



Laboratory for Usage-centered Software Engineering

Technical Paper

DRAFT - REVISION 2.0

Activity Modeling: Toward a Pragmatic Integration of Activity Theory with Usage-Centered Design

Larry L. Constantine, IDSA

Chief Scientist, Constantine & Lockwood, Ltd.

Director, Laboratory for Usage-centered Software Engineering

Abstract. Activity modeling is a systematic approach to organizing and representing the contextual aspects of tool use that is both well-grounded in an accepted theoretical framework and embedded within a proven design method. Activity theory provides the vocabulary and conceptual framework for understanding the human use of tools and other artifacts. Usage-centered design provides the methodological scaffolding for applying activity theory in practice. In this Technical Paper, activity theory and usage-centered design are outlined and the connections between the two are highlighted. Simple extensions to the models of usage-centered design are introduced that together succinctly model the salient and most essential features of the activities within which tool use is embedded. Although not intended as a tutorial, examples of Activity Maps, Activity Profiles, and Participation Maps are provided.

Introduction

Activity theory is a way of describing and characterizing the structure of human activity of all kinds. First introduced by Russian psychologists Rubinshtein, Leontiev, and Vigotsky in the early part of the last century, activity theory has more recently gained increasing attention among interaction designers and others in the human-computer interaction and usability communities (see, for example, Gay and Hembrooke, 2004). Interest was given a significant boost when Donald Norman suggested activity-theory and activity-centered design as antidotes to some of the putative ills of “human-centered design” (Norman, 2005). Norman, who has been credited with coining the phrase “user-centered design,” suggested that too much attention focused on human users may be harmful, that to design better tools designers need to focus

DRAFT - PLEASE DO NOT COPY WITHOUT PERMISSION - DRAFT

This work was undertaken in part under the Activity Modeling Initiative of the Laboratory for Usage-centered Software Engineering, Department of Mathematics and Engineering, University of Madeira, Funchal, Portugal. The author wishes to thank the many people who provided feedback on earlier drafts, but particularly Donald Norman for his suggestions and encouragement and my colleagues Nuno Nunes, Pedro Campos, Leonel Nóbrega, and Eduardo Fermé for their generous support and feedback.

© 2006, L. L. Constantine

not so much on users as on the activities in which users are engaged and the tasks they seek to perform within those activities.

Although many researchers and practitioners claim to have used or been influenced by activity theory in their work (see, for example, Nardi, 1999), it is often difficult to trace precisely where or how the results have actually been shaped by activity theory. In many cases, even detailed case studies report results that seem only distantly related, if at all, to the use of activity theory.

Contributing to the lack of precise and traceable impact is that activity theory, despite its name, is not truly a formal and proper theory. Better characterized as a conceptual framework, activity theory comprises a collection of concepts and categories for communicating about activity coupled to diverse assertions—posited but largely untested—about the nature of human activity. Rich in vocabulary but somewhat lacking in rigor, it is perhaps better described as a philosophy of analysis and design, a philosophy that emphasizes understanding the larger context of activities within which designed artifacts are and will be used.

For designers, the great potential of activity theory is to provide an organized and consistent way to investigate, describe, and understand the larger context of activity within which use of a software tool or other artifact is embedded. For this potential to be fully realized, however, the somewhat vague formulations and expressions of activity theory need to be made more precise and accessible.

Some attempts have been made to systematize and operationalize activity theory for purposes of informing the design process. For example, the Activity Checklist originally developed by Kaptalinin, Nardi, and Macaulay (1999) has recently been transformed into a somewhat more precise form as an Activity Interview (Duignan, Noble, and Biddle, 2006). Nevertheless, even its most ardent proponents acknowledge the problems of putting activity theory into practice: “These general principles help orient thought and research, but they are somewhat abstract when it comes to the actual business of working on a design or performing an evaluation” (Kaptalinin, Nardi, and Macaulay, 1999).

The purpose of this paper is to introduce activity modeling, a systematic approach to representing activities that is intended to make it easier for practicing designers to capture essential insight and understanding about the context of activity and to reflect this understanding in their designs. Specifically, activity modeling provides a link between activity theory and essential use cases (Constantine, 1995), a widely used task modeling technique and one of the core models of usage-centered design (Constantine and Lockwood, 1999). Usage-centered design itself has been viewed as providing already established and effective methods for putting activity-centered design into practice and for overcoming some of the stated shortcomings of human-centered design (Norman, 2006).

The development of activity modeling was spurred by recognized limitations in both activity theory and usage-centered design. Activity theory has generated little in the way of systematic modeling techniques or straightforward methods connecting activity to interaction design. Usage-centered design has, for its part, lacked clear, concise constructs for representing the contextual or collective aspects of work. In undertaking to overcome these weaknesses, the aim has been to create an easily grasped modeling language anchored in a consistent, coherent vocabulary of well-defined concepts that link task modeling based on essential use cases to the established conceptual foundation of activity theory.

Activity modeling is intended as a tool to capture and succinctly represent the salient information regarding activities that is most relevant to interaction design. The goal is first and foremost a practical design tool to serve practicing designers, rather than a comprehensive framework for research or academic analysis. As such, the focus is on pragmatics over rigor, on systematic rather than completely formal techniques.

Before elaborating the technique of activity modeling, it is appropriate to briefly review activity theory from a design perspective and to provide an overview of usage-centered design.

Activity Theory

Activity theory in its most elaborate and fully articulated forms can be rather daunting, but the basic tenets are fairly straightforward. The essentials of activity theory can be summarized by a couple of simple diagrams supported by brief explanations.

In a formulation that has been widely replicated, the structure of human activity is represented schematically as in the diagram of Figure 1. The original perspective, represented by the upper triangle in the diagram, was that human activity is performed by agents (subject) motivated toward solution of a problem or by a purpose (object or motive) mediated by tools (artifacts) within a transformational process yielding a result (outcome). Engeström (1999) elaborated this perspective by adding the elements in the bottom half of the diagram, implying that all activity takes place in a social context (community) with differentiated responsibilities (roles or division of labor) constrained by socio-cultural and procedural factors (rules).

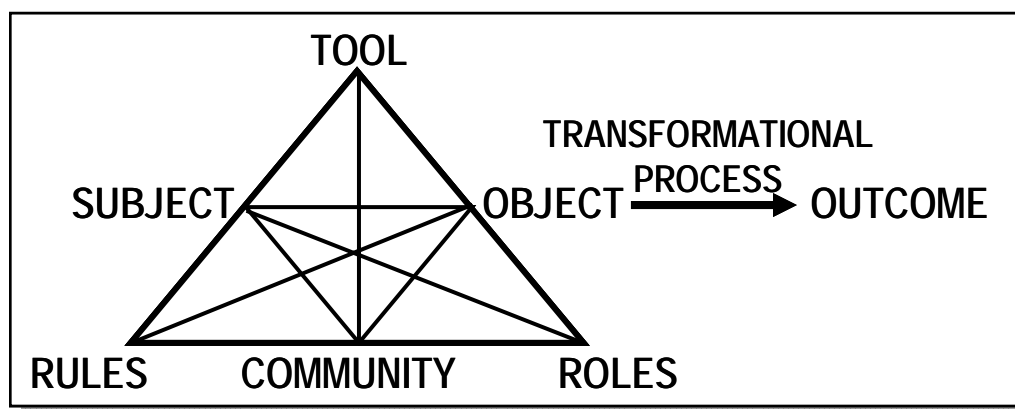


Figure 1 - The structure of human activity (adapted from Engeström, 1999)

Activity theory further characterizes human activity as hierarchical. As suggested by Figure 2, activity can be understood at three levels of analysis: activity, action, and operation. Activity consists of collections of actions directed toward goals that contribute to or are related to the purpose of the activity. Actions in turn comprise operations, conscious or non-conscious, adapted to emerging conditions in service of the goals of the actions.

