**EVALUATION TECHNIQUES**

* **Evaluation** 
  + tests usability and functionality of system
  + occurs in laboratory, field and/or in collaboration with users
  + evaluates both design and implementation
  + should be considered at all stages in the design life cycle

**2.11.1 Goals of Evaluation**

Evaluation has three main goals:

* to assess the extent and accessibility of the system‘s functionality
* to assess users‘ experience of the interaction
* to identify any specific problems with the system

The system‘s functionality is important in that it must accord with the user‘s requirements. In other words, the design of the system should enable users to perform their intended tasks more easily. This includes not only making the appropriate functionality available within the system, but making it clearly reachable by the user in terms of the actions that the user needs to take to perform the task. It also involves matching the use of the system to the user‘s expectations of the task. For example, if a filing clerk is used to retrieving a customer‘s file by the postal address, the same capability (at least) should be provided in the computerized file system. Evaluation at this level may also include measuring the user‘s performance with the system, to assess the effectiveness of the system in supporting the task.

In addition to evaluating the system design in terms of its functional capabilities, it is important to assess the user‘s experience of the interaction and its impact upon him. This includes considering aspects such as how easy the system is to learn, its usability and the user‘s satisfaction with it. It may also include his enjoyment and emotional response, particularly in the case of systems that are aimed at leisure or entertainment. It is important to identify areas of the design that overload the user in some way, perhaps by requiring an excessive amount of information to be remembered, for example. Much evaluation is aimed at measuring features such as these.

The final goal of evaluation is to identify specific problems with the design. These may be aspects of the design which, when used in their intended context, cause unexpected results, or confusion amongst users. This is, of course, related to both the functionality and usability of the design (depending on the cause of the problem). However, it is specifically concerned with identifying trouble-spots which can then be rectified.

**2.11.2 Evaluation through expert analysis**

There are four approaches to expert analysis:

* Cognitive walkthrough,
* Heuristic evaluation,
* The use of models
* Use of previous work **Cognitive walkthrough,**

1. A specification or prototype of the system. It doesn‘t have to be complete, but it should be fairly detailed. Details such as the location and wording for a menu can make a big difference.
2. A description of the task the user is to perform on the system. This should be a representative task that most users will want to do.
3. A complete, written list of the actions needed to complete the task with the proposed system.
4. An indication of who the users are and what kind of experience and knowledge the valuators can assume about them.

Given this information, the evaluators step through the action sequence (identified in item 3 above) to critique the system and tell a believable story about its usability. To do this, for each action, the evaluators try to answer the following four questions for each step in the action sequence.

1. **Is the effect of the action the same as the user’s goal at that point?** Each user action will have a specific effect within the system. Is this effect the same as what the user is trying to achieve at this point? For example, if the effect of the action is to save a document, is

‗saving a document‘ what the user wants to do?

1. **Will users see that the action is available?**

Will users see the button or menu item, for example, that is used to produce the action? This is not asking whether they will recognize that the button is the one they want. This is merely asking whether it is visible to them at the time when they will need to use it.

Instances where the answer to this question might be ‗no‘ are, for example, where a VCR remote control has a covered panel of buttons or where a menu item is hidden away in a submenu.

1. **Once users have found the correct action, will they know it is the one they need?**

Complements the previous question. It is one thing for a button or menu item to be visible, but will the user recognize that it is the one he is looking for to complete his task? Where the previous question was about the visibility of the action, this one is about whether its meaning and effect is clear.

1. **After the action is taken, will users understand the feedback they get?**

If you now assume that the user did manage to achieve the correct action, will he know that the has done so? Will the feedback given be sufficient confirmation of what has actually happened? This is the completion of the execution–evaluation interaction cycle In order to determine if they have accomplished their goal, users need appropriate feedback. It is vital to document the cognitive walkthrough to keep a record of what is good **Heuristic evaluation**

Nielsen‘s ten heuristics are:

1. **Visibility of system status** Always keep users informed about what is going on, through appropriate feedback within reasonable time. For example, if a system operation will take some time, give an indication of how long and how much is complete.
2. **Match between system and the real world** The system should speak the user‘s language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in natural and logical order.
3. **User control and freedom** Users often choose system functions by mistake and need a clearly marked ‗emergency exit‘ to leave the unwanted state without having to go through an extended dialog. Support undo and redo.
4. **Consistency and standards** Users should not have to wonder whether words, situations or actions mean the same thing in different contexts. Follow platform conventions and accepted standards.
5. **Error prevention** Make it difficult to make errors. Even better than good error messages is a careful design that prevents a problem from occurring in the first place.
6. **Recognition rather than recall** Make objects, actions and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use** Allow users to tailor frequent actions. Accelerators – unseen by the novice user – may often speed up the interaction for the expert user to such an extent that the system can cater to both in experienced and experienced users.

**The use of models**

A third expert-based approach is the use of models. Certain cognitive and design models provide a means of combining design specification and evaluation into the same framework. For example, the GOMS (goals, operators, methods and selection) model predicts user performance with a particular interface and can be used to filter particular design options. Similarly, lower-level modeling techniques such as the keystroke-level model provide predictions of the time users will take to perform low-level physical tasks. Design methodologies, such as design rationale also have a role to play in evaluation at the design stage. Design rationale provides a framework in which design options can be evaluated. By examining the criteria that are associated with each option in the design, and the evidence that is provided to support these criteria, informed judgments can be made in the design. Dialog models can also be used to evaluate dialog sequences for problems, such as unreachable states, circular dialogs and complexity.

**Using previous studies in evaluation**

A final approach to expert evaluation exploits this inheritance, using previous results as evidence to support (or refute) aspects of the design. It is expensive to repeat experiments continually and an expert review of relevant literature can avoid the need to do so. It should be noted that experimental results cannot be expected to hold arbitrarily across contexts. The reviewer must therefore select evidence carefully, noting the experimental design chosen, the population of participants used, the analyses performed and the assumptions made.

**2.11.3 User Participation**

The different approaches to evaluation through user participation. These include empirical or experimental methods, observational methods, query techniques, and methods that use physiological monitoring, such as eye tracking and measures of heart rate and skin conductance.

**2.11.4 Empirical methods**

One of the most powerful methods of evaluating a design or an aspect of a design is to use a controlled experiment. This provides empirical evidence to support a particular claim or hypothesis. It can be used to study a wide range of different issues at different levels of detail. Any experiment has the same basic form. The evaluator chooses a hypothesis to test, which can be determined by measuring some attribute of participant behavior. A number of experimental conditions are considered which differ only in the values of certain controlled variables. Any changes in the behavioral measures are attributed to the different conditions. Within this basic form there are a number of factors that are important to the overall reliability of the experiment, which must be considered carefully in experimental design. These include the participants chosen, the variables tested and manipulated, and the hypothesis tested. **2.12 Universal design**

**2.12.1 Design principles**

These principles give us a framework in which to develop universal designs.

1. **Equitable use**: the design is useful to people with a range of abilities and appealing to all. No user is excluded or stigmatized. Wherever possible, access should be the same for all; where identical use is not possible, equivalent use

should be supported. Where appropriate, security, privacy and safety provision should be available to all.

1. **Flexibility** in use: the design allows for a range of ability and preference, through choice of methods of use and adaptivity to the user‘s pace, precision and custom.
2. **The system be simple and intuitive to use**, regardless of the knowledge, experience, language or level of concentration of the user. The design needs to support the user‘s expectations and accommodate different language and literacy skills. It should not be unnecessarily complex and should be organized to facilitate access to the most important areas. It should provide prompting and feedback as far as possible.
3. **Perceptible information**: the design should provide effective communication of information regardless of the environmental conditions or the user‘s abilities. Redundancy of presentation is important: information should be represented in different forms or modes (e.g. graphic, verbal, text, touch). Essential information should be emphasized and differentiated clearly from the peripheral content. Presentation should support the range of devices and techniques used to access information by people with different sensory abilities.
4. **Tolerance for error:** minimizing the impact and damage caused by mistakes or unintended behavior. Potentially dangerous situations should be removed or made hard to reach. Potential hazards should be shielded by warnings. Systems should fail safe from the user‘s perspective and users should be supported in tasks that require concentration.
5. **Low physical effort**: systems should be designed to be comfortable to use, minimizing physical effort and fatigue. The physical design of the system should allow the user to maintain a natural posture with reasonable operating effort. Repetitive or sustained actions should be avoided.
6. **Requires size and space for approach and use**: the placement of the system should be such that it can be reached and used by any user regardless of body size, posture or mobility. Important elements should be on the line of sight for both seated and standing users. All physical components should be comfortably reachable by seated or standing users. Systems should allow for variation in hand size and provide enough room for assistive devices to be used. These seven principles give us a good starting point in considering universal design. They are not all equally applicable to all situations, of course. For example, principles six and seven would be vital in designing an information booth but less important in designing word- processing software. But they provide a useful checklist of considerations for designers, together with guidelines on how each principle can be achieved.

**2.12.2 Multimodal Interaction**

Providing access to information through more than one mode of interaction is an important principle of universal design.Such design relies on multi-modal interaction. there are five senses: sight, sound, touch, taste and smell. Sight is the predominant sense for the majority of people, and most interactive systems consequently use the visual channel as their primary means of presentation, through graphics, text, video and animation. However, sound is also an important channel, keeping us aware of our surroundings, monitoring people and events around us, reacting to sudden noises, providing clues and cues that switch our attention from one thing to another. It can also have an emotional effect on us, particularly in the case of music. Music is almost completely an auditory experience, yet is able to alter moods, conjure up visual images,

Evoke atmospheres or scenes in the mind of the listener. Touch, too, provides important information: tactile feedback forms an intrinsic part of the operation of many common tools – cars, musical instruments, pens, anything that requires holding or moving. It can form a sensuous bond between individuals, communicating a wealth of non-verbal information. Taste and smell are often less appreciated (until they are absent) but they also provide useful information in daily life: checking if food is bad, detecting early signs of fire, noticing that manure has been spread in a field, pleasure.

**2.12.3 Designing for diversity**

1. **Designing for users with disability**

Visual Impairment Hearing Impairment Physical Impairment Speech impairment

1. **Designing for different age groups**

Older people Children

1. **Designing for cultural differences**

References

1. Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, “Human Computer Interaction”, 3rd Edition, Pearson Education, 2004

**MODELS AND THEORIES**

**UNIT 3 MODELS AND THEORIES**

Cognitive models - Socio-Organizational issues and stake holder requirements – Communication and collaboration models - Hypertext, Multimedia and WWW.

**3.1 COGNITIVE MODELS**

Cognitive models represent users of interactive systems.

* Hierarchical models represent a user‗s task and goal structure.
* Linguistic models represent the user–system grammar.
* Physical and device models represent human motor skills.
* Cognitive architectures underlie all of these cognitive models.

**3.1.1 GOAL AND TASK HIERARCHIES**

To achieve this goal we divide it into several subgoals, say gathering the data together, producing the tables and histograms, and writing the descriptive material. Concentrating on the data gathering, we decide to split this into further subgoals: find the names of all introductory HCI textbooks and then search the book sales database for these books. Similarly, each of the other subgoals is divided up into further subgoals, until some level of detail is found at which we decide to stop. We thus end up with a hierarchy of goals and subgoals.

We can go on decomposing tasks until we get down to the individual hand and eye movements of the user, or we can stop at a more abstract level. Where do we start? In a similar way, we can start our analyses at different points in the hierarchy of goals. At the extreme we could extend our analysis to larger and larger goals: ‗light hob‗ is a subgoal of

‗boil peas‗ and so on to goals such as ‗have my dinner‗, ‗feed‗ and ‗stay alive‗.

These two questions are issues of granularity, and both of the methods described below leave this to some extent in the hands of the designer. Different design issues demand different levels of analysis. However, both methods operate at a relatively low level; neither would attempt to start with such an abstract goal as ‗produce a report‗ which will involve real creativity and difficult problem solving. Instead they confine themselves to more routine learned behavior. This most abstract task is referred to as the unit task. The unit task does not require any problem-solving skills on the part of the user, though it frequently demands quite sophisticated problem-solving skills on the part of the designer to determine them. What do we do when there are several ways of solving a problem, or if the solutions to two subgoals interact? Users will often have more than one way to achieve a goal and there must be some way of representing how they select between competing solutions.

**GOMS**

The GOMS model of Card, Moran and Newell is an acronym for Goals, Operators, Methods and Selection

**Goals** These are the user‗s goals, describing what the user wants to achieve. GOMS the goals are taken to represent a ‗memory point‗ for the user, from which he can evaluate what should be done and to which he may return should any errors occur.

**Operators** These are the lowest level of analysis. They are the basic actions that the user must perform in order to use the system. They may affect the system (for example, press the

‗X‗ key) or only the user‗s mental state (for example, read the dialog box). There is still a degree of flexibility about the granularity of operators; we may take the command level

‗issue the SELECT command‗ or be more primitive: ‗move mouse to menu bar, press center mouse button .

**Methods** There are typically several ways in which a goal can be split into subgoals. For instance, in a certain window manager a currently selected window can be closed to an icon either by selecting the ‗CLOSE‗ option from a pop-up menu, or by hitting the ‗L7‗ function key. In GOMS these two goal decompositions are referred to as methods, so we have the CLOSE-METHOD and the L7-METHOD:

GOAL: ICONIZE-WINDOW

. [select GOAL: USE-CLOSE-METHOD

. . MOVE-MOUSE-TO-WINDOW-HEADER

. . POP-UP-MENU

. . CLICK-OVER-CLOSE-OPTION

GOAL: USE-L7-METHOD

. . PRESS-L7-KEY]

The dots are used to indicate the hierarchical level of goals. Selection From the above snippet we see the use of the word select where the choice of methods arises. GOMS does not leave this as a random choice, but attempts to predict which methods will be used. This typically depends both on the particular user and on the state of the system and details about the goals.

Rule 1: Use the CLOSE-METHOD unless another rule applies.

Rule 2: If the application is ‗blocks‗ use the L7-METHOD.

The goal hierarchies described in a GOMS analysis are almost wholly below the level of the unit task defined earlier. A typical GOMS analysis would therefore consist of a single high- level goal, which is then decomposed into a sequence of unit tasks, all of which can be further decomposed down to the level of basic operators:

GOAL: EDIT-MANUSCRIPT

GOAL: EDIT-UNIT-TASK repeat until no more unit tasks The goal decomposition between the overall task and the unit tasks would involve detailed understanding of the user‗s problem-solving strategies and of the application domain.

**Cognitive complexity theory**

Cognitive complexity refer to the number of mental structures an individual uses, how abstract they are and how they interact to shape his discernment or an individual difference variable linked with a wide range of communication skills and associated abilities.

Individuals with high cognitive complexity have the capacity to analyze a situation to discern various constituent elements and explore connections and possible relationships among the elements. These individuals think in a multidimensional way. The assumption of the complexity theory is that the more an event can be differentiated and parts considered in novel relationships, the more sophisticated the response and successful the solution. Whereas less complex individuals can be trained to understand a complicated set of detailed differentiations for a specific context, highly complex individuals are highly flexible in creating distinctions in new situations.

Individuals with high cognitive complexity are open to new information, attracted to other individuals of high complexity, highly flexibility, socially influential, problem solvers, strategic planners, highly creative, effective communicators and generally good leaders.

**Problems and extensions of goal hierarchies**

The formation of a goal hierarchy is largely a post hoc technique and runs a very real risk of being defined by the computer dialog rather than the user. One way to rectify this is to produce a goal structure based on pre-existing manual procedures and thus obtain a natural hierarchy . GOMS defines its domain to be that of expert use, and thus the goal structures that are important are those which users develop out of their use of the system. The conceptual framework of goal hierarchies and user goal stacks can be used to express interface issues, not directly addressed by the notations above. For instance, we can use this to examine in more detail the closure problem with early automated teller machines (ATMs) mentioned in the Design Focus box These early ATMs gave the customers the money before returning their cards. Unfortunately, this led to many customers leaving their cards behind. This was despite on-screen messages telling them to wait. This is referred to as a problem of closure. The user‗s principal goal is to get money; when that goal is satisfied, the user does not complete or close the various subtasks which still remain open:

GOAL: GET-MONEY

. GOAL: USE-ATM

. . INSERT-CARD

. . ENTER-PIN

. . ENTER-AMOUNT

. . COLLECT-MONEY

<< outer goal now satisfied goal stack popped >>

. . COLLECT-CARD – subgoal operators missed

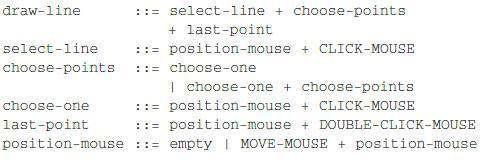
**3.1.2 LINGUISTIC MODELS**

The user‗s interaction with a computer is often viewed in terms of a language, so it is not surprising that several modeling formalisms have developed centered around this concept. BNF grammars are frequently used to specify dialogs.

The models here, although similar in form to dialog design notations, have been proposed with the intention of understanding the user‗s behavior and analyzing the cognitive difficulty of the interface.

**BNF**

Representative of the linguistic approach is Reisner‗s use of Backus–Naur Form (BNF) rules to describe the dialog grammar . This views the dialog at a purely syntactic level, ignoring the semantics of the language. BNF has been used widely to specify the syntax of computer programming languages, and many system dialogs can be described easily using BNF rules. For example, imagine a graphics system that has a line-drawing function. To select the function the user must select the ‗line‗ menu option. The line-drawing function allows the user to draw a polyline, that is a sequence of line arcs between points. The user selects the points by clicking the mouse button in the drawing area. The user double clicks to indicate the last point of the polyline.



The aims in the description are of two types: non-terminals, shown in lower case, and terminals, shown in upper case. Terminals represent the lowest level of user behavior, such as pressing a key, clicking a mouse button or moving the mouse. Non-terminals are higher-level abstractions. The non-terminals are defined in terms of other non-terminals and terminals by a definition of the form name ::= expression The ‗::=‗ symbol is read as ‗is defined as‗. Only non-terminals may appear on the left of a definition. The right-hand side is built up using two operators ‗+‗ (sequence) and ‗|‗ (choice). For example, the first rule says that the non- terminal draw-line is defined to be select-line followed by choose-points followed by lastpoint. All of these are non-terminals, that is they do not tell us what the basic user actions are. The second rule says that select-line is defined to be position mouse (intended to be over the ‗line‗ menu entry) followed by CLICK-MOUSE. This is our first terminal and represents the actual clicking of a mouse.

Position-mouse is, we look at the last rule. This tells us that there are two possibilities for position-mouse (separated by the ‗|‗ symbol). One option is that position-mouse is empty – a special symbol representing no action. That is, one option is not to move the mouse at all. The other option is to doa MOVE-MOUSE action followed by position-mouse. This rule is recursive, and this second position-mouse may itself either be empty or be a MOVE-MOUSE action followed by position-mouse, and so on. That is, position-mouse may be any number of MOVE-MOUSE actions whatsoever. Similarly, choose-points is defined recursively, but this time it does not have the option of being empty. It may be one or more of the non-terminal choose-one which is itself defined to be (like select-line) position-mouse followed by

CLICK-MOUSE.

The BNF description of an interface can be analyzed in various ways. One measure is to count the number of rules. The more rules an interface requires to use it, the more complicated it is. This measure is rather sensitive to the exact way the interface is described. For example, we could have replaced the rules for choose points and choose-one with the single definition choose-points ::= position-mouse + CLICK-MOUSE | position-mouse + CLICK-MOUSE + choose-points.

