Công nghệ yêu cầu

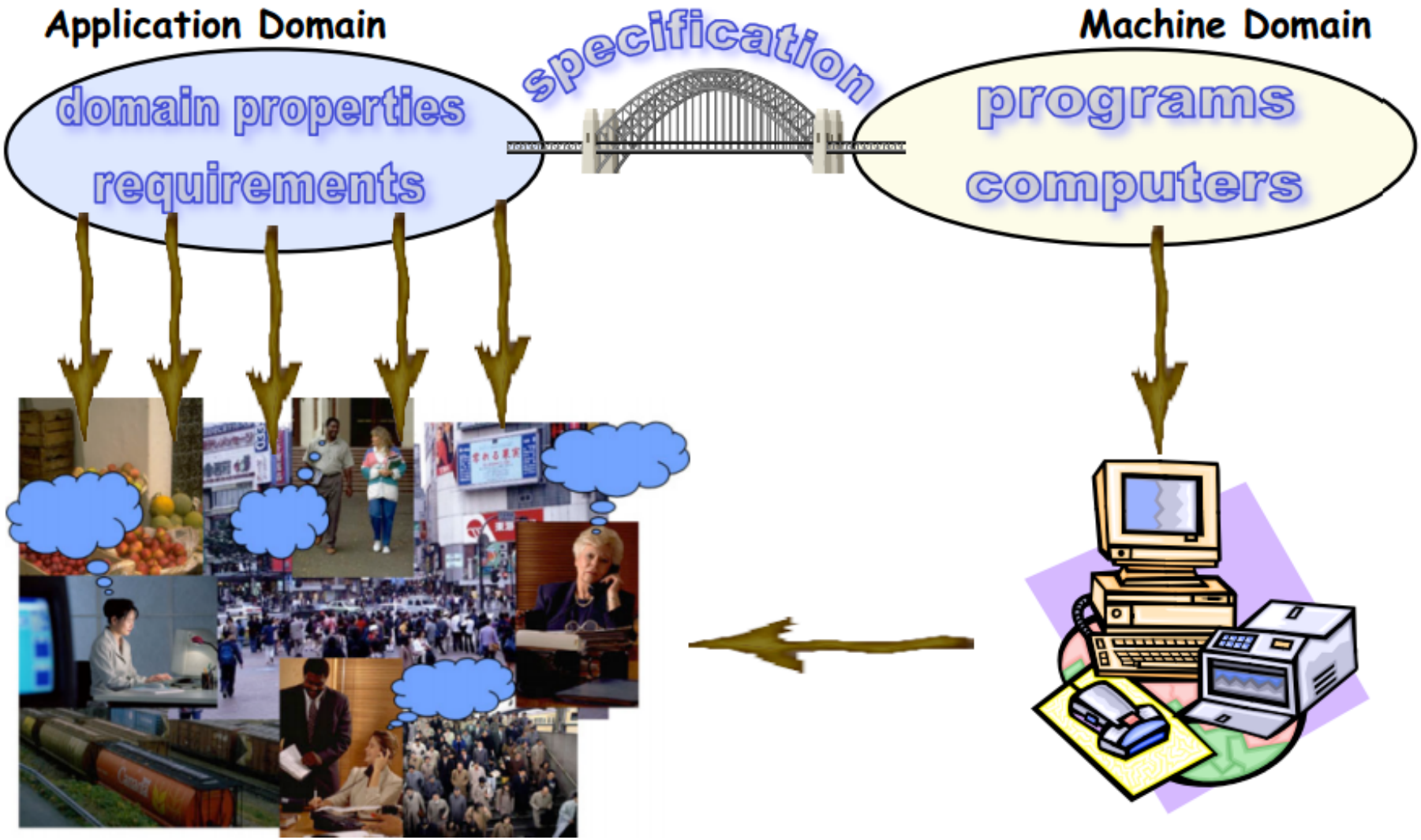
Bộ môn CNPM - CNTT&TT

Ngày 22 tháng 1 năm 2019

Chất lượng

- **Công nghệ phần mềm có mặt ở khắp mọi nơi**
  - Tác động hầu hết đến các khía cạnh của cuộc sống
  - Kinh nghiệm thì còn hạn chế
- **Phần mềm được thiết kế với một mục đích nào đó**
  - Nếu nó không được thực hiện tốt
    - Người thiết kế chưa thật sự thấu hiểu mục đích
    - Sử dụng phần mềm cho mục đích khác với dự định
  - Phân tích yêu cầu nhằm xác định chính xác mục đích này
  - Không hiểu đầy đủ về mục đích sẽ dẫn đến phần mềm kém chất lượng
- **Mục đích được tìm thấy từ hoạt động của con người**
  - Ví dụ: Mục đích của hệ thống ngân hàng xuất phát từ hoạt động kinh doanh và nhu cầu của khách hàng (ATM,...)
  - Mục đích thường phức tạp