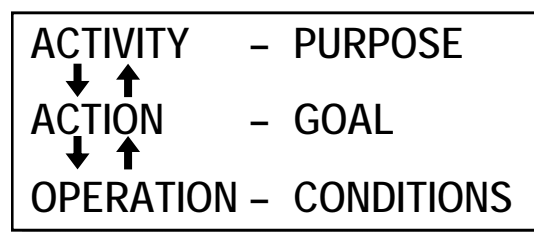


Figure 2 - Hierarchical nature of activity (adapted from Engeström, 1999).

Activity theory sees all human activity as mediated by tools and, whether significantly or fortuitously, places tools at the very apex of the structure of activity. This is precisely the perspective of usage-centered design (Constantine and Lockwood, 1999). For designed artifacts to be most effective as tools, they must be suited to the operational context in which they are actually used and deployed. Most importantly, this requires that the design fit with the purpose(s) of the activities within which the use by subjects takes place. At a more detailed

level, it also requires that designed artifacts effectively support the combined actions by which these purposes are advanced. Thirdly, a well designed artifact takes into account the community of participants, their roles, and the rules regulating their activity. These are arguably among the most important aspects of the immediate context of activity.

This brief synopsis does not, of course, do justice to the richness of activity theory. Elucidating the conflicts among activities and among competing goals are considered to be essential to understanding how activity takes place and evolves. Activity theory also posits that oft repeated actions gradually become operationalized to become automatic or partially automatic operations. For design purposes, it is useful to highlight some additional aspects of activities that, while not being obvious from the more compact formulations of activity theory, can have strong design implications. In particular, activities take place in and over time, within a particular physical and social setting, and are performed in characteristic manners, styles, or patterns. These are useful considerations for the designer to take into account in designing tools to support activity.

Usage-Centered Design

Because usage-centered design is probably more widely known among software engineers than in the HCI community (where it is sometimes confused with *user-centered design*), a brief review is appropriate. More detailed descriptions can be found elsewhere (Constantine and Lockwood, 1999; 2002)

Usage-centered design is a model-driven process for user interface and interaction design that takes its name from its primary focus on use or usage rather than on users per se. It is a systematic process for deriving a user interface design from a series of interrelated abstract models representing user roles, tasks, and interface contents. In this process, the content and organization of a user interface is derived more or less directly from a fine-grained task model, which in turn is grounded in a well-defined model of the relationships between users and the system being designed. What distinguishes usage-centered design from most mainstream user-centered approaches is a matter of emphasis and focus, but collectively these differences in degree can add up to substantial and significant differences in practice and in results (Constantine, 2004).

While it is certainly common for designers and design processes to compile information about users, tasks, and the content of user interfaces, usage-centered design is distinguished by the high level of abstraction of its models and the straightforward way in which these are interconnected. The most popular techniques for compiling and conveying information about users, for instance, are, in contrast, concrete and realistic rather than abstract; personas (Cooper and Reimann, 2003) and user profiles (Hackos and Redish, 1998) are probably the best known examples. In the case of personas, the pursuit of realism even includes construction of a hypothetical personal history, background, personality, and frequently even augmentation with photographs (Pruitt and Aldin, 2006). By contrast, *usage-centered design* carries salient information about users in the highly condensed form of user roles representing abstract relationships between users and the system being designed (Constantine, 2006).

Similarly, in *user-centered design*, tasks are often modeled using scenarios (Carroll, 1995), most commonly expressed in the form of plausible story narratives. In contrast, *usage-centered design* models user tasks as use cases (Cockburn, 2001; Jacobson et al., 1992), a construct originating in software engineering. A special form of use case, the so-called task case or essential use case (Constantine, 1994, 1995) was invented to serve the needs of user interface and interaction design by distilling interaction to its simplest, abstract essence. Task cases became the core of the model-driven process that is now known as usage-centered design.

Most designers rely heavily on paper prototypes, mockups, or other more or less realistic sketches or drawings of actual user interfaces (Snyder, 2003) to express and develop user interface designs. Usage-centered design relies on abstract prototypes to model the

organization and functional content of user interfaces without regard to details of appearance or behavior (Constantine, 1998; 2003).

The use of abstraction has many advantages for designers. It makes it easier to defer decisions and avoid premature preoccupation with details. It helps focus the designer's attention on essentials, promoting resolution of large-scale or architectural issues before becoming immersed in low-level details. Experience has also shown that such abstract models encourage inventive, creative thinking about solutions to design problems. Most crucially, highly abstract models allow complex problems to be described more succinctly without sacrificing critical information.

Because of their connection with activity theory, the core models of user roles and task cases will be described in somewhat greater detail.

User Roles

Users who interact with a system are referred to as actors, a term borrowed from software engineering. Actors play roles. A user role is an abstraction representing a relationship between users and a system. In its simplest form, a role can be described by the context in which it is performed, the characteristic manner in which it is performed, and by the evident design criteria for effective support of performance of the role (Constantine, 2006). An example is shown below for a user working in the ticketing window reserved for "today's performances only" at a performing arts center.

R01 - Current-Sales-and-Ticketing Role

context (of role performance): isolated in booth, likely facing queue of customers; final step in performance promotion and sales

characteristics (of role performance): relatively simple task performed repeatedly with some training, prior experience likely; performed under some time pressure, which increases as show time approaches

criteria (for support of role performance): simplified, highly efficient interaction; foolproof means of identifying customer, guarantee that all the right tickets are dispensed and received by the customer

A typical application will involve a number of distinct roles representing the various relationships a user can assume in interaction with the application. Roles can, of course, be shared by any number of different occupants or incumbents, and a given user may occupy different roles at different times, even sometimes shifting rapidly between roles. For example, a visitor to a Web site may begin in the role of Indifferent-Curious-Information-Seeker and switch to the role of Engaged-Potential-Purchaser.

Usage-centered design models roles rather than users for two reasons. First, the characteristics of the role, the relationship to the system, has a more immediate and direct relevance for interaction design than do characteristics of the person playing the role. Second, the relationship to any given system represents a small subset of all possible aspects of the user. Interaction with the system takes place over a channel with relatively limited bandwidth which is restricted to a small subset of user behaviors that take place in specific settings. Because of its narrow focus on the most salient issues, a user role model can thus be substantially more compact than many alternative user models (Constantine, 2006). This simplicity has contributed to the popularity of usage-centered design for the streamlined and accelerated processes known as agile development (Constantine, 2002; Constantine and Lockwood, 2002; Patton, 2002).

Task Cases

A task case represents a single, discrete user intention in interaction with a system that is complete and meaningful to a user in some role. A task case is a specially structured form of a

use case, one that is expressed in so-called essential form (Constantine, 1995; McMenamin and Palmer, 1984), that is, abstract, simplified, and independent of assumptions about technology or implementation. Task cases are written as an abstract dialog representing user intentions and system responsibilities. This form focuses on the essence of a task stripped of assumptions about how it might be performed with or supported by a particular user interface design. For example, a task supporting the Current-Sales-and-Ticketing Role described earlier might be:

T01 - <u>Issuing-Held-Ticket(s) for Performance(s)</u>	
USER INTENTIONS	SYSTEM RESPONSIBILITIES
2. provide customer identification	1. request customer identification
4. confirm by selection	3. provide confirming details
	5. print tickets with in-process and completion notification

Task cases are typically small and focused on a highly specific user goal, yielding a fine-grained model of user activity. A complete task model comprises a collection of such task cases interrelated in a variety of ways. For example, a task case can include or make use of other task cases to form more complex combinations, a task case can be extended by others that alter or interrupt its performance, or a task case can express a specialized variation of another.

