1. A persons interaction with the outside world occurs through information being received and sent input and output. In an interaction with a computer the user receives information that is output by the computer, and responds by providing input to the computer the users output becomes the computers input and vice versa. Input in the human occurs mainly through the senses and output through the motor control of the effectors.

2. Sensory buffers

Short-term memory or working memory

Long-term memory

3. Design is defined as achieving Goals within constraints and encompasses work tasks data design, architectural design, interface design and component-level design and create a design model or design specification.

4. Requirements , Analysis and Design , Iteration and prototyping , Implementation and Deployment

5. A Cognitive model is the designers intended mental model for the user of the system a set of ideas about how it is organized and operates.

6. Analyze and design user interfaces and new user-interface technologies, created software tools and development environment to facilitate the construction of graphical user interfaces, pioneered the user of voice and video in user interfaces, hypertext links, interactive tutorials and context sensitive help systems.

7. Services , Applications , Application Frameworks , Operating Systems , Platforms , Devices , Aggregators , Networks , Operators

8. Services include tasks such as accessing the Internet, sending a text message, or being able to get a location basically, anything the user is trying to do.

9. The grid is a handy tool for planning out interesting moments during a drag and drop interaction. It serves as a checklist to make sure there are no holes in the interaction.

10. Placeholder targeting - Most explicit way to preview the effect. Midpoint boundary - Requires the least drag effort to move modules around.

11. b. . It is the information that explains why a computer system is the way it is , including its structural or architectural description and its functional or behavioral description. In this sense, design rationale does not fit squarely into the software life cycle described in this chapter as just another phase or box. Rather, design rationale relates to an activity of both reflection (doing design rationale) and documentation (creating a design rationale) that occurs throughout the entire life cycle. It is beneficial to have access to the design rationale for several reasons In an explicit form, a design rationale provides a communication mechanism among the members of a design team so that during later stages of design and/or maintenance it is possible to understand what critical decisions were made, what alternatives were investigated (and, possibly, in what order) and the reason why one alternative was chosen over the others. This can help avoid incorrect assumptions later. Accumulated knowledge in the form of design rationales for a set of products can be reused to transfer what has worked in one situation to another situation which has similar needs. The design rationale can capture the context of a design decision in order that a different design team can determine if a similar rationale is appropriate for their product. The effort required to produce a design rationale forces the designer to deliberate more carefully about design decisions. The process of deliberation can be assisted by the design rationale technique by suggesting how arguments justifying or discarding a particular design option are formed. In the area of HCI, design rationale has been particularly important, again for several reasons There is usually no single best design alternative. More often, the designer is faced with a set of trade-offs between different alternatives. For example, a graphical interface may involve a set of actions that the user can invoke by use of the mouse and the designer must decide whether to present each action as a button on the screen, which is always visible, or hide all of the actions in a menu which must be explicitly invoked before an action can be chosen. The former option maximizes the operation visibility but the latter option takes up less screen space. It would be up to the designer to determine which criterion for evaluating the options was more important and then communicating that information in a

