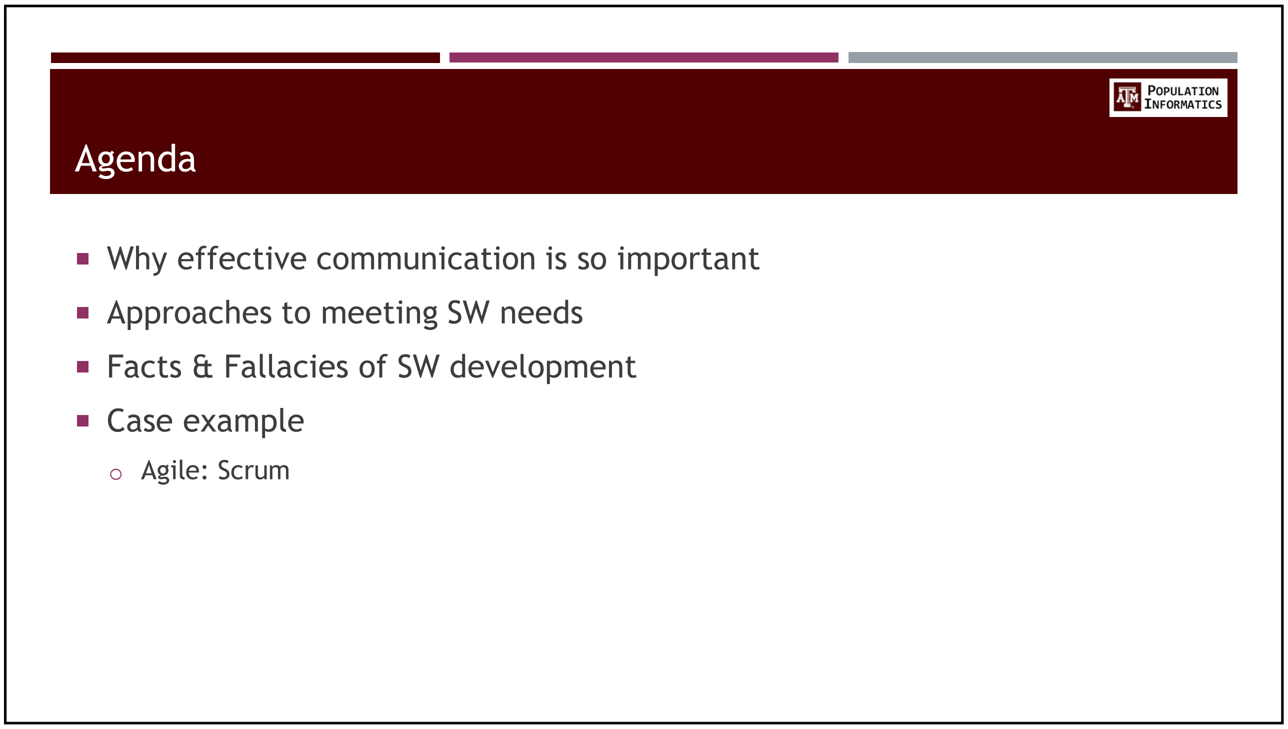
Managing Software Needs (Acquisition and Development)

Hye-Chung Kum (kum@tamu.edu)
Associate Professor
Population Informatics Lab (https://pinformatics.org/)