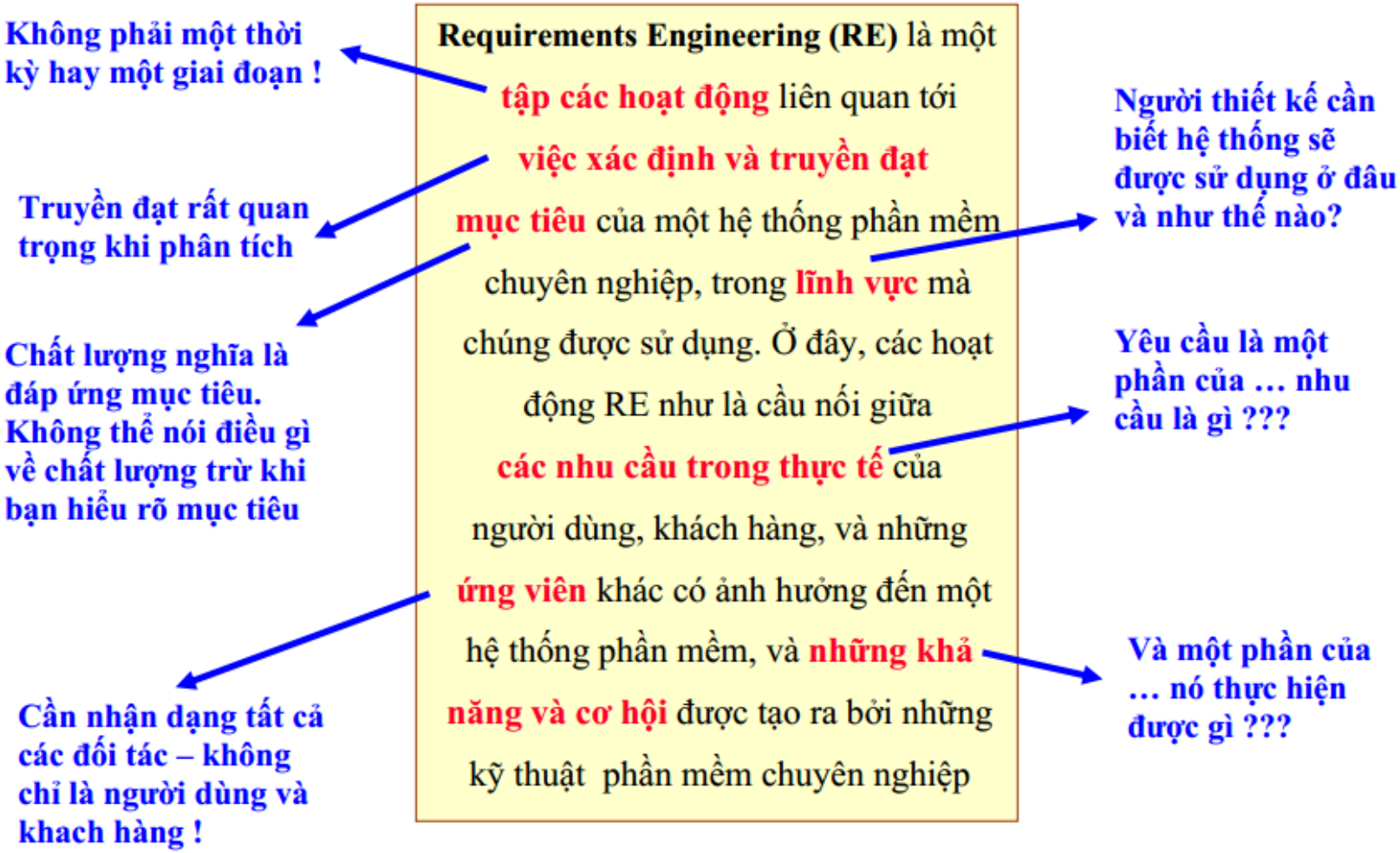
Thách thức


Application Domain
Machine Domain
specification
domain properties
requirements
programs
computers

Hệ thống mềm

- **Các loại phần mềm**
  - Ví dụ: Các chức năng lõi trong hệ điều hành, dịch vụ mạng, ...
  - Có quan hệ ổn định về mặt chức năng thông qua giao diện kỹ thuật
  - Chú ý: Hệ thống có thể tác động bởi hoạt động của con người. Ví dụ: URL, ...
- **Các hệ thống quản lý - Control Systems**
  - Ví dụ: điều hành quy trình bay, tiến trình công nghiệp.
  - Hầu hết các yêu cầu được xác định bởi thông qua các qui trình tự nhiên.
  - Cách thức giao tiếp thường mang tính quyết định
  - Ví dụ: Tai nạn tàu vũ trụ Arian 5 - France
- **Các hệ thống thông tin - Information Systems**
  - Ví dụ: Tự động văn phòng, phần mềm hỗ trợ kinh doanh, web services,...
  - Các hệ thống này phải gắn liền với hoạt động mà chúng hỗ trợ
