PROTOTYPING

DR. KRISHNENDU GUHA

ASSISTANT PROFESSOR (ON CONTRACT)

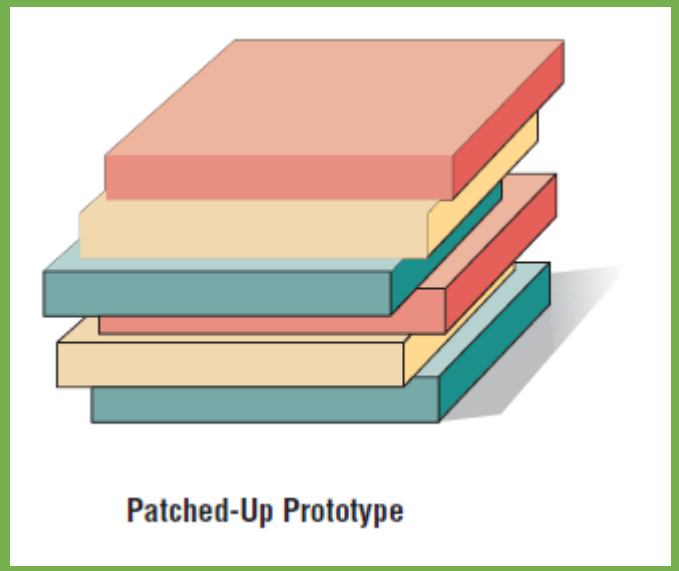
NATIONAL INSTITUTE OF TECHNOLOGY (NIT), JAMSHEDPUR

EMAIL: KRISHNENDU.CA@NITJSR.AC.IN

INTRODUCTION

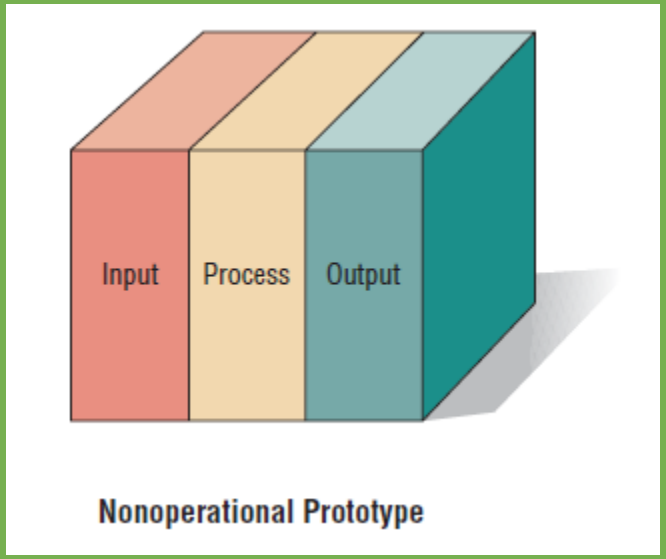
- As the systems analyst presenting a prototype of the information system, we are keenly interested in the reactions of users and management to the prototype.
- We want to know in detail how they react to working with the prototype and how good the fit is between their needs and the prototyped features of the system.
- Reactions are gathered through observation, interviews, and feedback sheets (possibly questionnaires)
- Information gathered in the prototyping phase allows the analyst to set priorities and redirect plans inexpensively, with a minimum of disruption.
- Because of this feature, prototyping and planning go hand-in-hand.

KINDS OF PROTOTYPES



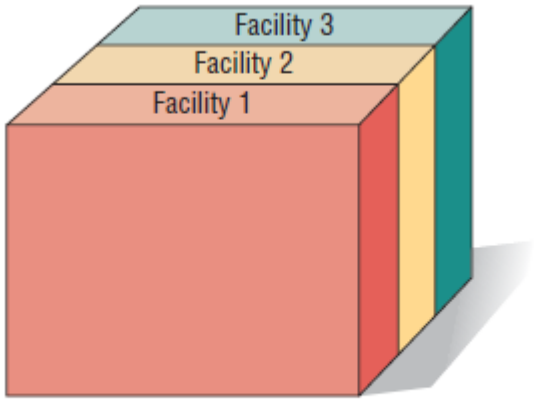
- **PATCHED-UP PROTOTYPE.**
- The first kind of prototyping has to do with constructing a system that works but is patched up or patched together.
- In engineering this approach is referred to as breadboarding: creating a patched-together, working model of an (otherwise microscopic) integrated circuit.