Course URL: http://pinformatics.org/phpm631

3

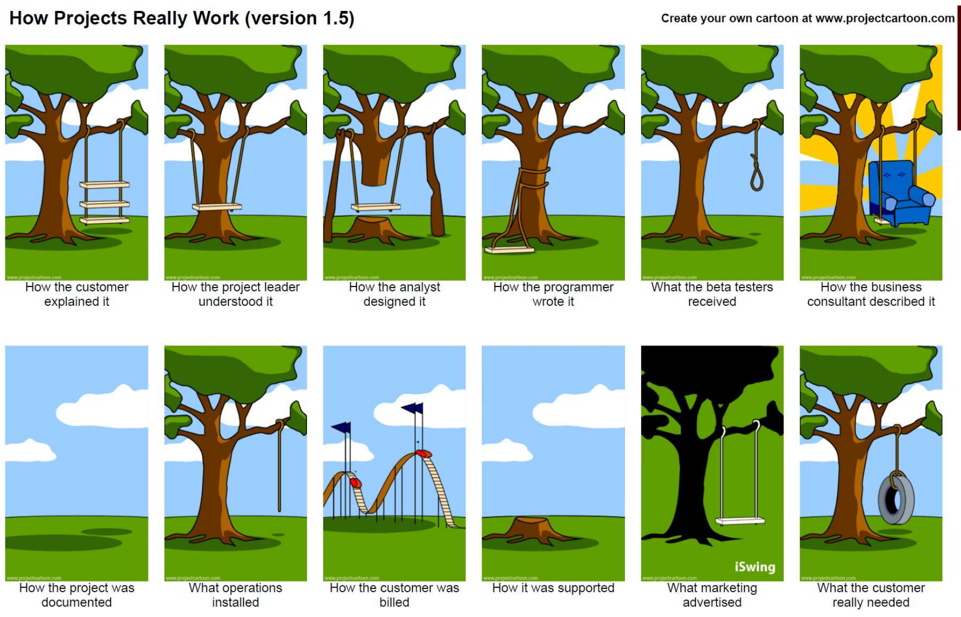
3



Agenda

- Why effective communication is so important
- Approaches to meeting SW needs
- Facts & Fallacies of SW development
- Case example
  - Agile: Scrum

4


How Projects Really Work (version 1.5)
Create your own cartoon at www.projectcartoon.com
How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
What the beta testers received
How the business consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
iSwing
What marketing advertised
What the customer really needed

- Whose job is it to figure out what hospitals really need ?
- Who can?

http://projectcartoon.com/cartoon/2

5

Systems Development Life Cycle (SDLC)

- System Acquisition Process
  - Planning and Analysis
  - Design
- System Implementation Process
  - Implementation
  - Support and Evaluation

6

6

Agenda

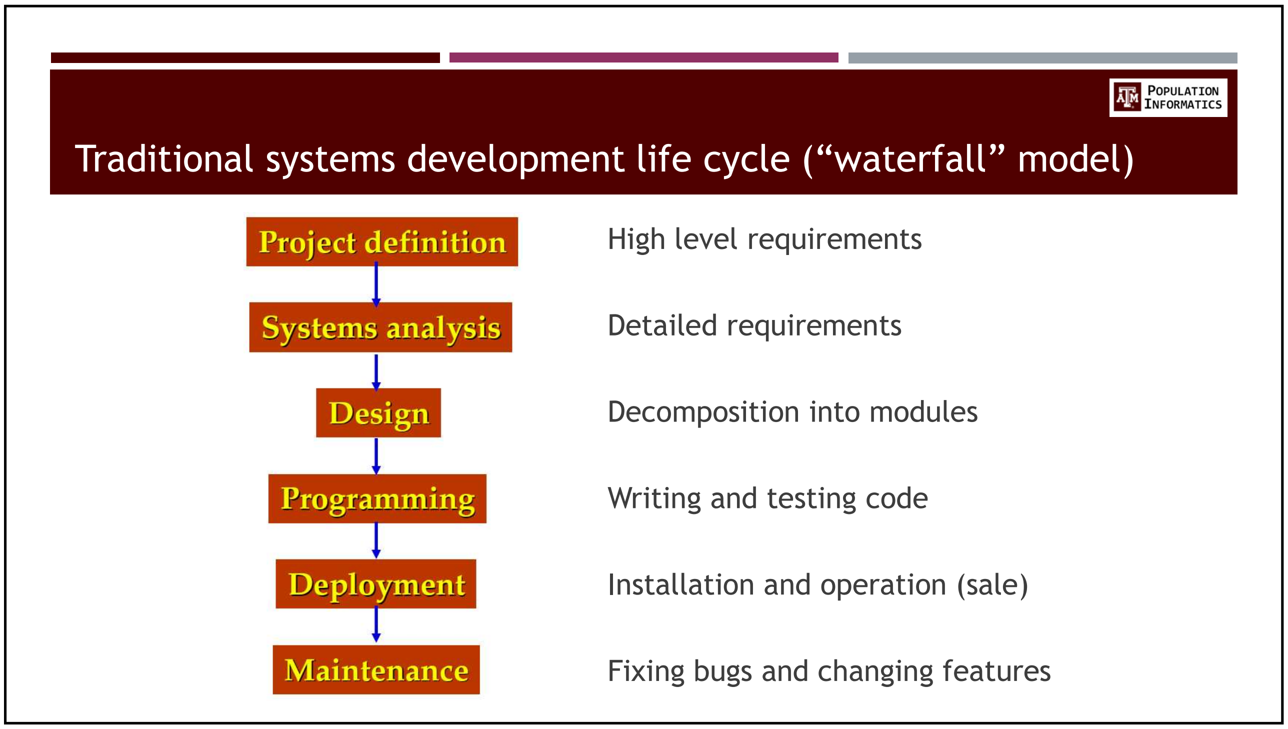
- Why effective communication is so important
- Systems Development Life Cycle (SDLC)
- Approaches to meeting SW needs
- Facts & Fallacies of SW development
- Case example
  - Agile: Scrum