  - Thiết kế phần mềm phải đi từ hoạt động của con người.

Requirements Engineering



Hậu quả của sai sót

- **Giá để sửa lỗi**
  - Một tiến trình phát triển phần mềm gồm:

    Phân tích yêu cầu -> Thiết kế->Lập trình->Kiểm thử phát triển->Kiểm thử chấp nhận->Vận hành

  - Giá sửa lỗi ngày càng tăng vào thời điểm phát hiện chúng trong tiến trình
  - Ví dụ: Một lỗi về phần tích yêu cầu phải trả giá 100 lần cao hơn lỗi chương trình.

Hậu quả của sai sót

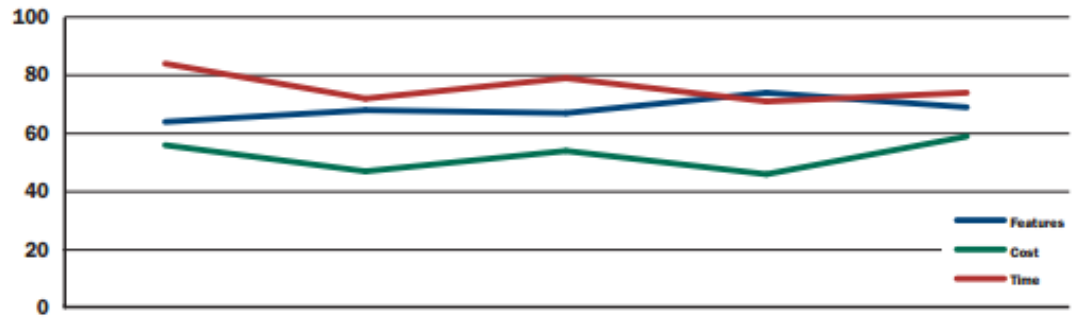
- **Một số nguyên nhân thất bại**
  - Thống kê các dự án phần mềm của nhóm Standish.

	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Hậu quả của sai sót

- **Một số nguyên nhân thất bại**
  - Thống kê các dự án phần mềm của nhóm Standish.

Time and cost overruns, plus percentage of features delivered from CHAOS research for the years 2004 to 2012.