More complex scenarios or workflow can be expressed by constructing composite task cases that include (by reference) any number of other task cases in the structured narrative. These so-called workflow task cases can be useful for modeling relatively predictable, orderly tasks that are well-understood in detail but are not well suited to expressing combinations of individual tasks that may be performed in many different, largely unpredictable ways.

Toward Integration

Activity theory and usage-centered design are clearly connected at more than one point. The participation of actors in activities and the hierarchical nature of performance of activities represent the most important points of intersection. Activity theory provides a coherent way of understanding and modeling actors as tool users engaged with other participants and artifacts.

Perhaps most importantly, activities constitute an elegant scheme for aggregation of task cases into larger, loosely structured collections related by common purposes rather than by explicit or detailed interconnections. This approach to aggregation of task cases is particularly appealing because it is not an ad hoc modeling mechanism introduced to solve a problem in task modeling but comes already embedded in a larger body of knowledge and insight about human action.

The integration of activity theory with usage-centered design is less a matter of adding new concepts to the method as it is providing a stronger and more coherent organization to existing ones. The foundations of usage-centered design rest firmly on the bedrock of activity and action, but environment and context became the backdrop for a focus on tasks as central to a model-driven design process. As originally conceived, operational context was represented by an operational model, a loosely bound collection of so-called operational profiles: the incumbent profile, proficiency profile, interaction profile, information profile, environment profile, and others (Constantine and Lockwood, 1999). Over time, some of this information migrated into the user role model, but otherwise it remained largely unconnected with the other models of usage-centered design. Activity theory offers a single, simple framework capable of tying all this information together coherently.

Vocabulary

Both activity theory and usage-centered design have established vocabularies of specialized terminology. Unfortunately, the terminology is incompatible and can be confusing at points. For example, activity theory typically uses the term object, rather than objective, to refer to the motive for or purpose of an activity. Although semantically correct, this less common usage conflicts with the software engineering lexicon, in which object is a software component defined by attributes and operations.

In building a bridge between activity theory and usage-centered design every attempt has been made to keep original vocabulary intact as much as possible while avoiding confusion and conflict. One way to smooth the linguistic integration is by allowing alternative terms for the same concept used from different perspectives. Thus the three levels of analysis can still be referred to in the traditional way as activity, action, and operation when speaking generically, but when focusing specifically on interactive activity, that is, activity in interaction with a system being designed or analyzed, the three levels are referred to as activity, *task* and operation to maintain consistency with usage-centered design.









For design purposes it is also important to distinguish participants who actually interact directly with the user interface from those who are not engaged with the system or whose interaction is indirect, that is, mediated by other participants who do have direct contact with the user interface. Participants who are direct users are called actors, the well established term in both usage-centered design and software engineering, while other participants are referred to as players. Clearly, whether a participant (subject) is an actor (user participant) or a player (other participant) can depend on the choice of the system of reference (the system to be designed or analyzed), the defined boundary of that system, and the activity frame of reference. For example, a customer on the telephone is an actor (direct user) with respect to the voice menu for initial contact but a player (indirect, mediated user) when speaking with a sales agent who has direct access to an ordering system. Players can, of course, be actors with respect to other systems.

Notation

The purpose of a notation is to enable the construction of compact representations that serve as repositories of insight and understanding and that facilitate rapid comprehension, clear and efficient communication, and detailed analysis. The notation introduced here for activity modeling is an extension of the notation long used in usage-centered design, which, in turn, is related to the Unified Modeling Language (UML) widely used in software engineering (Fowler and Scott, 1997). The objective is a simple notation that expresses clear distinctions where needed with minimal additions. The notation for activity modeling summarized in Table 1 adds to the established notation already used in usage-centered design four new symbols for activities, actions, artifacts, and non-actor participants.

It is important to keep in mind that these models are being introduced to maximize utility and efficiency in representing activity context for interaction design purposes rather than for software engineering. The notation has not, therefore, been forced to fit, however awkwardly, within the constraints of UML at the expense of facile expression. An artifact, for example, could be argued to be an object in the object-oriented software engineering sense and, therefore, representable by the extant UML symbol for an object. However, an artifact is not a software object but an actual real-world physical entity of interest not for its modeling in software but for its part in some human activity. Making all objects, whether modeled in software or existing in the real world, look alike does not serve the purpose of facile expression and easy comprehension.

Table 1 - Extended usage-centered design notation for activity modeling

Symbol	Name	Description
	actor, user actor	activity participant interacting with the system of reference
	role, user role	relationship between an actor and the system of reference
	system actor	non-human system (software or hardware) interacting with the system of reference
	player*	activity participant not interacting with the system of reference (but often an actor with other systems)
	artifact, tool*	any artifact employed within an activity
	activity*	collection of actions or tasks undertaken for some purpose
	task, task case	action by an actor in interaction with the system of reference for some goal within an activity
	action*	action by a player for some goal within an activity

* new notation introduced for activity modeling

Activity Modeling

In as much as usage-centered design is an already proven design method that has been widely and successfully practiced (see, for example, Constantine and Lockwood, 2002; Strope, 2003; Windl, 2002) for more than a decade, the objective in introducing systematic activity modeling is refinement rather than wholesale replacement. In particular, every effort has been made to add value without sacrificing the economy of expression of the established usage-centered models.

To incorporate systematic activity modeling into usage-centered design the following additions and alterations have been made:

- An Activity Model defines and describes activities and their interrelationships.
- The Role Profile that describes user roles is modified to connect roles explicitly to the activities within which the roles are embedded.
- The Task Model is elaborated to incorporate actions in relation to other participants and artifacts and to connect task cases explicitly to activities.

The activity model itself includes three parts: an Activity Map that identifies relevant activities and their interrelationships (including, optionally, the aggregation of task cases into activities), a collection of Activity Profiles describing the salient aspects of the relevant activities, and a Participation Map showing the involvement of actors with the system, with other artifacts, and with other participants.

Activity Map

An Activity Map represents activities relevant to the design problem and the interrelationships among them. The most relevant are, of course those activities that include interaction with the system of reference, which are referred to as proximate activities. Proximate activities define the immediate context of use, how individual tasks are combined into larger, more complex, interdependent collections. For example, telephone ticket sales is an activity that may involve a complex and changing mix of answering simple questions, helping customers find events of possible interest, responding to requests for specific tickets, taking credit card information, and the like.

Activities can contain or include other activities. Thus, for example, the larger activity of telephone sales might be clarified by breaking it down into two more focused activities: ticket selling and inquiry handling.

Even activities that do not involve interaction with the system of reference may in some cases impact a design and be relevant for defining and understanding the context of use. If, for example, actors are involved in activities with other participants that compete for their time and attention, this has implications for presentation and interaction design. In such cases, the ability to suspend or interrupt interaction at any arbitrary point might be required, and presentation design may need to make it easy for actors to recognize where they are and where they left off in a process.

Activities that are connected in time can be related in a number of different ways. They may be either independent or coordinated in some way. They may be concurrent or consecutive. If concurrent, they may be coordinated (or synchronized), independent, or interleaved, that is alternated. (This last relationship is particularly common because a single actor can seldom engage in more than one activity at a time, which results in otherwise independent activities becoming coordinated in time.) If consecutive, activities may or may not overlap in time. An activity can compete with another activity because it shares common participants or resources. More indirectly, an activity can be affected by an adjacent activity with which it has no relationship other than both activities take place within the same setting.

The most commonly meaningful relationships among activities in an Activity Map are listed in Table 2. This is not intended as a complete listing but only to offer examples of the kinds of relationships that have proved useful for modeling real-world problems.