**Task–action grammar**

Measures based upon BNF have been criticized as not ‗cognitive‗ enough. They ignore the advantages of consistency both in the language‗s structure and in its use of command names and letters. Task–action grammar (TAG)

**3.1.3 THE CHALLENGE OF DISPLAY-BASED SYSTEMS**

hierarchical and grammar-based techniques were initially developed when most interactive systems were command line, or at most, keyboard and cursor based. There are significant worries, therefore, about how well these approaches can generalize to deal with more modern windowed and mouse-driven interfaces. Pressing a cursor key is a reasonable lexeme, but moving a mouse one pixel is less sensible. In addition, pointer-based dialogs are more display oriented. Clicking a cursor at a particular point on the screen has a meaning dependent on the current screen contents. This problem can be partially resolved by regarding operations such as ‗select region of text‗ or ‗click on quit button‗ as the terminals of the grammar. If this approach is taken, the detailed mouse movements and parsing of mouse events in the context of display information (menus, etc.) are abstracted away. Goal hierarchy methods have different problems, as more display-oriented systems encourage less structured methods for goal achievement. Instead of having well-defined plans, the user is seen as performing a more exploratory task, recognizing fruitful directions and backing out of others. Typically, even when this exploratory style is used at one level,

WRITE\_LETTER

. FIND\_SIMILAR\_LETTER

. COPY\_IT

. EDIT\_COPY

**3.1.4 PHYSICAL AND DEVICE MODELS**

**Keystroke-level model**

The human motor system is well understood. KLM (Keystroke-Level Model) uses this understanding as a basis for detailed predictions about user performance. It is aimed at unit tasks within interaction – the execution of simple command sequences, typically taking no more than 20 seconds. Examples of this would be using a search and replace feature, or changing the font of a word. It does not extend to complex actions such as producing a diagram. The assumption is that these more complex tasks would be split into subtasks (as in GOMS) before the user attempts to map them into physical actions.

**The task is split into two phases:**

Acquisition of the task, when the user builds a mental representation of the task;

Execution of the task using the system‗s facilities.

During the acquisition phase, the user will have decided how to accomplish the task using the primitives of the system, and thus, during the execution phase, there is no high-level mental activity – the user is effectively expert. KLM is related to the GOMS model, and can be thought of as a very low-level GOMS model where the method is given.

The model decomposes the execution phase into five different physical motor operators, a mental operator and a system response operator:

K Key stroking, actually striking keys, including shifts and other modifier keys.

B Pressing a mouse button.

P Pointing, moving the mouse (or similar device) at a target.

H Homing, switching the hand between mouse and keyboard.

D Drawing lines using the mouse.

M Mentally preparing for a physical action.

R System response which may be ignored if the user does not have to wait for it, as in copy typing.

The execution of a task will involve interleaved occurrences of the various operators. For instance, imagine we are using a mouse-based editor. If we notice a single character error we will point at the error, delete the character and retype it, and then return to our previous typing point. This is decomposed as follows:

1. Move hand to mouse H[mouse]
2. Position mouse after bad character PB[LEFT]
3. Return to keyboard H[keyboard]
4. Delete character MK[DELETE]
5. Type correction K[char]
6. Reposition insertion point H[mouse]MPB[LEFT]

**3.1.5 COGNITIVE ARCHITECTURES**

The concept of taking a problem and solving it by divide and conquer using subgoals is central to GOMS. CCT assumes the distinction between long- and short-term memory, with production rules being stored in long-term memory and ‗matched‗ against the contents of short-term (or working) memory to determine which ‗fire‗. The values for various motor and mental operators in KLM were based on the Model Human Processor (MHP) architecture of Card, Moran and Newell. Another common assumption, itis the distinction between linguistic levels – semantic, syntactic and lexical – as an architectural model of the user‗s understanding.

The problem space model Rational behavior is characterized as behavior that is intended to achieve a specific goal. This element of rationality is often used to distinguish between intelligent and machine-like behavior. In the field of artificial intelligence (AI), a system exhibiting rational behavior is referred to as a knowledge-level system. A knowledge- level system contains an agent behaving in an environment. The agent has knowledge about itself and its environment, including its own goals. It can perform certain actions and sense information about its changing environment. As the agent behaves in its environment, it changes the environment and its own knowledge. We can view the overall behavior of the knowledge-level system as a sequence of environment and agent states as they progress in time. The goal of the agent is characterized as a preference over all possible sequences of agent/environment states. The search proceeds by moving from one state to another possible state by means of operations or actions, the ultimate goal of which is to arrive at one of the desired states. This very general model of computation is used in the ordinary task of the programmer. Once she has identified a problem and a means of arriving at the solution to the problem (the algorithm), the programmer then represents the problem and algorithm in a programming language, which can be executed on a machine to reach the desired state. The architecture of the machine only allows the definition of the search or problem space and the actions that can occur to traverse that space. Termination is also assumed to happen once the desired state is reached.

The new computational model is the problem space model, based on the problem- solving work of Newell and Simon at Carnegie–Mellon University. A problem space consists of a set of states and a set of operations that can be performed on the states. Behavior in a problem space is a two-step process. First, the current operator is chosen based on the current state and then it is applied to the current state to achieve the new state. The problem space must represent rational behavior, and so it must characterize the goal of the agent. A problem space represents a goal by defining the desired states as a subset of all possible states. Once the initial state is set, the task within the problem space is to find a sequence of operations that form a path within the state space from the initial state to one of the desired states, whereupon successful termination occurs.

We can highlight four different activities that occur within a problem space: goal formulation, operation selection, operation application and goal completion. The relationship between these problem space processes and knowledge-level activity is key. Perception that occurs at the knowledge level is performed by the goal formulation process, which creates the initial state based on observations of the external environment. Actions at the knowledge level are operations in the problem space which are selected and applied. The real knowledge about the agent and its environment and goals is derived from the state/operator information in the problem space. Because of the goal formulation process, the set of desired states indicates the knowledge-level goal within the problem space. The operation selection process selects the appropriate operation at a given point in time because it is deemed the most likely to transform the state in the problem space to one of the desired states; hence rational behavior is implied.

Interacting cognitive subsystems (ICS) provides a model of perception, cognition and action, but unlike other cognitive architectures, it is not intended to produce a description of the user in terms of sequences of actions that he performs. ICS provides a more holistic view of the user as an information-processing machine. The emphasis is on determining how easy particular procedures of action sequences become as they are made more automatic within the user.

ICS attempts to incorporate two separate psychological traditions within one cognitive architecture. On the one hand is the architectural and general-purpose information-processing approach of short-term memory research. On the other hand is the computational and representational approach characteristic of psycholinguistic research and AI problem-solving literature.

The architecture of ICS is built up by the coordinated activity of nine smaller subsystems: five peripheral subsystems are in contact with the physical world and four are central, dealing with mental processes. Each subsystem has the same generic structure. A subsystem is described in terms of its typed inputs and outputs along with a memory store for holding typed information. It has transformation functions for processing the input and producing the output and permanently stored information. Each of the nine subsystems is specialized for handling some aspect of external or internal processing. For example, one peripheral subsystem is the visual system for describing what is seen in the world.

**1.2 SOCIO-ORGANIZATIONAL ISSUES AND STAKEHOLDER REQUIREMENTS**

There are several organizational issues that affect the acceptance of technology by users and that must therefore be considered in system design:

o systems may not take into account conflict and power relationships o those who benefit may not do the work o not everyone may use systems.

In addition to generic issues, designers must identify specific stakeholder requirements within their organizational context.

* Socio-technical models capture both human and technical requirements.
* Soft systems methodology takes a broader view of human and organizational issues.
* Participatory design includes the user directly in the design process.
* Ethnographic methods study users in context, attempting to take an unbiased perspective.

**3.2.1 ORGANIZATIONAL ISSUES**

**Cooperation or conflict?**

The term ‗computer-supported cooperative work‗ (CSCW) seems to assume that groups will be acting in a cooperative manner. This is obviously true to some extent;even opposing football teams cooperate to the extent that they keep (largely) within the rules of the game, but their cooperation only goes so far. People in organizations and groups have conflicting goals, and systems that ignore this are likely to fail spectacularly.

Imagine that an organization is already highly computerized, the different departments all have their own systems and the board decides that an integrated information system is needed. The production manager can now look directly at stocks when planning the week‗s work, and the marketing department can consult the sales department‗s contact list to send out marketing questionnaires.

The storekeeper always used to understate stock levels slightly in order to keep an emergency supply, or sometimes inflate the quoted levels when a delivery was due from a reliable supplier. Also, requests for stock information allowed the storekeeper to keep track of future demands and hence plan future orders. The storekeeper has now lost a sense of control and important sources of information. Members of the sales department are also unhappy: their contacts are their livelihood. The last thing they want is someone from marketing blundering in and spoiling a relationship with a customer built up over many years.

Some of these people may resort to subverting the system, keeping ‗sanitized‗ information online, but the real information in personal files.

**Changing power structures**

The identification of stakeholders will uncover information transfer and power relationships that cut across the organizational structure. Indeed, all organizations have these informal networks that support both social and functional contacts. The official lines of authority and information tend to flow up and down through line management.

The physical layout of an organization often reflects the formal hierarchy: each department is on a different floor, with sections working in the same area of an office. If someone from sales wants to talk to someone from marketing then one of them must walk to the other‗s office. Their respective supervisors can monitor the contact.

In face-to-face conversation, the manager can easily exert influence over a subordinate: both know their relative positions and this is reflected in the patterns of conversation and in other non-verbal cues. Email messages lose much of this sense of presence and it is more difficult for a manager to exercise authority. The ‗levelling‗ effect even makes it possible for subordinates to direct messages ‗diagonally‗ across the hierarchy, to their manager‗s peers, or, even worse, to their manager‗s manager!

**The invisible worker**

The ability to work and collaborate at a distance can allow functional groups to be distributed over different sites. This can take the form of cross-functional neighbourhood centers, where workers from different departments do their jobs in electronic contact with their functional colleagues. If the approach in an organization is ‗management by presence‗, that is you know someone is working because they are in the office, then there is no way a remote worker is going to be trusted. If, on the other hand, the style is ‗management by objectives‗, that is you know your subordinates are working because they are doing their jobs and producing results, then remote working is not so problematical.

**Who benefits?**

In these systems the sender has to do work in putting information into fields appropriately, but it is the recipient who benefits. Another example is shared calendars. The beneficiary of the system is a manager who uses the system to arrange meeting times, but whose personal secretary does the work of keeping the calendar up to date. Subordinates are less likely to have secretarial support, yet must keep up the calendar with little perceived benefit. Of course, chaos results when a meeting is automatically arranged and the subordinates may have to rearrange commitments that have not been recorded on the system. The manager may force use by edict or the system may simply fall into disuse. Many such groupware systems are introduced on a ‗see if it works‗ basis,

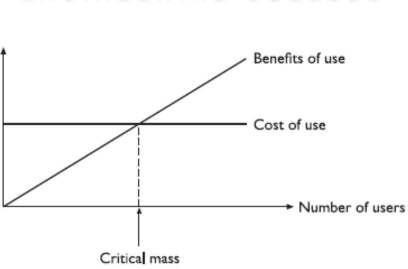
**Free rider problem**

A system may still not function symmetrically, which may be a problem, particularly with shared communication systems. One issue is the free rider problem. Take an electronic conferencing system. If there is plenty of discussion of relevant topics then there are obvious advantages to subscribing and reading the contributions. When considering writing a contribution, the effort of doing so may outweigh any benefits.

The total benefit of the system for each user outweighs the costs, but for any particular decision the balance is overturned. A few free riders in a conference system are often not a problem, as the danger is more likely from too much activity. In addition, in electronic conferences the patterns of activity and silence may reflect other factors such as expertise. It is easy for the number of free riders gradually to increase and the system slide into disuse. It is hard to enforce equal use, except by restrictive schemes such as round-robin contributions (everyone contributes something however short). In the real world, such problems are often solved by social pressure, and the free rider reacts to the collective censure of the group.

Increasing the visibility of participants‗ contributions might also help these social mechanisms. **Critical mass**

When telephones were only in public places, their use as a form of pervasive interpersonal communication was limited. However, once a large number of people have telephones in their homes it becomes worthwhile paying to have a telephone installed. In cost/benefit terms, the early subscribers probably have a smaller benefit than the cost. Only when the number of subscribers increases beyond the critical mass does the benefit for all dominate the cost .



**Figure: Cost/benefit of system use**

The telephone was useful for subgroups before it became beneficial for all. Even when only a small proportion of the population had personal telephones, they still formed a significant proportion of their social group, so these cliques of use could grow gradually over time.

**Automating processes – workflow and BPR**

Organizations have many such processes, and workflow systems aim to automate much of the process using electronic forms, which are forwarded to the relevant person based on pre-coded rules. Some workflow systems are built using special purpose groupware, often based on a notation for describing the desired workflow.

The rigid form of a typical workflow system is an example of global structuring. The

danger with any form of global structuring is that it may conflict with or inhibit more informal and less structured patterns of activity which also contribute to the organization‗s free running.

A more radical approach to organizational processes is found in business process re- engineering (BPR). Traditionally, organizations have been structured around functions: sales, accounts, stores, manufacturing. However, the purpose of an organization can be seen in terms of key business processes. The ordering/delivery process described above is a typical and important example. In BPR these processes are recorded and analyzed. Problems in the current process are noted and the whole process may be redesigned in order to make the path of the process more efficient. For example, instead of sending an order to the accounts department to approve, a list of customer credit limits could be given to the sales executives. They could then check the credit rating of the customer whilst on the phone and only forward the order to accounts if there are any unusual problems.

**Evaluating the benefits**

The benefits from cooperative systems, especially organization-wide systems such as email or electronic conferencing, are in terms of job satisfaction or more fluid information flow. Some, such as the video wall , are expected primarily to help social contact within the organization. It may be possible to measure contentment and job satisfaction using attitude questionnaires.

**3.2.2 CAPTURING REQUIREMENTS**

**Who are the stakeholders?**

A stakeholder can be defined as anyone who is affected by the success or failure of the system. It can be useful to distinguish different categories of stakeholder, and the following categorization from the CUSTOM approach is helpful for this: **Primary stakeholders** are people who actually use the system – the end-users.

**Secondary stakeholders** are people who do not directly use the system, but receive output from it or provide input to it (for example, someone who receives a report produced by the system).

**Tertiary stakeholders** are people who do not fall into either of the first two categories but who are directly affected by the success or failure of the system .Facilitating stakeholders are people who are involved with the design, development and maintenance of the system.

The aim of the design team is to meet the needs of as many stakeholders as possible.

The reality is that usually stakeholder needs are in conflict with each other. Sometimes this

does not matter: a company is unlikely to be too concerned that its competitors‗ requirement to maintain advantage over it is under threat by the new system.

**Socio-technical models**

The socio-technical systems view came about to counter this technology-centric position, by stressing that work systems were composed of both human and machine elements and that it was the interrelationship between these that should be central.

Socio-technical models for interactive systems are therefore concerned with technical, social, organizational and human aspects of design. They recognize the fact that technology is not developed in isolation but as part of a wider organizational environment. It is important to consider social and technical issues side by side so that human issues are not overruled by technical considerations.

The key focus of the socio-technical approach is to describe and document the impact of the introduction of a specific technology into an organization. Methods vary but most attempt to capture certain common elements:

* The problem being addressed: there is a need to understand why the technology is being proposed and what problem it is intended to solve.
* The stakeholders affected, including primary, secondary, tertiary and facilitating, together with their objectives, goals and tasks.
* The workgroups within the organization, both formal and informal.
* The changes or transformations that will be supported.
* The proposed technology and how it will work within the organization.
* External constraints and influences and performance measures.

**CUSTOM methodology**

CUSTOM is a socio-technical methodology designed to be practical to use in small organizations. It is based on the User Skills and Task Match (USTM) approach, developed to allow design teams to understand and fully document user requirements. CUSTOM focusses on establishing stakeholder requirements: all stakeholders are considered, not just the end- users.

It is applied at the initial stage of design when a product opportunity has been identified, so the emphasis is on capturing requirements. It is a forms-based methodology, providing a set of questions to apply at each of its stages.

There are six key stages to carry out in a CUSTOM analysis:

1. Describe the organizational context, including its primary goals, physical characteristics, political and economic background.
2. Identify and describe stakeholders. All stakeholders are named, categorized (as primary, secondary, tertiary or facilitating) and described with regard to personal issues, their role in the organization and their job. For example, CUSTOM addresses issues such as stakeholder motivation, disincentives, knowledge, skills, power and influence within the organization, daily tasks and so on.
3. Identify and describe work-groups. A work-group is any group of people who work together on a task, whether formally constituted or not. Again, work-groups are described in terms of their role within the organization and their characteristics.
4. Identify and describe task–object pairs. These are the tasks that must be performed, coupled with the objects that are used to perform them or to which they are applied.
5. Identify stakeholder needs. Stages 2–4 are described in terms of both the current system and the proposed system. Stakeholder needs are identified by considering the differences between the two. For example, if a stakeholder is identified as currently lacking a particular skill that is required in the proposed system then a need for training is identified.
6. Consolidate and check stakeholder requirements. Here the stakeholder needs list is checked against the criteria determined at earlier stages.

**Open System Task Analysis (OSTA)**

OSTA attempts to describe what happens when a technical system is introduced into an organizational work environment. Like CUSTOM, OSTA specifies both social and technical aspects of the system. However, whereas in CUSTOM these aspects are framed in terms of stakeholder perspectives, in OSTA they are captured through a focus on tasks.

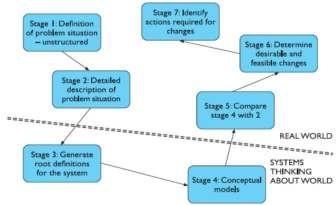
**OSTA has eight main stages:**

* 1. The primary task which the technology must support is identified in terms of users‗ goals.

Task inputs to the system are identified. These may have different sources and forms that may constrain the design.

* 1. The external environment into which the system will be introduced is described, including physical, economic and political aspects.
  2. The transformation processes within the system are described in terms of actions performed on or with objects.
  3. The social system is analyzed, considering existing work-groups and relationships within and external to the organization.
  4. The technical system is described in terms of its configuration and integration with other systems.
  5. Performance satisfaction criteria are established, indicating the social and technical requirements of the system.
  6. The new technical system is specified.

**Soft systems methodology**

Soft systems methodology (SSM) arises from the same tradition but takes a view of the organization as a system of which technology and people are components. There is no assumption of a particular solution: the emphasis is rather on understanding the situation fully.

**Participatory design**

Participatory design is a philosophy that encompasses the whole design cycle. It is design in the workplace, where the user is involved not only as an experimental subject or as someone to be consulted when necessary but as a member of the design team. Users are therefore active collaborators in the design process, rather than passive participants whose involvement is entirely governed by the designer. The argument is that users are experts in the work context and a design can only be effective within that context if these experts are allowed to contribute actively to the design process. In addition, introduction of a new system is liable to change the work context and organizational processes, and will only be accepted if these changes are acceptable to the user. Participatory design therefore aims to refine system requirements iteratively through a design process in which the user is actively involved.

Participatory design has three specific characteristics. It aims to improve the work environment and task by the introduction of the design. This makes design and evaluation context or work oriented rather than system oriented. Secondly, it is characterized by collaboration: the user is included in the design team and can contribute to every stage of the design. Finally, the approach is iterative: the design is subject to evaluation and revision at each stage.

The participatory design process utilizes a range of methods to help convey information between the user and designer. They include

**Brainstorming** This involves all participants in the design pooling ideas. This is informal and relatively unstructured although the process tends to involve ‗on the- structuring of the ideas as they materialize.