	2004	2006	2008	2010	2012
TIME	84%	72%	79%	71%	74%
COST	56%	47%	54%	46%	59%
FEATURES	64%	68%	67%	74%	69%

Hậu quả của sai sót

- **Một số nguyên nhân thất bại**
  - Thống kê các dự án phần mềm của nhóm Standish.

Factors of Success	Points
Executive management support	20
User involvement	15
Optimization	15
Skilled resources	13
Project management expertise	12
Agile process	10
Clear business objectives	6
Emotional maturity	5
Execution	3
Tools and infrastructure	1

Hậu quả của sai sót

- **Lỗi yêu cầu có thể phải trả giá đắt nếu không phát hiện và sửa chữa sớm trong tiến trình phát triển.**
- **Báo cáo của Boehm và Papaccio (1988)** ước lượng chi phí cho việc sửa lỗi phần mềm như sau:

Phân tích yêu cầu **(1$)** ⇒ Thiết kế **(5$)** ⇒ Lập trình **(10$)**
⇒ Kiểm thử **(20$)** ⇒ Triển khai hệ thống **(>200$)**

Chú ý

Cần dành thời gian tìm hiểu kỹ vấn đề trong lĩnh vực của chúng và thu thập chính xác yêu cầu trong giai đoạn đầu

Mục tiêu của phân tích yêu cầu

- **Tập trung chú ý rằng có một "vấn đề" cần được giải quyết**
  - không bằng lòng với trạng thái công việc hiện tại
  - một cơ hội kinh doanh mới
  - một cơ hội để tiết kiệm chi phí, thời gian, tài nguyên,...
- **Nhà phân tích yêu cầu là một tác nhân của sự thay đổi**

Phân tích yêu cầu cần đạt được gì?