In considering whether to include an activity in an activity model or to define a relationship between activities, the sole issue is relevance: Does it make a demonstrable difference or have an arguable impact on the design? The objective is not to be exhaustive or all encompassing but to model what matters. In general, activities can be rank ordered on the relevance for interaction design. From most relevant to least, these are:

1. proximate activities (the immediate activity context within which use occurs)
2. competing activities also involving the same actor(s)
3. competing activities involving shared resources in common with proximate activities
4. adjacent activities in the same setting but otherwise unrelated to proximate activities

Activities are represented in an Activity Map by the block shape shown in Table 1. A line or arrow connecting one activity to another represents a relationship. Relationships can be labeled with qualifiers to specify the relationship more precisely. An example of an Activity Map using this notation is shown in Figure 3. This map represents some of the core activities for a retail sales context within the appliance department of a large retailer. As seen here, braces (curly brackets) can be freely used to visually simplify representation of relationships.

Table 2 - Relationships between activities.

Relationship Qualifier	Explanation*
contains	[includes] activity is composed of other sub-activities
coordinated	[synchronized] activities are coordinated/synchronized by some means
concurrent	activities occur over common time span, not further qualified
synchronized	[coordinated]
unsynchronized	independent, concurrent but not coordinated
interleaved	alternating
consecutive	sequential activities, not further qualified
precedes	strictly sequential
overlaps	activity finishes after another starts
competing	activities conflict or interfere, not further qualified
common participants	participant(s) (optionally identified) overlap
shared artifacts	some resources (optionally identified) are shared
adjacent	activities occur within same setting (place and time)

* Alternate terms are shown in brackets.

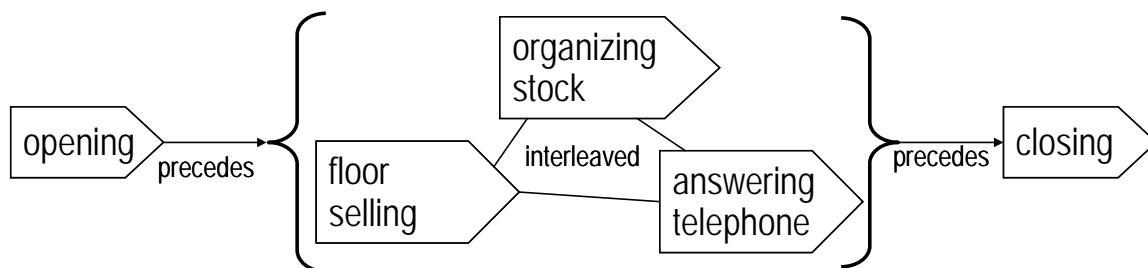


Figure 3 - Example of an Activity Map in a retail sales context

In addition to the qualifiers listed in Table 2, other terms or descriptions appropriate to the context can be used as needed. For example, the flow of information, participants, or artifacts between activities can be modeled where known and where relevant using small arrows running alongside the line or arrow for the relationship. An Activity Map can also be expanded into a combined Activity-Task Map that includes the aggregation of task cases into activities. An example of this can be found in Figure 6, to be discussed later.

Activity Profiles

For purposes of informing interaction design, activities are described by Activity Profiles. An Activity Profile is not intended to capture in fullest detail everything known or knowable about

an activity but rather to organize in compact form the salient aspects of an activity that are most likely to be relevant to shaping the user interface design. This condensed formulation is organized under four easily remembered headings: Purpose, Place and Time, Participation, and Performance. A fifth heading, Product, is not, strictly speaking, part of the activity description but serves as a holding place for any evident design implications that follow from the understanding of the activity. An Activity Profile can, of course, be augmented by additional narrative description or supporting documentation or models.

Purpose refers to the motives or objectives for the activity, what it is all about. An activity can have more than one purpose and the purpose may in some cases be different for different participants. The purpose of a soccer game may be different for players on the field and for fans in the stand, for example. Purpose may also differ depending on whether an internal or external view is taken.

Place and Time (the setting) refers to where, when, and under what conditions the activity takes place, which can include both the physical and the social setting of the activity as well as the duration, schedule, frequency or other temporal aspects of the performance. The setting within which activity takes place is not always known or easily described. On-line shopping, for example, might take place within an office cubicle, in the living room of a home, or in a wi-fi equipped coffee shop. Such variations in the setting can be included if they are of significance for design and can be described at an appropriate level.

Participation refers to those engaged in the activity and the artifacts with which they are involved. Participants include actors engaged with the system of reference along with the roles they play as well as other players not engaged with the system. Artifacts include the physical and conceptual tools employed in the activity, including information sources, references, and other resources. Relationships and interactions among participants and between participants and artifacts as well as the division of responsibilities among participants also fall under this rubric. Participation can also be defined by reference to a Participation Map (see below).

Performance refers to the characteristic manner or style in which the activity is performed including how it might be coordinated with or otherwise related to other identified activities. Included under this heading are the rules, formal or informal, that shape and govern the performance of the activity. Relationships with other activities can also be defined by reference to an Activity Map.

Participation Map

One of the most straightforward ways to model an activity is through a simple map of the “playing field,” a representation of the participants and their relationships with each other and with the various artifacts involved in the activity. Such a Participation Map, as it is called, gives a quick overview of the context within which system use takes place. For design purposes, it is useful to distinguish participants who are engaged in interaction with the system of reference, the tool being designed, from other participants. Carrying over the already established terminology, the former are referred to as user actors, or more simply, actors. Other systems that interact with the system of reference are called system actors. The term player is introduced to refer to all other participants.

The Participation Map takes the form of a simple diagram, such as the one shown in Figure 4. The notation employs simple but distinctive iconic representations to distinguish actors, roles, players, system actors, and artifacts. Interconnecting lines identify the interrelationships and can be decorated to represent the flow of information or material.