design rationale. Even if an optimal solution did exist for a given design decision, the space of alternatives is so vast that it is unlikely a designer would discover it. In this case, it is important that the designer indicates all alternatives that have been investigated. Then later on it can be determined if she has not considered the best solution or had thought about it and discarded it for some reason. In project management, this kind of accountability for design is good. The usability of an interactive system is very dependent on the context of its use. The flashiest graphical interface is of no use if the end-user does not have access to a high-quality graphics display or a pointing device. Capturing the context in which a design decision is made will help later when new products are designed. If the context remains the same, then the old rationale can be adopted without revision. If the context has changed somehow, the old rationale can be reexamined to see if any rejected alternatives are now more favorable or if any new alternatives are now possible. There are 3 types Process-oriented design rationale Hierarchical structure to a design rationale is created. A root issue is identified which represents the main problem or question that the argument is addressing. Various positions are put forth as potential resolutions for the root issue, and these are depicted as descendants in the IBIS hierarchy directly connected to the root issue. Each position is then supported or refuted by arguments, which modify the relationship between issue and position. The hierarchy grows as secondary issues are raised which modify the root issue in some way. Each of these secondary issues is in turn expanded by positions and arguments, further sub-issues, and so on. A graphical version of IBIS has been defined by Conklin and Yakemovic called IBIS (pronounced gibbiss), which makes the structure of the design rationale more apparent visually in the form of a directed graph which can be directly edited by the creator of the design rationale. Issues, positions and arguments are nodes in the graph and the connections between them are labeled to clarify the relationship between adjacent nodes. So, for example, an issue can suggest further sub-issues, or a position can respond to an issue or an argument can support a position. The gIBIS structure can be supported by a hypertext tool to allow a designer to create and browse various parts of the design rationale Design space analysis The design space is initially structured by a set of questions representing the major issues of the design. Since design space analysis is structure oriented, it is not so important that the questions recorded are the actual questions asked during design meetings. Rather, these questions represent an agreed characterization of the issues raised based on reflection and understanding of the actual design activities. Questions in a design space analysis are therefore similar to issues in IBIS except in the way they are captured. Options provide alternative solutions to the question. They are assessed according to some criteria in order to determine the most favorable option. The key to an effective design space analysis using the QOC notation is deciding the right questions to use to structure the space and the correct criteria to judge the options. The initial questions raised must be sufficiently general that they cover a large enough portion of the possible design space, but specific enough that a range of options can be clearly identified. It can be difficult to decide the right set of criteria with which to assess the options. Another structure-oriented technique, called Decision Representation Language (DRL), developed by Lee and Lai, structures the design space in a similar fashion to QOC, though its language is somewhat larger and it has a formal semantics. The questions, options and criteria in DRL are given the names decision problem, alternatives and goals. QOC assessments are represented in DRL by a more complex language for relating goals to alternatives. The sparse language in QOC used to assess an option relative to a criterion (positive or negative assessment only) is probably insufficient, but there is a trade-off involved in adopting a more complex vocabulary which may prove too difficult to use in practice The advantage of the formal semantics of DRL is that the design rationale can be used as a computational mechanism to help manage the large volume of information. For example, DRL can track the dependencies between different decision problems, so that subsequent changes to the design rationale for one decision problem can be automatically propagated to other dependent problems Psychological design rationale This psychological design rationale has been introduced by Carroll and Rosson , and before we describe the application of the technique it is important to understand some of its theoretical background. People use computers to accomplish some tasks in their particular work domain, as we have seen before. When designing a new interactive system, the designers take into account the tasks that users currently perform and any new ones that they may want to perform. This task identification serves as part of the requirements for the new system, and can be done through empirical observation of how people perform their work currently and presented through informal language or a more formal task analysis language. When the new system is implemented, or becomes an artifact, further observation reveals that in addition to the required tasks it was built to support, it also supports users in tasks that the designer never intended. Once designers understand these new tasks, and the associated problems that arise between them and the previously known tasks, the new task definitions can serve as requirements for future artifacts The purpose of psychological design rationale is to support this natural task artifact cycle of design activity. The main emphasis is not to capture the designers intention in building the artifact. Rather, psychological design rationale aims to make explicit the consequences of a design for the user, given an understanding of what tasks he intends to perform. Previously, these psychological consequences were left implicit in the design, though designers would make informal claims about their systems (for example, that it is more natural for the user, or easier to learn). The first step in the psychological design rationale is to identify the tasks that the proposed system will address and to characterize those tasks by questions that the user tries to answer in accomplishing them. The main task the system is to support is learning how Smalltalk works. In learning about the programming environment, the programmer will perform tasks that help her answer the questions What can I do that is, what are the possible operations or functions that this programming environment allows How does it work that is, what do the various functions do How can I do this that is, once I know a particular operation I want to perform, how do I go about programming it

12. b. Requirements what is wanted the first stage is establishing what exactly is needed. As a precursor to this it is usually necessary to find out what is currently happening. For example, how do people currently watch movies What sort of personal appliances do they currently use There are a number of techniques used for this in HCI interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly. Analysis-The results of observation and interview need to be ordered in some way to bring out