- An example in information systems is a working model that has all the necessary features but is inefficient.
- In this instance of prototyping, users can interact with the system, getting accustomed to the interface and types of output available.
- The retrieval and storage of information may be inefficient, however, because programs were written rapidly with the objective of being workable rather than efficient.



Nonoperational Prototype

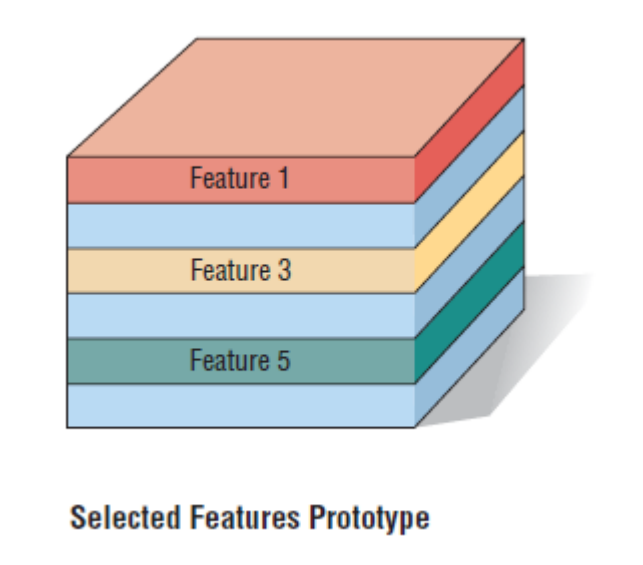
- **NON-OPERATIONAL PROTOTYPE.**
- The second conception of a prototype is that of a nonworking scale model that is set up to test certain aspects of the design.
- The size and shape of the auto are precise, but the car is not operational.
- In this case only features of the automobile essential to wind tunnel testing are included.
- A nonworking scale model of an information system might be produced when the coding required by the applications is too extensive to prototype but when a useful idea of the system can be gained through the prototyping of the input and output only.
- In this instance, processing, because of undue cost and time, would not be prototyped.
- Users could still make decisions on the utility of the system, based on their use of prototyped input and output.



First-of-a-Series Prototype

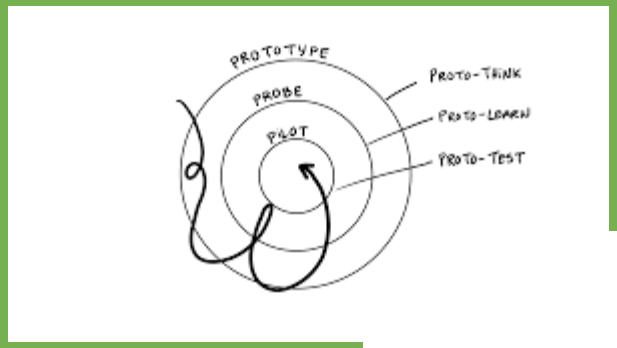
- **FIRST-OF-A-SERIES PROTOTYPE.**
- A third conception of prototyping involves creating a first fullscale model of a system, often called a pilot.
- An example is prototyping the first airplane of a series, then seeing if it flies before building a second.
- The prototype is completely operational and is a realization of what the designer hopes will be a series of airplanes with identical features.

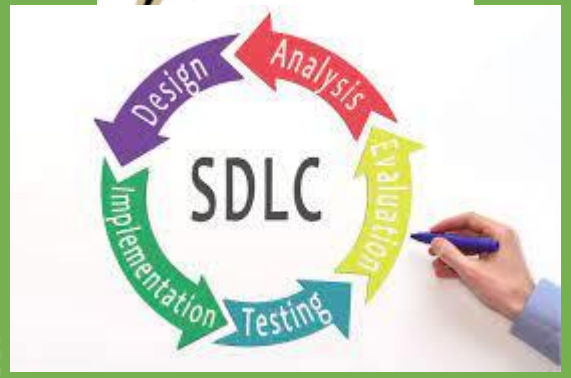
- This type of prototyping is useful when many installations of the same information system are planned.
- The full-scale working model allows users to experience realistic interaction with the new system, but it minimizes the cost of overcoming any problems that it presents.