**Storyboarding** : Storyboards can be used as a means of describing the user‗s day-to-day activities as well as the potential designs and the impact they will have.

**Workshops** These can be used to fill in the missing knowledge of both user and designer and provide a more focussed view of the design. They may involve mutual enquiry in which both parties attempt to understand the context of the design from each other‗s point of view. The designer questions the user about the work environment in which the design is to be used, and the user can query the designer on the technology and capabilities that may be available. This establishes common ground between the user and designer and sets the foundation for the design that is to be produced. The use of role play can also allow both user and designer to step briefly into one another‗s shoes.

**Pencil and paper exercises** These allow designs to be talked through and evaluated with

very little commitment in terms of resources. Users can ‗walk through‗ typical tasks using paper mock-ups of the system design. This is intended to show up discrepancies between the user‗s requirements and the actual design as proposed. Such exercises provide a simple and cheap technique for early assessment of models.

**Effective Technical and Human Implementation of Computer-based Systems (ETHICS)** ETHICS methodology, stakeholders are included as participants in the decision making process. ETHICS considers the process of system development as one of managing change: conflicts will occur and must be negotiated to ensure acceptance and satisfaction with the system. If any party is excluded from the decision-making process then their knowledge and contribution is not utilized and they are more likely to be dissatisfied. However, participation is not always complete. Mumford recognizes three levels of participation:

**Consultative** – the weakest form of participation where participants are asked for their opinions but are not decision makers.

**Representative** – a representative of the participant group is involved in the decision making process.

**Consensus** – all stakeholders are included in the decision-making process.

The usual practice is that design groups are set up to include representatives from each stakeholder group and these groups make the design decisions, overseen by a steering committee of management and employee representatives.

1. **Make the case for change.** Change for its own sake is inappropriate. If a case cannot be made for changing the current situation then the process ends and the system remains as it is.
2. **Identify system boundaries**. This focusses on the context of the current system and its interactions with other systems, in terms of business, existing technology, and internal and external organizational elements. How will the change impact upon each of these?
3. **Describe the existing system**, including a full analysis of inputs and outputs and the various other activities supported, such as operations, control and coordination.
4. **Define key objectives,** identifying the purpose and function of each area of the organization.
5. **Define key tasks:** what tasks need to be performed to meet these objectives?
6. **Define key information needs,** including those identified by analysis of the existing system and those highlighted by definition of key tasks.
7. **Diagnose efficiency needs,** those elements in the system that cause it to underperform or perform incorrectly. If these are internal they can be redesigned out of the new system; if they are external then the new system must be designed to cope with them.
8. **Diagnose job satisfaction needs,** with a view to increasing job satisfaction where it is low.
9. **Analyze likely future changes,** whether in technology, external constraints (such as legal requirements), economic climate or stakeholder attitudes. This is necessary to ensure that the system is flexible enough to cope with change.
10. **Specify and prioritize objectives based on efficiency,** job satisfaction and future needs. All stakeholders should be able to contribute here as it is a critical stage and conflicting priorities need to be negotiated. Objectives are grouped as either primary (must be met) or secondary

**Ethnographic methods**

Ethnography is based on very detailed recording of the interactions between people and between people and their environment. It has a special focus on social relationships and how they affect the nature of work. The ethnographer does not enter actively into the situation, and does not see things from a particular person‗s viewpoint. However, an aim is to be uncultured, to understand the situation from within its own cultural framework. Culture here means that of the particular workgroup or organization, rather than that of society as a whole. Ethnographers try to take an unbiased and open-ended view of the situation. They report and do not like to speculate, so it is often unclear how well their approach can contribute to the design of new systems.

**Contextual inquiry**

Contextual inquiry has much in common with the ethnographic tradition: it studies the user in context, trying to capture the reality of his work culture and practice. However, it is also an approach rooted in practice and it differs in a number of significant ways from pure ethnographic study: the intention is to understand and to interpret the data gathered, and rather than attempting to take an open-ended view, the investigator acknowledges and challenges her particular focus. In addition, the explicit aim is to design a new system, whereas in a pure ethnographic study, it would be open ended.

The model of contextual inquiry is of the investigator being apprenticed to the user to learn about his work. Interviews take place in the workplace so that the objects, artifacts and relationships of the work can be better understood. Examples of work are collected and both verbal and non-verbal communication is studied. The idea is to be as comprehensive in the data gathering as possible and to be concrete. Another central notion of contextual inquiry is that of partnership: the user is the expert in the workplace and is therefore encouraged to lead the investigation. the investigator is not a passive observer. Her objective is to gain a shared understanding of how the work happens and, to do so, she questions meaning and offers interpretations of what she observes. The aim is to draw out the implications of comments and actions and understand (rather than assume) what they really mean. In order to do this honestly and effectively the investigator must know her focus – her pre-existing beliefs and assumptions about the situation – and be prepared to challenge and adjust them in the face of new information.

**A number of models of the work are developed to capture what is important in the user’s work situation:**

The sequence model elaborates the steps required to complete a specific task, as well as the triggers that initiate that sequence of steps.

* + - The physical model maps the physical work environment and how it impacts upon work practice, for example, an office plan showing where different work activities happen.
    - The flow model shows the lines of coordination and communication between the user and other participants within and outside the workplace.
    - The cultural model reflects the influences of work culture and policy and shows the scope of these influences. This may include official or unofficial codes of behavior, common expectations (which may or may not be explicit) and value systems.
    - The artifact model describes the structure and use of a particular artifact within the work process.

**1.3 COMMUNICATION AND COLLABORATION MODELS**

All computer systems, single-user or multi-user, interact with the work-groups and organizations in which they are used.

• We need to understand normal human–human communication:

– face-to-face communication involves eyes, face and body – conversation can be analyzed to establish its detailed structure.

• This can then be applied to text-based conversation, which has:

* + - * reduced feedback for confirmation
      * less context to disambiguate utterances
      * slower pace of interaction but is more easily reviewed.

• Group working is more complex than that of a single person:

* + - * it is influenced by the physical environment – experiments are more difficult to control and record
      * field studies must take into account the social situation.

**3.3.1 FACE-TO-FACE COMMUNICATION**

Face-to-face contact is the most primitive form of communication – primitive, that is, in terms of technology.

**Transfer effects and personal space**

When we come to use computer-mediated forms of communication, we carry forward all our expectations and social norms from face-to-face communication. People are very adaptable and can learn new norms to go with new media. However, success with new media is often dependent on whether the participants can use their existing norms. The rules of face- to-face conversation are not conscious, so, when they are broken, we do not always recognize the true problem. We may just have feeling of unease, or we may feel that our colleague has been rude.

**Eye contact and gaze**

Normal conversation uses eye contact extensively, if not as intently. Our eyes tell us whether our colleague is listening or not; they can convey interest, confusion or boredom.

Sporadic direct eye contact (both looking at one another‗s eyes) is important in establishing a sense of engagement and social presence. People who look away when you look at them may seem shifty and appear to be hiding something. Furthermore, relative frequency of eye contact and who ‗gives way‗ from direct eye contact is closely linked to authority and power.

**Gestures and body language**

When the participants are in the same room, the existence of electronic equipment can interfere with the body language used in normal face-to-face communication. The fact that attention is focused on keyboard and screen can reduce the opportunities for eye contact.

Also, large monitors may block participants‗ views of one another‗s bodies, reducing their ability to interpret gestures and body position. Most computer-supported meeting rooms recess monitors into the desks to reduce these problems.

**Back channels, confirmation and interruption**

It is easy to think of conversation as a sequence of utterances: A says something, then B says something, then back to A. This process is called turn-taking and is one of the fundamental structures of conversation. However, each utterance is itself the result of intricate negotiation and interaction. Consider the following transcript:

**Alison:** Do you fancy that film . . . er . . . ‗The Green‗ um . . . it starts at eight. **Brian**: Grea

The nods, grimaces, shrugs of the shoulder and small noises are called back channels. They feed information back from the listener to the speaker at a level below the turn-taking of the conversation. The existence of back channels means that the speaker can afford to be slightly vague, adding details until it is obvious that the listener understands. Imagine making no response as someone talks to you, no little ‗yes‗es, no nods or raised eyebrows. You could answer questions and speak in turn, but not use back channels. It is likely that your colleague would soon become very uncomfortable, possibly rambling on with ever more detailed explanations, looking for some sign of understanding:

**Do you fancy that film . . . er . . . ‘The Green’ um . . . the one with Charles Dermot in**

**. . . you know with that song, er and the black cat on the poster . . . uhh**

**Turn-taking**

Starting to speak in the middle of someone‗s utterance can be rude, but one can say something like ‗well uh‗ accompanied by a slight raising of the hands and a general tensing of the body and screwing of the eyes. This tells the speaker that you would like to interrupt, allowing a graceful transition. In this case, the listener requested the floor. Turn-taking is the process by which the roles of speaker and listener are exchanged. Back channels are often a crucial part of this process.

The role of ‗um‗s and ‗ah‗s is very important. They can be used by either participant during the gap to claim the turn. So, if Brian wanted to respond in the middle of the utterance, but had not yet framed his utterance, he might begin ‗um the one . . .‗. As it was, Brian did not respond, so Alison starts ‗er‗ which says to Brian ‗I‗m going to continue, but I‗m thinking‗. Alternatively, Alison could have started to ‗er‗ as soon as she had said the word

‗film‗. This would have told Brian not to interrupt. These turn-offering gaps are just the places where the speaker expects some back channel response even if no turn exchange takes place. A total lack of response will be taken, depending on the circumstances, as assent to the speaker, or perhaps as lack of understanding.

**3.3.2 CONVERSATION**

Conversational analyses are sociological and psychological understandings of conversation.

**Basic conversational structure**

Imagine we have a transcript of a conversation, recalling from that the production of such a transcript is not a simple task. For example, a slightly different version of Alison and Brian‗s conversation may look like this: **Alison:** Do you fancy that film?

**Brian:** The uh (500 ms) with the black cat – ‗The Green whatsit‗?

**Alison:** Yeah, go at uh . . . (looks at watch – 1.2 s) . . . 20 to?

This transcript is quite heavily annotated with the lengths of pauses and even Alison‗s action of looking at her watch. it certainly lacks the wealth of gesture and back channel activity that were present during the actual conversation.

Transcripts may be less well documented, perhaps dropping the pause timings, or more detailed, adding more actions, where people were looking and some back channelling. Whilst thinking about the structure of conversation, the transcript above is sufficient.

The most basic conversational structure is turntaking. On the whole we have an alternating pattern: Alison says something, then Brian, then Alison again. The speech within each turn is called an utterance. There can be exceptions to this turn-taking structure even within two- party conversation.

**Context**

Take a single utterance from a conversation, and it will usually be highly ambiguous if not meaningless: ‗the uh with the black cat – ―The Green whatsit‖‗. Each utterance and each fragment of conversation is heavily dependent on context, which must be used to disambiguate the utterance. We can identify two types of context within conversation: **internal context** – dependence on earlier utterances. For example, when Brian says ‗masses‗ in the last transcript, this is meaningful in the light of Alison‗s question ‗and lots of chocolate?‗. This in turn is interpreted in the context of Brian‗s original offer of gateau. **external context –** dependence on the environment. For example, if Brian had said simply

‗do you want one?‗, this could have meant a slice of gateau, or, if he had been holding a bottle, a glass of wine, or, if accompanied by a clenched fist, a punch on the nose.

**Topics, focus and forms of utterance**

Alison began the conversation with the topic of roses. Brian shifts to the related, but distinct, topic of greenfly. However, for some reason Alison has missed this shift in focus, so when she makes her second utterance, her focus and Brian‗s differ, leading to the breakdown in communication. The last two utterances are a recovery which re-establishes a shared dialog focus.

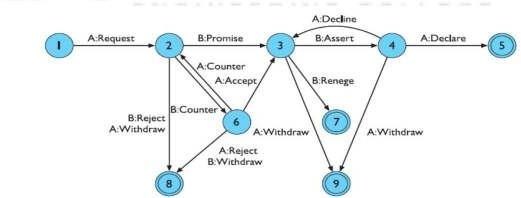
**Breakdown and repair**

When Alison and Brian were talking about Brian‗s roses, they failed to maintain a shared focus. Brian tried to interpret Alison‗s utterance in terms of his focus and failed, or rather the meaning in that focus was unusual – greenfly are the symbol of the English summer? He then questioned Alison and the confusion was cleared. This correction after breakdown is called repair.

**Speech act theory**

Speech act theory, has been both influential and controversial in CSCW. Not only is it an analytic technique, but it has been used as the guiding force behind the design of a commercial system, The basic premise of speech act theory is that utterances can be characterized by what they do. If you say ‗I‗m hungry‗, this has a certain propositional meaning – that you are feeling hungry. However, depending on who is talking and to whom, this may also carry the meaning ‗get me some food‗ – the intent of the statement is to evoke an action on the part of the hearer. Speech act theory concerns itself with the way utterances interact with the actions of the participants. The act of saying the words changes the state of the couple. Other acts include promises by the speaker to do something and requests that the hearer do something. These basic acts are called illocutionary points.

Individual speech acts can contribute to a conversation. The basic structure of conversations can then be seen as instances of generic conversations. One example of such a generic structure is a conversation for action (CfA).



**3.3.3 TEXT BASED COMMUNICATION**

Text-based communication is familiar to most people, in that they will have written and received letters. However, the style of letter writing and that of face-to face communication are very different. The text-based communication in groupware systems is acting as a speech substitute, and, thus, there are some problems adapting between the two media.

There are four types of textual communication in current groupware:

**discrete** – directed message as in email. There is no explicit connection between different messages, except in so far as the text of the message refers to a previous one.

**linear** – participants‗ messages are added in (usually temporal) order to the end of a single transcript. **non-linear** – when messages are linked to one another in a hypertext fashion. **spatial** – where messages are arranged on a two-dimensional surface.

**Back channels and affective state**

One of the most profound differences between face-to-face and text-based communication is the lack of fine-grained channels. Much of the coordination of face-toface conversation depends on back channels and interpretation of the listener‗s expressions. Text- based communication loses these back channels completely. speaker would pause to seek back channel confirmation or to offer the floor, the text ‗speaker‗ must either continue regardless, or finish the message, effectively passing the turn.

These normally convey the affective state of the speaker (happy, sad, angry, humorous) and the illocutionary force of the message (an important and urgent demand or a deferential request). Email users have developed explicit tokens of their affective state by the use of ‗flaming‗ and ‗smilies‗, using punctuation and acronyms; for example:

:-) – smiling face, happy

:-( – sad face, upset or angry ;-) – winking face, humorous LOL – laughing out loud.

**Grounding constraints**

This grounding process is linked strongly with the types of channels through which the conversants communicate. Clark and Brennan describe the properties of these channels in terms of grounding constraints. These include:

**Co-temporality** – an utterance is heard as soon as it is said (or typed); **simultaneity** – the participants can send and receive at the same time; **sequence** – the utterances are ordered.

These are all constraints which are weaker in text-based compared with face-to-face interaction.

In a text-based system, different participants can compose simultaneously, but they lack cotemporality. As we saw, even if the messages appear as they are produced, they will not be read in real time. In addition, the messages may only be delivered when complete and even then may be delayed by slow communications networks.

**Turn-taking**

In a pair of participants, turn-taking is simple; first one person says something, then the other. The only problem is deciding exactly when the exchange should happen. With three or more participants, turn-taking is more complex. They must decide who should have the next turn. This is resolved by face-to-face groups in a number of ways. First, the conversation may, for a period, be focused on two of the parties, in which case normal two- party turn-taking holds. Secondly, the speaker may specifically address another participant as the utterance is finished, either implicitly by body position, or explicitly: ‗what do you think Alison?‗ Finally, the next speaker may be left open, but the cotemporality of the audio channel allows the other participants to negotiate the turn. Basically, whoever speaks first, or most strongly, gets in. These mechanisms are aided by back channels, as one of the listeners may make it clear that she wants to speak. In this case, either the speaker will explicitly pass the turn (the second option above), or at least the other listeners are expecting her to speak. In addition, the movement between effective two-party conversation (the first option) and open discussion will be mediated by back channel messages from the other participants.

In an unstructured text-based conversation the third option is not available, nor, of course, are the back channels. Paired conversation is quite common and the second option, explicitly naming the next speaker, is possible. This naming is not particularly natural unless a direct question is being asked. In both options, the absence of back channels makes it difficult for another listener to interrupt the conversation. Some systems use more structured mechanisms to get round these problems, perhaps having a round-robin protocol (each participant ‗speaks‗ in turn) or having a queue of turnrequests. Whether the strictures of such mechanisms are worse than the problems of occasional breakdown depends very much on the context and is a matter of opinion.

**Context and deixis**

Utterances are highly ambiguous and are only meaningful with respect to external context, the state of the world, and internal context, the state of the conversation. Both of these are problems in text-based communication.

The very fact that the participants are not co-present makes it more difficult to use external context to disambiguate utterances. This is why many groupware systems strive so hard to make the participants‗ views the same; that is, to maintain WYSIWIS (‗what you see is what I see‗). Whatever the means of direct communication, remote participants have difficulty in using deictic reference. They cannot simply say ‗that one‗, but must usually describe the referrant: ‗the big circle in the corner‗. If their displays are not WYSIWIS then they must also ensure that their colleague‗s display includes the object referred to and that the description is unambiguous. Asynchronous participants have even more problems with deixis as there is no opportunity for their colleagues to clarify a reference (without extremely lengthy exchanges). The objects referred to by a message may have changed by the time someone comes to read it! Similarly, group pointers are not really an option, but one can use methods of linking the conversation to its context, either by embedding it within the objects as annotations or by having hypertext links between the conversation and the object. The trouble does not end with external context; there are also problems with deictic reference to internal context. In speech, the context is intimately connected to linear sequence and adjacency. As we have seen, even in linear text transcripts, overlap breaks the strict sequentiality of the conversation, and thus causes problems with indexicals and with context in general.

1. Alison: Brian‗s got some lovely roses.
2. Brian: I‗m afraid they‗re covered in greenfly.
3. Clarise: I‗ve seen them, they‗re beautiful.

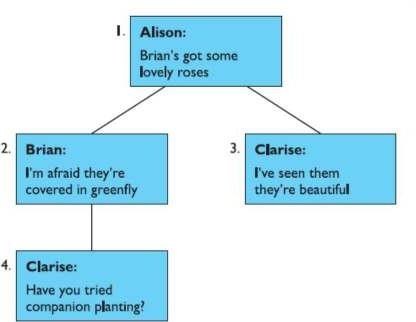


Fig: Hypertext conversation structure

**Pace and granularity**

The term pace is being used in a precise sense above. Imagine a message being composed and sent, the recipient reading (or hearing) the message and then composing and sending a reply. The pace of the conversation is the rate of such a sequence of connected messages and replies. Clearly, as the pace of a conversation reduces, there is a tendency for the granularity to increase. To get the same information across, you must send more per message. However, it is not as easy as that. We have seen the importance of feedback from listener to speaker in clarifying meaning and negotiating common ground. Even most monologs are interactive in the sense that the speaker is constantly looking for cues of comprehension in the listener. Reducing the pace of a conversation reduces its interactivity.

In a hypertext-based system one can expand several branches of a conversation tree, but in speech or in a linear text transcript the conversation follows one branch. To overcome these limitations, people adopt several coping strategies. The simplest strategy is just to avoid conversation. This can be done by delegating parts of a task to the different participants. Each participant can then perform much of the task without communication. They must still communicate for large-scale strategic decisions, but have significantly reduced the normal communications. Of course, this approach reduces communication by reducing collaboration. More interesting in a cooperative work setting are two coping strategies which increase the chunk size of messages in order to reduce the number of interactions required to complete a task. These strategies are frequently seen in both text-based conferences and in letter writing.