7

Meeting Software Needs
Approaches to software development

- Traditional systems development life cycle
- Prototyping
- Packaged software: off the self
- End-user development: in-house
- Outsourcing
- Open source
- Cloud - SaaS: Software as a Service

8

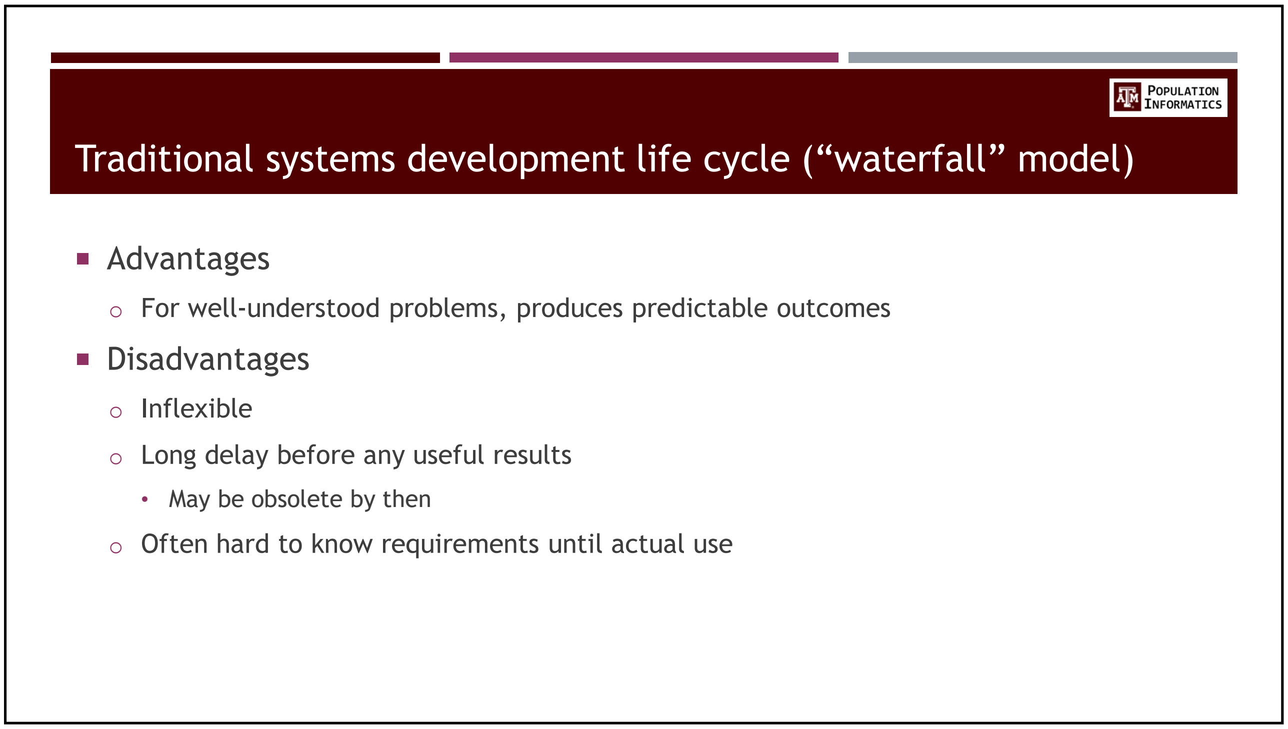


Traditional systems development life cycle (“waterfall” model)