- **SELECTED FEATURES PROTOTYPE.**
- A fourth conception of prototyping concerns building an operational model that includes some, but not all, of the features that the final system will have.
- When prototyping information systems in this way, some, but not all, essential features are included.
- For example, users may view a system menu on a screen that lists six features: add a record, update a record, delete a record, search a record for a key word, list a record, or scan a record. In the prototyped system, however, only three of the six may be available for use, so that the user may add a record (feature 1), delete a record (feature 3), and list a record (feature 5).
- User feedback can help analysts understand what is working and what isn't.
- It can also help with suggestions on what features to add next.
- When this kind of prototyping is done, the system is accomplished in modules so that if the features that are prototyped are evaluated by users as successful, they can be incorporated into the larger, final system without undertaking immense work in interfacing.
- Prototypes done in this manner are part of the actual system. They are *not* just a mock-up as in nonoperational prototyping considered previously.

PROTOTYPING AS AN ALTERNATIVE TO THE SDLC



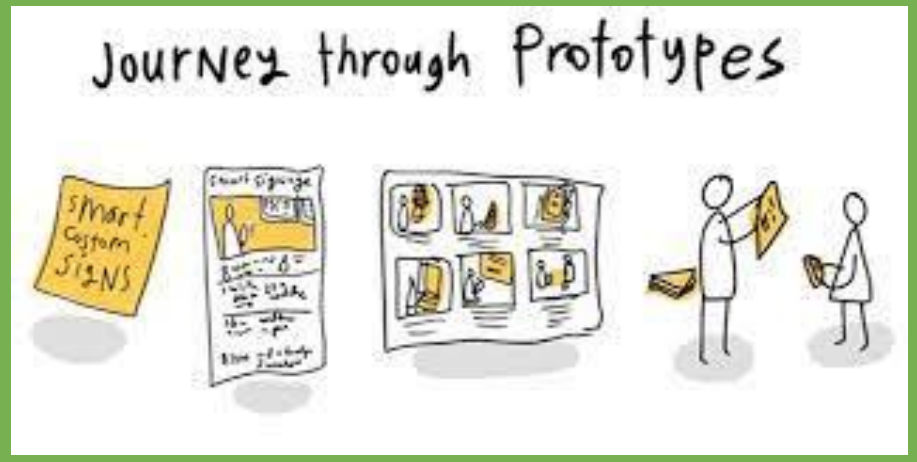


- The first concern is the extended time required to go through the development life cycle.
- As the investment of analyst time increases, the cost of the delivered system rises proportionately.

- The second concern about using the SDLC is that user requirements change over time.
- During the long interval between the time that user requirements are analyzed and the time that the finished system is delivered, user requirements are evolving.
- Thus, because of the extended development cycle, the resulting system may be criticized for inadequately addressing current user information requirements.

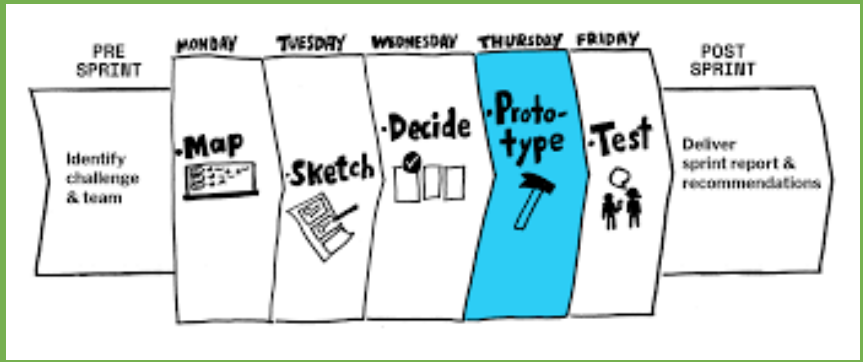
- To overcome these problems, some analysts propose that prototyping be used as an alternative to the SDLC.
- When prototyping is used in this way, the analyst effectively shortens the time between ascertainment of human information requirements and delivery of a workable system

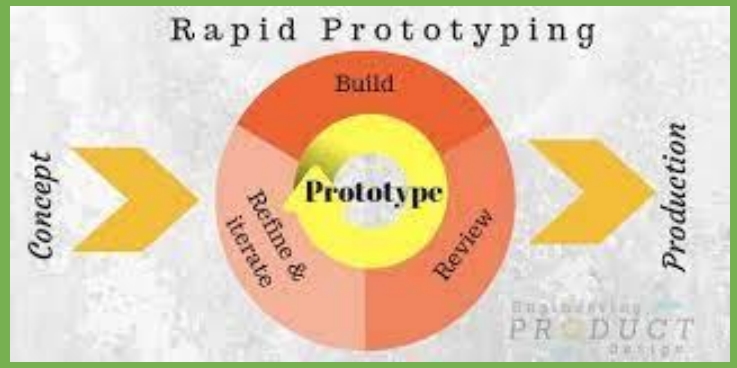
DEVELOPING A PROTOTYPE



- The first step of prototyping is to estimate the costs involved in building a module of the system.
- If costs of programmers' and analysts' time as well as equipment costs are within the budget, building of the prototype can proceed.

