, and re-engineering purposes. (b) The analysis of a system to identify its components and their dependencies to extract and create system abstractions and design information. The original system is not altered. However, additional knowledge about the system is produced. Opposite of "forward engineering".

Robustness (of the User Interface). The capacity of the UI to prevent users and system errors, as well as the capacity of the UI to increase the chance of successful task accomplishment. [Amodeus 95].

Selection. [syn] Restriction. [comment] A selection is the same as a restriction (mathematical term), but is more frequently used in the domain of database engineering.

Situation recognition. The identification of the current context of use.

State. Assignment of values to names representing the observables of a system [Amodeus 95].

State vector (of a component, of a system). Names that define the state of a component, of a system [Amodeus 95].

System Task. In ConcurTaskTrees, [syn] Application task.

Target. Triple of the form "e, p, u" where e is an element of the environments set considered for the interactive system, p is an element of the platforms set considered for the interactive system, u is an element of the users set for the interactive system.

Target change. A change of at least one element of the triple "e, p, u".

Target environment. The class of environments envisioned for an interactive system.

Target platform. The class of platforms envisioned for an interactive system.

Target aware (UI). [syn] context aware UI.

Target sensitive (UI). [syn] context sensitive UI.

Target user. The archetypal set of end-users envisioned for the system.

Task. [dangling] (a) A goal, together with some procedure or set of actions that will achieve the goal. (b) An activity that should be performed in order to reach a goal.

Task category: In CTT, definition of how the performance of the task is allocated (it can be user, system, interactive, abstract).

Task domain concept. A concept identified by task analysis as relevant to the user to accomplish tasks in that domain. [syn] Domain concept.

Task model. Description of a set of tasks and their relationships

Task operator. An operator that denotes relationships between tasks.

Task presentation set. Set of tasks supported by one presentation.

Task type. Indication of the semantic effect obtained by the performance of a task (e.g. selection, show info, …).

Transient description. [syn] Transient model.

Transient model. Within a multi-target development environment, an intermediate description used in the process of producing multi-target UI's.

Transition task. When performed, a Basic task that triggers a new presentation.

Translation. The operation that transforms a description intended for a particular target into a description of the same class but aimed at a different target.

User Interface. The software component of an interactive system that allows users to observe and manipulate domain concepts. In the Arch model, it is composed of the Functional Core Adaptor, the Dialogue Controller, the Logical Presentation Component and the Physical Presentation Component [Arch 92].

UI dialogue. See Dialogue of the UI.

UI distribution. See Distribution of the UI.

UI migration. See Migration of the UI.

Task migration. Dynamic transfer of task performance between agents (whether these agents be humans or computational).

Transition UI. Feedback provided to the user during the adaptation of the UI to changes of context of use.

Usability. The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [ISO 91].

Usage cost (of an interactor). Measures the system resources as well as the human resources the interactor requires.

User. A short cut for "target user".

User model. A description of a set of elements that characterise user preferences and knowledge. It can be dynamically updated.

User task. In ConcurTaskTrees, a task performed by the user without the interactive system (i.e., an internal cognitive activity, such as making a decision).

Workspace. [syn] Interaction space.

3. Abbreviations

AIO: Abstract Interaction Object.

CCT: ConcurTaskTrees

CIO: Concrete Interaction Object.

DC: Dialogue Controller.

FC: Functional Core.

FCA: Functional Core Adaptor.

LPC: Logical Presentation Component.

LW: Logical Window.

MUI: Multi-target User Interface.

PPC: Physical Presentation Component.

PU: Presentation Unit.

UI: User Interface.

4.Mathematical definitions

Composition of functions If $f:A \rightarrow B$ and $g: B \rightarrow C$ are two functions, then the *composition of f and g*, written as *f o g*, is defined as:

$$f \circ g = \{(x,y) \dagger \forall x \in A, \exists y \in C: y=g(f(x)) \}$$

that is $f \circ g = \{(x,y) \dagger \forall x \in A, \exists y \in C, z \in B : y=g(z) \text{ where } z=f(x)\}$. In this definition, *f* and *g* can be composed because *domain(g)=co-domain(f)*. The reverse composition *g o f* is not necessarily valid since *domain(f)* $\neq$ *co-domain(g)*.

Equivalence. A relation that is reflexive, symmetric and transitive. Relation *R* is transitive iff $R(x,y) \wedge R(y,z) \Rightarrow R(x,z)$. Relation *R* is symmetric iff $R(x,y) \Rightarrow R(y,x)$. Relation *R* is reflexive iff $R(x,x) \forall x \in R$.