key issues and communicate with later stages of design models, which are a means to capture how people carry out the various tasks that are part of their work and life. we will look at scenarios, rich stories of interaction, which can be used in conjunction with a method like task analysis or on their own to record and make vivid actual interaction. These techniques can be used both to represent the situation as it is and also the desired situation. Design-Well, this is all about design, but there is a central stage when you move from what you want, to how to do it. There are numerous rules, guidelines and design principles. We need to record our design choices in some way and there are various notations and methods to do this, including those used to record the existing situation Iteration and prototyping Humans are complex and we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements.. Some forms of evaluation can be done using the design on paper, but it is hard to get real feedback without trying it out. Most user interface design therefore involves some form of prototyping, producing early versions of systems to try out with real users. Implementation and deployment Finally, when we are happy with our design, we need to create it and deploy it. This will involve writing code, perhaps making hardware, writing documentation and manuals everything that goes into a real 5.4 User focus 197 system that can be given to others Golden rule of Design. Part of the understanding we need is about the circumstances and context of the particular design problem. However, there are also more generic concepts to understand. The designs we produce may be different, but often the raw materials are the same. This leads us to the golden rule of design Interaction design basics In the case of a physical design this is obvious. Look at a chair with a steel frame and one with a wooden frame. They are very different often the steel frames are tubular or thin L or H section steel. In contrast wooden chairs have thicker solid legs. If you made a wooden chair using the design for a metal one it would break; if you made the metal one in the design for the wooden one it would be too heavy to move. For Human Computer Interaction the obvious materials are the human and the computer. That is we must n understand computers limitations, capacities, tools, platforms n understand people psychological, social aspects, human error. e must understand the fundamental materials of humancomputer interaction in order to design areas. For example, the way you fit seats and windows into an airplanes hull affects the safety and strength of the aircraft as a whole.

13. a. The organizational issues that affect the acceptance and relevance of information and communication systems. These factors often sit outside the system as such, and may involve individuals who never use it. 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. 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. However, the official lines of authority and information tend to flow up and down through line management. New communications media may challenge and disrupt these formal managerial structures. The physical layout of an organization often reflects the formal hierarchy An email system has no such barriers; it is as easy to chat to someone in another department as in your own. Face-to-face conversation, the manager can easily exert influence over a subordinate Technology can be an important vector of social change, but if violent reaction is to be avoided, the impact of the technology must be assessed before it is introduced. In the short term, solutions must be carefully matched to the existing social and organizational structures. 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. Alternatively, distributed groupware can allow the true home-based teleworker to operate on similar terms to an office-based equivalent. The ecological and economic advantages of such working practices are now becoming well established, and it seems that communications and CSCW technology can overcome many of the traditional barriers. Free rider problem. In

economics, the free rider problem occurs when those who benefit from resources, goods, or ervices do not pay for them, which results in an under-provision of those goods or services. The free rider problem is the question of how to limit free riding and its negative effects in these situations. The free rider problem may occur when property rights are not clearly defined and imposed. The free rider problem is common among public goods. These are goods that have two characteristics non-excludability non-paying consumers cannot be prevented from using it and non-rivalry when you consume the good, it does not reduce the amount available to others. The potential for free riding exists when people are asked to voluntarily pay for a public good. Critical mass A critical mass is the smallest amount of fissile material needed for a sustained nuclear chain reaction. The critical mass of a fissionable material depends upon its nuclear properties (specifically, the nuclear fission cross-section), its density, its shape, its enrichment, its purity, its temperature, and its surroundings. The concept is important in nuclear weapon design. 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. 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. Evaluating the benefits We have seen several problems that can arise from the mismatch between information systems and organizational and social factors. 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. The benefits are difficult to quantify, but, over time, it has become clear that the competitive edge of information technology is necessary for survival in the modern world.

14. b. . Application Context Once the application medium is decided upon, it is time to look at the application context, or the appropriate type of application to present to the user in order for the user to process and understand the information presented and complete her goals. Where the application medium

refers mostly to the technical approach of creating an application, the application context deals with the user experience. Actually context is the surroundings in which information is processed, and the application user experience is no different. Applications can be presented in a variety of ways, ranging from a simple task-based utility to an experience meant to consume the users focus and attention. There of course is no right or wrong direction only what is best for your user. In fact, nothing says that you cant use multiple application contexts within the same application. it is best to present only one application context so as to avoid confusing the user. If you think it best for your app to mix contexts, then give the user the option to switch between them; for example, some smartphones allow for an orientation change, so if the device is rotated to landscape mode, your app switches from an informative view to a utility view, or maybe from a locale view to an immersive view.