- **Guidelines for Developing a Prototype**
- Once the decision to prototype has been made, four main guidelines must be observed :
- **1.** Work in manageable modules.
- **2.** Build the prototype rapidly.
- **3.** Modify the prototype in successive iterations.
- **4.** Stress the user interface.

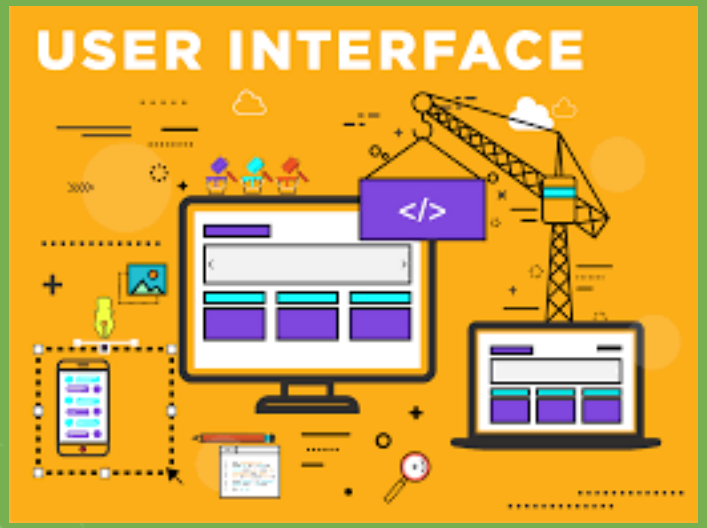




- **WORKING IN MANAGEABLE MODULES.**
- A manageable module is one that allows users to interact with its key features but can be built separately from other system modules.
- Module features that are deemed less important are purposely left out of the initial prototype

- **BUILDING THE PROTOTYPE RAPIDLY.**
- Speed is essential to the successful prototyping of an information system.
- Analysts can use prototyping to shorten this gap by using traditional information-gathering techniques to pinpoint salient information requirements, and then quickly make decisions that bring forth a working model.
- In effect the user sees and uses the system very early in the SDLC instead of waiting for a finished system to gain hands-on experience.
- Putting together an operational prototype both rapidly and early in the SDLC allows the analyst to gain valuable insight into how the remainder of the project should go.
- By showing users very early in the process how parts of the system actually perform, rapid prototyping guards against overcommitting resources to a project that may eventually become unworkable.

- **MODIFYING THE PROTOTYPE.**
- A third guideline for developing the prototype is that its construction must support modifications.
- Making the prototype modifiable means creating it in modules that are not highly interdependent.
- If this guideline is observed, less resistance is encountered when modifications in the prototype are necessary.
- The prototype is generally modified several times, going through several iterations.
- Changes in the prototype should move the system closer to what users say is important.
- Each modification necessitates another evaluation by users.
- The prototype is not a finished system.



- **STRESSING THE USER INTERFACE.**
- Because what you are really trying to achieve with the prototype is to get users to further articulate their information requirements, they must be able to interact easily with the system's prototype.
- They should be able to see how the prototype will enable them to accomplish their tasks. For many users the interface is the system. It should not be a stumbling block.
- Although many aspects of the system will remain undeveloped in the prototype, the user interface must be well developed enough to enable users to pick up the system quickly and not be put off.
- Online, interactive systems using GUI interfaces are ideally suited to prototypes.

DISADVANTAGES OF PROTOTYPING

- The first is that it can be quite difficult to manage prototyping as a project in the larger systems effort.
- The second disadvantage is that users and analysts may adopt a prototype as a completed system when it is in fact inadequate and was never intended to serve as a finished system.
- Analysts need to work to ensure that communication with users is clear regarding the timetable for interacting with and improving the prototype.
- The analyst needs to weigh these disadvantages against the known advantages when deciding whether to prototype, when to prototype, and how much of the system to prototype.