POPULATION INFORMATICS

Project definition	High level requirements
Systems analysis	Detailed requirements
Design	Decomposition into modules
Programming	Writing and testing code
Deployment	Installation and operation (sale)
Maintenance	Fixing bugs and changing features

9



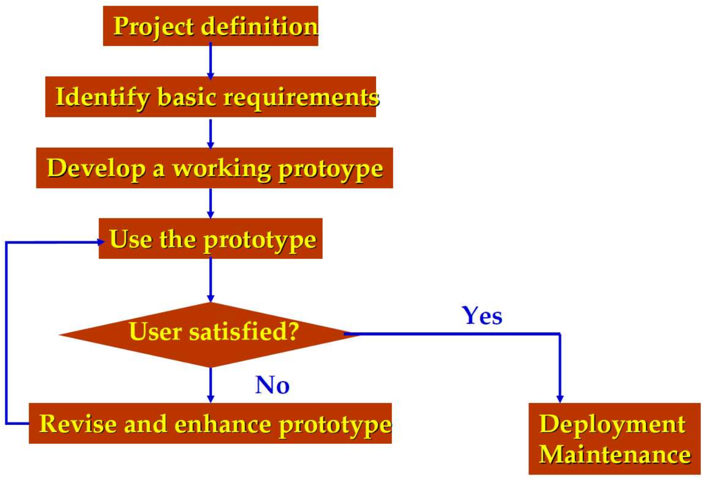
Traditional systems development life cycle (“waterfall” model)

POPULATION INFORMATICS

- Advantages
  - For well-understood problems, produces predictable outcomes
- Disadvantages
  - Inflexible
  - Long delay before any useful results
    - May be obsolete by then
  - Often hard to know requirements until actual use

10

Prototyping ("Iterative" model) Agile Method


Project definition
Identify basic requirements
Develop a working protoype
Use the prototype
User satisfied?
Yes
No
Revise and enhance prototype
Deployment
Maintenance

11

Prototyping ("Iterative" model) Agile Method

- Advantages
  - Especially useful when exact requirements are hard to know in advance
    - user interfaces
    - decision systems
    - electronic commerce?
  - Encourages user involvement
- Disadvantages
  - Hard to predict and control outcomes reliably
  - If repeated, significant reimplementations are needed, can be very expensive
  - May result in systems that are inefficient, unreliable, or hard to maintain

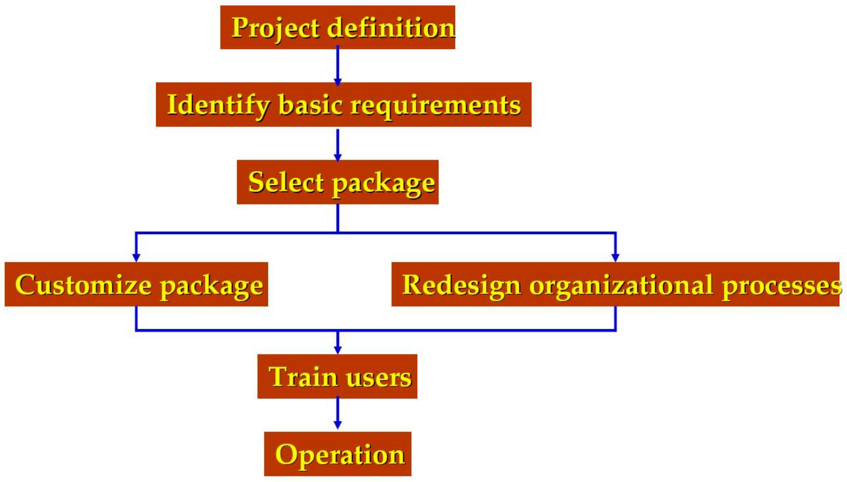
12

Agile Manifesto

- Philosophy
  - Adaptive vs predictive
  - Iterative vs waterfall
  - Code vs documentation
- Scrum: Do sprints
  - sprints are short two-to-three week design or development cycles via a repeatable process where the team works on designing and/or developing specific user stories. Depending on the size of the application being built, there may be many sprints. But the rapid iterations move the project forward quickly and allow the team to focus on the needs of the end user.
- Extreme programming, lean software development, test driven
- prototyping