The first of these coping strategies is multiplexing. Basically, the conversants hold several conversations in parallel, each message referring to several topics. In terms of the conversation tree, this corresponds to going down several branches at once.

Linear text vs. Hypertext

Multiplexed messages can be represented as updates to several parts of the hypertext, thus reducing the likelihood of breakdown and lost topics. In addition, if the messages themselves can be mini-hypertexts, then eager messages listing several possible courses of action can be explicitly represented by the message.

Even static hypertexts, which have been carefully crafted by their authors, can be difficult to navigate. A hypertext that is created ‗on the fly‗ is unlikely to be comprehensible to any but those involved in its creation. Conklin and Begeman, themselves associated with the hypertext based argumentation tool gIBIS, conclude that ‗traditional linear text provides a continuous, unwinding thread of context.

For the asynchronous reader trying to catch up with a conversation, a linear transcript is clearly easier, but it is precisely in more asynchronous settings where overlap in linear text is most likely to cause confusion.

**3.3.4 GROUP WORKING**

Group behavior is more complex still as we have to take into account the dynamic social relationships during group working. We will begin by looking at several factors which affect group working, and then discuss the problems of studying group working .

**Group dynamics**

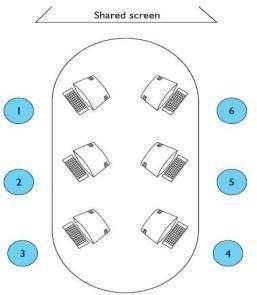
organizational relationships such as supervisor/supervisee are relatively stable, the roles and relationships within a group may change dramatically within the lifetime of a task and even within a single work session. For example, studies of joint authoring have found that roles such as author, co-author and commentator change throughout the lifetime of a document. This means that systems, such as co-authoring systems, which use a formal concept of role, must allow these roles to change together with the socially defined roles.

A person may be an author of a book or paper, but never write the words in it, acting instead as a source of ideas and comments. A particular case of this is the biographical story where the individual concerned and a professional writer co-author the book, but only the professional author writes. A co-authoring system such as Quilt would call the non-writing author a

‗commentator‗ or a ‗reviewer‗, but not an ‗author‗. One can imagine some of the social friction such naming will cause.

**Physical layout**

The designers of Capture Lab, an eight-person meeting room, considered all these features and many other subtle effects. However, the users still had some difficulty in adapting to the power positions in the electronic meeting room. At first sight, the electronic meeting room is not unlike a normal conference room. If the shared screen is a whiteboard or an overhead projector, then the most powerful position is toward the front of the room . Managers would normally take this seat as they can then easily move to the whiteboard or overhead projector to point out some item and draw the group‗s attention.



**Distributed cognition**

Traditional views talk about the movement of information between working memory and long-term memory: it is not so difficult then to regard bits of paper, books and computer systems as extensions to these internal memory systems. Similarly, many models of human cognition regard the mind as a set of interacting subsystems. The step to regarding several people as involved in joint thinking is not difficult.

**1.4 HYPERTEXT, MULTIMEDIA AND THE WORLD WIDE WEB**

* Hypertext allows documents to be linked in a nonlinear fashion.
* Multimedia incorporates different media: sound, images, and video.
* The World Wide Web is a global hypermedia system.
* Animation and video can show information that is difficult to convey statically.
* Applications of hypermedia include online help, education and e-commerce.
* Design for the World Wide Web illustrates general hypermedia design, but also has its own special problems.
* Dynamic web content can be used for simple online demonstration

**3.4.1 Hypertext.**

* The term hypertext means certain extra capabilities imparted to normal or standard text.
* Technical documentation consists often of a collection of independent information units.
* It consists of cross references which lead to multiple searches at different places for the reader.
* Hypertext is text which is not constrained to be linear and it contains links to other texts which is known as hyperlinks.
* Hypertext is mostly used on World Wide Web for linking and navigating through different web pages.
* A hypertext consists of two different parts: Anchor and link
* An anchor or node is an entry point to another document. In some cases instead of a text an image a video or some other non-textual element.
* A link or pointer provide connection to other information unit known as target documents. **3.4.2 Multimedia** refers to using computers to integrate text, graphics, animation, audio, and video into one application. Most multimedia applications are interactive, so that users may choose the material to view, define the order in which it is presented, and obtain feedback on their actions.

Interactivity also makes multimedia very suitable for video games, electronic newspapers and magazines, electronic books and references, simulations, virtual reality, and computer-based training. Multimedia applications can be created by using a multimedia authoring software.

Many multimedia applications are also deliverable via the World Wide Web.

**Graphics**

A graphic is a digital representation of information such as a drawing, a chart, or a photograph.Graphics were the first media used to enhance the originally text-based Internet. Two of the more common graphical formats on the Web are JPEG and GIF. Other graphical formats such as BMP and TIFF have larger file sizes, and may require special viewer software to display on the Web. To reduce download times for graphics, some Web sites use thumbnails, which is a smaller version of a larger graphical image that a user may click to display the fullsized image.

**Audio**

Audio can be music, speech, or any other sound. Common audio formats include WAV, MID, and MP3. Some Web sites use streaming audio, which allows a user to listen to the sound as it downloads to the computer. Two accepted standards for streaming audio on the Web are Windows Media Player and RealAudio.

**Video**

Video consists of full-motion images that are played back at various speed. Most video is also accompanied with audio. MPEG is a popular video compression standard defined by the Moving Picture Experts Group (MPEG). Streaming video allows a user to view longer or live video images as they download to the computer from the Web. Two popular streaming video formats are Windows Media Player and RealVideo.

**Animation** is the appearance of motion that is created by displaying a series of still images in rapid sequence. Animated GIF is a popular type of animation format, which combines several images into a single GIF file.

**Multimedia Authoring Software**

Multimedia authoring software combines text, graphics, animation, audio, and video into an application. Multimedia is widely used in video games, electronic newspapers and magazines, electronic books and references, simulations, virtual reality, and computer-based training. Popular multimedia authoring software includes Macromedia AuthorWare, Macromedia Director, and Macromedia Flash. Multimedia computers have facilities for handling sound and video as well as text and graphics. Most computers are now sold with a multimedia capacity.

**3.4.3 Web - World Wide Web**

The Web, or World Wide Web, is basically a system of Internet servers that support specially formatted documents. The documents are formatted in a markup language called HTML (HyperText Markup Language) that supports links to other documents, as well as graphics, audio, and video files.

This means you can jump from one document to another simply by clicking on hot spots. Not all Internet servers are part of the World Wide Web.

The Internet is a worldwide collection of networks that links millions of businesses, government offices, educational institutions, and individuals. Data is transferred over the Internet using servers, which are computers that manage network resources and provide centralized storage areas, and clients, which are computers that can access the contents of the storage areas. The data travels over communications lines. Each computer or device on a communications line has a numeric address called an IP (Internet protocol) address, the text version of which is called a domain name. Every time you specify a domain name, a DNS (domain name system) server translates the domain name into its associated IP address, so data can route to the correct computer.

An Internet service provider (ISP) provides temporary Internet connections to individuals and companies. An online service provider (OSP) also supplies Internet access, in addition to a variety of special content and services. A wireless service provider (WSP) provides wireless Internet access to users with wireless modems or Web-enabled handheld computers or devices.

Employees and students often connect to the Internet through a business or school network that connects to a service provider. For home or small business users, dial-up access provides an easy and inexpensive way to connect to the Internet. With dial-up access, you use a computer, a modem, and a regular telephone line to dial into an ISP or OSP. Some home and small business users opt for newer, high-speed technologies. DSL (digital subscriber line) provides high-speed connections over a regular copper telephone line. A cable modem provides high-speed Internet connections through a cable television network.

The World Wide Web (WWW or Web) consists of a worldwide collection of electronic documents called Web pages. A browser is a software program used to access and view Web pages. Each Web page has a unique address, called a URL (Uniform Resource Locator), that tells a browser where to locate the Web page. A URL consists of a protocol, domain name, and sometimes the path to a specific Web page or location on a Web page. Most URLs begin with http://, which stands for hypertext transfer protocol, the communications standard that enables pages to transfer on the Web.

A search engine is a software program you can use to find Web sites, Web pages, and Internet files. To find a Web page or pages, you enter a relevant word or phrase, called search text or keywords, in the search engine‗s text box. Many search engines then use a program called a spider to read pages on Web sites and create a list of pages that contain the keywords. Any Web page that is listed as the result of the search is called a hit. Each hit is a link that can be clicked to display the associated Web site or Web page.

There are six basic types of Web pages. An advocacy Web page contains content that describes a cause, opinion, or idea. A business/marketing Web page contains content that promotes or sells products or services. An informational Web page contains factual information. A news Web page contains newsworthy material including stories and articles relating to current events, life, money, sports, and the weather. A portal Web page offers a variety of Internet services from a single, convenient location. A personal Web page is maintained by a private individual who normally is not associated with any organization.

Many exciting Web pages use multimedia. Multimedia refers to any application that integrates text with one of the following elements: graphics, sound, video, virtual reality, or other media elements.

A graphic is a digital representation of information such as a drawing, chart, or photograph. Two common file formats for graphical images on the Web are JPEG (JointPhotographic Experts Group) and GIF (Graphics Interchange Format), which use compression techniques to reduce the size of graphics files and thus speed downloading.

Animation is the appearance of motion created by displaying a series of still images in rapid sequence. One popular type of animation, called an animated GIF, uses computer animation and graphics software to combine several images into a single GIF file.

Audio is music, speech, or any other sound. A common format for audio files on the Web is MP3, a popular technology that compresses audio. More advanced Web audio applications use streaming audio, which transfers audio data in a continuous and even flow, allowing users to listen to the sound as it downloads. Video consists of full-motion images that are played back at various speeds. Video files often are quite large in size. The Moving Pictures Experts Group (MPEG) defines a popular video compression standard. Streaming video allows you to view longer or live video images as they are downloaded.

Virtual reality (VR) is the use of computers to simulate a real or imagined environment that appears as a three-dimensional (3-D) space. A VR world is an entire 3-D site that contains infinite space and depth.

A variety of services are used widely on the Internet, including e-mail, FTP, newsgroups and message boards, mailing lists, chat rooms, and instant messaging. E-mail (electronic mail) is the transmission of messages and files via a computer network. You use an e-mail program to create, send, receive, forward, store, print, and delete messages. To receive messages, you need an e-mail address, which is a combination of a username and a domain name that identifies a user.

FTP (File Transfer Protocol) is an Internet standard that allows you to upload and download files with other computers on the Internet. An FTP server is a computer that allows you to use FTP to upload files to, and download files from, an FTP site. With anonymous FTP, anyone can transfer some, if not all, available files. A newsgroup is an online area in which users conduct written discussions about a particular subject. The computer that stores and distributes newsgroup messages is called a news server. You use a program called a newsreader to access a newsgroup, read previously entered messages (called articles), and add (post) messages of your own.

A thread consists of the original article and all subsequent related replies. In a moderated newsgroup, a moderator reviews articles and posts them, if appropriate. A message board is a popular Web-based type of discussion group that does not require a newsreader and typically is easier to use than a newsgroup. A mailing list is a group of e-mail names and addresses given a single name. To add your e-mail name and address to a mailing list you subscribe to it; to remove your name, you unsubscribe.

A chat is real-time (meaning everyone involved in the chat is online at the same time) typed conversation that takes place on a computer. A location on an Internet server that permits users to chat is called a chat room. Some chat rooms support voice chats and video chats, where you can hear or see others and they can hear or see you as you chat. A chat client is a program on your computer that allows you to connect to a chat server and start a chat session. Instant messaging (IM) is a real-time Internet communications service that notifies you when one or more people are online and then allows you to exchange messages or join a private chat room.

References

1. Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, “Human Computer Interaction”,

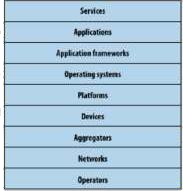
3rd Edition, Pearson Education, 2004

**UNIT 4 MOBILE HCI**

Mobile Ecosystem: Platforms, Application frameworks - Types of Mobile Applications: Widgets, Applications, Games - Mobile Information Architecture, Mobile 2.0, Mobile Design: Elements of Mobile Design, Tools.

**4.1 The Mobile Ecosystem**

Mobile is an entirely unique ecosystem and, like the Internet, it is made up of many different parts that must all work seamlessly together. With mobile technology, the parts are different, and because you can use mobile devices to access the Internet, that means that not only do you need to understand the facets of the Internet, but you also need to understand the mobile ecosystem.



**4.1.1 Operators**

The base layer in the mobile ecosystem is the operator. Operators go by many names, depending on what part of the world you happen to be in or who you are talking to. Operators can be referred to as Mobile Network Operators (MNOs); mobile service providers, wireless carriers, or simply carriers; mobile phone operators; or cellular companies. In the mobile community, we officially refer to them as operators, though in the United States, there is a tendency to call them Carriers. Operators are what essentially make the entire mobile ecosystem work. They are the gatekeepers to the kingdom. They install cellular towers, operate the cellular network, make services (such as the Internet) available for mobile subscribers, and they often maintain relationships with the subscribers, handling billing and support, and offering subsidized device sales and a network of retail stores. The operator‘s role in the ecosystem is to create and maintain a specific set of wireless services over a reliable cellular network. That‘s it. However, to grow the mobile market over the past decade, the operator has been required to take a greater role in the mobile ecosystem, doing much more than just managing the network. For example, they have had to establish trust with subscribers to handle the billing relationship and to offer.

**4.1.2 Networks**

Operators operate wireless networks. Remember that cellular technology is just a radio that receives a signal from an antenna. The type of radio and antenna determines the capability of the network and the services you can enable on it. You‘ll notice that the vast majority of networks around the world use the GSM standard, using GPRS or GPRS EDGE for 2G data and UMTS or HSDPA for 3G. We also have CDMA (Code Division Multiple Access) and its 2.5G hybrid CDMA2000, which offers greater coverage than its more widely adopted rival. So in places like the United States or China, where people are more spread out, CDMA is a great technology. It uses fewer towers, giving subscribers fewer options as they roam networks.

**4.1.3 Devices**

What you call phones, the mobile industry calls handsets or terminals. These are terms that I think are becoming outdated with the emergence of wireless devices that rely on operator networks, but do not make phone calls. The number of these ―other‖ devices is a small piece of the overall pie right now, but it‘s growing rapidly. Let‘s focus on the biggest slice of the device pie—mobile phones. As of 2008, there are about 3.6 billion mobile phones currently in use around the world; just more than half the planet‘s population has a mobile phone Most of these devices are feature phones, making up the majority of the marketplace. Smartphones make up a small sliver of worldwide market share and maintain a healthy percentage in the United States and the European Union; smartphone market share is growing with the introduction of the iPhone and devices based on the Android platform. As next- generation devices become a reality, the distinction between feature phones and smartphones will go away. In the next few years, feature phones will largely be located in emerging and developing markets.

**4.2 PLATFORMS**

A mobile platform‗s primary duty is to provide access to the devices. To run software and services on each of these devices, you need a platform, or a core programming language in which all of your software is written. Like all software platforms, these are split into three categories: licensed, proprietary, and open source.

**4.2.1 Licensed**

Licensed platforms are sold to device makers for nonexclusive distribution on devices. The goal is to create a common platform of development Application Programming Interfaces (APIs) that work similarly across multiple devices with the least possible effort required to adapt for device differences, although this is hardly reality. Following are the licensed platforms:

**Java Micro Edition (Java ME)**

Java ME is by far the most predominant software platform of any kind in the mobile ecosystem. It is a licensed subset of the Java platform and provides a collection of Java APIs for the development of software for resource constrained devices such as phones.

**Binary Runtime Environment for Wireless (BREW)**

BREW is a licensed platform created by Qualcomm for mobile devices, mostly for the U.S. market. It is an interface-independent platform that runs a variety of application frameworks, such as C/C++, Java, and Flash Lite.

**Windows Mobile**

Windows Mobile is a licensable and compact version of the Windows operating system, combined with a suite of basic applications for mobile devices that is based on the Microsoft Win32 API.

**LiMo**

LiMo is a Linux-based mobile platform created by the LiMo Foundation. Although Linux is open source, LiMo is a licensed mobile platform used for mobile devices. LiMo includes SDKs for creating Java, native, or mobile web applications using the WebKit browser framework.

**4.2.2 Proprietary**

Proprietary platforms are designed and developed by device makers for use on their devices. They are not available for use by competing device makers. These include: Palm **Palm** uses three different proprietary platforms. Their first and most recognizable is the Palm OS platform based on the C/C++ programming language; this was initially developed for their Palm Pilot line, but is now used in low-end smartphones such as the Centro line. As Palm moved into higher-end smartphones, they started using the Windows Mobile-based platform for devices like the Treo line. The most recent platform is called webOS, is based on the WebKit browser framework, and is used in the Prē line.

**BlackBerry**

Research in Motion maintains their own proprietary Java-based platform, used exclusively by their BlackBerry devices.

**iPhone**

Apple uses a proprietary version of Mac OS X as a platform for their iPhone and iPod touch line of devices, which is based on Unix.

**4.2.3 Open Source**

Open source platforms are mobile platforms that are freely available for users to download, alter, and edit. Open source mobile platforms are newer and slightly controversial, but they are increasingly gaining traction with device makers and developers. Android is one of these platforms. It is developed by the Open Handset Alliance, which is spearheaded by Google. The Alliance seeks to develop an open source mobile platform based on the Java programming language.

**4.3 APPLICATION FRAMEWORKS**

Application frameworks often run on top of operating systems, sharing core services such as communications, messaging, graphics, location, security, authentication, and many others.

**Java**

Applications written in the Java ME framework can often be deployed across the majority of Java-based devices, but given the diversity of device screen size and processor power, cross-device deployment can be a challenge.

**S60**

The S60 platform, formerly known as Series 60, is the application platform for devices that run the Symbian OS. S60 is often associated with Nokia devices—Nokia owns the platform—but it also runs on several non-Nokia devices. S60 is an open source framework.

S60 applications can be created in Java, the Symbian C++ framework, or even Flash Lite.

**BREW**

Applications written in the BREW application framework can be deployed across the majority of BREW-based devices, with slightly less cross-device adaption than other frameworks. **Flash Lite**

Adobe Flash Lite is an application framework that uses the Flash Lite and ActionScript frameworks to create vector-based applications. Flash Lite applications can be run within the Flash Lite Player, which is available in a handful of devices around the world.

Flash Lite is a promising and powerful platform, but there has been some difficulty getting it on devices. A distribution service for applications written in Flash Lite is long overdue.

**Windows Mobile**

Applications written using the Win32 API can be deployed across the majority of Windows Mobile-based devices. Like Java, Windows Mobile applications can be downloaded and installed over the air or loaded via a cable-connected computer.

**Cocoa Touch**

Cocoa Touch is the API used to create native applications for the iPhone and iPod touch. Cocoa Touch applications must be submitted and certified by Apple before being included in the App Store. Once in the App Store, applications can be purchased, downloaded, and installed over the air or via a cable-connected computer.

**Android SDK**

The Android SDK allows developers to create native applications for any device that runs the Android platform. By using the Android SDK, developers can write applications in C/C++ or use a Java virtual machine included in the OS that allows the creation of applications with Java, which is more common in the mobile ecosystem.

**Web Runtimes (WRTs)**

Nokia, Opera, and Yahoo! provide various Web Runtimes, or WRTs. These are meant to be miniframeworks, based on web standards, to create mobile widgets. Both

Opera‗s and

Nokia‗s WRTs meet the W3C-recommended specifications for mobile widgets.