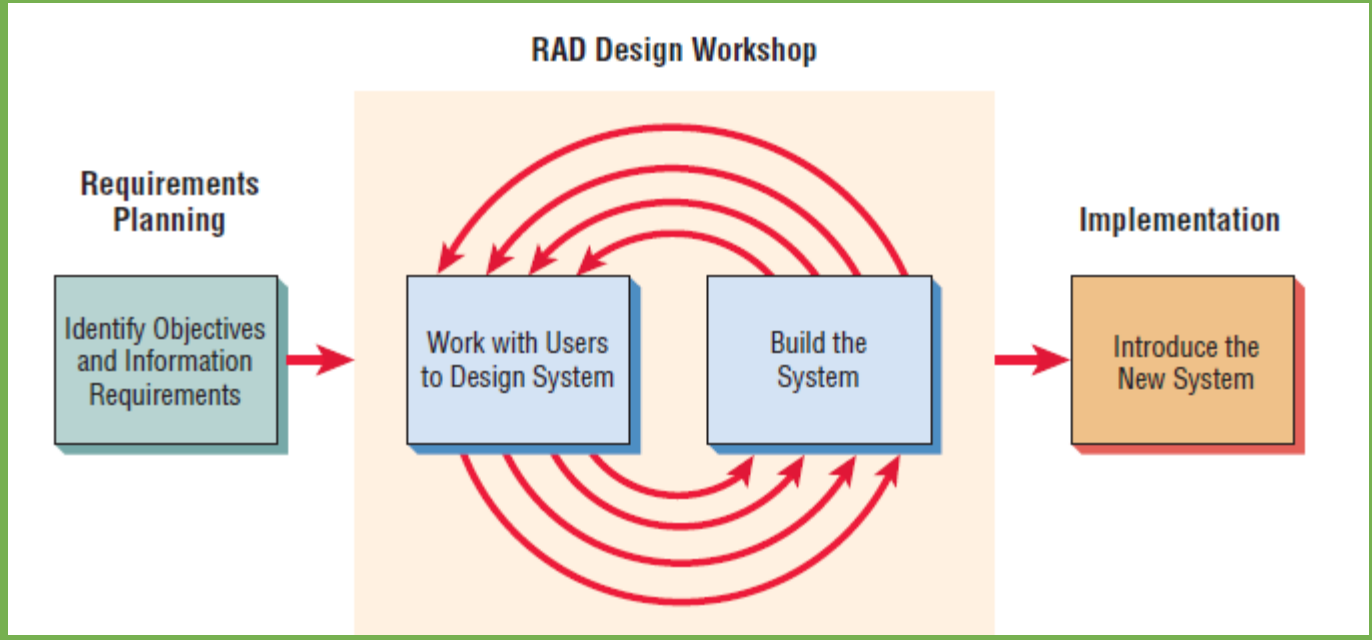
ADVANTAGES OF PROTOTYPING

- The three major advantages of prototyping are
- the potential for changing the system early in its development,
- the opportunity to stop development on a system that is not working, and
- the possibility of developing a system that more closely addresses users' needs and expectations.

- Successful prototyping depends on early and frequent user feedback, which analysts can use to modify the system and make it more responsive to actual needs.
- As with any systems effort, early changes are less expensive than changes made late in the project's development