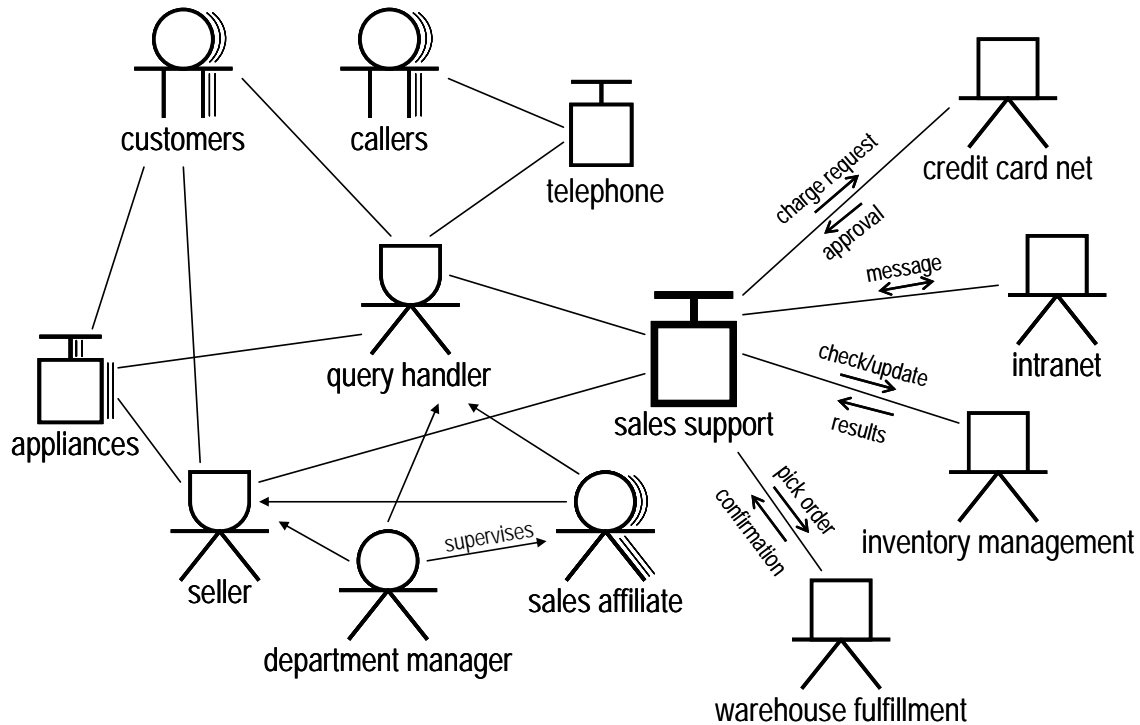


Figure 4 - Example of Participation Map for retail selling situation

The Participation Map would typically include only participants and artifacts having a salient connection with actors (or roles) within proximate activities, either directly or as mediated by artifacts. Where the information is relevant, the Participation Map can be supplemented by more detailed descriptions of artifacts or of participants.

The form of Participation Map shown in Figure 4, referred to as a system-centered participation map, is the most typical as it compactly represents the context of use for the system of reference. It is possible to add activities directly to such a diagram, but if there are multiple activities the resulting visual clutter can hinder interpretation. Which form of Participation Map is used depends upon the application and what information and perspective are deemed most useful.

System Actors and Artifacts (tools, resources) are, of course, closely related and can be easily confused. System Actors, like User Actors, interact with the system of reference but unlike Artifacts, they are not directly used by User Actors. In Figure 4, for example, the credit card network interacts with the sales support system but, unlike the telephone, is not itself a tool used directly by any of the actors or players. In some cases, a system might be modeled either way, depending on the purpose and the context of the model.

Activity-Based User Roles

From an activity theory perspective, user roles are played by Actors within activities. User roles are connected to activity theory by modifying the Role Profile to include information about activities. The content of this revised Role Profile is organized under three headings: Activity, Background, and Characteristics.

Activity refers, of course, to the activity within which the role is played. If the activity is defined elsewhere by an Activity Profile, then it can be referred to by name. Otherwise it is briefly described in terms of purpose, place (physical and social context) and time, and participation, including salient artifacts.

Background refers to the background characteristics of the performers of the role in terms of experience, training, education, system knowledge and domain knowledge, distribution of performance skills, and orientation or attitudes of performers

Characteristics refers to performance characteristics, such as frequency, regularity, intensity, complexity, and predictability of performance. In some cases this may overlap with or repeat aspects of the Activity Profile, particularly if there is only one Role for a single Actor in the activity.

A fourth rubric, *Design*, serves as a holding place for evident design implications for effective support of the role.

An activity-centered Role Profile for the Current-Sales-and-Ticketing Role resembles the description given earlier but places greater emphasis on the larger context within which the role is played:

R01 - Current-Sales-and-Ticketing Role

Activity (in which role is performed): selling and delivering tickets for today's events; part of performing arts promotion, sales, and presentation (purpose: sell and deliver tickets efficiently and accurately; place: isolated in booth outside main entrance, likely facing queue of customers; time: before show time, final step in activity; participants: seller, customer, queued customers, milling crowd, supervisor; artifacts: ticketing system, tickets in ticket printer, phone to supervisor)

Background (of role performers): some training and experience expected, may or may not have domain (performing arts) knowledge

Characteristics (of role performance): relatively simple task performed repeatedly under some pressure, increasing as show time approaches; governed by business rules for normal sales and exception handling as well as informal rules of courtesy and customer focus

Design (implications for support of role performance): needs simplified, highly efficient interaction, foolproof means of identifying customer, guarantee that all the right tickets dispensed and received

It is worth emphasizing again that the goal in activity modeling is efficient expression, as represented in this Role Profile, not comprehensive description.

Activity-Based Task Modeling

Task cases (essential use cases) as employed in usage-centered design represent the second level of the activity hierarchy. A task model based on task cases provides a fine-grained view of user intentions and interactions within an activity. The task model is extended to integrate with the activity model in two ways: by connecting tasks to the activities within which they are embedded and by elaborating the task model to incorporate non-interactive actions. In this context, actions refer to goal-directed interactions among actors or players and between them and artifacts other than the system of reference. Actions are represented by a distinct symbol (the barred ellipse seen in Table 1), a variation of the symbol already generally used to represent task cases.

The Task Map, a model used to represent the interrelationships among task cases in usage-centered design, can be extended to incorporate activities and actions. For complex problems, however, a single diagram combining activities, tasks, and actions along with lines representing all their relationships can become too complicated visually. The combined model can be simplified by omitting the relationships among tasks and actions to focus on the aggregation into activities, which in many cases is the primary interest. An example of such a combined Activity-Task Map is shown in Figure 5. The aggregation of tasks and actions into activities can also be easily expressed in matrix form, with activities as columns and actions and tasks as

rows. The choice of representation depends on the complexity of the problem and the goals of modeling.



Figure 5 - Example of partial Activity-Task Map for retail selling.

The narrative body that defines a task case in detail can also, in turn, be extended to include, as appropriate, references to external actions involving other players. For example, in the ticket selling application, the interaction with the customer can be incorporated into the narrative as shown in the task case below. An external action is indicated by bracketing with vertical bars (as in the barred ellipse symbol), as for operations 2, 5, and 8 in the example below. Named actions, like tasks, are identified with underlined names. The former can be defined further with their own narratives or other description where warranted by the complexity and relevance of the action. Whether such attention to detail is worthwhile for a particular problem depends largely on whether there are potential interface design implications.

T01 - <u>Issuing-Held-Ticket(s) for Performance(s)</u>	
USER INTENTIONS	SYSTEM RESPONSIBILITIES
2. <u>getting identification from customer</u>	1. request customer identification
3. provide customer identification	4. provide confirming details
5. <u>confirming details with customer</u>	7. print tickets with in-process and completion notification
6. confirm by selection	10. note as delivered
8. give tickets to customer	
9. confirm delivery	

Design Implications

The manifest design implications regarding activities and the roles played within them are captured and carried as part of the Activity Profiles and Role Profiles. Such implications, which are almost invariably idiosyncratic to the particular problem, can be expanded or altered whenever the need arises. In many cases, design implications will be clear when profiles are first constructed, but in others the significance for design may not become apparent until sometime later in the design and development process.

Some general implications of activity modeling have become clear from use so far. The activities within which task cases are embedded form a powerful and crucial guide to designing a sound architecture for the user interface. All the tasks that constitute a given activity need to be supported through closely connected interaction contexts within the user interface. Tasks that are part of a common activity are likely to be used together or in more or less closely coupled yet often unpredictable ways. If the features that enable the performance of those tasks are widely separated on the user interface, then the system will be both harder to use and more difficult to learn in the first place. A more fully elaborated activity model, in which the finer structure of activities is expressed through a complete Activity Map, will provide more detailed guidance for the interface architecture. The relationships among activities can also be helpful in organizing the user interface. Sequential activities can be supported by discrete facilities of the user interface presented separately to the user, while concurrent activities need integrated facilities.

As noted earlier, not all activities are important to model and factor into design considerations. Proximate activities that include actor interaction with the system of reference are apt to be most relevant, while adjacent activities that merely share the same setting are much less likely to be relevant.