Utility Context The most basic application context is the utility, or a simple user experience metaphor that is meant to address short, task-based scenarios. Information is meant to be presented in a minimal fashion, often using the least amount of user input as possible. An example of a utility might be a calculator, weather forecast, unit conversion, stocks, world clock, and so on. In each of these cases, the user enters a small amount of information into the utility, like a simple equation, a city, or a stock symbol, and either by performing a small background task or fetching information online, the utility is able to present data to the user in the desired context. The goal of the utility is to give users at-aglance information, therefore offering users a minimal design aesthetic, focusing the design around the content in view, and often using larger type and a sparse layout.It would be easy to mistake utilities for widgets, given that widgets are a component of a user interface that operates in a particular way But utilities can be much more than widgets; they are not merely an extension of the user experience, but are applications in their own right that can establish their own look and feel separate from the established user experience. Use utilities for short, simple tasks, at-a-glance information, when there is limited content to display, and when combined with an immersive context to create dual-mode applications.

Locale Context The locale context is a newer application type that is still being defined by the mobile community, but we are certainly seeing some clear patterns of how to create locale applications. As more location information is being published online, and more devices add GPS to pinpoint the users location, locale is becoming an excellent data point to pivot information around. For example, I can use location to display the Cafes nearest to my current location.
Locale applications almost always have at least one thing in common a map on which they plot the requested data points visually. At the very least, they list items in order of distance, with the nearest item first and the farthest last. The users goal is to find information relative to his present location, and content should always be designed with this in mind. When creating locale apps, it is important to ensure that the users present location is always clearly identified, as well as a means of adding data to it. This could be another location, in the case of finding point-to-point directions, or it could be a keyword query to find people, places, or things nearby. Use locale applications for location-based applications, applications that might contain a dynamic map, and listing multiple location-based points of interest. d. Informative Applications The informative application is an application context in which the one and only goal is to provide information, like a news site, an online directory, a marketing site, or even a mobile commerce site, where the key task of the user is to read and understand and it is not necessary to interact. For example, remember that most mobile tasks are short and are often undertaken during brief idle periods. The user doesnt have much extra time and the task can be interrupted at any moment. In the case of a mobile news site, provide the user with the option to mark a page or story to be read later. With an online directory, allow the user to flag favorite entries. With a marketing site, allow users to enter the shortest possible contact information, like their phone number or email. And with a mobile commerce site, allow users to save items to a wishlist to review and purchase later. The theme here is that although reading information is a simple task, it usually creates a complex chain of events that can be anticipated. With mobile applications, we want to avoid forcing the users to input too much information with their mobile devices, which is more difficult and takes more time than it would on another medium such as a desktop or laptop

computers. Instead, we want to look for ways we can interconnect experiences, having users use the informative context to filter to the most desirable information when they have a moment, and allowing them to interact with it later, when they have more time, from the medium of their choice.
Use informative applications when users need information to gain an understanding or perform a physical task, for applications that can be used in multimedia contexts such as desktop and mobile, for information-heavy applications, and for marketing or promotional applications.

15. b. 6 principles for Designing Rich Web Experiences Make it right Allow input wherever you have output Shorten the length of interaction Make objects directly actionable Keep it lightweight Stay on the page Offer an invitation Show transitions React immediately Good example of inline editings are discoverability, complex editing and blending modes. Group editing is based on symmetry of interaction and discoverability vs readability. Overlays Dialog overlay Detail overlay Input overlay Considerations The Apple technique signifies that we have entered a special editing mode. When the icons become wiggly, it is not a large intuitive leap that the icons have become loose and thus we can rearrange them. Discoverability Admittedly, the feature is not very discoverable. But it can be argued that it is straightforward once discovered. However, pressing the home button deactivates the rearranging mode. This really should operate more like a toggle. A better way to exit the wiggly mode would be to press and hold down on a wiggly icon. It follows the idea that you are pushing the icon back into its fixed place. Since deactivation is not the same mechanism as activation, it is a little hard to figure out how to go back into the normal display mode. Visual noise Putting edit links on each module can be visually noisy. An alternative approach is to use the Group Edit pattern to place an edit link at the page level that turns on edit links for each module. When the Done Editing link is clicked, the links for each module are hidden. Again the trade-off is between visual noise and discoverability. 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. 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 a simple hole can be exposed where the object will be placed when dropped. A good example from the desktop world is Apples iPhoto. In a slideshow, you can easily rearrange the order of photos with drag and drop. Dragging the photo left or right causes the other photos to shuffle open a drop spot. Collected Selection and actions When Yahoo! Photos was working its way through an early design of its Photo Gallery the plan was to show all photos in a single, continuous scrolling page In a long virtual list, the selection model is simple. Photos are shown in a single page and selection is easily understood in the context of this single page. However, due to performance issues, the design was changed. Instead of a virtual page, photos had to be chunked into pages. In order to support Collected Selection, Yahoo! Photos introduced the concept of a tray into the interface. On any page, photos can be dragged into the tray. The tray keeps its contents as the user moves from page to page. So, adding a photo from page one and three more from page four would yield four items in the tray. As a nice touch, the tray would make itself visible (by sliding into view) even when the user was scrolled down below the fold.