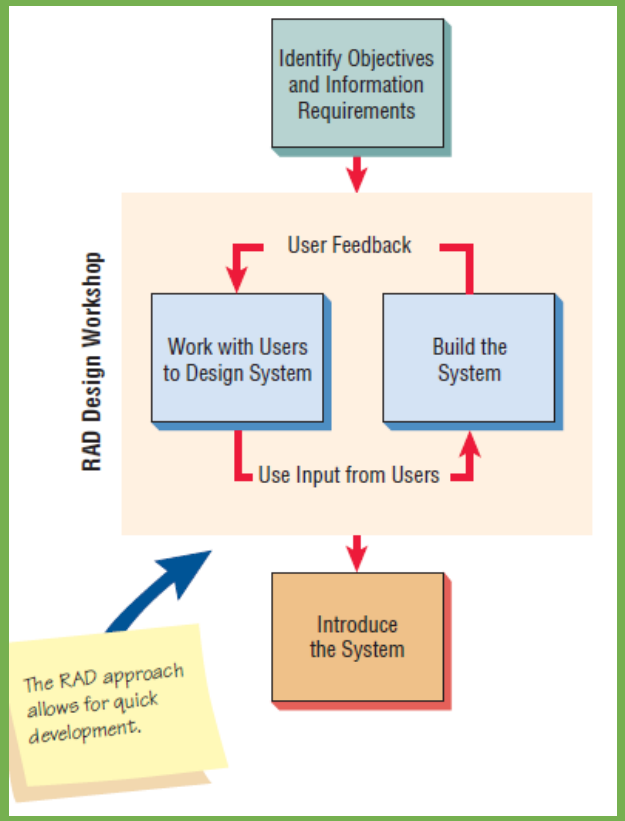
RAPID APPLICATION DEVELOPMENT

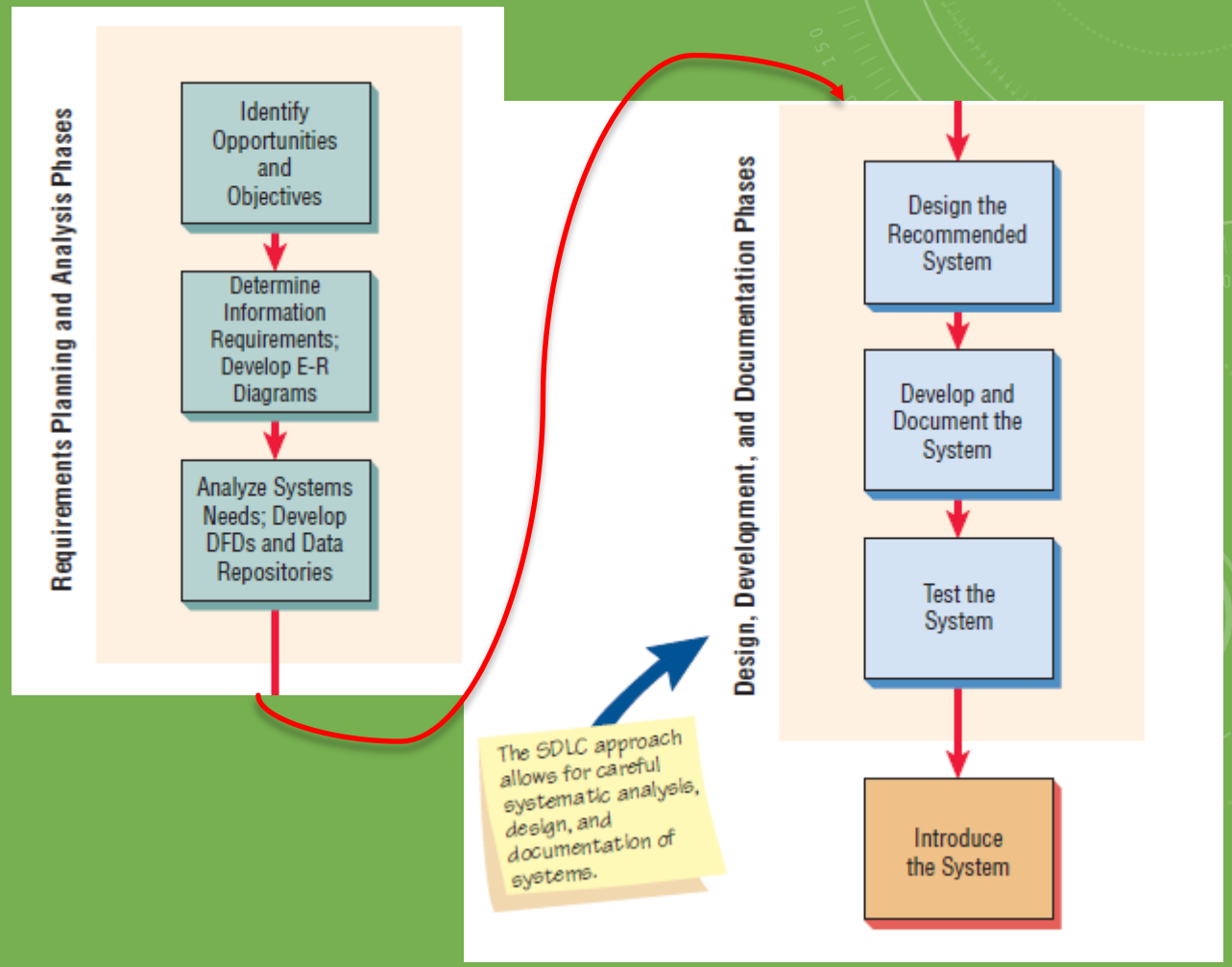
- Rapid application development (RAD) is an object-oriented approach to systems development that includes a method of development as well as software tools.
- Some developers are looking at RAD as a helpful approach in new ecommerce, Web-based environments in which so-called first-mover status of a business might be important.