In usage-centered design, the contents of the user interface derive more or less directly from the task model. Clusters of closely related tasks are initially assumed to require support through features in a shared interaction context, or a small set of closely connected ones. When externally directed actions are added into the mix, the derivation can change. The action-task formulation of Issuing Held Tickets for Performances shown above, for example, highlights the close connection between what identification information is requested by the system on the one hand and the exchange between the ticket agent actor and the customer player on the other, which is also true of the presentation of confirming details and the exchange to confirm with the customer. Planning or scripting the external exchanges and organizing the presentation design to correspond can contribute to efficient use and conformance to business rules. For example, business policies may favor using a purchase confirmation number because it uniquely identifies a particular set of tickets for a particular customer even though the customer is more likely to have other forms of identification or confirmation. This suggests a design that leads with a field for confirmation number coupled with a script that asks if the customer knows the confirmation number.

Process Implications

The modest elaboration of usage-centered design to accommodate activity modeling does not radically alter the process. In overview, the process begins with a focus on users and other participants to clarify the immediate activity context. Actors, roles, players (other participants) and artifacts are identified and characterized, along with system actors. The immediate context is modeled by the Participation Map and Role Profiles. On the basis of those models, activities and their constituent actions and tasks are elaborated as an integrated model that includes an Activity-Task Map and Task Cases. The Activity Model guides the development of a Navigation Map to express the overall architecture of the user interface and the derivation of Abstract Prototypes expressing the content and organization of the parts of the user interface. Details of the presentation and interaction design are then determined to complete the design.

In outline, the logic of this process is represented in Figure 6. It conveniently breaks down into two focuses: activity modeling and solution modeling. The former expresses the problem from an activity-centered perspective while the latter expresses the design that follows from expression of the problem. The entire process is model-driven and linked end-to-end by common threads captured in the models. Thus, any given feature of the user interface design can be simply traced to some part of the abstract prototype that derived directly from some element of the task cases that, in turn, support some one or more user roles played by actors participating in modeled activities.

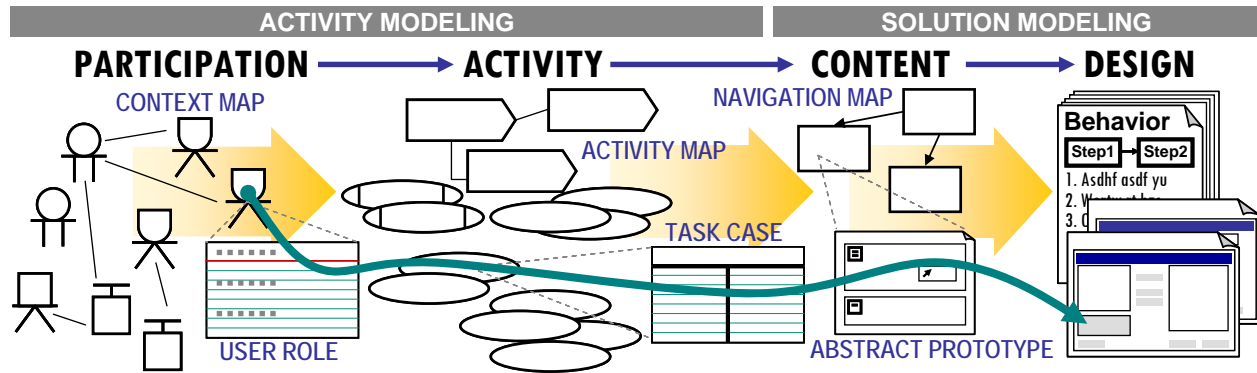


Figure 6 - Logical overview of usage-centered design with activity modeling.

Although the logical connections suggest a straightforward sequential process, in practice the process is iterative and much more nonlinear. In particular, the models under activity modeling, owing to their strong interdependence, are often developed more or less concurrently, with the focus of attention repeatedly shifting among them.

Application

The purpose here is to provide a worked out example of activity modeling for a real-world problem. The problem chosen for illustration is the design of the UInspect system, a software tool to support collaborative usability inspections (Constantine, 2005; Lockwood and Constantine, 2003), particularly the recording of defects during an inspection. A collaborative usability inspection is a structured review organized to identify usability defects in a design, prototype, or working system. Under the leadership of a Lead Reviewer, a User or User Surrogate following an inspection scenario collaborates with other Reviewers to identify usability problems according to a strict protocol. Each of the participants has an assigned role and responsibilities governed by explicit inspection rules. (For more details on collaborative inspections and the rules governing them, see Constantine, 2005; Lockwood and Constantine, 2003.)

Activity Profile: Collaborative Usability Inspection Session

Purpose: to efficiently and effectively locate, describe, and categorize usability defects in a design, prototype, or working system

Place and Time: typically in a conference room or other isolated setting at a scheduled time, lasting 1-3 hours, typically under project schedule pressure; social setting mixes insiders (designers and developers) with outsiders (users or user surrogates, usability experts)

Participation: typically 6-12 people, 1-3 end users or user surrogates, Lead Reviewer, Inspection Recorder, Reviewers (designers or developers), System Driver; optional Continuity Reviewer, Usability Specialist; See references for role responsibilities and rules of conduct (summarized by Lead Reviewer, optionally on poster). Artifacts include: design, prototype, or system being inspected;

inspection scenario; UInspect tool with inspection record; optional projector or large-screen monitor, equipment to run prototype or system being inspected

Performance: intense, sometimes tense interaction, tending to come in bursts; process is carefully orchestrated and divided into distinct phases; location and multi-part descriptions for as many as 100 defects per hour must be recorded accurately and in sufficient detail to support later review and correction; Recorder role particularly pressured; Lead Reviewer important to maintaining focus, preventing unproductive debate and discussion

The system-centered Participation Map in Figure 7 provides an overview of the activity context. The immediate context of the inspection proper is enclosed in the dashed line. This highlights the fact that all players are involved in identifying defects but that all defects flow through the inspection recorder. The unlabeled “couples” shown on some relationships represent the dominant flow of information. In this model, the same actor, the Recorder, is shown as playing three distinct roles: Prep/Configuring, Session Recording, and Record Refining. The first and last of these are associated with distinct activities represented in the Activity-Task Map of Figure 8, which shows the relationships among the activities connected with conducting a collaborative usability inspection, including both preparation and follow-up activities. For simplicity, this combined model shows the aggregation of tasks and actions into activities but not the relationships among actions and interactive tasks.