**WebKit**

WebKit is a browser technology, so applications can be created simply by using web technologies such as HTML, CSS, and JavaScript. WebKit also supports a number of recommended standards not yet implemented in many desktop browsers. Applications can be run and tested in any WebKit browser, desktop, or mobile device.

**The Web**

The Web is the only application framework that works across virtually all devices and all platforms. Although innovation and usage of the Web as an application framework in mobile has been lacking for many years, increased demand to offer products and services outside of operator control, together with a desire to support more devices in shorter development cycles, has made the Web one of the most rapidly growing mobile application platforms to date.

**4.4 Types of Mobile Applications**

**4.4.1 Mobile Application Medium Types**

The mobile medium type is the type of application framework or mobile technology that presents content or information to the user. It is a technical approach regarding which type of medium to use; this decision is determined by the impact it will have on the user experience.

**SMS**

The most basic mobile application you can create is an SMS application. Although it might seem odd to consider text messages applications, they are nonetheless a designed experience. Given the ubiquity of devices that support SMS, these applications can be useful tools when integrated with other mobile application types. Typically, the user sends a single keyword to a five-digit short code in order to return information or a link to premium content. For example, sending the keyword ―freebie‖ to a hypothetical short code ―12345‖ might return a text message with a coupon code that could be redeemed at a retail location, or it could include a link to a free ringtone. SMS applications can be both ―free,‖ meaning that there is no additional charge beyond the text message fees an operator charges, and

―premium,‖ meaning that you are charged an additional fee in exchange for access to premium content. The most common uses of SMS applications are mobile content, such ringtones and images, and to interact with actual goods and services. Some vending machines can dispense beverages when you send them an SMS; SMS messages can also be used to purchase time at a parking meter or pay lot. A great example of how SMS adds incredible value would be Twitter, where users can receive SMS alerts from their friends and post to their timeline from any mobile device.

**4.4.2 Mobile websites**

Mobile website is a website designed specifically for mobile devices, not to be confused with viewing a site made for desktop browsers on a mobile browser. Mobile websites are characterized by their simple ―drill-down‖ architecture, or the simple presentation of navigation links that take you to a page a level deeper.



Mobile websites often have a simple design and are typically informational in nature, offering few—if any—of the interactive elements you might expect from a desktop site. Mobile websites have made up the majority of what we consider the mobile web for the past decade, starting with the early WML-based sites (not much more than a list of links) and moving to today‘s websites, with a richer experience that more closely resembles the visual aesthetic users have come to expect with web content. Though mobile websites are fairly easy to create, they fail to display consistently across multiple mobile browsers—a trait common to all mobile web mediums. The mobile web has been gradually increasing in usage over the years in most major markets, but the limited experience offered little incentive to the user. Many compare the mobile web to a 10-year-old version of the Web: slow, expensive to use, and not much to look at. As better mobile browsers started being introduced to device platforms like the iPhone and Android, the quality of mobile websites began to improve dramatically, and with it, usage improved. For example, in just one year, the U.S. market went from being just barely in the top five consumers of the mobile web to number one, largely due to the impact of the iPhone alone. Pros

* They are easy to create, maintain, and publish.
* They can use all the same tools and techniques you might already use for desktop sites.
* Nearly all mobile devices can view mobile websites.

Cons

The cons of mobile websites are:

* They can be difficult to support across multiple devices.
* They offer users a limited experience.
* Most mobile websites are simply desktop content reformatted for mobile devices.

**4.4.3 Mobile Web Widgets**

Largely in response to the poor experience provided by the mobile web over the years, there has been a growing movement to establish mobile widget frameworks and platforms. For years the mobile web user experience was severely underutilized and failed to gain traction in the market, so several operators, device makers, and publishers began



creating widget platforms (Figure) to counter the mobile web‗s weaknesses.

Figure: An example mobile web widget

I initially saw mobile web widgets as another attempt by the mobile industry to hype a technology that no one wants. I liked to quiz mobile web widget advocates about what makes mobile web widgets different than what we can do with the mobile web. A component of a user interface that operates in a particular way. The ever-trusty Wikipedia defines a web widget this way:

A portable chunk of code that can be installed and executed within any separate HTML based web page by an end user without requiring additional compilation.

Between these two definitions is a better answer:

A mobile web widget is a standalone chunk of HTML-based code that is executed by the end user in a particular way. Mobile web widgets are small web applications that can‗t run by themselves; they need to be executed on top of something else. I think one reason for all the confusion around what is a mobile web widget is that this definition can also encompass any web application that runs in a browser. Opera Widgets, Nokia Web Run Time (WRT), Yahoo! Blueprint, and Adobe Flash Lite are all examples of widget platforms that work on a number of mobile handsets. Using a basic knowledge of HTML (or vector graphics in the case of Flash), you can create compelling user experiences that tap into device features and, in many cases, can run while the device is offline.

**Pros**

The pros of mobile web widgets are:

* They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
* They can be simple to deploy across multiple handsets.
* They offer an improved user experience and a richer design, tapping into device features and offline use.

**Cons**

The cons of mobile web widgets are:

* They typically require a compatible widget platform to be installed on the device.
* They cannot run in any mobile web browser.
* They require learning additional proprietary, non-web-standard techniques.

**4.4.4 Mobile Web Applications**

Mobile web applications are mobile applications that do not need to be installed or compiled on the target device. Using XHTML, CSS, and JavaScript, they are able to provide an application-like experience to the end user while running in any mobile web browser. By Application -like experience, I mean that they do not use the drill-down or page metaphors in which a click equals a refresh of the content in view. Web applications allow users to interact with content in real time, where a click or touch performs an action within the current view.

The history of how mobile web applications came to be so commonplace is interesting, and is one that I think can give us an understanding of how future mobile trends can be assessed and understood. Shortly after the explosion of Web 2.0, web applications like Facebook, Flickr, and Google Reader hit desktop browsers, and there was discussion of how to bring those same web applications to mobile devices. The Web 2.0 movement brought user-centered design principles to the desktop web, and those same principles were sorely needed in the mobile web space as well.

The challenge, as always, was device fragmentation. The mobile browsers were years behind the desktop browsers, making it nearly impossible for a mobile device to render a comparable experience. While XHTML support had become fairly commonplace across devices, the rendering of CSS2 was wildly inconsistent, and support for Java- Script, necessary or simple DHTML, and Ajax was completely nonexistent. To make matters worse, the perceived market demand for mobile web applications was not seen as a priority with many operators and device makers. It was the classic chickenor- the-egg scenario. What had to come first, market demand to drive browser innovation or optimized content to drive the market? With the introduction of the first iPhone, we saw a cataclysmic change across the board.

Using WebKit, the iPhone could render web applications not optimized for mobile



devices as perfectly usable, including DHTML- and Ajax-powered content. Developers quickly got on board, creating mobile web applications optimized mostly for the iPhone (Figure). The combination of a high-profile device with an incredibly powerful mobile web browser and a quickly increasing catalog of nicely optimized experiences created the perfect storm the community had been waiting for.

Usage of the mobile web exploded with not just users of the iPhone, but users of other handsets, too. Because Web applications being created for the iPhone were based on web standards, they actually worked reasonably well on other devices. Operators and device makers saw that consumers wanted not just the mobile web on their handsets, but the regular Web, too.

**Pros:**

The pros of mobile web applications are:

* They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
* They are simple to deploy across multiple handsets.
* They offer a better user experience and a rich design, tapping into device features & offline use.
* Content is accessible on any mobile web browser.

**Cons:**

The cons of mobile web applications are:

* The optimal experience might not be available on all handsets.
* They can be challenging (but not impossible) to support across multiple devices.
* They don‗t always support native application features, like offline mode, location lookup, file system access, camera, and so on.

**4.4.5 Games**

The most popular of all media available to mobile devices. Technically games are really just native applications that use the similar platform SDKs to create immersive experiences (Figure). But I treat them differently from native applications for two reasons: they cannot be easily duplicated with web technologies, and porting them to multiple mobile platforms is a bit easier than typical platform-based applications.



Seeing as how we have yet to see these types of gaming experiences appear on the desktop using standard web technologies, I believe we are still a few years out from seeing them on mobile devices. Adobe‗s Flash and the SVG (scalable vector graphics) standard are the only way to do it on the Web now, and will likely be how it is done on mobile devices in the future, the primary obstacle being the performance of the device in dealing with vector graphics. The reason games are relatively easy to port (―relatively‖ being the key word), is that the bulk of the gaming experience is in the graphics and actually uses very little of the device APIs. The game mechanics are the only thing that needs to adapted to the various platforms. Like in console gaming, there are a great number of mobile game porting shops that can quickly take a game written in one language and port it to another.

These differences, in my mind, are what make mobile games stand apart from all other application genres—their capability to be unique and difficult to duplicate in another application type, though the game itself is relatively easy to port. Looking at this model for other application areas—namely, the mobile web—could provide helpful insight into how we create the future of mobile web applications.

**Pros:** The pros of game applications are:

* They provide a simple and easy way to create an immersive experience.
* They can be ported to multiple devices relatively easily.

**Cons:** The cons of game applications are:

* They can be costly to develop as an original game title.
* They cannot easily be ported to the mobile web.

**4.5 MOBILE INFORMATION ARCHITECTURE**

What Is Information Architecture?

The structural design of shared information environments

* The combination of organizations, labelling, search, and navigation systems within websites and intranets
* The art and science of shaping information products and experiences to support usability and find ability
* An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape

**Information architecture**

The organization of data within an informational space. In other words, how the user will get to information or perform tasks within a website or application.

**Interaction design**

The design of how the user can participate with the information present, either in a direct or indirect way, meaning how the user will interact with the website of application to create a more meaningful experience and accomplish her goals.

**Information design**

The visual layout of information or how the user will assess meaning and direction given the information presented to him.

**Navigation design**

The words used to describe information spaces; the labels or triggers used to tell the users what something is and to establish the expectation of what they will find.

**Interface design**

The design of the visual paradigms used to create action or understanding.

The role of information architecture is played by a variety of people, from product managers to designers and even developers. To make things more confusing, information architecture can be called many different things throughout the design and development process. Words like intuitive, simple, findable, usable, or the executive favourite easy to- use—all describe the role that information architects play in creating digital experiences.

The visual design of your product, what frameworks you use, and how it is developed are integral to the success of any product, but the information architecture stands apart as being the most crucial element of your product. It is the first line of scrimmage—the user‗s first impression of your product. Even if you have the best design, the best code, and the best backend service, if the user cannot figure out how to use it, she will fail and so will your product.

**Mobile Information Architecture**

Information architecture has become a common discipline in the web industry; unfortunately, the mobile industry like software has only a handful of specialized mobile information architects. Although mobile information architecture is hardly a discipline in its own right, it certainly ought to be. This is not because it is so dissimilar from its desktop cousin, but because of context, added technical constraints, and needing to display on a smaller screen as much information as we would on a desktop.

The role of a mobile information architect would be to interpret this content to the mobile context. Do you use the same structure, or sections? Do you present the same information above the fold? If so, how should that be prioritized? How does the user navigate to other areas? Do you use the same visual and interaction paradigms, or invent new ones? And if you do start to invent new paradigms, will you lose the visual characteristics of what users expect?

**Keeping It Simple**

When thinking about your mobile information architecture, you want to keep it as simple as possible.

**Support your defined goals**

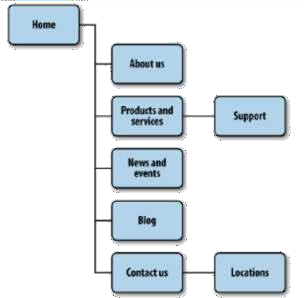
If something doesn‗t support the defined goals, lose it. Go back to your user goals and needs, and identify the tasks that map to them. Find those needs and fill them.

**Clear, simple labels**

Good trigger labels, the words we use to describe each link or action, are crucial in Mobile. Words like products or services aren‗t good trigger labels. Users have a much higher threshold of pain when clicking about on a desktop site or application, hunting and pecking for tasty morsels. Mobile performs short, to-the-point, get-it-quick, and get-out types of tasks. What is convenient on the desktop might be a deal breaker on mobile.

**4.5.1 Site Maps**

Relationship of content to other content and provide a map for how the user will travel through the informational space Limit opportunities for mistakes In Figure, you can see a poorly designed mobile information architecture that too closely mimics its desktop cousin; it was not designed with the mobile user in mind.

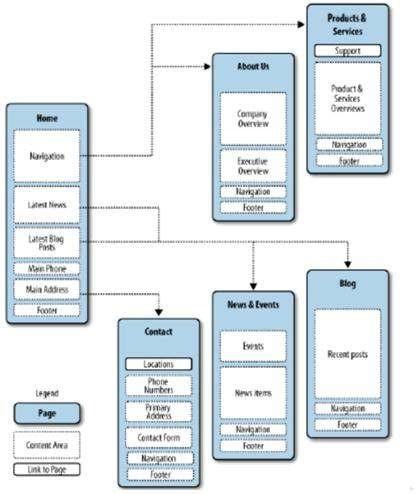


In the mobile context, tasks are short and users have limited time to perform them. And with mobile websites, we can‗t assume that the users have access to a reliable broadband connection that allows them to quickly go back to the previous page. In addition, the users more often than not have to pay for each page view in data charges. So not only do they pay cash for viewing the wrong page by mistake, they pay to again download the page they started from: we can‗t assume that pages will be cached properly.

**Confirm the path by teasing content**

Information-heavy sites and applications often employ nested or drill-down

architectures, forcing the user to select category after category to get to their target. To reduce risking the user‗s time and money, we want to make sure we present enough information for the user to wade through our information architecture successfully. On the Web, we take these risks very lightly, but with mobile, we must give our users a helping hand. We do this by teasing content within each category— that is, providing at least one content item per category.



**4.5.2 Clickstreams**

Clickstream is a term used for showing the behavior on websites, displaying the order in which users travel through a site‗s information architecture, usually based on data gathered from server logs. Clickstreams are usually historical, used to see the flaws in your information architecture, typically using heat-mapping or simple percentages to show where your users are going. I‗ve always found them to be a useful tool for researching large websites. The maps the ideal path the user will take to perform common tasks. Being able to visually lay out the path users will take gives you a holistic or bird‗s-eye view of your mobile information architecture, just as a road map does. When you can see all the paths next to each other and take a step back, you start to see shortcuts and how you can get users to their goal faster or easier, as shown in Figure. Just create user or process flows, such as the esoteric diagram shown in Figure, which is made up of boxes and diamonds that look more like circuit board diagrams than information architecture. If that is what your team prefers, then by all means, flow away. Personally, I like to present all of my information architecture deliverables from the perspective of the user, using the same metaphors she will use to make her way through my information architecture in this case, either a screen or page metaphor.

**4.5.3 Wireframes**

The next information architecture tool at our disposal is wireframes. Wireframes are

a way to lay out information on the page, also referred to as information design. Site maps show how our content is organized in our informational space; wireframes show how the user will directly interact with it. Wireframes are like the peanut butter to the site map jelly in our information architecture sandwich. It‗s the stuff that sticks. Wireframes like the one in Figure serve to make our information space tangible and useful. Wireframes to be one of the most valuable information deliverables to communicate my vision for how a site or app will work, the challenge is that a diagram on a piece of paper doesn‗t go a long way toward describing how the interactions will work. Most common are what I call ―in-place interactions or areas where the user can interact with an element without leaving the page. This can be done with Ajax or a little show/hide JavaScript. These interactions can include copious amounts of annotation, describing each content area in as much length as you can fit in the margins of the page.

**4.5.4 Prototyping**

Prototypes might sound like a scary (or costly) step in the process. Some view them as redundant or too time-consuming, preferring to jump in and start coding things. But as with wireframes, I‗ve found that each product we‗ve built out some sort of prototype has saved both time and money.

**Paper prototypes**

The most basic level we have is paper prototyping: taking our printed-out wireframes or even drawings of our interface, like the one shown in Figure, and putting them in front of people.



**Context prototype**

The next step is creating a context prototype (Figure). Take a higher-end device that enables you to load full-screen images on it. Take your wireframes or sketches and load them onto the device, sized to fill the device screen. Leave the office. Go for a walk down to your nearest café. Or get on a bus or a train. Pay particular attention to what you are thinking and your physical behavior while you are using your interface and then write it down. If you are brave and don‗t have strict nondisclosure issues, ask the people around you to use it, too. I wouldn‗t bother with timing interactions or sessions, but try to keep an eye on a clock to determine how long the average session is.

**4.6 Mobile 2.0**

**The Web as a platform** for the mobile context, this means ―write once, deploy everywhere,‖ moving away from the costly native applications deployed over multiple frameworks and networks.

**Harnessing collective intelligence** this isn‗t something the mobile community has done much of, but projects like WURFL—an open source repository of device profiles provided by the community—is exactly what mobile needs more of.

**Data is the next Intel inside** Mobile takes this principle several steps further. It can include the data we seek, the data we create, and the data about or around our physical locations. **End of the software release cycle** Long development and testing cycles heavily weigh on mobile projects, decreasing all hopes of profitability. Shorter agile cycles are needed to make mobile development work as a business. Releasing for one device, iterating, improving, and then releasing for another is a great way to ensure profitability in mobile.

**Lightweight programming models** Because mobile technology is practically built on enterprise Java, the notion of using lightweight models is often viewed with some skepticism. But decreasing the programming overhead required means more innovation occurs faster.

**Software above the level of a single device** This effectively means that software isn‗t just about computers anymore. We need to approach new software as though the user will demand it work in multiple contexts, from mobile phones to portable gaming consoles and e- book readers.

**Rich user experiences a** great and rich user experience helps people spend less time with the software and more time living their lives. Mobile design is about enabling users to live their lives better.

**4.6.1 The Convergence of the Web and Mobile**

It is obvious that in the minds of many, Mobile 2.0 is the Web. At this point, the mobile web has always been viewed as a second-class citizen within the mobile ecosystem, for many reasons, as discussed later. Mobile is already a medium, but the consensus is that by leveraging the power of the Web, integrating web services into the mobile medium is the future of mobile development. When the iPhone exploded onto the scene, it increased the usage of the mobile web by its users to levels never seen before. The spur of new mobile web apps created just for the iPhone doubled the number of mobile websites available in under a year. If Web 2.0 taught us that the Web is the platform, then Mobile 2.0 tells us that mobile will be the primary context in which we leverage the Web in the future.

**4.6.2 The Mobile Web Browser**

If the future of mobile is the Web, then it only makes sense that the mobile web browser is the next killer app of mobile. Again, this is something we saw confirmed with WebKit in the iPhone and later in Android. However, of particular concern is how device fragmentation factors into mobile browsers. For example, how can we expect developers to support more than 30 different mobile browsers? A fellow panelist from the Mozilla Minimo project offered a potential solution in consolidation—that we will see only a few mobile browsers in the future; specifically, browsers built on Mozilla, Opera, Internet Explorer, and WebKit technologies. At the time, I thought that prediction was too focused on smart phones, but in the years since, the line between smart phone and feature phone seems to be going away, so this prediction is fairly accurate. But the single biggest challenge in mobile remains device fragmentation. The mobile browser enables us to penetrate the problem by not having our content locked so specifically to the device abilities, screen size, and form factor, but device fragmentation still causes old, outdated browsers to remain in the market long after they should be put out to pasture. What appears to be solving browser fragmentation is actually the iPhone. The Mobile Safari browser included with the iPhone provided such an excellent web experience on a mobile device that it drove use of the mobile web to huge levels, which means big profits for the operators. This also means that the mobile web is no longer a secondclass citizen. In the post-iPhone market, all new devices are judged by the quality of their mobile web browser. Operators know it and therefore are demanding better

**4.6.3 JavaScript**

If you are going to provide mobile web applications, you have to have a mobile web

browser that supports Ajax, or, as it is technically known, XML Http Request. It makes a lot of those cool interactions in your web browser work via the capability to load content asynchronously into your browser view. But it isn‘t just Ajax; it is JavaScript, a web technology that has largely been ignored with most mobile web browsers. Ajax is great, but just being able to do a little show hide or change a style after you click or touch it goes a long way toward improving the user experience. This is probably where mobile web browsers fall behind desktop browsers the most. Because they both support XHTML and CSS relatively well, JavaScript has been a nogo in mobile for years. In order for mobile web apps to rival native applications, you have to support some JavaScript. Modern mobile browsers have made much progress over the last few years, but there is still plenty of work to be done. For example, accessing the device capabilities like the phone book or file system with JavaScript doesn‘t work in a consistent way. These problems still need to be solved in order to truly reap the benefits of the Web.

