Introduction to Web Engineering SENG2050/6050

Requirement Engineering

Requirement Engineering

- *Requirements Engineering (RE)* – the principles, methods, & tools for eliciting, describing, validating, and managing project goals and needs.
- Given the complexity of Web apps, RE is a critical initial stage, but often poorly executed.
- What are the consequences?
    - Inadequate software architectures
    - "Unforeseen" problems
        - Budget overruns
        - Production delays
        - "That's not what I asked for"
    - Low user acceptance

What is a Requirement?

- Definition of *requirement:*
  - 1) Solves a user’s problem
  - 2) Must be met or possessed by the system to satisfy a formal agreement
  - 3) Documented representation of conditions in 1 and 2
- Keys to requirement definition:
  - Negotiation
  - Scenario-based discovery
  - Clear definition of context and constraints

Two Types of Requirements

- Divide them into two groups:
  - *Functional* – describes the capability’s purpose (e.g., the ability to transfer money between user accounts.)
  - *Non-functional* – describes the capability’s properties (e.g., the Home Page must load within 5 seconds on a dial-up connection.)

Stakeholder

- Customer
- User
- Developer
- For Web apps extremely relevant:
    - Content providers
    - Domain experts
- Goals:
    - Requirements on a higher level of abstraction.
    - Means to define a **shared vision.**

Examples

- *Web app must be available on Sep. 1, 2015. (Customer)*
- *Web app must be able to handle up to 2500 concurrent users. (Customer, quality related)*
- *J2EE should be used as a development platform. (Developer)*
- *Data transactions must be secured. (User, quality related)*
- *The user interface must allow having different layouts for different groups of customers. (Customer)*
- *An arbitrary user must be able to find the desired product within 3 minutes.*
- *An arbitrary user must be able to find the desired product within 3 clicks.*

Requirement Engineering

- Not by simple questionnaires.
- Requirements are a result of a joint learn and consensus finding process
- Various methods:
  - Scenario based
  - Multi-criteria decision processes
  - Moderation techniques
  - Interviews
  - Document analysis

Elicitation & Negotiation

- Identify and involve (if possible) the *stakeholders*
    - Those that directly influence the requirements
    - Customers, users, developers
- What are their expectations?
    - May be misaligned or in conflict.
    - May be too narrowly focused or unrealistic.
- Why is the web application being developed in the first place?

Challenges with Stakeholders

- Users don’t know what they want
    - They may not know what is technically possible (may expect too much, or not expect much)
- Lack of commitment
    - Sometimes they have other work on their minds
        - Deadlines that are more important than a system that isn’t going live for 6 months
- Ever-expanding requirements
    - This relates to users not knowing what they want
    - When they see what is possible, they want more
- Communication delays
    - Client only works Monday, Tuesday, and Wednesday
    - Developer only allocated Wednesday, Thursday, and Friday to the project
- Users don’t take part in reviews
    - Again they may be too busy
- Users don’t understand the process
    - May ask for features to be implemented mid-sprint, or after some key related functionality has already been implemented

Challenges with Developers

- Users and engineers/developers speak different “languages”.
- The tendency to fit the requirements into an existing model
- Engineers & developers are also asked to do RE, but sometimes lack negotiating skills and domain knowledge.

Specifics in Web Engineering

- Is RE for the Web really that different than RE for conventional software?
- Top 6 distinguishing characteristics

  1) Multidisciplinary teams

  2) Unavailability of stakeholders

  3) Rapidly changing requirements & constraints

  4) Unpredictable operational environment

  5) No manual for the user interface

  6) Content Management

Requirements Gathering

- Where do the requirements come from?
  - Project initiator
  - Online Survey
  - Feedback
  - Similar products
    - We have this old program that only runs on Windows 95. Can you make it into a web application?
      - Functionality is already well-defined
  - Etc.