Idempotence $\forall fi \in FI , \forall i \in \{1,\ldots,n\}: fi = f(fi) = f(f(fi)) = \ldots = f_{\ldots \text{i times}}(f(fi)) = f \circ f \circ_{\text{i times}} f(fi) \Rightarrow f$ is idempotent. For instance, $f = (rei_f \circ abs_c) \Rightarrow f$ is idempotent. Similarly, $g = (rei_c \circ abs_a)$ and $h = (rei_a \circ abs_{tm})$ are idempotent.

Inverse function Let *f* and *g* be two functions. If f(g(x)) = x and g(f(x)) = x, then *g* is the *inverse of f* and *f* is the *inverse of g*. The inverse function of a function *f* is denoted f^{-1}. In this case, *f* is said to be *inversible*.

Mapping. A *mapping* $M:A \rightarrow B$ is represented as a set

$$M = \{(x,y) \dagger x \in A \text{ and } y \in B\}$$

where A is called the *domain of M*, written *domain(M)* and B is called the *range of M*, written *range(M)*. A *m-to-1* mapping *M* associates one or more than one element in *domain(M)* with each element in *range(M)*. An *onto* mapping *M* associates elements of A with all elements of B. A *function f* is then defined as a *m-to-1* and *onto* mapping. If *f* is a function, then the range of *f* is also called the co-domain of *f*, written *co-domain(f)*.

Restriction A *restriction* of a function $f:A \rightarrow B$, denoted as $\sigma_{cond}(f)$, is a function such that

$$\sigma_{cond}(f) = \{(x,y) \dagger \forall x \in A, \exists y \in B: y=f(x) \text{ and } cond(x,y) = true\}$$

where *cond* is any first-order predicate.

5. Mathematical Symbols

And (conjunction)	∧
Belongs to	∈
Composition	o
For All	∀
Iff	if and only if (⇔)
Implies	⇒
Is equivalent to	⇔
Maps into	→
Or (disjunction)	∨
Selection	σ
Set	{}
Such that	: , †
There exists	∃

6. References

[Amodeus 95] Salber, D., Coutaz, J., Nigay, L., Faconti, G., Paterno, F., Duke, D., & Harrison, M., "The System Modelling Glossary", Amodeus project document SM/WP26, 1995. http://www.mrc-cbu.cam.ac.uk/amodeus

[Arch 92] Arch, "A Metamodel for the Runtime Architecture of An Interactive System", The UIMS Developers Workshop, SIGCHI Bulletin, 24(1), ACM, 1992.

[Bouillon 02a] Bouillon, L., Vanderdonckt, J., & Souchon, N., "Recovering Alternatives Presentation Models of a Web Page with VAQUITA", Chapter 27, Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002), Kluwer Academics Pub., Dordrecht, 2002, pp. 311-322.

http://www.isys.ucl.ac.be/bchi/publications/2002/Bouillon-CADUI2002.pdf

[Bouillon 02b] Bouillon, L., Vanderdonckt, J., & Eisenstein, J., "Model-Based Approaches to Reengineering Web Pages", in Proceedings of 1st International Workshop on Task Models and Diagrams for user interface design TAMODIA'2002 (Bucharest, 18-19 July 2002), Academy of Economic Studies of Bucharest, INFOREC Printing House, Bucharest, 2002, pp. 86-95.

[Bouillon 02c] Bouillon, L. & Vanderdonckt, J., "Retargeting Web Pages to other Computing Platforms with Vaquita", Proceedings of the IEEE 9th Working Conference on Reverse Engineering WCRE'2002 (Richmond, 28 October-1 November 2002), IEEE Computer Society Press, Los Alamitos, 2002, to appear. http://www.isys.ucl.ac.be/bchi/publications/2002/Bouillon-WCRE2002.pdf

[Calvary 01] Calvary, G., Coutaz, J., & Thevenin, D., "A Unifying Reference Framework for the Development of Plastic User Interfaces", in Proceedings of IFIP WG2.7 (13.2) Working Conference EHCI'2001 (Toronto, May 2001), M. Reed Little & L. Nigay (Eds.), Springer Verlag Publ., LNCS 2254, pp.173-192.