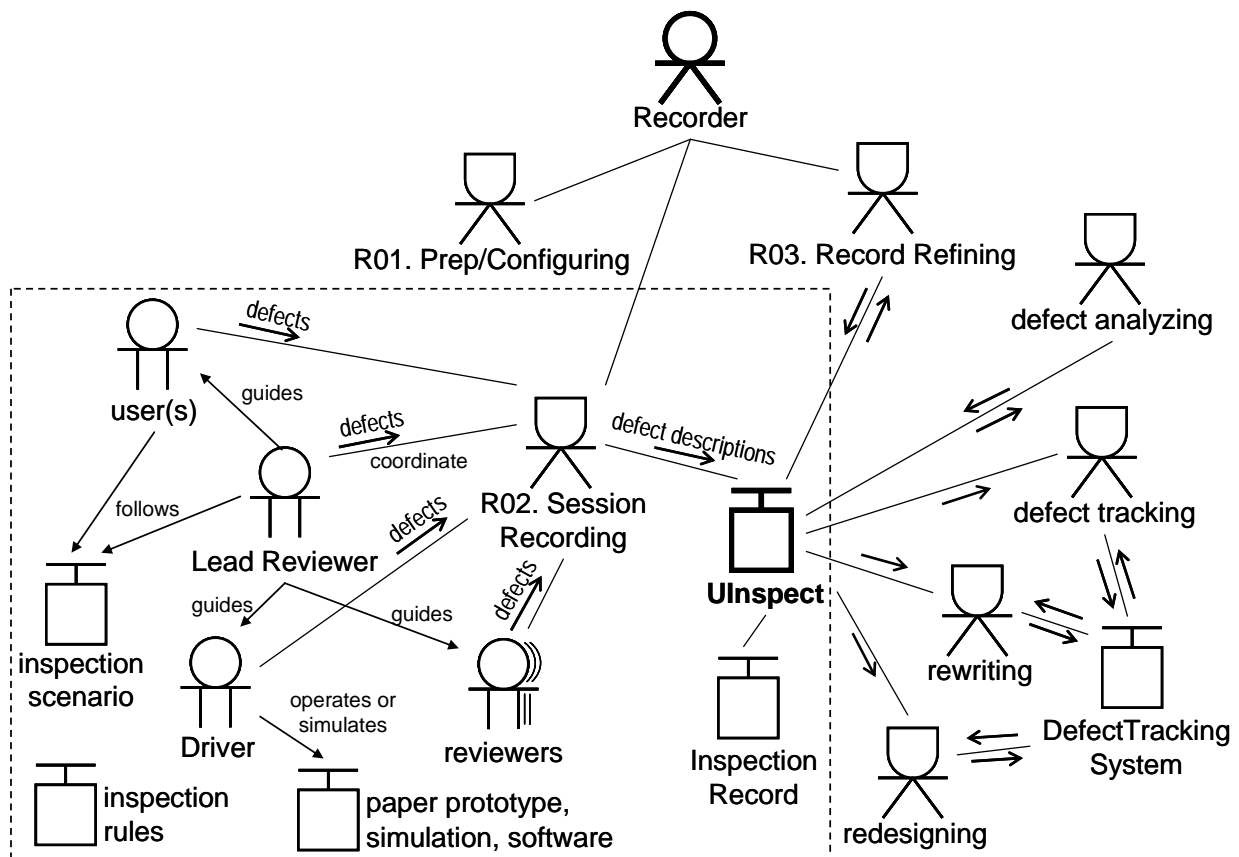


Figure 7 - System-centered Participation Map for collaborative usability inspections.

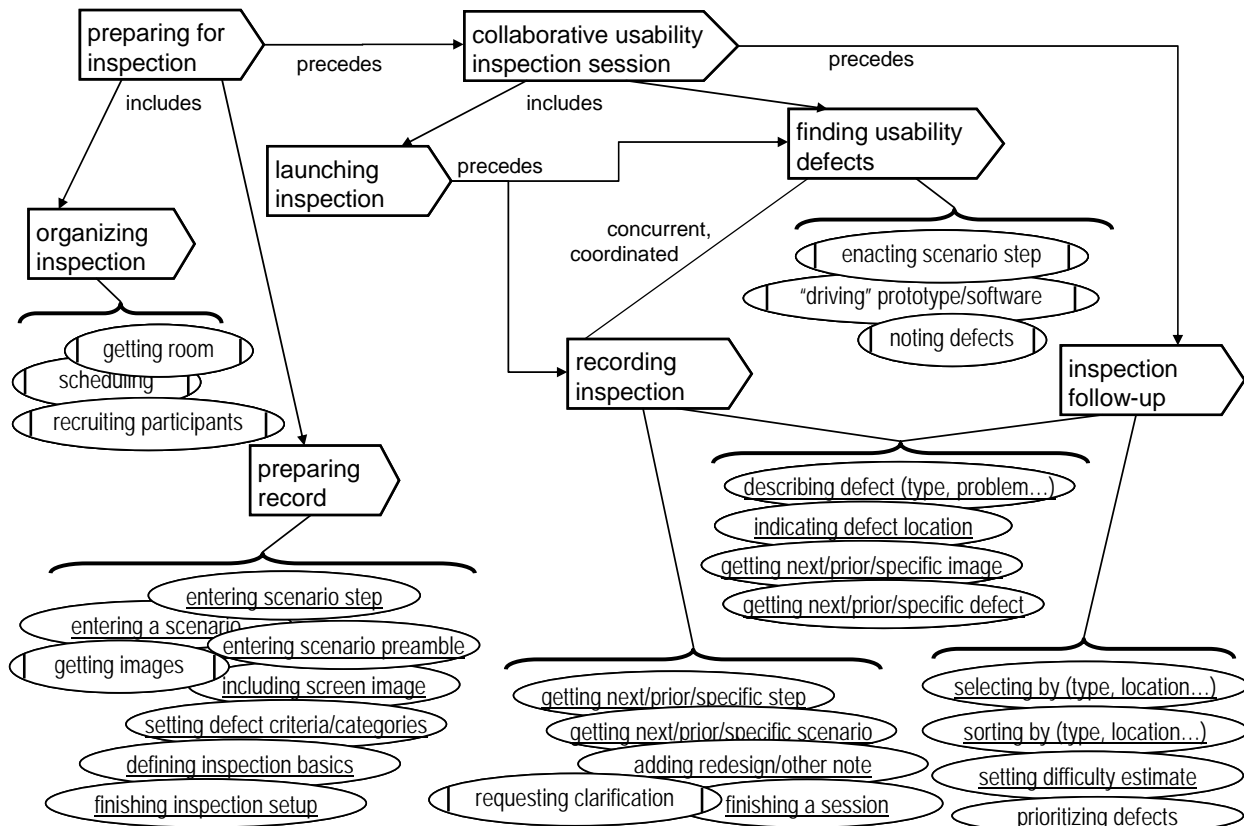


Figure 8 - Activity-Task Map for the UInspect collaborative usability inspections tool.

The pivotal role of session recorder can be described with a Role Profile.

R02 - Session-Recording Role

Activity (in which role is performed): Collaborative Usability Inspection Session.

Purpose: quickly, accurately, and efficiently capture usability defects identified

Background (of role performers): trained and educated technical professional, probably member of design/development group (not non-technical or administrative); some familiarity with UInspect system and object being inspected (at least from Preparing for Inspection activity); some knowledge of basic user interface and usability concepts required, familiarity with inspection process desirable; role may be played repeatedly but infrequently, expert performance in using the UInspect tool is unlikely

Characteristics (of role performance): multi-part repetitive recording task requiring rapid interaction and quick judgment under some pressure; close coordination with Lead Reviewer; competing attention between verbal reports, comments, and recording activity

Design (implications for support of role performance): speed and accuracy are foremost requiring maximum visibility of recording options and categories, simplified means for marking screenshots or design images; need ability to quickly navigate forward and back or to specific defect record, UI image, or scenario step; system must allow incomplete and blank entries with ability to edit on the fly

Discussion

The integration provided by activity modeling is unlikely to appeal to everyone. Longstanding activity theorists may disdain the distillation of an elaborate body of discourse into a few simple profiles and diagrams. Designers already working comfortably within a usage-centered perspective may object to complicating a straightforward process with additional models. Designers immersed in more elaborate ethnographic approaches to analysis and design may consider the models of activity modeling too spare and simplistic.

Vocal objections to activity modeling are likely also to come from the software engineering community, particularly that segment most closely tied to the Unified Modeling Language. Although it is no doubt possible to find ways to shoehorn activity modeling into the procrustean bed of UML, this proposal makes no attempt to do so for two reasons. First, the purposes of activity modeling and the professional constituency it is intended to serve are quite distinct from those of UML. Moreover, UML is particularly deficient in supporting visual and interaction design of user interfaces. Although it is possible to force fit user interface design problems and concerns into the UML, the results are rarely satisfying, particularly from the perspective of the interests and common practices of interaction designers and other related professionals, which is precisely the constituency targeted by activity modeling. Second, UML itself is objectionable from a human-factors perspective (Henderson-Sellers and Constantine, 1995a; 1995b), making it particularly inappropriate as a standard of reference for models directed toward human-factors professionals.