Additional Requirement Types for WE

- Data Requirements
  - How is information stored and managed?
    - Cloud?
    - Local database?
    - 4 servers in different geographical locations that synchronize hourly, etc.?
- Interface Requirements
  - How will the user be interacting with the application?
    - Will they use their phones, a desktop, or both?
- Navigational Requirements
  - How will the user be navigating through the application?
    - Will they bookmark sections of the application?
- Personalization Requirements
  - Should the user be able to apply themes
- Transactional Requirements
  - How the application behaves internally?
    - Will data be saved to disk after every user action, or will we use some top-secret method for saving data

Non-Functional Requirement Types

- Quality
  - Functionality, Usability, Portability
  - Reliability, Efficiency, Maintainability
- System Environment
- User Interface
    - Self-explanatory & intuitive
    - Usage-centered design
- Evolution
- Project Constraints

Requirements Analysis

- Functional outcomes
  - What are the processes the system is going to perform?
- Data outcomes
  - What types of data must the system handle?
- Operational outcomes
  - What are the operational constraints on the system?
    - Limited server resources available
    - Limited bandwidth, etc.

Requirements Analysis

- Output specification
  - Usually a wordy description of all the processes the system should support, and the high-level business components involved
  - Includes descriptions of the “actors” (people and third-party systems) involved
  - Written in the language of the client
  - Supplemented with diagrams describing the relationships between processes and/or actors

Requirements Analysis

- Specification includes client imposed constraints
  - Types of hardware the application must run on
  - Legacy systems the application must support/migrate
  - Response times – e.g. the server must respond to a search request within 2 seconds under “normal” load
    - How do you define normal load?
  - Scalability – how will the system handle higher-than-normal load?
  - Security requirements – Cinema booking system vs. e-tax system

Requirements Analysis

- Some constraints are influenced by the types of users identified
  - E.g. do we expect users to access the application using their smart phones?
- Specification can also contain an initial testing strategy
  - E.g. Internet banking is deemed successful if users can transfer money between their accounts without the response time and scalability constraints

Requirements Analysis

- Requirements Analysis is usually performed in a top-down fashion
  - Identify very general business processes
  - Refine each process into its component steps
  - Identify components common to many processes and abstract
  - Use diagrams to show relationships between processes and components

Validation

- Iterative feedback from stakeholders is essential
  - Is the requirement feasible?
  - Do the results meet stakeholders’ expectations?
- Common tools:
  - Online surveys
  - Usability testing: cognitive walkthroughs & prototypes

Requirements Management

- Requirements are not static and changes are natural.
- Requirements management includes
  - Adding new requirements
    - Businesses can change during the development lifecycle
    - The application should change with them
  - Modifying existing requirements
  - Management of inter-dependencies
    - If we stopped offering Echo360, the EchoCenter link in blackboard will become redundant

Documentation – Traditional RE

- 4 categories of notation
  - *Stories* – Plain-language scenarios; understandable to non-technical persons.
  - *Itemized Requirements* – Plain-language lists of requirements
  - *Formatted Requirements* – Accurately-defined, but allow for plain-language descriptions
    - Ex. Use case scenarios in UML
  - *Formal Specifications* – Expressed in formal syntax & semantics; rarely used in Web applications.

Documentation – RE for Web Apps

- So, what's best for a Web development project?
  - Formatted requirements (i.e. use cases) and stories are heavily used.
  - Low to medium accuracy is suitable for Web apps; formal specifications very rarely required.
  - Keep effort for eliciting and managing requirements low.
  - Scalability is (most likely) important.

Good Requirements Specifications

- Correct
  - Correspond to actual need
- Unambiguous
  - Can be interpreted only in one way
- Complete
  - Any external imposed requirement should be included
- Consistent
  - Conflicting requirements should be avoided

Good Requirements Specifications

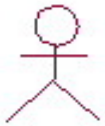
- Ranked for importance and/or stability
  - Requirements are not equally important
  - Requirements are not equally stable
- Verifiable
  - It’s possible to use a cost-effective process to check it
- Modifiable
  - Can be restructured quickly
  - Adopt cross reference
  - Requirements are clearly separated
- Traceable
  - Can be tracked from originating design documentation

Use-Case Diagrams

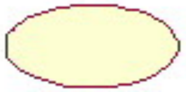
- A use case describes the interaction between External users of a software product (actors) and the software product itself
- Describing a set of user scenarios
- Capturing user requirements
- Contract between end user and software developers

Use-Case Diagrams

- **<u>Actors:</u>** A role that a user plays with respect to the system, including human users and other systems. e.g., inanimate physical objects (e.g. robot); an external system that needs some information from the current system.
- **<u>Use case:</u>** A set of scenarios that describing an interaction between a user and a system, including alternatives.
- **<u>System boundary</u>**: rectangle diagram representing the boundary between the actors and the system.