**4.6.4 Mobile User Experience**

Traditionally, the user experience available in the mobile web has been like using a website from 1995: mostly text-based, difficult to use, and ugly as sin. This isn‘t to say that the user experience of mobile applications has been much better, but it used to be that if you wanted a good experience, you built a native app. Descriptions within the industry range from the honest ―the mobile user experience is utterly horrid,‖ to the sales pitch of ―look at these cool things you can do,‖ to the optimistic ―the mobile user experience is the future!‖ These polar attitudes toward the mobile user experience are somewhat ironic, given that the mobile user experience was largely ignored for close to a decade. People in mobile treat the user experience like a chicken-and-egg scenario: bad input/output of the user experience prevents adoption, but designing a shiny user experience with bells and whistles will bring them in droves. Device APIs usually force you to use their models of user experience, meaning that you have to work in the constraints of the API. This is good for creating consistent experiences for that platform, but these experiences don‘t translate to others. For example, would you take an iPhone app design and put it on an Android device? The user experience for these devices is similar but still remains different. The beauty of the

Web, literally, is that you can design whatever experience you want, for better or worse. You are in control of the presentation and can establish your own visual metaphors. The problem has been that traditionally complex (which often equates to good) user experiences haven‘t been possible on mobile devices. Modern mobile web browsers, as they come closer to their desktop counterparts, remove this distinction, giving us the same canvas on mobile devices that we have for the desktop. This means that creating mobile experiences just got a whole lot easier. It also means we can have a consistent user experience across multiple mediums.

**4.7 MOBILE DESIGN**

When building mobile experiences, it is impossible to create a great experience without three ingredients: context, information architecture, and visual design. The visual design of your experience is the direct representation of everything underneath; it is the first impression the user will have. A great design gives the user high expectations of your site or application; a poor design leads to lower expectations. Users‘ expectations translate to value and trust. In the mobile space, where content is often ―free‖ (they still need to pay for data charges), users often have low expectations, due to the limitations of the canvas. Confusing site structures and slow download speeds reinforce those initial low expectations. In the downloadable application space, where application design can be much more robust, users must purchase applications almost sight unseen. For example, they may see just a small screenshot of your application or game. But if the application doesn‘t meet the higher expectations of the design, your application downloads will drop like a stone. The design, that first impression, determines right from the start if the user will spend five seconds, five minutes, or five hours with your product. This leads us to the most significant challenge in mobile design: creativity. You can‘t always be as creative as you want to be. Many devices just can‘t support complex designs for every channel; for example, on many lower-end devices, the mobile web experience may just be a list of links. But every device has the capability to create a bestin- device experience; it just depends on how you take advantage of the application medium and context that you plan to use.

**4.7.1 Interpreting Design**

Days were spent creating print designs and advertisements, defining precisely what each design would be, down to the picas and points. The method of design is meant for creating a vision for how to communicate information or ideas in his head, and then duplicating that on the printed page. Precise designs might look better, but they can be brutal to implement. More flexible designs might not be much to look at, but they work for the most users, or the lowest common denominator. But more than that, our backgrounds and our training can actually get in the way of creating the best design for the medium. We like to apply the same rules to whatever the problem in front of us might be. In mobile design, you interpret what you know about good design and translate it to this new medium that is both technologically precise and at times incredibly unforgiving, and you provide the design with the flexibility to present information the way you envision on a number of different devices. In the end, the graphic designer and I scrapped the work, and he provided me his pica perfect designs as giant images, which I turned into a series of massive image maps.

**4.7.2 The Mobile Design Tent-Pole**

In Hollywood, executives like to use the term ―tent-pole‖ to describe their movies and television shows. The term has dual meanings: one is business, and the other creative. The business goal of a tent-pole production is to support or prop up the losses from other productions. However, to create a tent-pole production, the creators involved must make an artistic work that they know will appeal to the largest possible audience, providing something for everyone. You probably know tent-pole movies as ―blockbusters‖; in television, they are known as ―anchor‖ shows. Trying to reach the widest possible audience poses a problem. Hollywood is learning with great pains that with so many entertainment options and with today‘s audience being so hard to reach through traditional advertising channels, tent-pole productions are failing. As the number of social niches increases, it becomes difficult to satisfy the specific tastes of each social group. What one group finds hysterically funny, several other groups might find offensive. Today, tent-pole productions often come off as bland and half-hearted, failing to appease anyone. One of the most interesting examples of how the tide turned in entertainment is with the animated films of Disney versus those of Pixar. For years, Disney produced tentpole family fare quite successfully. But as competition increased, notably from Pixar, Disney films would spend millions to create stale and dated films, losing audiences and revenue. Meanwhile, Pixar found that their movies could be successful by avoiding the traditional storytelling formats of animated film, which Disney essentially defined. Instead, Pixar based their stories around specific emotional themes and was able to connect with audiences of all ages, in multiple cultures and across multiple niches. In 2006, Disney acquired Pixar, making its top executives the new leaders of all Disney creative projects. Although Disney technically acquired Pixar, I‘ve always looked at it as the other way around. Disney realized that it needed to be more Pixar and less Disney in order to grow and adapt to today‘s changing audiences and niches. This is something that Pixar was doing correctly. Back in the world of mobile design, the de facto strategy is to create tent-pole products. Like the old days of Disney, the strategy is to sink millions into creating tent-pole products, or products that support the largest number of devices that no one will ever use. They are creatively stale, they lack inspiration, and they simply exist with no meaningful purpose to the user. They make the same mistake Disney made, thinking that it could simply put something on the market that might not be the best quality, but because it carried the Disney name, people would buy it. To have a successful mobile design, you have to think like Pixar. Find that emotional connection, that fundamental need that serves many audiences, many cultures, and many niches and design experiences. Too often, designers simply echo the visual trends of the day, mimicking the inspiration of others—I‘m certainly guilty of it. But with mobile design, once you find that essential thing, that chewy nougat we call ―context‖ that lives at the center of your product, then you will find ample inspiration of your own to start creating designs that translate not only across multiple devices, but across multiple media. Sure, there are countless examples of poorly designed mobile products that are considered a success. You only need to look as far as the nearest mobile app store to find them. This is because of the sight unseen nature of mobile commerce. Traditionally, you couldn‘t demo— or in some cases even see— screenshots of a game or mobile application before you bought it. You simply had to purchase it and find out if it was any good. Apple‘s App Store quickly changed that. You can clearly see that the best-selling games and applications for the iPhone are the ones with the best designs.



**THE ELEMENTS OF MOBILE DESIGN**

Good design requires three abilities: the first is a natural gift for being able to see visually how something should look that produces a desired emotion with the target audience. The second is the ability to manifest that vision into something for others to see, use, or participate in. The third knows how to utilize the medium to achieve your design goals.

Six elements of mobile design that you need to consider, starting with the context and layering in visual elements or laying out content to achieve the design goal. Then, you need to understand how to use the specific tools to create mobile design, and finally, you need to understand the specific design considerations of the mobile medium.

**4.8.1 Context**

I won‗t belabor the point except to say that context is core to the mobile experience. As the designer, it is your job to make sure that the user can figure out how to address context using your app. Make sure you do your homework to answer the following questions:

* Who are the users? What do you know about them? What type of behavior can you assume or predict about the users?
* What is happening? What are the circumstances in which the users will best absorb the content you intend to present?
* When will they interact? Are they at home and have large amounts of time? Are they at work where they have short periods of time? Will they have idle periods of time while waiting for a train, for example?
* Where are the users? Are they in a public space or a private space? Are they inside or outside? Is it day or is it night?
* Why will they use your app? What value will they gain from your content or services in their present situation?
* How are they using their mobile device? Is it held in their hand or in their pocket?
* How are they holding it? Open or closed? Portrait or landscape?

The answers to these questions will greatly affect the course of your design. Treat these questions as a checklist to your design from start to finish.

**4.8.2 Message**

Message is the overall mental impression you create explicitly through visual design. I like to think of it as the holistic or at times instinctual reaction someone will have to your design. If you take a step back, and look at a design from a distance, what is your impression? Or conversely, look at a design for 30 seconds, and then put it down. What words would you use to describe the experience?

Branding shouldn‗t be confused with messaging. Branding is the impression your company name and logo gives—essentially, your reputation. Branding serves to reinforce the message with authority, not deliver it. In mobile, the opportunities for branding are limited, but the need for messaging is great. With such limited real estate, the users don‗t care about your brand, but they will care about the messaging, asking themselves questions like, ―What can this do for me? or ―Why is this important to me?

**Yahoo!**

Yahoo! sort of delivers a message. This app provides a clean interface, putting a focus on search and location, using color to separate it from the news content. But I‗m not exactly sure what it is saying. Words you might use to describe the message are crisp, clean, and sharp.

**ESPN**

The ESPN site clearly is missing a message. It is heavily text-based, trying to put a lot of content above the fold, but doesn‗t exactly deliver a message of any kind. If you took out the ESPN logo, you likely would have indifferent expectations of this site; it could be about anything, as the design doesn‗t help set expectations for the user in any way. Words you might use to describe the message: bold, cluttered, and content-heavy.

**Disney**

Disney creates a message with its design. It gives you a lot to look at—probably too much— but it clearly tries to say that the company is about characters for a younger audience. Words you might use to describe the message: bold, busy, and disorienting.

**Wikipedia**

The Wikipedia design clearly establishes a message. With a prominent search and text-heavy layout featuring an article, you know what you are getting with this design. Words you might use to describe the message: clean, minimal, and text-heavy.

**Amazon**

Amazon sort of creates a message. Although there are some wasted opportunities above the fold with the odd ad placement, you can see that it is mostly about products (which are improved even more if you scroll down). Words you might use to describe the message: minimal but messy, product-heavy, and disorienting.

**4.8.3 Look and Feel**

Look and feel is used to describe appearance, as in I want a clean look and feel or I want a usable look and feel. The problem is: as a mobile designer, what does it mean? And how is that different than messaging? I think of look and feel in a literal sense, as something real and tactile that the users can look at, and then feel something they can touch or interact with. Look and feel is used to evoke action how the user will use an interface. Messaging is holistic, as the expectation the users will have about how you will address their context. It is easy to confuse the two, because feel can be interpreted to mean our emotional reaction to design and the role of messaging.

I often find myself explaining the look and feel with the word because, with a cause- and-effect rationale for design decisions, as in The user will press this button because...‖ or The user will go to this screen because… followed by a reason why a button or control is designed a certain way. Establishing a look and feel usually comes from wherever design inspiration comes from. However, your personal inspiration can be a hard thing to justify. Therefore we have design patterns, or documented solutions to design problems, sometimes referred to as style guides. On large mobile projects or in companies with multiple designers, a style guide or pattern library is crucial, maintaining consistency in the look and feel and reducing the need for each design decision to be justified.



**4.8.4 Layout**

Layout is an important design element, because it is how the user will visually process the page, but the structural and visual components of layout often get merged together, creating confusion and making your design more difficult to produce. The first time layout should rear its head is during information architecture. In fact, I prefer to make about 90 percent of my layout decisions during the information architecture period. I ask myself questions like: where should the navigation go on the page or screen? What kind of navigation type should I use? Should I use tabs or a list? What about a sidebar for larger screens? All of these should be answered when defining the information architecture and before you begin to design. Design is just too subjective of an issue. If you are creating a design for anyone but yourself, chances are good that there will be multiple loosely-basedon- experience opinions that will be offered and debated.

Where the design opinions of the CEO or Chief Marketing Officer (CMO) might influence a design direction more than, say, the Creative Director or Design Director. By defining design elements like layout prior to actually applying the look and feel, you can separate the discussion. As a self-taught designer, I started out in this business making designs for my own projects. I could just put pen to paper and tweak it to my heart‗s content. If I wanted to radically change the layout, I could. When I started my mobile design career with my first mobile company more than a decade ago, I realized that this approach didn‗t work. The majority of comments that reviewers would make were about the layout. They focused on the headers, the navigation, the footer, or how content blocks are laid out, and so on. But their feedback got muddied with the look and feel, the colors, and other design elements.

Reviewers do make remarks like I like the navigation list, but can you make it look more raised? Most designers don‗t hear that; they hear the navigation isn‗t right, do it again. But, with this kind of feedback, there are two important pieces of information about different types of design. First, there is confirmation that the navigation and layout are correct. Second, there is a question about the look and feel. Because designers hear Do it again, they typically redo the layout, even though it was actually fine.

Creating mobile designs in an environment with multiple reviewers is all about getting the right feedback at the right time. Your job is to create a manifestation of a shared vision. Layout is one of the elements you can present early on and discuss independently. People confuse the quality and fidelity of your deliverables as design. By keeping it basic, you don‗t risk having reviewers confuse professionalism with design. The irony is that as I become more adept at defining layouts, I make them of increasingly lower fidelity. For example, when I show my mobile design layouts as wireframes during the information architecture phase, I intentionally present them on blueprint paper, using handwriting fonts for my annotations. It also helps to say that this is not a design; it is a layout, so please give me feedback on the layout.

**4.8.5 Color**

The fifth design element, color, is hard to talk about in a black-and-white book.

Maybe it is fitting, because it wasn‗t that long ago that mobile screens were available only inblack and white well, technically, it was black on a green screen). These days, we have nearly the entire spectrum of colors to choose from for mobile designs.

The most common obstacle you encounter when dealing with color is mobile screens, which come in a number of different color or bit depths, meaning the number of bits (binary digits) used to represent the color of a single pixel in a bitmapped image. When complex designs are displayed on different mobile devices, the limited color depth on one device can cause banding, or unwanted posterization in the image.

For an example of posterization, the technical term for when the gradation of tone is replaced with regions of fewer tones, see in Figure 8-10 how dramatically the color depth can affect the quality of a photo or gradient, producing banding in several parts in the image.

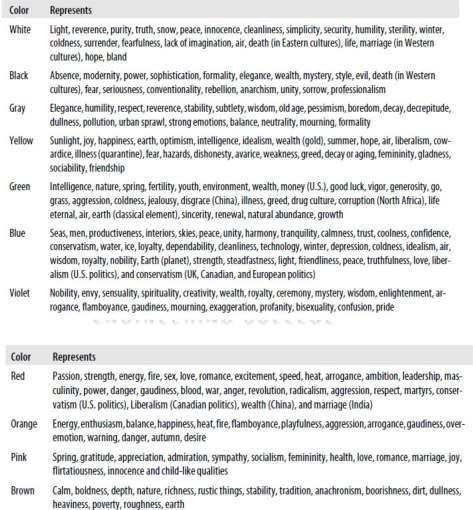


**Color palettes**

Defining color palettes can be useful for maintaining a consistent use of color in your mobile design. Color palettes typically consist of a predefined number of colors to use throughout the design. Selecting what colors to use varies from designer to designer, each having different techniques and strategies for deciding on the colors. I‗ve found that I use three basic ways to define a color palette:

**Sequential**

In this case, there are primary, secondary, and tertiary colors. Often the primary color is reserved as the ―brand color or the color that most closely resembles the brand‗s using a color wheel.



**Adaptive**

An adaptive palette is one in which you leverage the most common colors present in a supporting graphic or image. When creating a design that is meant to look native on the device, I use an adaptive palette to make sure that my colors are consistent with the target mobile platform.

**Inspired**

This is a design that is created from the great pieces of design you might see online, as shown in Figure below, or offline, in which a picture of the design might inspire you. This could be anything from an old poster in an alley, a business card, or some packaging.

When I sit down with a new design, I thumb through some of materials to create an inspired palette. Like with the adaptive palette, you actually extract the colors from the source image, though you should never ever use the source material in a design.

**4.9 MOBILE DESIGN TOOLS**

Mobile design requires understanding the design elements and specific tools. The closest thing to a common design tool is Adobe Photoshop, though each framework has a different method of implementing the design into the application. Some frameworks provide a complete interface toolkit, allowing designers or developers to simply piece together the interface, while others leave it to the designer to define from scratch.



**Designing for the Right Device**

The truly skilled designer doesn‗t create just one product—she translates ideas into experiences. The spirit of your design should be able to be adapted to multiple devices. What device suits this design best? What market niche would appreciate it most? What devices are the most popular within that niche? The days of tent-poles are gone. Focus instead on getting your best possible experience to the market that will appreciate it most. It might not be the largest or best long-term market, but what you will learn from the best possible scenario will tell you volumes about your mobile product‗s potential for success or failure.

This knowledge will help you develop your porting and/or adaptation strategy, the most expensive and riskiest part of the mobile equation. iPhone users consume more mobile content and products than the average mobile user. This platform has an easy-to-learn framework and excellent documentation, for both web and native products, and an excellent display and performance means. Although iPhone users might not be the majority of your market, the ability to create the best possible design and get it in front of those users presents the least expensive product to produce with the lowest risk.

With a successful single device launch, you can start to adapt designs from the best possible experience to the second best possible experience, then the third, and fourth, and so on. The best possible experience is how it should be, so it serves as a reference point for how we will adapt the experience to suit more devices.

**Designing for Different Screen Sizes**

Mobile devices come in all shapes and sizes. Choice is great for consumers, but bad for design. It can be incredibly difficult to create that best possible experience for a plethora of different screen sizes. For example, your typical feature phone might only be 140 pixels wide, whereas your higher-end smartphone might be three to four times wider. Landscape or portrait? Fixed width or fluid? Do you use one column or two? These are common questions that come up when thinking about your design on multiple screen sizes. The bad news is that there is no simple answer. How you design each screen of content depends on the scope of devices you look to support, your content, and what type of experience you are looking to provide. The good news is that the vast majority of mobile device screens share the same vertical or portrait orientation, even though they vary greatly in dimension.

**References**

1. Brian Fling, “Mobile Design and Development”, First Edition, O’Reilly Media Inc., 2009

**WEB INTERFACE DESIGN**

**UNIT 5 WEB INTERFACE DESIGN**

Designing Web Interfaces - Drag & Drop, Direct Selection, Contextual Tools, Overlays, Inlays and Virtual Pages, Process Flow. Case Studies.

**User Interface Design Basics**

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

**Choosing Interface Elements**

Users have become familiar with interface elements acting in a certain way, so try to be consistent and predictable in your choices and their layout. Doing so will help with task completion, efficiency, and satisfaction.

**Interface elements include but are not limited to:**

Input Controls: buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons Informational Components: tooltips, icons, progress bar, notifications, message boxes, modal windows Containers: accordion

There are times when multiple elements might be appropriate for displaying content.

When this happens, it‗s important to consider the trade-offs. For example, sometimes elements that can help save you space, put more of a burden on the user mentally by forcing them to guess what is within the dropdown or what the element might be.