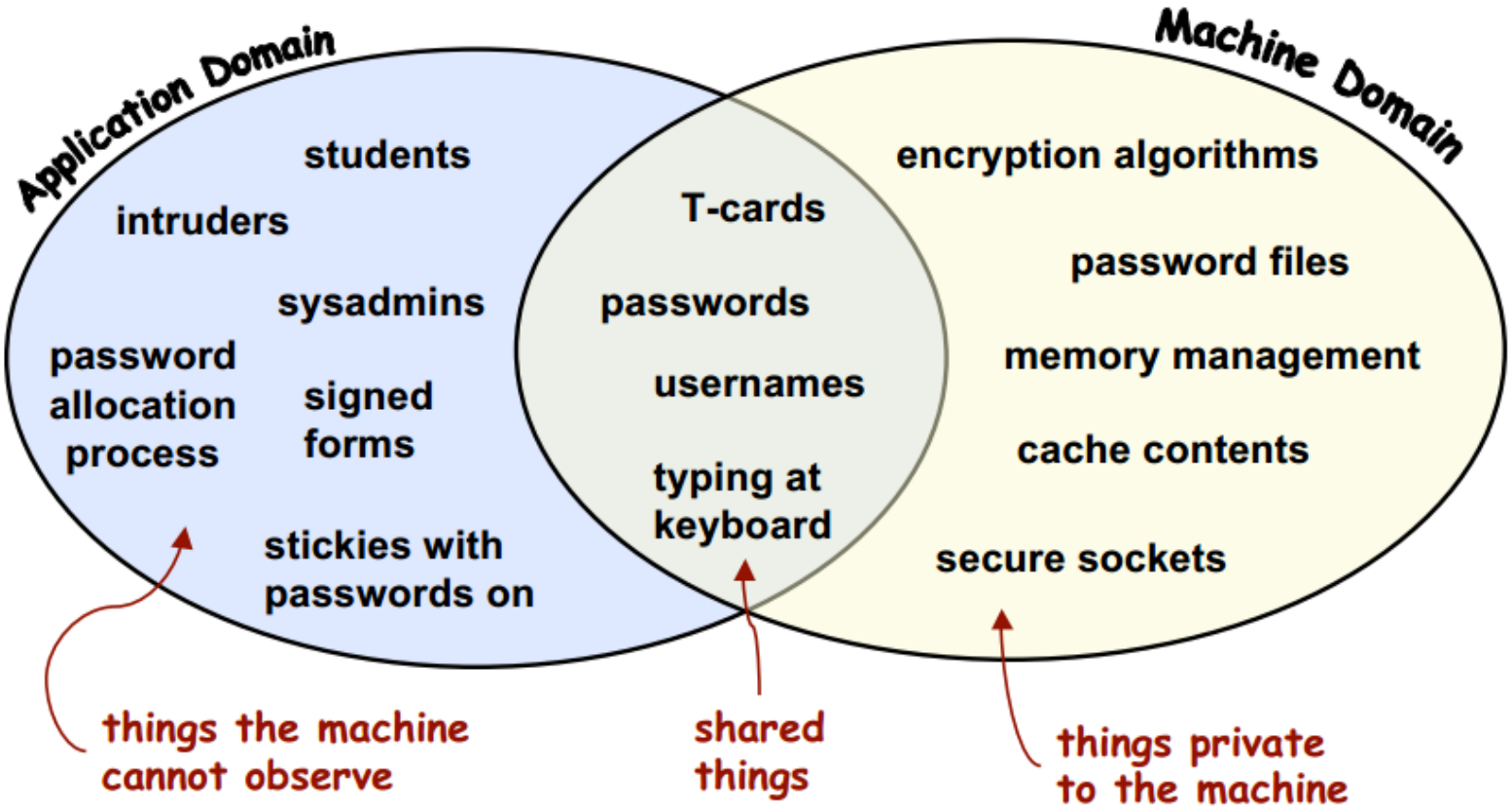
- **Định nghĩa được vấn đề**
  - (Which)Vấn đề nào cần được giải quyết?
    Xác định ranh giới vấn đề - Boundaries
  - (Where)Vấn đề ở đâu?
    Hiểu ngữ cảnh/ phạm vi vấn đề - Context/Problem Domain
  - (Whose)Vấn đề của ai?
    Định nghĩa Đối tác - Stakeholders
  - (Why)Tại sao cần được giải quyết?
    Định nghĩa Mục tiêu đối tác - stakeholders' Goals
  - (How)Hệ thống phần mềm sẽ hỗ trợ như thế nào?
    Thu thập kịch bản - Scenarios
  - (When)Khi nào vấn đề cần giải quyết?
    Định nghĩa các ràng buộc phát triển - Development Constraints
  - (What)Điều gì ngăn chặn việc giải quyết chúng?
    Định nghĩa tính khả thi và rủi ro - Feasibility and Risk
- **Là chuyên gia trong phạm vi vấn đề**

Một số khảo sát về RE

- **RE không cần thiết phải theo một tiến trình tuần tự**
  - Không cần phải viết mô tả vấn đề trước mô tả giải pháp
    - Viết lại mô tả vấn đề có thể giúp ích ở các giai đoạn phát triển
  - Các hoạt động RE tiếp tục xuyên suốt tiến trình phát triển
- **Khai báo vấn đề sẽ không hoàn hảo**
  - Các mô hình RE thì chỉ gần đúng với thực tế
    - Sẽ chứa sự thiếu chính xác và không nhất quán
    - Sẽ bỏ sót một số thông tin.
    - Các nhà phân tích luôn làm giảm bớt những rủi ro sẽ có trong vấn đề thực. . .
- **Việc hoàn chỉnh một sự đặc tả có thể không mang lại lợi nhuận**
  - Phân tích yêu cầu có giá của nó
  - Đối với những dự án khác nhau, cân bằng lợi nhuận cũng khác nhau
- **Khai báo vấn đề không khi nào được xem là cố định**
  - Thay đổi thì chắc chắn sẽ xảy ra, vì thế phải dự kiến trước

Mô tả vấn đề

- Ví dụ: Ngăn chặn việc truy cập trái phép từ các máy tính



Yêu cầu là gì?