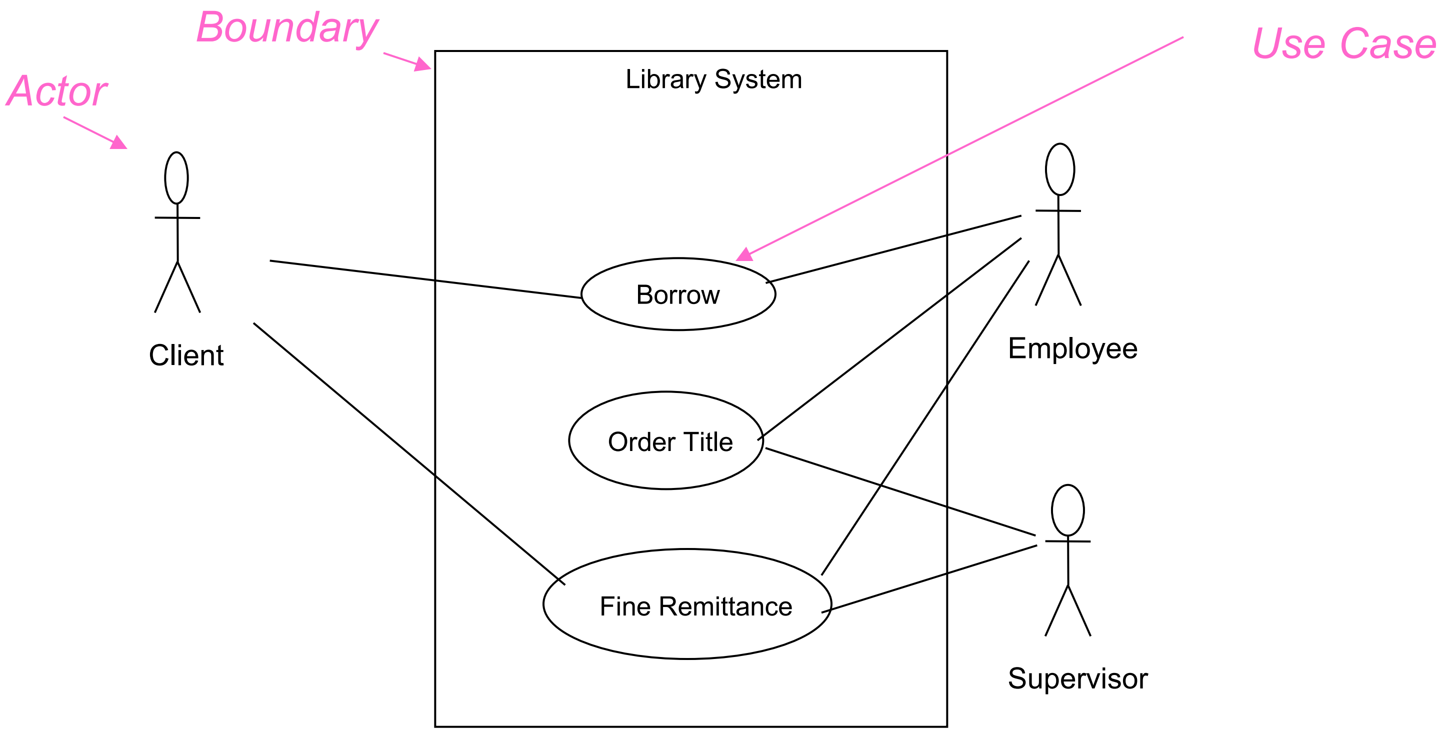


Actor



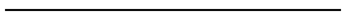
Use Case

Use-Case Diagrams

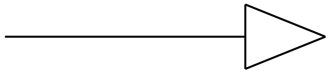


Use-Case Diagrams

- **Association:** communication between an actor and a use case; Represented by a solid line.