Activity modeling bears some similarity to other techniques in current analysis and design practice, particularly business process modeling, and it might be argued that activity modeling is already covered by such techniques. However, both the objectives and the targeted users of business process modeling and activity modeling are distinct. Business process modeling is intended to serve the needs of business analysts and requirements engineers capturing precise definitions of key business processes in terms of step-by-step workflow, well-defined decision criteria, and the exact flow of control, information, and resources that must be reflected in software. For our purposes, then, business processes may be thought of as special cases of the broader construct of human activities. Activity modeling serves design professionals trying to capture the broad design implications of human activities as loose and relatively unconstrained and often unpredictable combinations of tasks toward the end of creating a more useful and usable products. The differing needs and focuses of attention of these two activities undertaken for differing purposes require different tools and techniques.

Regrettably, the term activity is also used in UML, where it refers to the behavior of a system expressed as a set of control and data flows. A UML activity diagram is used to represent the dynamic view of a system as the flow of control and data from activity to activity (Booch, Rumbaugh, Jacobson, 2005). Business process modeling borrows the notation of UML activity diagrams to create business process diagrams which thus represent the workflow, inputs, and outputs of business processes (Penker and Eriksson, 2001).

The primary purpose of activity modeling is to provide practicing designers of any ilk with a practical tool for expressing the most salient aspects of the human activity context within which use of a system is embedded. Through relatively modest extensions and minimal changes in process, the method of usage-centered design can be firmly anchored in activity theory thereby improving the practice of interaction design and expanding the scope of application of activity theory.

No claim is made that activity modeling in its present form is complete. Although the current formulation was developed through an extensive retrospective analysis of many projects and has been refined through generous feedback from early adapters and other colleagues, only long-term application in a variety of settings will expose weak or missing elements or reveal which elements are most often of significance in the design process. In putting this revised process to use, designers have the opportunity to contribute to the refinement of both theory and practice, both models and methods.

References

- Booch, G., Rumbaugh, J., and Jacobson, I. (2005) *The Unified Modeling Language User Guide: Second Edition*. Boston: Addison-Wesley.
- Brown, R. B. K., Hyland, P., & Piper, I. C. (2005) "Eliciting and Specifying Requirements for Highly Interactive Systems Using Activity Theory." *Interact 2005 Proceedings - IFIP WG 2.7 / 13.4 - User Interface Engineering*.
- Carroll, J. M., ed. (1995) *Scenario-Based Design*. New York: Wiley.
- Cockburn, A. (2001) *Writing Effective Use Cases*. Boston: Addison-Wesley.
- Constantine, L. L. (1994) "Essentially Speaking," *Software Development*, 2 (11). Reprinted in L. L. Constantine, *The Peopleware Papers*. Upper Saddle River, NJ: Prentice Hall, 2001.
- Constantine, L. L. (1995) "Essential Modeling: Use Cases for User Interfaces." *Interactions*, 2 (2): 34-46, April 1995.
- Constantine, L. L. (1998) "Abstract Prototyping," *Software Development*, 6 (10), October. Reprinted in S. Ambler and L. Constantine, eds., *The Unified Process Elaboration Phase*. San Francisco: CMP Books, 2000.
- Constantine, L. L. (2002) "Process Agility and Software Usability," *Information Age*, August 2002.
- Constantine, L. L. (2003) "Canonical abstract prototypes for abstract visual and interaction design." In J. Jorge, N. Jardim Nunes, and J. Falcao e Cunha, Eds. *Interactive Systems: Design, Specification, and Verification*. Proceedings, 10th International Workshop, DSV-IS 2003, Funchal, Madeira Island, Portugal, 11-13 June 2003. Lecture Notes in Computer Science, Vol. 2844. ISBN: 3-540-20159-9 Springer-Verlag.
- Constantine, L. L. (2004) "Beyond User-Centered Design and User Experience." *Cutter IT Journal*, 17, 2: 2-11.
- Constantine, L. L. (2005) "Peer Reviews for Usability," *Cutter IT Journal*, 18 (1), January 2005.
- Constantine, L. L. (2006) "Users, Roles, and Personas." In J. Pruitt and T. Adlin (eds.) *The Persona Lifecycle: Keeping People in Mind Throughout Product Design*. San Francisco: Morgan-Kaufman.
- Constantine, L. L., and Lockwood, L. A. D. (1999) *Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design*. Reading, MA: Addison-Wesley, 1999.
- Constantine, L. L., and Lockwood, L. A. D. (2002) "Usage-Centered Engineering for Web Applications," *IEEE Software*, 19 (2), March/April 2002, pp 42-50.
- Cooper, A., and Reimann, R. M. (2003) *About Face 2.0: The Essentials of Interaction Design*. New York: Wiley.
- Duignan, M., Noble, J., & Biddle, R. (2006) "Activity theory for design." *Proceedings, HWID 2006*. University of Madeira.
- Engeström, Y., Miettinen, R. & Punamäki, R-L. (Eds.) (1999). *Perspectives on Activity Theory*. Cambridge University Press.
- Fowler, M., and Scott, K. (1997) *UML Distilled*. Reading, MA: Addison-Wesley.
- Gay, G. & Hembrooke, H. (2004) *Activity-Centered Design*. MIT Press.
- Hackos, J. T., and Redish, J. C. (1998) *User and Task Analysis for Interface Design*. New York: Wiley.
- Henderson-Sellers, B., and Constantine, L. L. (1995a) "Notation Matters. Part 1: Framing the Issues," *Report on Object Analysis and Design*, 2 (3):25-29, September-October.
- Henderson-Sellers, B., and Constantine, L. L. (1995b) "Notation Matters. Part 2: Applying the Principles," *Report on Object Analysis and Design*, 2 (4):25-27, November-December.
- Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading, MA: Addison-Wesley, 1992.
- Kaptalinin, V., Nardi, B. A., & Macaulay, C. (1999) "The activity checklist." *Interactions* 6, 4: 27-39.
- Lockwood, L. A. D., and Constantine, L. L. (2003) "Usability by Inspection: Collaborative Techniques for Software and Web Applications." In L. Constantine (ed.) *forUSE 2003 Performance by Design: Proceedings of the Second International Conference on Usage-Centered Design*. Rowley, MA: Ampersand Press, 2003.

- McMenamin, S. M., & Palmer, J. (1984) *Essential Systems Analysis*. Englewood, Cliffs, NJ: Prentice Hall.
- Nardi, B. (ed.) (1996) *Context and Consciousness*. MIT Press.
- Norman, D. (2005) "Human-Centered Design Considered Harmful." *Interactions*, 12, 4: 14-19
- Norman, D. (2006) Private communication.
- Patton, J. "Hitting the target: adding interaction design to agile software development." In *Proceedings, Conference on Object Oriented Programming Systems Languages and Applications*. New York: ACM Press.
- Penker, M., and Eriksson, H. (2001) *Business Modeling With UML: Business Patterns at Work*. New York: Wiley.
- Snyder, C. (2003) *Paper Prototyping*. San Francisco: Morgan-Kaufmann.
- Strope, J. (2003) "Designing for Breakthroughs in User Performance." In L. Constantine, ed., *Performance by Design: Proceedings of forUSE 2003, the Second International Conference on Usage-Centered Design*. Rowley, MA: Ampersand Press.
- Windl, H. (2002) "Designing a Winner: Creating STEP 7 lite with Usage-Centered Design." In L. Constantine, ed., *forUSE 2002: Proceedings of the First International Conference on Usage-Centered Design*. Rowley, MA: Ampersand Press.