13

Packaged software



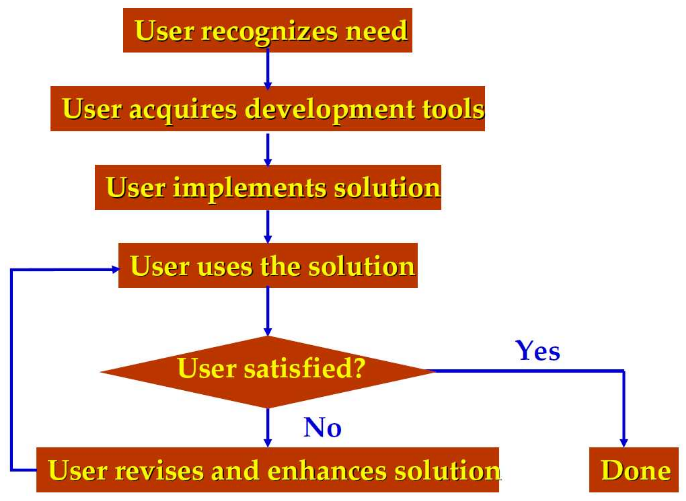
14

Packaged software

- Advantages
  - By amortizing development and maintenance costs over many organizations, it is possible to get superior solutions at much lower cost
- Disadvantages
  - Customizing software can be very time-consuming and expensive
  - May have to change organization to fit software, rather than vice versa

15

End-user development (in house development)



16

End-user development (in house development)

- Advantages
  - Can be *much* faster
  - Improved requirements determination
  - Increased user involvement and satisfaction
- Disadvantages
  - Often, users lack the right implementation skills
  - Many problems can’t be solved within the limitations of the tools
  - Lack of quality assurance and standards for programs and data
    - Security requirements introduce barriers
  - Lack of sharing of programs and data
  - Reduced opportunity for reuse of results

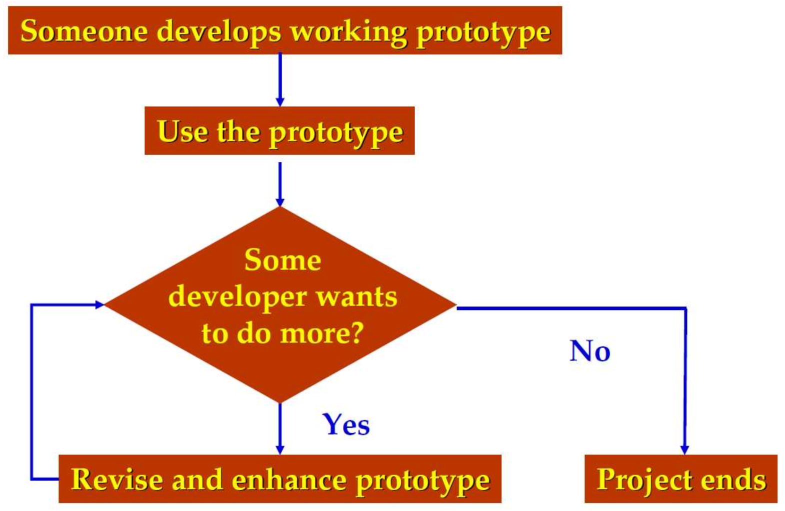
17

Outsourcing

- Contract out the performance of any or all of the above steps to another firm
- Advantages
  - Economies of scale
  - Flexibility
  - Predictability
  - Freeing up human resources and capital
- Disadvantages
  - Loss of control
  - Vulnerability of strategic information
  - Dependency

18

Open source


Someone develops working prototype
Use the prototype
Some developer wants to do more?
No
Yes
Revise and enhance prototype
Project ends

19

Open source

- Advantages
  - Usually lower cost
  - Sometimes easier to adapt “packaged” software to own needs
  - “Philosophically” appealing to many people
- Disadvantages
  - Usually lower quality support (Often higher quality software for lower cost)
  - Only a few kinds of software are currently available in this format (Linux operating system, Apache web server, etc.)

20

Cloud: Software as a Service (SaaS)

- Cloud based: gmail
- Advantages
  - Do not have to maintain hardware/software
  - Economies of scale
  - Predictability
  - Freeing up human resources and capital
- Disadvantages
  - Loss of control
  - Dependency
  - No access when network is down
  - In the long run may be expensive. Kind of like leasing car.