- **Đặc tính lĩnh vực (Domain Properties D)**
  - Những thứ có trong lĩnh vực ứng dụng cho dùng chúng ta có thiết kế hệ thống dự định không
- **Các yêu cầu (Requirement D)**
  - Những thứ có trong lĩnh vực ứng dụng mà dùng chúng ta muốn trở thành hiện thực bằng cách thực hiện hệ thống dự định
  - Rất nhiều trong chúng mà máy tính không thể truy cập được
- **Sự đặc tả (Specification S)**
  - Là sự mô tả hành vi mà chương trình phải làm để đáp ứng các yêu cầu

Đáp ứng mục tiêu

- **Hai tiêu chuẩn kiểm tra tính chính xác (verification)**
  - Chương trình (Program) chạy trên một máy tính (Computer) cụ thể phải đáp ứng các đặc tả (Specification)
  - Đặc tả (Specification) được cho trong thuộc tính của lĩnh vực (Domain Properties) thỏa mãn các yêu cầu (Requirements)
- **Hai tiêu chuẩn kiểm chứng sự hoàn thiện (validation)**
  - Đã xem xét và hiểu tất cả các yêu cầu quan trọng chưa?
  - Đã xem xét và hiểu tất cả các lĩnh vực liên quan chưa?

Ví dụ

- Requirement R
  - Phản lực chỉ có thể xảy ra khi máy bay đang chạy trên đường băng
- Domain Properties D
  - Xung lực bánh xe xảy ra khi và chỉ khi các bánh xe bật ra
  - Các bánh xe bật ra khi và chỉ khi nó chạy trên đường băng
- Specification S
  - Phản lực có thể xảy ra khi và chỉ khi có xung lực bánh xe
- **Kiểm tra**
  - Phần mềm cho máy bay, P, thực thi trên máy tính trong buồng lái của máy bay, C, có hoàn toàn chính xác như đặc tả, S?
  - S, trong ngữ cảnh của giả thuyết D, có đáp ứng R?
- **Kiểm chứng**
  - Giả thuyết của chúng ta, D, về lĩnh vực có thật chính xác? Có thiếu gì không?
  - Yêu cầu, R, có thật sự cần thiết? Có thiếu gì không?

Quản lý dự án

- **Một nhà quản lý dự án có thể kiểm soát 4 khía cạnh**
  - Tài nguyên: Có thể tăng thêm, tiện ích, nhận lực
  - Thời gian: Có thể tăng thời gian, trì hoản,...
  - Sản phẩm: Có thể giảm chức năng, ví dụ: các yêu cầu phức tập, không cần thiết,...
  - Rủi ro: Có thể quyết định các rủi ro nào chấp nhận được
- **Để thực hiện điều này, nhà quản lý phải theo dõi**
  - Công sức: Cần tốn công sức nhiều thế nào? Tiêu hao bao nhiêu?
  - Thời gian: Lịch biểu được mong đợi ra sao? Còn bao lâu nữa?
  - Kích cỡ: Kế hoạch vấn đề lớn như thế nào? Phải thiết kế ra sao?
  - Hạn chế: Đã tạo ra bao nhiêu lỗi? Bao nhiêu lần phát hiện lỗi?

Chúng ta không thể kiểm soát được cái mà mình không thể đo lường

Các kiểu dự án

- **Các lý do khởi đầu cho một dự án phát triển phần mềm**
  - Problem-driven: sự cạnh tranh, sự khủng hoảng,...
  - Change-driven: nhu cầu mới, sự lớn mạnh, thay đổi doanh nghiệp hoặc môi trường,...
  - Opportunity-driven: bùng nổ một kỹ thuật mới,...
  - Legacy-driven: một phần của kế hoạch trước đó, công việc chưa hoàn thành,...
- **Các kiểu quan hệ với khách hàng**
  - Customer-specific: một khách hàng với vấn đề cụ thể
    - Có thể là một công ty khác, với hợp đồng thỏa thuận
    - Có thể là một bộ phận trong cùng công ty
  - Market-based: hệ thống bán ra thị trường
    - Trong một số trường hợp, sản phẩm phải sinh ra khách hàng
    - Đội ngũ tiếp thị đóng vai trò như là khách hàng
  - Community-based: dự định như một tiện ích cho cộng đồng
    - Công cụ nguồn mở(open source), các công cụ cho nghiên cứu khoa học
    - Khách hàng tài trợ(nếu nhà tài trợ không chiếm giữ kết quả)
  - Hybrid: kết hợp những kiểu trên