16. b. One possible GOMS description of the goal hierarchy for this task is given below. Answers will vary depending on assumptions about the photocopier used as the model for the exercise. In this example, we will assume that the article is

to be copied one page at a time and that a cover over the imaging surface of the copier has to be in place before the actual copy can be made. GOAL PHOTOCOPY-PAPER . GOAL LOCATE-ARTICLE . GOAL PHOTOCOPY-PAGE repeat until no more pages . . GOAL ORIENT-PAGE . . . OPEN-COVER . . . SELECT-PAGE . . . POSITION-PAGE . . . CLOSE-COVER . . GOAL PRESS-COPY-BUTTON . . GOAL VERIFY-COPY . . . LOCATE-OUT-TRAY . . . EXAMINE-COPY . GOAL COLLECT-COPY . . LOCATE-OUT-TRAY . . REMOVE-COPY (outer goal satisfied!) . GOAL RETRIEVE-JOURNAL . . OPEN-COVER . . REMOVE-JOURNAL . . CLOSE-COVER The closure problem which appears in this example occurs when the copy of the article is removed from the photocopier out tray, satisfying the overall goal for the task. In the above description, however, the original journal article is still on the imaging surface of the photocopier, and the cover is closed. The user could easily forget to remove the journal. How could the photocopying procedure be revised to eliminate this problem One answer is to force the goal RETRIEVE-JOURNAL to be satisfied before COLLECT-COPY.

Buxton has developed a simple model of input devices , the three-state model, which captures some of these crucial distinctions. He begins by looking at a mouse. If you move it with no buttons pushed, it normally moves the mouse cursor about. This tracking behavior is termed state 1. Depressing a button over an icon and then moving the mouse will often result in an object being dragged about. This he calls state 2 . If instead we consider a light pen with a button, it behaves just like a mouse when it is touching the screen. When its button is not depressed, it is in state 1, and when its button is down, state 2. However, the light pen has a third state, when the light pen is not touching the screen. In this state the system cannot track the light pens position. This is called state 0 . A touchscreen is like the light pen with no button. While the user is not touching the screen, the system cannot track the finger that is, state 0 again. When the user touches the screen, the system can begin to track state 1. So a touchscreen is a state 01 device whereas a mouse is a state 12 device. As there is no difference between a state 02 and a state 01 device, there are only the three possibilities we have seen. The only additional complexity is if the device has several buttons, in which case we would have one state for each button 2left, 2middle, 2right. One use of this classification is to look at different pointing tasks, such as icon selection or line drawing, and see what state 012 behavior they require. At first, the model appears to characterize the states of the device by the inputs available to the system. So, from this perspective, state 0 is clearly different from states 1 and 2. However, if we look at the state 12 transaction, we see that it is symmetric with respect to the two states. In principle, there is no reason why a program should not decide to do simple mouse tracking whilst in state 2 and drag things about in state 1. That is, there is no reason until you want to type something! The way we can tell state 1 from state 2 is by the activity of the user. State 2 requires a button to be pressed, whereas state 1 is one of relative relaxation (whilst still requiring handeye coordination for mouse movement). There is a similar difference in tension between state 0 and state 1. It is well known that Fitts law has different timing constants for different devices. Recall that Fitts law says that the time taken to move to a target of size S at a distance D is a +b log2(D/S +1) The constants a and b depend on the particular pointing device used and the skill of the user with that device. However, the insight given by the three-state model is that these constants also depend on the device state. In addition to the timing, the final accuracy may be affected. Experiments to calculate Fitts law constants in states 1 and 2 have shown that these differences do exist.