21

Agenda

- Why effective communication is so important
- Approaches to meeting SW needs
- Facts & Fallacies of SW development
- Case example
  - Agile: Scrum

22

Problems with software development

- Computerworld magazine
  - “Nearly one-third of all projects fail”
  - “More than half come in over budget”
  - “Only 16% of all projects come in on time and on budget”
  - Survey of 8000 projects from 385 companies.
- Key factor for success or failure:
  - “User involvement/input”

23

Facts about Software Development

- Facts
  - The most important factor in software development is the quality of the programmers.
  - The best programmers are up to 28 times better than the worst.
  - Adding people to a late project makes it later.
  - One of the most common causes of runaway projects is poor estimation.
  - The other most common cause of runaway projects is unstable requirements.
  - Requirements errors are the most expensive to fix during production.
  - Maintenance typically consumes 40 to 80 percent of software costs.
  - Enhancements represent roughly 60 percent of maintenance costs.

Adapted from Robert L. *Glass, Facts and Fallacies of Software Engineering*, Addison Wesley, 2003

24

Fallacies about Software Development

- Fallacies
  - Software needs more methodologies.
  - You teach people how to program by showing them how to write programs.

Adapted from Robert L. *Glass, Facts and Fallacies of Software Engineering*, Addison Wesley, 2003

25

Main message

- IT systems (software, hardware, network, etc) have to come together to meet organizational goals
- Coming together should never be a haphazard process
- It should be engineered
- IT Systems are all about understanding tradeoffs
  - Computer Systems can be FAST, CHEAP, or RELIABLE
  - But not all at once. Pick any two. (You can have fast and reliable, if $$ is not an issue)
  - Goldilocks principle: Not too hot, not too cold
- Must know organizations objectives and desired system properties to decide

26

26

Take Away 1
Approaches to software development

- Traditional systems development life cycle
- Prototyping
- Packaged software
- End-user development
- Outsourcing
- Open source
- Cloud - SaaS: Software as a Service

27

Take Away 2
Traditional (waterfall) vs Agile (iterative)

- Waterfall (Traditional)
- Know the process
- Advantages
  - For well-understood problems, produces predictable outcomes
- Disadvantages
  - Inflexible
  - Long delay before any useful results
    - May be obsolete by then
  - Often hard to know requirements until actual use

- Iterative (Agile)
- Know the process
- Advantages
  - Especially useful when exact requirements are hard to know in advance
    - user interfaces
    - decision systems
    - electronic commerce?
  - Encourages user involvement
- Disadvantages
  - Hard to predict and control outcomes reliably
  - If repeated, significant reimplementations are needed, can be very expensive
  - May result in systems that are inefficient, unreliable, or hard to maintain

28

Take Away 3
Facts & Fallacies about SW Development

POPULATION INFORMATICS

- Facts
  - The most important factor in software development is the quality of the programmers.
  - The best programmers are up to 28 times better than the worst.
  - Adding people to a late project makes it later.
  - One of the most common causes of runaway projects is poor estimation.
  - The other most common cause of runaway projects is unstable requirements.
  - Requirements errors are the most expensive to fix during production.
  - Maintenance typically consumes 40 to 80 percent of software costs.
  - Enhancements represent roughly 60 percent of maintenance costs.
- Fallacies
  - Software needs more methodologies.
  - You teach people how to program by showing them how to write programs.
- Adapted from Robert L. Glass, Facts and Fallacies of Software Engineering, Addison Wesley, 2003

29

Break (please read assignment 4)

30

Presentation:

- Peer Review
- How well was the technology described in the presentation ?
- How well did you understand the three take aways for hospital managers?
- How well did the presentation discuss the relevance and impact of the technology on health care?
  - Great
  - Good
  - Reasonable
  - Bad

31

31

Next week

- Read Chapter 6 & 8
- Quiz 5
- Assignment 4 Part 1 Due
- Lab 4 due

32

32

Definition: IT Architecture

- Architecture: blueprint
- Platform:
- Infrastructure: the actual system
- "In creating an infrastructure, an organization will implement platforms and be guided by its IT architecture."