[Calvary 02] Calvary,G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M., Vanderdonckt, J., "Plasticity of User Interfaces: A Revised Reference Framework", in Proceedings of 1st International Workshop on Task Models and Diagrams for user interface design TAMODIA'2002 (Bucharest, 18-19 July 2002), Academy of Economic Studies of Bucharest, INFOREC Printing House, Bucharest, 2002, pp. 127-134.

[Csaba 97] Csaba, L., "Experience with User Interface Reengineering – Transferring DOS panels to Windows", in Proceedings of 1st Euromicro Working Conference on Software Maintenance and Reengineering CSMR'97 (Berlin, 17-19 March 1997), IEEE Computer Society Press, Los Alamitos, 1997.

[D1.1 02] Calvary, G., Coutaz, J., Thevenin, D., Bouillon, L., Florins, M., Limbourg, Q., Souchon, N., Vanderdonckt, J., Marucci, L., Paternò, F., &

Santoro, C., "The CAMELEON Reference Framework", Deliverable D1.1, CAMELEON Project, 2002.

[Gram 96] Gram, C. & Cockton, G. (Eds.), "Design Principles for Interactive Software", IFIP WG 2.7 (13.4), Chapman & Hall Publ., London, 1996.

[IEEE Terminology] IEEE Terminology in Reverse Engineering, IEEE. http://www.cc.gatech.edu/reverse/bibliography/terminology.html

[ISO 91] ISO9241-11 Ergonomic requirement for office works with VDT's – guidance on usability. Technical report, International Standard Organisation, 1991.

[Nigay 95] Nigay, L. & Coutaz, J., "A Generic Platform for Addressing the Multimodal Challenge", in Proceedings of ACM Conf. on Human Aspects in Computing Systems CHI'95 (Denver, May 1995), ACM Press, New York, pp. 98-105.

[Paternò 99] Paternò, F., "Model-based Design and Evaluation of Interactive Applications", Springer Verlag, Berlin, 1999.

[Paternò 02] Paternò, F. & Santoro C., "One Model, Many Interfaces", Chapter 13, Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002), Kluwer Academics Pub., Dordrecht, 2002, pp. 143-154.

[Puerta 99] Puerta, A.R. & Eisenstein, J., "Towards a General Computational Framework for Model-Based Interface Development Systems", in Proc. of ACM Conf. on Int. User Interfaces IUI'99 (Los Angeles, January 1999), ACM Press, New York, 1999, pp. 171-178. http://www.arpuerta.com/pubs/iui99.htm

[Stroulia 00] Stroulia, E., Thomson, J., & Situ, Q., "Constructing XML-speaking wrappers for WEB Applications: Towards an Interoperating WEB", in Proceedings of the IEEE 7th Working Conference on Reverse Engineering WCRE'2000 (Brisbane, 23-25 November 2000), IEEE Computer Society, Los Alamitos, 2000.

[STSC] http://www.stsc.hill.af.mil/reng/defin.asp

[Thevenin 01] Thevenin, D., "Adaptation en Interaction Homme-Machine: le cas de la Plasticité", Ph.D. thesis, Grenoble, France, 2001. http://iihm.imag.fr/publs/2001

[Vanderdonckt 93] Vanderdonckt, J. & Bodart, F., Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection, in Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel & T. White (Eds.), ACM Press, New York, 1993, pp. 424-429. http://www.acm.org/pubs/articles/proceedings/chi/169059/p424-vanderdonckt/p424-vanderdonckt.pdf

[Vanderdonckt 99] Vanderdonckt, J., & Berquin, P., "Towards a Very Large Model-based Approach for User Interface Development", in Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99 (Edimburg, 5-6 September 1999), N.W. Paton & T. Griffiths (éds.), IEEE Computer Society Press, Los Alamitos, 1999, pp. 76-85.

http://www.isys.ucl.ac.be/bchi/publications/1999/Vanderdonckt-UIDIS99b.pdf

[Vanderdonckt 01] Vanderdonckt, J., Bouillon, L., & Souchon, N., "Flexible Reverse Engineering of Web Pages with VAQUITA", in Proceedings of IEEE 8th Working Conference on Reverse Engineering WCRE'2001 (Stuttgart, 2-5 October 2001), IEEE Computer Society Press, Los Alamitos, 2001, pp. 241-248.

http://www.isys.ucl.ac.be/bchi/publications/2001/Vanderdonckt-WCRE2001.pdf

[Want 01] Want, R. & Shilit, B., "Expanding the Horizon of Location-Aware Computing", IEEE Computer, Vol. 34, No. 8, August 2001, pp. 31-34.