**Best Practices for Designing an Interface**

Everything stems from knowing your users, including understanding their goals, skills, preferences, and tendencies. Once you know about your user, make sure to consider the following when designing your interface:

**Keep the interface simple.**

The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging. Create consistency and use common UI elements. By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.

Be purposeful in page layout. Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability. Strategically use color and texture. You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage. Use typography to create hierarchy and clarity. Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scalability, legibility and readability.

Make sure that the system communicates what‗s happening. Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user. Think about the defaults. By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.

**5.1 DRAG AND DROP**

**5.1.1 Interesting Moments**

At first blush, drag and drop seems simple. Just grab an object and drop it somewhere. But, as always, the devil is in the details. There are a number of individual states at which interaction is possible. We call these microstates interesting moments:

* How will users know what is draggable?
* What does it mean to drag and drop an object?
* Where can you drop an object, and where is it not valid to drop an object?
* What visual affordance will be used to indicate draggability?
* ill valid and invalid drop targets be signified?
* Do you drag the actual object?
* Or do you drag just a ghost of the object?
* Or is it a thumbnail representation that gets dragged?

* What visual feedback should be used during the drag and drop interaction? What makes it challenging is that there are a lot of events during drag and drop that can be used as opportunities for feedback to the user. Additionally, there are a number of elements on the page that can participate as actors in this feedback loop.

**The Events:** There are at least 15 events available for cueing the user during a drag and drop interaction:

**Page Load:** Before any interaction occurs, you can pre-signify the availability of drag and drop. For example, you could display a tip on the page to indicate draggability.

**Mouse Hover:** The mouse pointer hovers over an object that is draggable. **Mouse Down:** The user holds down the mouse button on the draggable object. **Drag Initiated:** After the mouse drag starts (usually some threshold— 3 pixels).

**Drag Leaves Original Location:** After the drag object is pulled from its location or object that contains it.

**Drag Re-Enters Original Location:** When the object re-enters the original location.

**Drag Enters Valid Target:** Dragging over a valid drop target.

**Drag Exits Valid Target**: Dragging back out of a valid drop target.

**Drag Enters Specific Invalid Target:** Dragging over an invalid drop target.

**Drag Is Over No Specific Target:** Dragging over neither a valid or invalid target. Do you treat all areas outside of valid targets as invalid?

**Drag Hovers Over Valid Target :** User pauses over the valid target without dropping the object. This is usually when a spring loaded drop target can open up. For example, drag over a folder and pause, the folder opens revealing a new area to drag into.

**Drag Hovers Over Invalid Target:** User pauses over an invalid target without dropping the object. Do you care? Will you want additional feedback as to why it is not a valid target?

**Drop Accepted :** Drop occurs over a valid target and drop has been accepted.

**Drop Rejected**: Drop occurs over an invalid target and drop has been rejected. Do you zoom back the dropped object?

**Drop on Parent Container:** Is the place where the object was dragged from special?

Usually this is not the case, but it may carry special meaning in some contexts.

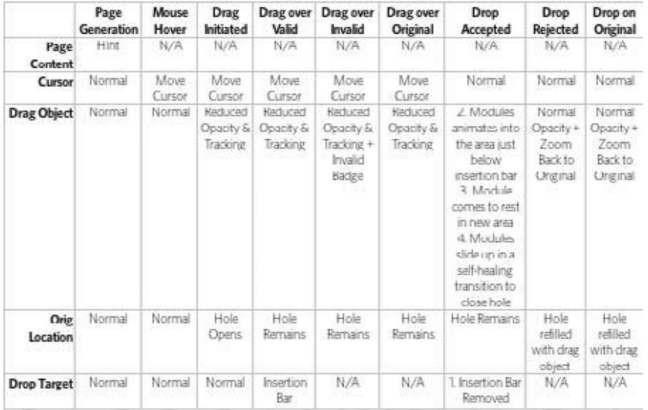
**The Actors :** During each event you can visually manipulate a number of actors. The page elements available include:

* Page (e.g., static messaging on the page)
* Cursor
* Tool Tip
* Drag Object (or some portion of the drag object, e.g., title area of a module)
* Drag Object‗s Parent Container
* Drop Target

Interesting Moments

Grid

That‗s 15 events times 6 actors. That means there are 90 possible interesting moment search requiring a decision involving an almost unlimited number of style and timing choices. You can pull all this together into a simple interesting moment‘s grid for Drag and Drop.



**5.1.3 Drag and Drop Module**

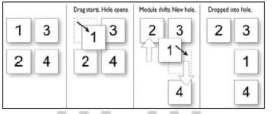
One of the most useful purposes of drag and drop is to allow the user to directly place objects where she wants them on the page. A typical pattern is Drag and Drop Modules on a page. Netvibes provides a good example of this interaction pattern

**Considerations**

Netvibes allows its modules to be rearranged with drag and drop. A number of interesting moments decide the specific interaction style for this site. Figure shows the interesting moment‘s grid for Netvibes. While dragging, it is important to make it clear what will happen when the user drops the dragged object. There are two common approaches to targeting a drop: • Placeholder target • Insertion target

**Placeholder target**

Netvibes uses a placeholder (hole with dashed outline) as the drop target. The idea (illustrated in Figure ) is to always position a hole in the spot where the drop would occur. When module 1 starts dragging, it gets ―ripped‖ out of the spot. In its place is the placeholder target (dashed outline). As 1 gets dragged to the spot between 3 and 4, the placeholder target jumps to fill in this spot as 4 moves out of the way.



The hole serves as a placeholder and always marks the spot that the dragged module will land when dropped. It also previews what the page will look like (in relation to the other modules) if the drop occurs there. For module drag and drop, the other modules only slide up or down within a vertical column to make room for the dragged module.

One complaint with using placeholder targets is that the page content jumps around a lot during the drag. This makes the interaction noisier and can make it harder to understand what is actually happening. This issue is compounded when modules look similar. The user starts dragging the modules around and quickly gets confused about what just got moved. One way to resolve this is to provide a quick animated transition as the modules move. It is important, however, that any animated transitions not get in the way of the normal interaction.

**Boundary-based placement.** Since most sites that use placeholder targeting drag the module in its original size, targeting is determined by the boundaries of the dragged object and the boundaries of the dragged-over object. The mouse position is usually ignored because modules are only draggable in the title (a small region). Both Netvibes and iGoogle take the boundary-based approach. But, interestingly, they calculate the position of their placeholders differently. In Netvibes, the placeholder changes position only after the dragged module‗s title bar has moved beyond the dragged-over module‗s title bar. In practice, this means if you are moving a small module to be positioned above a large module, you have to move it to the very top of the large module. In Figure you have to drag the small ―To Do List module all the way to the top of the ―Blog Directory module before the placeholder changes position.



**Insertion target**

Placeholder positioning is a common approach, but it is not the only way to indicate drop targeting. An alternate approach is to keep the page as stable as possible and only move around an insertion target (usually an insertion bar). A previous version of My Yahoo! Used the insertion bar approach as the dragged module was moved around.

**Drag distance**

Dragging the thumbnail around does have other issues. Since the object being dragged is small, it does not intersect a large area. It requires moving the small thumbnail directly to the place it will be dropped. With iGoogle, the complete module is dragged. Since the module will always be larger than the thumbnail, it intersects a drop target with much less movement. The result is a shorter drag distance to accomplish a move

**Drag rendering**

How should the dragged object be represented? Should it be rendered with a slight transparency (ghost)? Or should it be shown fully opaque? Should a thumbnail representation be used instead? **5.1.4 Drag and Drop List**

The Drag and Drop List pattern defines interactions for rearranging items in a list. 37

Signal‗s Backpackit allows to-do items to be rearranged with Drag and Drop List **Considerations** Backpackit takes a real-time approach to dragging items. Since the list is constrained, this is a natural approach to moving objects around in a list. You immediately see the result of the drag.

**Placeholder target**

This is essentially the same placeholder target approach we discussed earlier for dragging and dropping modules. The difference is that when moving an item in a list, we are constrained to a single dimension. Less feedback is needed. Instead of a ―ripped-out‖ area (represented earlier with a dotted rectangle), a simple hole can be exposed where the object will be placed when dropped.

**Insertion target**

Drag and Drop Modules, placeholder targeting is not the only game in town. You can also use an insertion bar within a list to indicate where a dropped item will land. Netflix uses an insertion target when movies are dragged to a new location in a user‗s movie queue

The upside to this approach is that the list doesn‗t have to shuffle around during drag. The resulting experience is smoother than the Backpack it approach. The downside is that it is not as obvious where the movie is being positioned. The insertion bar appears under the ghosted item. The addition of the brackets on the left and right of the insertion bar is an attempt to make the targeting clearer.

**Non–drag and drop alternative**

Besides drag and drop, the Netflix queue actually supports two other ways to move objects around: Edit the row number and then press the ―Update DVD Queue button. Click the ―Move to Top‖ icon to pop a movie to the top. Modifying the row number is straightforward. It‗s a way to rearrange items without drag and drop. The ―Move to Top button is a little more direct and fairly straightforward (if the user really understands that this icon means ―move to top). Drag and drop is the least discoverable of the three, but it is the most direct, visual way to rearrange the list. Since rearranging the queue is central to the

Netflix customer‗s satisfaction, it is appropriate to allow multiple ways to do so.

**Hinting at drag and drop**

When the user clicks the ―Move to Top button, Netflix animates the movie as it moves up. But first, the movie is jerked downward slightly and then spring-loaded to the top The combination of the downward jerk and then the quick animation to the top gives a subtle clue that the object is draggable. This is also an interesting moment to advertise drag and drop. After the move to top completes, a simple tip could appear to invite users to drag and drop. The tip should probably be shown only once, or there should be a way to turn it off. Providing an invitation within a familiar idiom is a good way to lead users to the new idiom. **Drag and Drop Object**

Another common use for drag and drop is to change relationships between objects. This is appropriate when the relationships can be represented visually. Drag and drop as a means of visually manipulating relationships is a powerful tool.Cogmap is a wiki for organizational charts. Drag and Drop Object is used to rearrange.

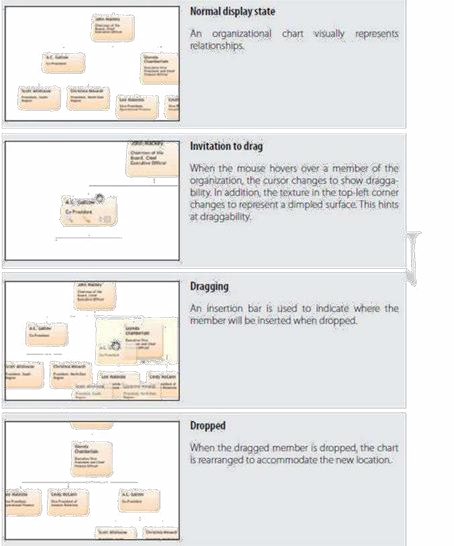
**Considerations**

When object relationships can be clearly represented visually, drag and drop is a natural choice to make these type of changes. Cogmap uses the target insertion approach. This allows the dragging to be nondistracting, since the chart does not have to be disturbed during targeting.

**5.1.6 Drag and Drop Action**

Drag and drop is also useful for invoking an action or actions on a dropped object. The Drag and Drop Action is a common pattern. Its most familiar example is dropping an item in the trash to perform the delete action.

Normally uploading files to a web application includes pressing the upload button and browsing for a photo. This process is repeated for each photo.



**Considerations**

This is not a trivial implementation. But it does clearly illustrate the benefit of drag and drop for operating on a set of files. The traditional model requires each photo to be selected individually for upload. Drag and drop frees you to use whatever browsing method is available on your system and then drop those photos for upload.

**Anti-pattern: Artificial Visual Construct**

Unfortunately, drag and drop can sometimes drive the design of an interface instead of being an extension of a natural interface. These interactions are almost always doomed, as they are the tail wagging the proverbial dog. Rating movies, books, and music is a common feature found on many sites. But what happens if you try to use drag and drop to rate movies?

**5.1.7 Drag and Drop Collection**

A variation on dragging objects is collecting objects for purchase, bookmarking, or saving into a temporary area. This type of interaction is called Drag and Drop Collection. Drag and drop is a nice way to grab items of interest and save them to a list. Drag and drop is a natural way collect items for purchase. It mimics the shopping experience in the real world. Grab an item. Drop it in your basket. This is fast and convenient once you know about the feature. However, as a general rule, you should never rely solely.

**The Challenges of Drag and Drop**

As you can see from the discussion in this chapter, Drag and Drop is complex. There are four broad areas where Drag and Drop may be employed: Drag and Drop Module, Drag and Drop List, Drag and Drop Object, and Drag and Drop Action. And in each area, there are a large number of interesting moments that may be handled in numerous ways. Being consistent in visual and interaction styles across all of these moments for all of these types of interactions is a challenge in itself. And keeping the user informed throughout the process with just the right amount of hints requires design finesse.

**5.2 DIRECT SELECTION**

* **Toggle Selection:** Checkbox or control-based selection.
* **Collected Selection:** Selection that spans multiple pages.
* **Object Selection:** Direct object selection.
* **Hybrid Selection:** Combination of Toggle Selection and Object Selection

**Toggle Selection**

The most common form of selection on the Web is Toggle Selection. Checkboxes and toggle buttons are the familiar interface for selecting elements on most web pages.



The way to select an individual mail message is through the row‗s checkbox. Clicking on the row itself does not select the message. We call this pattern of selection Toggle Selection since toggle-style controls are typically used for selecting items. Once items have been check- selected, actions can be performed on them. Usually these actions are performed on the selection by clicking on a separate button (e.g., the Delete button). Gmail is a good example of actions in concert with Toggle Selection.

Toggle Selection with checkboxes has some nice attributes:

* Clear targeting, with no ambiguity about how to select the item or deselect it.
* Straightforward discontinuous selection, and no need to know about Shift or Control key ways to extend a selection. Just click the checkboxes in any order, either in a continuous or discontinuous manner.
* Clear indication of what has been selected.

**Scrolling versus paging**

The previous examples were with paged lists. But what about a scrolled list? Yahoo! Mail uses a scrolled list to show all of its mail messages (Figure). While not all messages are visible at a time, the user knows that scrolling through the list retains the currently selected items. Since the user understands that all the messages not visible are still on the same continuous pane, there is no confusion about what an action will operate on—it will affect all selected items in the list. Sometimes the need for clarity of selection will drive the choice between scrolling and paging.

Making selection explicit

With Yahoo! Bookmarks you can manage your bookmarks by selecting bookmarked pages and then acting on them. The selection model is visually explicit.

**5.2.2 Collected Selection**

Toggle Selection is great for showing a list of items on a single page. But what happens if you want to collect selected items across multiple pages? Collected Selection is a pattern for keeping track of selection as it spans multiple pages. In Gmail, you can select items as you move from page to page. The selections are remembered for each page. If you select two items on page one, then move to page two and select three items, there are only three items selected. This is because actions only operate on a single page. This makes sense, as users do not normally expect selected items to be remembered across different pages.

**Considerations**

Gmail does provide a way to select all items across different pages. When selecting all items on a individual page (with the ―All‖ link), a prompt appears inviting the user to ―Select all 2785 conversations in Spam‖. Clicking that will select all items across all pages (Figure). The ―Delete Forever‖ action will operate on all 2785 conversations, not just the 25 selected on the page.



**Keeping the selection visible**

The real challenge for multi-page selection is finding a way to show selections gathered across multiple pages. You need a way to collect and show the selection as it is being created. Here is one way that Collected Selection comes into play. LinkedIn uses Collected Selection to add potential contacts to an invite list.

The list of potential invitees is shown in a paginated list on the lefthand side. Clicking the checkbox adds them to the invite list. The invite list becomes the place where selected contacts across multiple pages are remembered.

**Collected Selection and actions**

In the menu system it was hard to discern whether the user meant to operate on the selection (photos on the page could be selected through an Object Selection model) or on the collected items in the tray. To resolve this ambiguity, the drop-down menus contained two identical sets of commands. The first group of commands in the menu operated on the collected items in the tray. The second set of commands operated on the selected objects. Needless to say, this was confusing since it required the user to be fully aware of these two selection models when initiating a command.

One way to remove this ambiguity would have been to have a single set of commands that operated on either the tray or the photos—depending on which had the focus. This would require a way to select the tray and a way to deselect it (by clicking outside the tray). A possible approach would be to slightly dim the photo gallery when the tray is selected (causing it to clearly have the focus), and do the opposite when the tray is not the focus.

**Object Selection**

Object Selection, is when selection is made directly on objects within the interface.

Sometimes using a checkbox does not fit in with the style of interaction desired. Laszlo‗s WebTop mail allows the user to select messages by clicking anywhere in the row



**Considerations**

Desktop applications tend to use Object Selection. It is also natural that web-based mail applications that mimic desktop interactions employ this same style of selection. Instead of showing a control (like a checkbox), the object itself can be selected and acted on directly. Object Selection can be extended by holding down the Shift key while clicking on a different item. The Command key (Macintosh) or Control key (Windows) can be used to individually add items in a discontinuous manner. The downside to this approach is that it is not obvious to use the modifier keys for extending the selection. Toggle Selection‗s use of toggle buttons makes the selection extension model completely obvious.

**Desktop-style selection**

For now Object Selection is not as common on the Web. Given that most sites have been content-oriented, there have been few objects to select. Also, with the Web‗s simple event model, Object Selection was not easy to implement. In typical web pages, keyboard events have rarely made sense since they are also shared with the browser.

Object Selection interactions include ways to use the mouse to drag-select objects. Yahoo! Photos introduced this same type of object selection to its photo gallery (Figure below). Individually clicking on a photo selects it. Using the Shift key and clicking also extends the selection. In addition, using the Control key and clicking discontinuously selects photos. And like most desktop applications, you can drag a selection box around a group of items to add them to the selected set (in this case, photos).



**Hybrid Selection**

Mixing Toggle Selection and Object Selection in the same interface can lead to a confusing interface. Referring back to Yahoo! Bookmarks, you‗ll see an odd situation arise during drag and drop



Figure: In Yahoo! Bookmarks, one item is selected, but two items can be dragged by dragging on the unselected item.

**Considerations**

There are a few important issues to consider when using Hybrid Selection.

**Confusing two models**

One bookmark element is selected (notice the checkbox Toggle Selection). The second bookmark element (―Dr. Dobb‗s) is unselected (the checkbox is clear). In the right panel of clicking and dragging on the unselected bookmark element initiates a drag. The drag includes both the selected element and the unselected element. Since only one is shown as selected, this creates a confusing situation.

This occurs because three things are happening in the same space:

* Toggle Selection is used for selecting bookmarks for editing, deleting, etc.
* Object Selection is used for initiating a drag drop.
* Mouse click is used to open the bookmark on a separate page.

The problem is that more than one interaction idiom is applied to the same place on the same page. In this case, if you happen to try to drag, but instead click, you will be taken to a new page. And if you drag an unselected item, you now have two items selected for drag but only one shown as selected for other operations. This is definitely confusing. Simply selecting the item (automatically checking the box) when the drag starts would keep the selection model consistent in the interface. However, it might lead the user to expect a single click to also do the same (which it cannot since it opens the bookmark). So, mixing the two selection models together can be problematic. However, there is a way to integrate the Toggle Selection and Object Selection and have them coexist peacefully as well as create an improved user experience.

**CONTEXTUAL TOOLS**

**Interaction in Context**

Desktop applications separate functionality from data. Menu bars, toolbars, and palettes form islands of application functionality. Either the user chooses a tool to use on the data or makes a selection and then applies the tool. They were completely contentoriented. Rich tool sets were not needed for simply viewing and linking to content pages. Even in e- commerce sites like Amazon or eBay, the most functionality needed was the hyperlink and Submit button.