33

33

Assignment 4: Report Part 1 (Due 2/23)

- Team project
- Required Reading for this Assignment
  - Turner, A. M., Reeder, B., & Ramey, J. Scenarios, personas and user stories: User-centered evidence-based design representations of communicable disease investigations. Journal of Biomedical Informatics, 46(4), 575-584. doi: 10.1016/j.jbi.2013.04.006
  - Kneale L, Mikles S, Choi YK, Thompson H, Demiris G. Using scenarios and personas to enhance the effectiveness of heuristic usability evaluations for older adults and their care team. Journal of biomedical informatics. 2017 Sep 1;73:43-50.
    - IMPORTANT: Read the appendix which has examples of personas and scenarios

34

Assignment 4: Software Requirement

- Next week: Report Part 1
  - Team members
  - Informal Software Need Description
  - Concept (Draft)
    - 1 page summary
    - Include an architecture statement that is clearly labeled
- Two weeks: Report Part 2
  - Concept (Final)
    - 1 page summary (ok to be identical to report part 1, if no updates are needed)
    - A tweet (under 280 characters) – you don't have to actually tweet
  - Personas : define & analyze behavior
  - Scenarios
  - User Stories

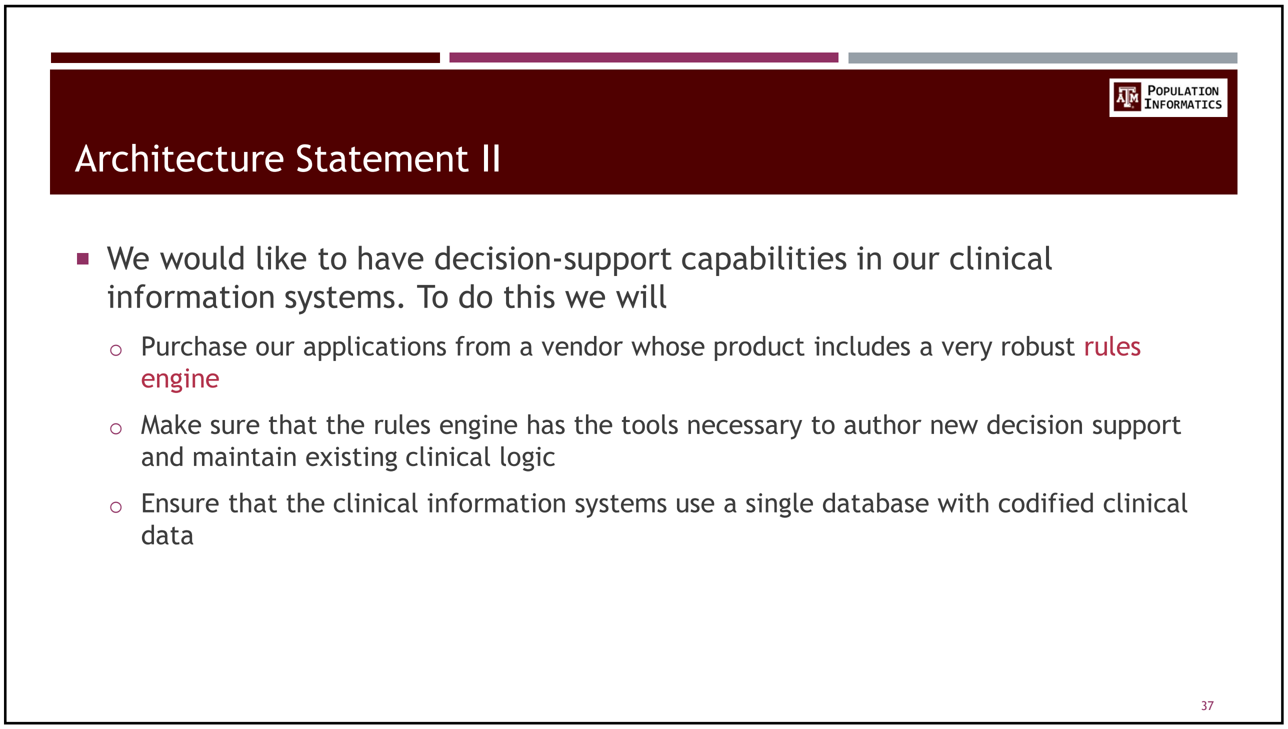
35

Architecture Statement I

- We would like to deliver an electronic health record to our small physician practices that is inexpensive, reliable, and easy to support. To do this we will
  - Run the application from our computer room, reducing the need for practice staff to manage their own servers and do tasks such as backups and applying application enhancements
  - Run several practices on one server to reduce the cost
  - Obtain a high speed network connection, and a back up connection, from our local telephone company to provide good application performance and improved reliability