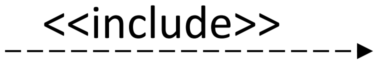


- **Generalization:** relationship between one general use case and a special use case (used for defining special alternatives); Represented by a line with a triangular arrow head toward the parent use case.

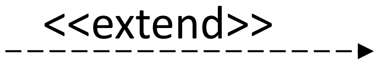


Use-Case Diagrams

- **Include:**
  - a dotted line labeled <<include>> beginning at the base use case and ending with an arrow pointing to the include use case.
  - The include relationship occurs when a chunk of behavior is similar across more than one use case.
  - Use “include” instead of copying the description of that behavior


<<include>>

- **Extend:**
  - A dotted line labeled <<extend>> with an arrow toward the base case.
  - The extending use case may add behavior to the base use case.
  - The base class declares “extension points” to define behavior that can be extended


<<extend>>

Use-Case Diagrams

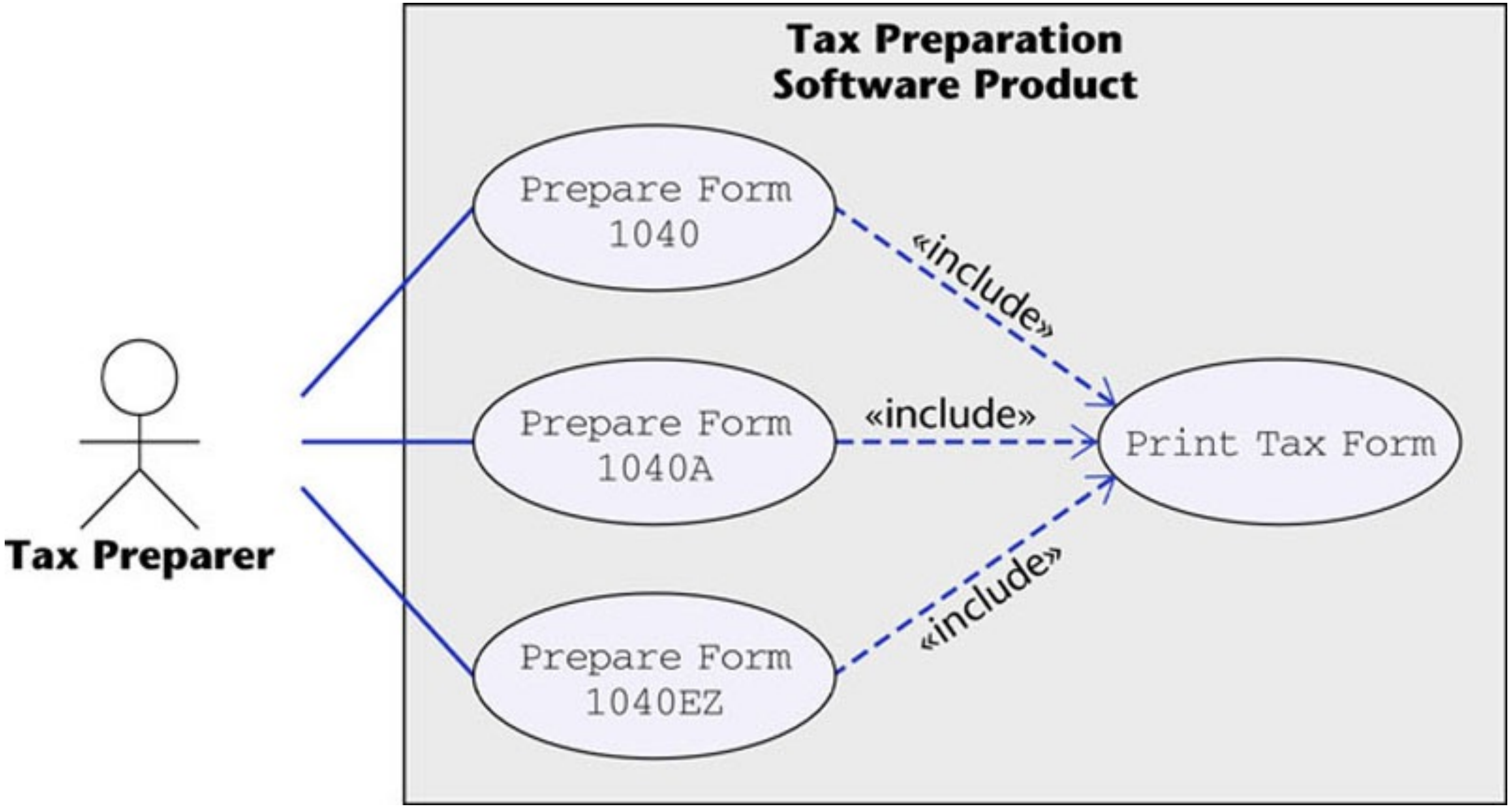


Figure 16.12

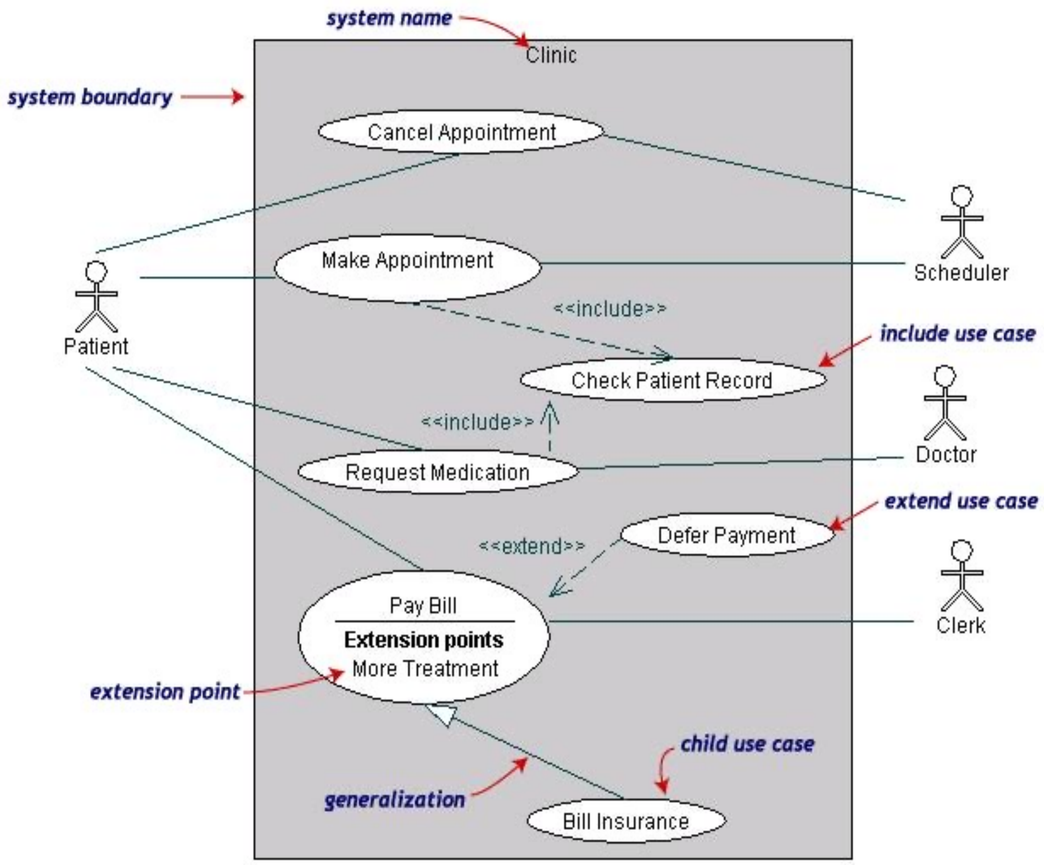
The McGraw-Hill Companies, 2005

Use-Case Diagrams

- Both **Make Appointment** and **Request Medication** include **Check Patient Record** as a subtask (include)

- The **extension point** is written inside the base case **Pay bill**; the extending class **Defer payment** adds the behavior of this extension point. (extend)