Chu kỳ sống của một dự án phần mềm

- **Các mô hình chu kỳ sống**
  - Rất hữu ích để so sánh các dự án trong ngữ cảnh chung
  - Không đủ chi tiết cho việc hoạch định dự án
- **Các ví dụ**
  - Các mô hình giai đoạn: Incremental, Evolutionary
  - Các mô hình tuần tự: Waterfall, V model
  - Lập bản mẫu nhanh (Rapid Prototyping)
  - Các mô hình vòng lặp: Spiral
  - Các mô hình linh hoạt (Agile Models): eXtreme Programming

Các mô hình linh hoạt (Agile Models)

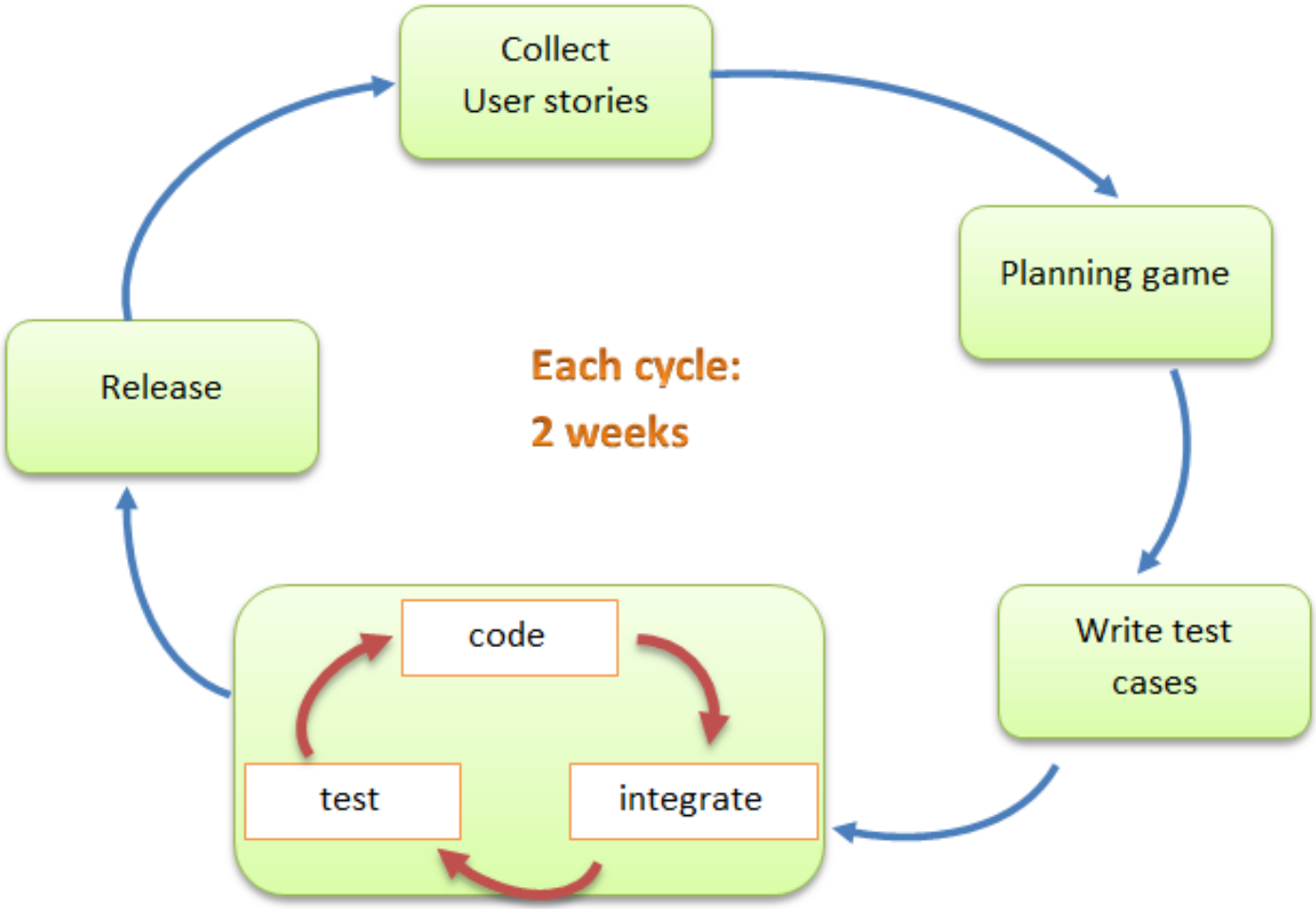
- **Lập luận cơ sở**
  - Giảm rào cản truyền thông
    - Người lập trình giao tiếp với khách hàng
  - Giảm tiếp cận nặng nề với tài liệu
    - Việc lập tài liệu thì tốn kém và giới hạn sử dụng
  - Có niềm tin giữa con người
    - Không cần thiết phải có các mô hình xử lý thật thu hút để thuyết phục cái sẽ làm
  - Đáp ứng được cho khách hàng
    - Hơn là tập trung vào việc ký hợp đồng
- **Điểm yếu**
  - Tin tưởng vào trí nhớ của người lập trình
    - Khó bảo trì
  - Tin tưởng vào truyền thông bằng miệng
    - Có thể thiếu rõ ràng
  - Chấp nhận duy nhất khách hàng đại diện
    - Các quan điểm khác nhau thì không thể đưa ra
  - Kế hoạch chỉ lập trong thời gian ngắn
    - Không có tầm nhìn xa