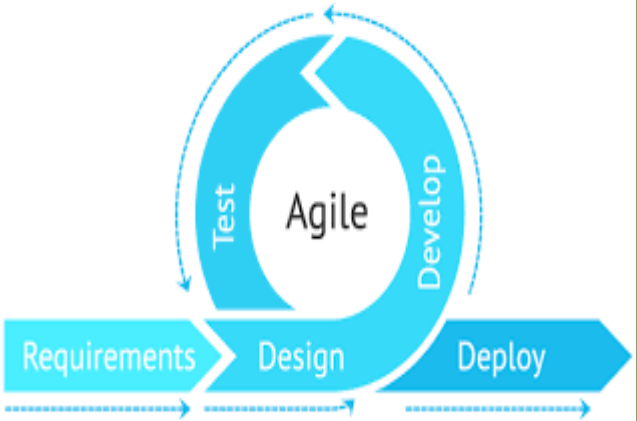
COMPARING RAD TO THE SDLC

- The ultimate purpose of RAD is to shorten the SDLC and in this way respond more rapidly to dynamic information requirements of organizations.





AGILE MODELING

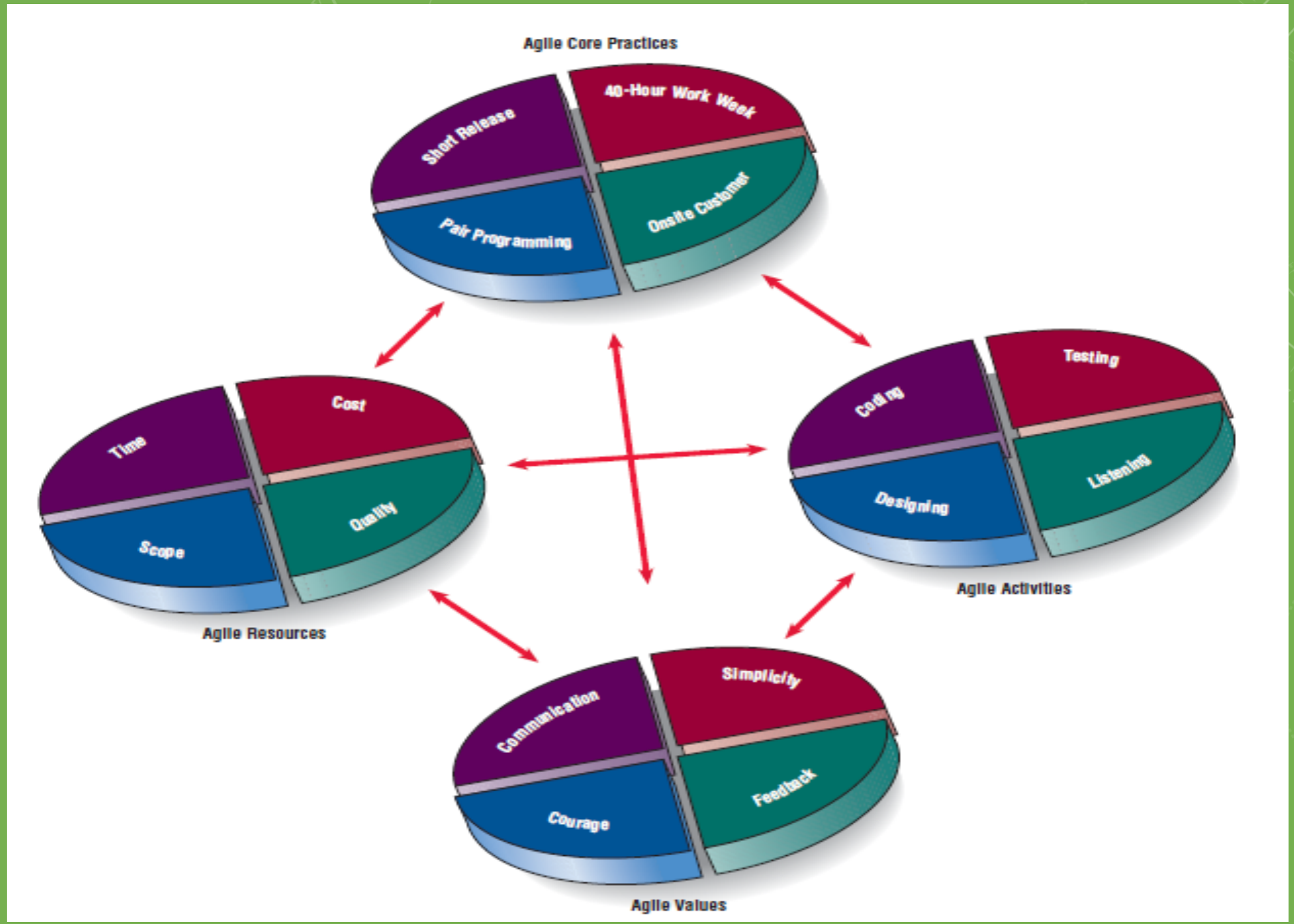


- Agile methods are a collection of innovative, user-centered approaches to systems development.
- Agile methods can be credited with many companies from a failing system that was designed using a structured methodology.

- The four values are
- Communication: Every human endeavor is fraught with possibilities for miscommunication. Systems projects that require constant updating and technical design are especially prone to such errors.
- Simplicity: When we are working on a software development project, our first inclination is to become overwhelmed with the complexity. Simplicity for software development means that we will begin with the simplest possible thing we can do.
- Feedback: Good, concrete feedback that is useful to the programmer, analyst, and customer can occur within seconds, minutes, days, weeks, or months, depending on what is needed, who is communicating, and what will be done with the feedback
- Courage: The value of courage has to do with a level of trust and comfort that must exist in the development team. It means not being afraid to throw out an afternoon or a day of programming and begin again if all is not right. It means being able to stay in touch with one's instincts (and test results) concerning what is working and what is not.

THE BASIC PRINCIPLES OF AGILE MODELING

- Agile principles were first described by Beck et al. and have evolved ever since. These principles can be expressed in a series of sayings such as:
  - **1.** Satisfy the customer through delivery of working software
  - **2.** Embrace change, even if introduced late in development
  - **3.** Continue to deliver functioning software incrementally and frequently
  - **4.** Encourage customers and analysts to work together daily
  - **5.** Trust motivated individuals to get the job done
  - **6.** Promote face-to-face conversation
  - **7.** Concentrate on getting software to work