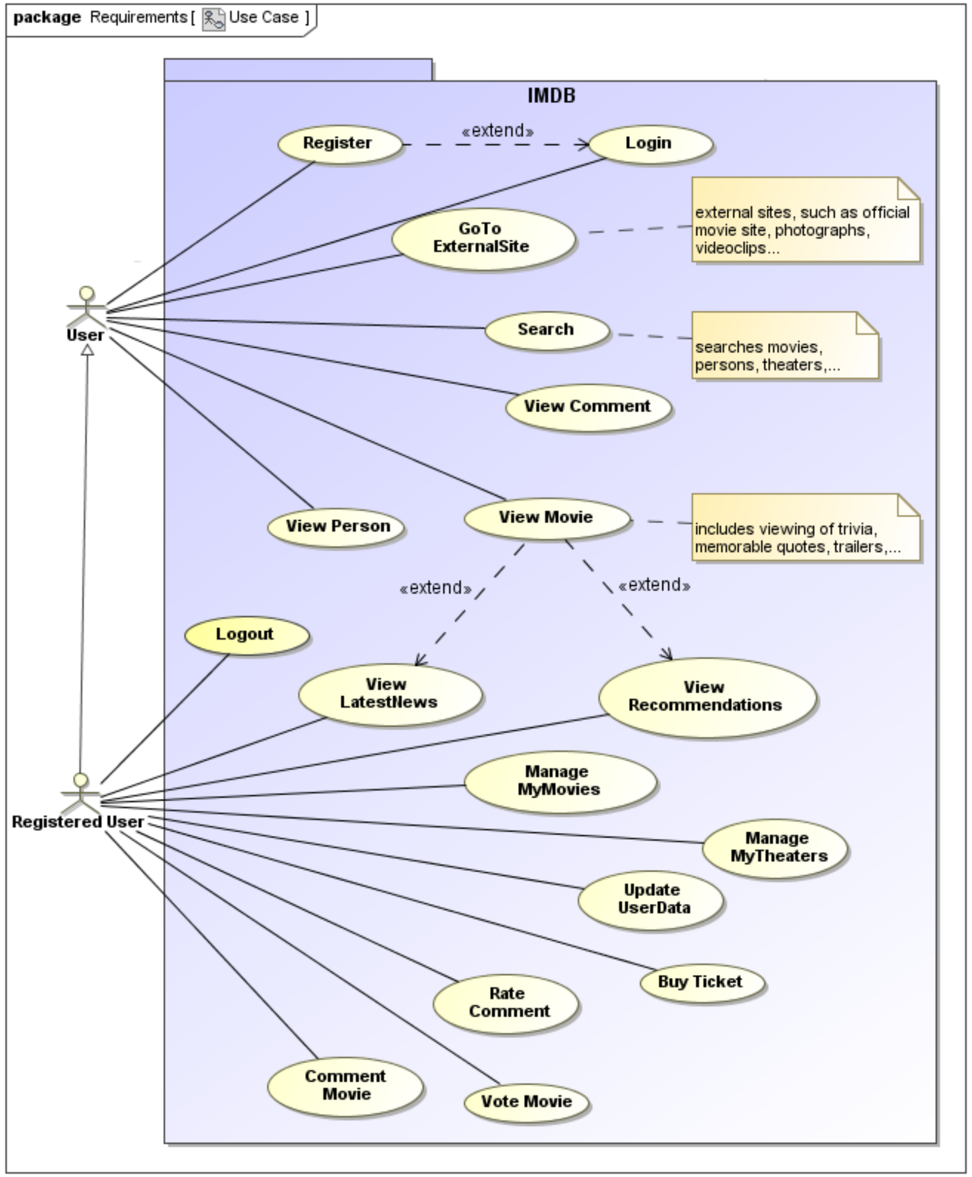
- **Pay Bill** is a parent use case and **Bill Insurance** is the child use case. (generalization)



(TogetherSoft, Inc)

Use-Case Diagrams

- IMDB



http://uwe.pst.ifi.lmu.de/exampleIMDB.html

UML Modelling for Web App

- UWE (UML based Web Engineering) http://uwe.pst.ifi.lmu.de/
- WebML https://en.wikipedia.org/wiki/WebML
- Various variation and extension of the UML.

Summary

- Know your Audience & Objectives
  - Balancing stakeholder interests
  - Focus on high-level requirements first.
- Elicitation & Negotiation is a learning process
- RE requires flexibility
  - Iterative changes should be expected
  - Be sure stakeholders understand this!
- Clear documentation is critical

THE END

QUESTIONS??

THANKS!!