Lập trình cực độ XP (Extreme Programming)

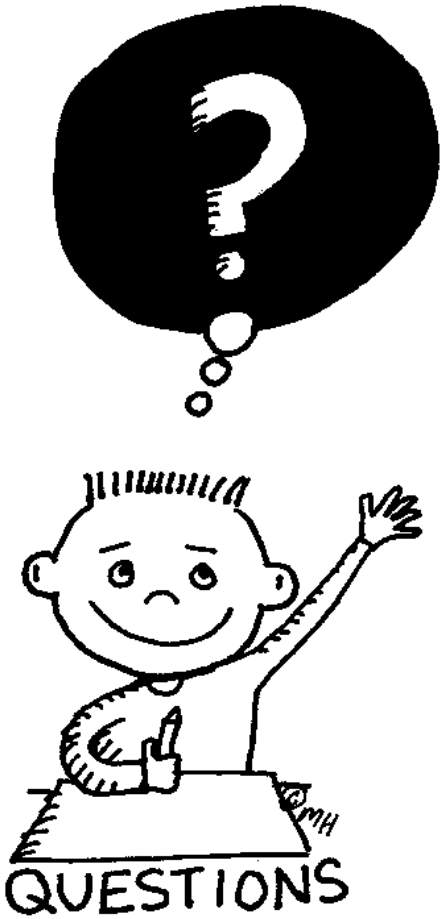
E.g. Lập trình cực độ
(XP - Extreme Programming)

- Thay vì viết đặc tả yêu cầu, thì sử dụng:
  - User story cards (Bản chức năng người dùng)
  - Khách hàng trực diện
- Lập trình cặp đôi (Pair Programming)
- Phát hành nhặt
  - E.g. mỗi 3 tuần
- Trò chơi kế hoạch (Planning Game)
  - Chọn lựa và đánh giá các user story cards vào lúc bắt đầu mỗi đợt phát hành
- Viết bản kiểm thử trước viết code
- Mã lệnh chương trình được thiết kế lập tức
- Tương tác liên tục
  - Tích hợp và kiểm thử mã lệnh vài lần trong một ngày

Lập trình cực độ XP (Extreme Programming)



http://cit.ctu.edu.vn


QUESTIONS