36

36

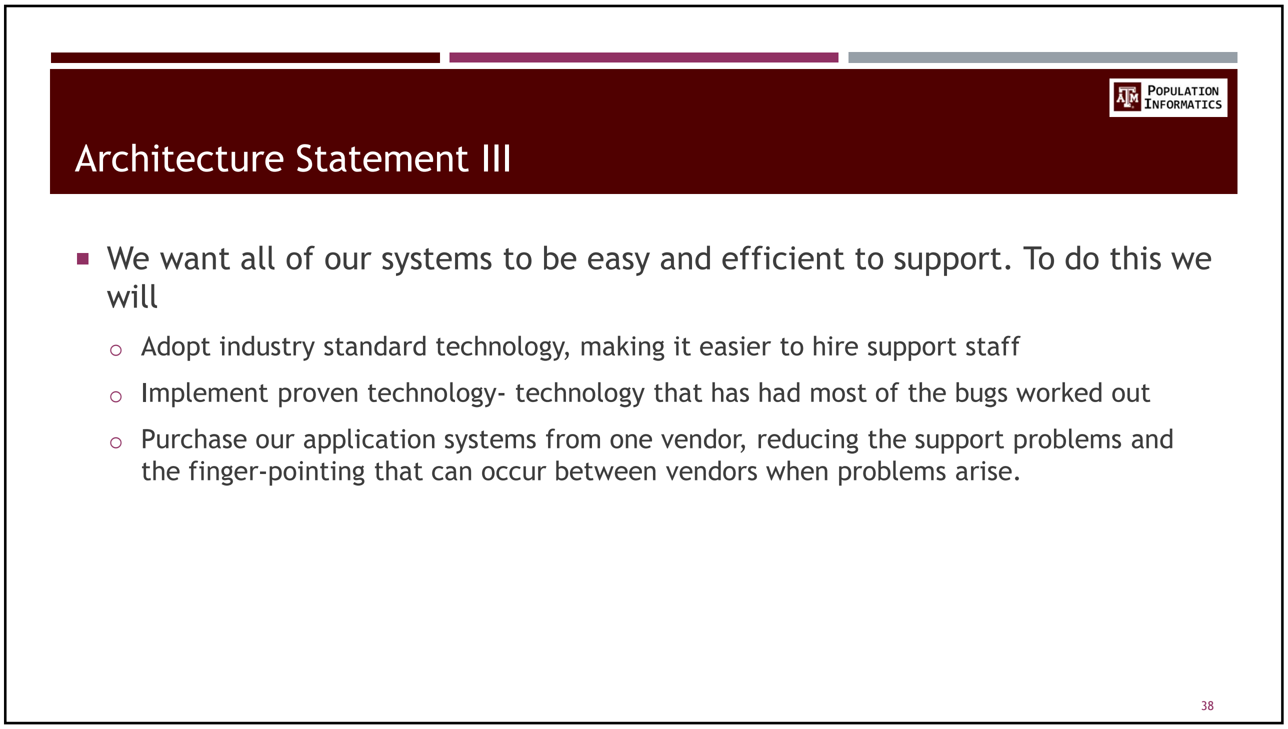


Architecture Statement II

- We would like to have decision-support capabilities in our clinical information systems. To do this we will
  - Purchase our applications from a vendor whose product includes a very robust rules engine
  - Make sure that the rules engine has the tools necessary to author new decision support and maintain existing clinical logic
  - Ensure that the clinical information systems use a single database with codified clinical data

37

37



Architecture Statement III

- We want all of our systems to be easy and efficient to support. To do this we will
  - Adopt industry standard technology, making it easier to hire support staff
  - Implement proven technology- technology that has had most of the bugs worked out
  - Purchase our application systems from one vendor, reducing the support problems and the finger-pointing that can occur between vendors when problems arise.

38

38