Touch-based interfaces were the stuff of research labs and, more recently, interesting You- Tube videos. But now they‗re as close as our phones. Most notably, the Apple iPhone brought touch to the masses. Gesture-based interfaces seemed even further out. Yet these became reality with the Nintendo Wii.

**5.3.2 Fitts’s Law**

Fitts‗s Law is an ergonomic principle that ties the size of a target and its contextual proximity to ease of use. Bruce Tognazzini restates it simply as:

The time to acquire a target is a function of the distance to and size of the target. In other words, if a tool is close at hand and large enough to target, then we can improve the user‗s interaction. Putting tools in context makes for lightweight interaction.

**5.3.3 Contextual Tools**

Contextual Tools are the Web‗s version of the desktop‗s right-click menus. Instead of having to right-click to reveal a menu, we can reveal tools in context with the content. We can do this in a number of ways:

* **Always-Visible Tools:** Place Contextual Tools directly in the content.
* **Hover-Reveal Tools:** Show Contextual Tools on mouse hover.
* **Toggle-Reveal Tools:** A master switch to toggle on/off Contextual Tools for the page.
* **Multi-Level Tools:** Progressively reveal actions based on user interaction.
* **Secondary Menus:** Show a secondary menu (usually by right-clicking on an object).

**5.3.4 Always-Visible Tools**

The simplest version of Contextual Tools is to use Always-Visible Tools. Digg is an example of making Contextual Tools always visible



The ―digg it button and Digg scorecard provide Always-Visible Tools next to each story. Clear call to action Why not hide the tools and only reveal them when the mouse is over the story? Since digging stories is central to the business of Digg, always showing the tool provides a clear call to action. There are other actions associated with news stories (comments, share, bury, etc.) but they are represented less prominently. In the case of Digg, the designers chose to show these at all times.

**Relative importance**

The ―digg it action is represented as a button and placed prominently in the context of the story. The ―bury it action is represented as a hyperlink along with other ―minor actions just below the story. The contrast of a button and a hyperlink as well as its placement gives a strong indication as to the relative importance of each action.

**Discoverability**

Discoverability is a primary reason to choose Always-Visible Tools. On the flip side, it can lead to more visual clutter. In the case of Digg and Netflix, there is a good deal of visual space given to each item (story, movie). But what happens when the items you want to act on are in a list or table? Generally Contextual Tools in a list work well when the number of actions is kept to a minimum. Gmail provides a single Always-Visible Tool in its list of messages—the star rating—for flagging emails Simply clicking the star flags the message as important. The unstarred state is rendered in a visually light manner, which minimizes the visual noise in the list.

**5.3.5 Hover-Reveal Tools**

One way to do this is to reveal the tools when the user pauses the mouse over an object. The Hover-Reveal Tools pattern is most clearly illustrated by 37 Signal‗s Backpackit (Figure below). To-do items may be deleted or edited directly in the interface.

The tools to accomplish this are revealed on mouse hover.

**Discoverability**

A serious design consideration for Hover-Reveal Tools is just how discoverable the additional functionality will be. While the Contextual Tools are revealed on hover, the checkbox is always visible for each to-do item. To check off an item, users have to move the mouse over it. When they do, they will discover the additional functionality.

**Contextual Tools in an overlay**

There are several actions available for a focused object. Instead of placing tools beside the object being acted on, the revealed tools can be placed in an overlay. However, there can be issues with showing contextual tools in an overlay:

1. Providing an overlay feels heavier. An overlay creates a slight contextual switch for the user‗s attention.
2. The overlay will usually cover other information—information that often provides context for the tools being offered.
3. Most implementations shift the content slightly between the normal view and the overlay view, causing the users to take a moment to adjust to the change.
4. The overlay may get in the way of navigation. Because an overlay hides at least part of the next item, it becomes harder to move the mouse through the content without stepping into a ―landmine.

**5.3.6 Toggle-Reveal Tools**

A variation on the two previous approaches is to not show any Contextual Tools until a special mode is set on the page. A good example of Toggle-Reveal Tools is in Basecamp‗s category editing



**Considerations**

Here are a few considerations to keep in mind when using Toggle-Reveal Tools.

**Soft mode**

Generally, it is a good thing to avoid specific modes in an interface. However, if a mode is soft it is usually acceptable. By soft we mean the user is not trapped in the mode. With Base camp, the user can choose to ignore the tools turned on. It just adds visual noise and does not restrict the user from doing other actions. This is a nice way to keep the interaction lightweight.

When would you use this technique? When the actions are not the main thing and you want to reduce visual noise. This fits the category example perfectly. Items are renamed or deleted occasionally. It is common, however, to want to click through and see the contents of a category (the category is always hyperlinked). Hence, make it readable and easily navigable in the normal case but still give the user a way to manage the items in context.

Google Reader could potentially be improved in this manner. In the current interface, clicking ―Manage Subscriptions takes the user to another page to edit subscriptions. One possible change is the addition of an edit button that toggles in a set of context tools for each subscription. This would allow the user to rename and unsubscribe without leaving the context of the reading pane.

**5.3.7 Multi-Level Tools**

Contextual Tools can be revealed progressively with Multi-Level Tools. Songza\* provides a set of tools that get revealed after a user clicks on a song. Additional tools are revealed when hovering over the newly visible tools

**Radial menus**

Radial menus\* such as in Songza have been shown to have some advantages over more traditional menus. First, experienced users can rely on muscle memory rather than having to look directly at the menu items. Second, the proximity and targeting size make the

menu easy to navigate since the revealed menu items are all equally close at hand (recall

Fitts‗s Law).

The one potential downside to this approach is that rating a song requires several steps: an initial click on the song, moving the mouse over the rate menu item, then clicking either the thumbs up or thumbs down option. If rating songs was an important activity, the extra effort might prevent some users from doing so. An alternate approach would be to replace ―rate directly with the thumbs up and the thumbs down options.

**Activation**

Another interesting decision Songza made was to not activate the radial menu on hover. Instead, the user must click on a song to reveal the menu. Activating on click makes the user intent more explicit. Making activation more explicit avoids the issues described earlier in the Hover and Cover anti-pattern. The user has chosen to interact with the song.

Conversely, with a mouse hover, it‗s never quite clear if the user meant to activate the menu or just happened to pause over a song title.

**Default action**

Playing a song requires moving to the top leaf. One possible solution would be to place the play option in the middle of the menu (at the stem) instead of in one of the leaves. Clicking once would activate the menu. Clicking a second time (without moving the mouse) would start playing the song. This interaction is very similar to one commonly used in desktop application: allowing a double-click to activate the first item (default action) in a right-click menu.

**5.4 OVERLAYS**

Overlays are really just lightweight pop ups. We use the term lightweight to make a clear distinction between it and the normal idea of a browser pop up. Browser pop ups are created as a new browser window Lightweight overlays are shown within the browser page as an overlay. Older style browser pop ups are undesirable because: Browser pop ups display a new browser window. As a result • these windows often take time and a sizeable chunk of system resources to create. Browser pop ups often display browser interface controls (e.g., a URL bar). Due to security concerns, in Internet Explorer 7 the URL bar is a permanent fixture on any browser pop-up window.



By using either Flash or Ajax-style techniques (Dynamic HTML), a web application can present a pop up in a lightweight overlay within the page itself. This has distinct advantages: Lightweight overlays are just a lightweight in-page object. They are inexpensive to create and fast to display. The interface for lightweight overlays is controlled by the •web application and not the browser. There is complete control over the visual style for the overlay. This allows the overlay to be more visually integrated into the application‘s interface



We will look at three specific types of overlays: Dialog Overlays, Detail Overlays, and Input Overlays.

**5.4.1 Dialog Overlay**

Dialog Overlays replace the old style browser pop ups. Netflix provides a clear example of a very simple Dialog Overlay. In the ―previously viewed movies for sale‖ section, a user can click on a ―Buy‖ button to purchase a DVD. Since the customer purchasing the DVD is a member of Netflix, all the pertinent shipping and purchasing information is already on record. The complete checkout experience can be provided in a single overlay

**5.4.2 Detail Overlay**

The second type of overlay is somewhat new to web applications. The Detail Overlay allows an overlay to present additional information when the user clicks or hovers over a link or section of content. Toolkits now make it easier to create overlays across different browsers and to request additional information from the server without refreshing the page. Taking another example from Netflix, information about a specific movie is displayed as the user hovers over the movie‘s box shot.

**5.4.3 Input Overlay**

Input Overlay is a lightweight overlay that brings additional input information for each field tabbed into. American Express uses this technique in its registration for premium card such as its gold card.

**5.5 INLAYS**

Information, or dialog with the user needs to be an overlay. Another approach is to inlay the information directly within the page itself. To distinguish from the pop-up overlay, we call these in-page panels Inlays.

**5.5.1 Dialog Inlay**

A simple technique is to expand a part of the page, revealing a dialog area within the page. The BBC recently began experimenting with using a Dialog Inlay as a way to reveal customization controls for its home page.

**5.5.2 List Inlay**

Lists are a great place to use Inlays. Instead of requiring the user to navigate to a new page for an item‘s detail or popping up the information in an Overlay, the information can be shown with a List Inlay in context. The List Inlay works as an effective way to hide detail until needed—while at the same time preserving space on the page for high-level overview information. Google Reader provides an expanded view and a list view for unread blog articles. In the list view, an individual article can be expanded in place as a List Inlay

**5.5.3 Detail Inlay**

A common idiom is to provide additional detail about items shown on a page. We saw this with the example of the Netflix movie detail pop up . Hovering over a movie revealed a Detail Overlay calling out the back-of-the-box information. Details can be shown inline as well. Roost allows house photos to be viewed in-context for a real estate listing with a Detail Inlay

**Inlay Versus Overlay?**

Use an overlay when there may be more than one place a • dialog can be activated from (the exception may be showing details for items in a list).

* + - * Use an overlay to interrupt the process.
      * Use an overlay if there is a multi-step process.
      * Use an inlay when you are trying to avoid covering information on the page needed

in the dialog.

* + - * Use an inlay for contextual information or details about one of many items (as in a list): a typical example is expanding list items to show detail.

**5.6 Virtual pages**

Overlays allow you to bring additional interactions or content in a layer above the current page. Inlays allow you to do this within the page itself. However, another powerful approach to keeping users engaged on the current page is to create a virtual page. That is to say, we create the illusion of a larger virtual page.

Patterns that support virtual pages include:

* + - * Virtual Scrolling
      * Inline Paging
      * Scrolled Paging
      * Panning
      * Zoomable User Interface

**5.6.1 Virtual Scrolling**

The traditional Web is defined by the ―page.‖ In practically every implementation of websites (for about the first 10 years of the Web‘s existence) pagination was the key way to get to additional content. Of course, websites could preload data and allow the user to scroll through it. However, this process led to long delays in loading the page. So most sites kept it simple: go fetch 10 items and display them as a page and let the user request the next page of content. Each fetch resulted in a page refresh. The classic example of this is Google Search. Each page shows 10 results. Moving through the content uses the now-famous

Google pagination control

Virtual Scrolling demonstrate three different ways to manage the virtual space:

* + - Yahoo! Mail creates the illusion that all data has been loaded up-front by having the scrollbar reflect the total virtual space.
    - Microsoft Live Search creates the virtual illusion as the user moves down through the search results.
    - And PicLens does the same with the caveat that it • shows a virtual window in the larger virtual space (by only providing a scroller control for where the user is and some before and after context).

**5.6.2 Inline Paging**

What if instead of scrolling through content we just wanted to make pagination feel less like a page switch? By only switching the content in and leaving the rest of the page stable, we can create an Inline Paging experience. This is what Amazon‘s Endless.com site does with its search results.

**5.6.3 Scrolled Paging**

Besides Virtual Scrolling and Virtual Paging, there is another option. You can combine both scrolling and paging into Scrolled Paging. Paging is performed as normal. But instead the content is ―scrolled‖ into view.

The Carousel pattern takes this approach. A Carousel provides a way to page-in more data by scrolling it into view. On one hand it is a variation on the Virtual Scrolling pattern. In other ways it is like Virtual Paging since most carousels have paging controls. The additional effect is to animate the scrolled content into view. Yahoo! Underground uses a Carousel to provide a way to page/scroll through articles.

**5.6.4 Virtual Panning**

One way to create a virtual canvas is to allow users the freedom to roam in two- dimensional space. A great place for Virtual Panning is on a map. Google Maps allows you to pan in any direction by clicking the mouse down and dragging the map around

**5.6.5 Zoomable user interface**

A Zoomable User Interface (ZUI) is another way to create a virtual canvas. Unlike panning or flicking through a flat, two-dimensional space, a ZUI allows the user to also zoom in to elements on the page. This freedom of motion in both 2D and 3D supports the concept of an infinite interface.Practically speaking, ZUIs have rarely been available in everyday software applications, much less on the Web. But with more advanced features added to Flash and the advent of Silverlight, this type of interface is starting to emerge and may be commonplace in the not-too-distant future.

**5.6.6 Paging Vs Scrolling**

Leading web designers and companies have taken different approaches to solving the same problems. Yahoo! Mail chose Virtual Scrolling. Gmail chose Inline Paging. How do you choose between paging and scrolling? While there are no hard and fast rules, here are some things to consider when making the decision:

* When the data feels ―more owned‖ by the user—in other words, the data is not transient but something users want to interact with in various ways. If they want to sort it, filter it, and so on, consider Virtual Scrolling (as in Yahoo! Mail).
* When the data is more transient (as in search results) and will get less and less relevant the further users go in the data, Inline Paging works well (as with the iPhone).
* For transient data, if you don‘t care about jumping around in the data to specific sections, Consider using Virtual Scrolling (as in Live Image Search).
* If you are concerned about scalability and performance, paging is usually the best choice. Originally Microsoft‘s Live Web Search also provided a scrollbar. However, the scrollbar increased server-load considerably since users are more likely to scroll than page.
* If the content is really continuous, scrolling is more natural than paging.
* If you get your revenue by page impressions, scrolling may not be an option for your business model.
* If paging causes actions for the content to become cumbersome, move to a scrolling model. This is an issue in Gmail. The user can only operate on the current page.

Changing items across page boundaries is unexpected. Changing items in a continuous scrolled list is intuitive.

**5.7 Process flow**

It has long been common practice on the Web to turn each step into a separate page. While this may be the simplest way break down the problem, it may not lead to the best solution. For some Process Flows it makes sense to keep the user on the same page throughout the process.

**5.7.1 Google blogger**

The popular site Google Blogger generally makes it easy to create and publish blogs. One thing it does not make easy, though, is deleting comments that others may leave on your blog. This is especially difficult when you are the victim of hundreds of spam comments left by nefarious companies hoping to increase their search ranking. Blogger forces you to delete these comments through a three-step process. Each step is an individual page, all punctuated with a page refresh.

My (Bill‘s) blog site was recently spammed. It turns out that my 100 or so articles all had 4 or more spam comments. That means that I had to delete more than 400 spam comments.

Given the way Google Blogger implemented comment deleting, I had to follow these steps for each comment on each blog article:

* + - 1. Scroll to find the offending comment.
      2. Click the trash icon to delete the comment.
      3. After page refreshes, click the ―Remove Forever‖ checkbox.
      4. Click the ―Delete Comment‖ button.
      5. After the page refreshes, click the link to return to my blog article.
      6. Repeat steps 1–5 for each article with spam comments.

It took 1,600 clicks, 1,200 page refreshes, 400 scroll operations, and several hours to finally rid myself of all of the spam comments. If the delete action could have been completed in the same page as the comments, that would have eliminated hundreds of clicks and well over a thousand page refreshes, and scrolling would have been all but eliminated. I would not have wasted all the mental energy to reorient myself after each page transition. And I would have been a much happier man. This is a common interaction flow on the Web. It turns out to be simpler to design and implement a process as a series of pages rather than a single interactive space.

**5.7.2 The magic principle**

Alan Cooper discusses a wonderful technique for getting away from a technologydriven approach and discovering the underlying mental model of the user. He calls it the ―magic principle.‖ Ask the question, ―What if when trying to complete a task the user could invoke some magic?‖ For example, let‘s look at the problem of taking and sharing photos.

The process for this task breaks down like this:

* Take pictures with a digital camera.
* Sometime later, upload the photos to a photo site like Flickr. This involves:

— Finding the cable.

— Starting iTunes.

— Importing all photos.

Using a second program, such as Flickr Uploadr, to upload the photos to Flickr.

- Copying the link for a Flickr set (which involves first locating the page for the uploaded set). Send the link in email to appropriate friends. If some magic were invoked, here is how it might happen:

* The camera would be event-aware. It would know that is your daughter‘s eighth birthday.
* When finished taking pictures of the event, the camera would upload the pictures to Flickr.
* Flickr would notify family and friends that the pictures of the birthday party are available.

**5.7.3 Interactive single page processor**

Consumer products come in a variety of shapes, sizes, textures, colors, etc. Online shoppers will not only have to decide that they want shoes, but do they want blue suede shoes? And what size and width do they want them in? In the end the selection is constrained by the available inventory. As the user makes decisions, the set of choices gets more and more limited. This type of product selection is typically handled with a multi-page workflow. On one page, the user selects a shirt and its color and size. After submitting the choice, a new page is displayed. Only when the user arrives at this second page does he find out that the ―true navy‖ shirt is not available in the medium size.

**5.7.4 Inline assistant process**

Another common place where multiple pages are used to complete a process is when adding items to a shopping cart. As mentioned earlier, Amazon provides the typical experience So what magic can we apply to move this from a multi-page experience to a single-page experience? Instead of thinking about the cart as a process, we can think about it as a real-world object. Given this mindset, the cart can be realized in the interface as an object and be made available on the page. The Gap employed an Inline Assistant Process pattern for its shopping cart when it re-launched its site a few years back.

**5.7.5 Dialog overlay process**

As mentioned before, any page switch is an interruption to the user‘s mental flow. In addition, any context switch is a chance for a user to leave the site. We seek an experience that has as little mental friction as possible. But sometimes the step-by-step flow is necessary. The Netflix approach just described uses a Dialog Overlay Process to encapsulate a multi- step flow inside a Dialog Overlay.. Overlays allow us to keep the context of the page yet present a virtual space to conduct a conversation with the user. Discover.com recently expanded its account section with a more detailed profile. The profile captures things like your payment date, mobile fraud alerts, paperless statements, and general contact information .The overlay pops up when you first enter your account.

**5.7.6 Configuration process**

Sometimes a Process Flow is meant to invoke delight. In these cases, it is the engagement factor that becomes most important. This is true with various Configurator Process interfaces on the Web. We can see this especially at play with car configurators. Porsche provides a configurator that allows users to build their own Porsche Being able to tweak the colors, wheels, interior, and options for a car and see the results in real time is an engaging experience. Live Previews allow the user to see the effect of his changes on a simulated version of the real thing.

**5.7.7 Static single page process**

The Apple example illustrates another way to get rid of multiple pages in a Process Flow. Just put the complete flow on one page in a Static Single-Page Process. The user sees all the tasks needed to complete the full process. This can be both good and bad. Seeing just one step to complete the process can encourage users to finish the task. But if the single step seems too long or too confusing, the user will most likely bail out of the process early. In other words, if placing all the tasks on a single page is enough to cause the user to bail out; it is not a good idea. In the case of the Apple store, each item is optionally set, and it‘s just a single click to include or exclude an item from the purchase. EBay provides two ways to sell an item. An introductory panel gathers the description of the item for sale. The ―Customize your listings…‖ option takes the user through a traditional multi-page process.

**References**

1. Bill Scott and Theresa Neil, “Designing Web Interfaces”, First Edition, O’Reilly, 2009