  - **8.** Encourage continuous, regular, and sustainable development
  - **9.** Adopt agility with attention to mindful design
  - **10.** Support self-organizing teams
  - **11.** Provide rapid feedback
  - **12.** Encourage quality
  - **13.** Review and adjust behavior occasionally, and
  - **14.** Adopt simplicity.


Agile Core Practices
Short Release
40-Hour Work Week
Pair Programming
Onsite Customer
Agile Resources
Time
Cost
Scope
Quality
Agile Activities
Coding
Testing
Designing
Listening
Agile Values
Communication
Simplicity
Courage
Feedback


Short releases allow systems to evolve
Agile values are crucial to success
Pair programming enhances overall quality
Lessons Gained from Adopting Agile Methods
Balanced resources and activities support project goals
Onsite customers are mutually beneficial
40-hour workweek improves effectiveness

DAVIS AND NAUMANN'S STRATEGIES FOR IMPROVING EFFICIENCY CAN BE IMPLEMENTED USING TWO DIFFERENT DEVELOPMENT APPROACHES

Strategies for Improving Efficiency in Knowledge Work	Implementation Using Structured Methodologies	Implementation Using Agile Methodologies
Reduce interface time and errors	Adopting organizational standards for coding, naming, etc.; using forms	Adopting pair programming
Reduce process learning time and dual processing losses	Managing when updates are released so the user does not have to learn and use software at the same time	Ad hoc prototyping and rapid development
Reduce time and effort to structure tasks and format outputs	Using CASE tools and diagrams; using code written by other programmers	Encouraging short releases
Reduce nonproductive expansion of work	Project management; establishing deadlines	Limiting scope in each release
Reduce data and knowledge search and storage time and costs	Using structured data gathering techniques, such as interviews, observation, sampling	Allowing for an onsite customer
Reduce communication and coordination time and costs	Separating projects into smaller tasks; establishing barriers	Timeboxing
Reduce losses from human information overload	Applying filtering techniques to shield analysts and programmers	Sticking to a 40-hour workweek


thank
you