CS 445 – Software Engineering Final

Instructions - You've heard of the mystery of the 39 steps? Here are the 39 questions with 8 steps – and hopefully no mystery to any of you:

1. Each of you is to take ownership of and answer two questions, AS WELL AS question #39 (on Pivotal Tracker: place your name as owner and press START). First come, first served! Fork and clone the final project on GitHub.  
   Because each of you will 'own' two questions, there is little risk of major conflict EXCEPT for the refactor step (see #5) and for question #39. Work out among yourselves what the proper steps should be (self-organize!!)
2. You must collaborate on all but question #39, and you must list your collaborators both in pivotal tracker and in your question document
   1. Every questions must list a minimum of two collaborators
   2. Everyone must collaborate on a minimum of two questions (not your own)
   3. You may request specific collaborators on Piazza
   4. You may request to be a collaborator for questions of which you are NOT an owner on Piazza also
   5. The owner of the question decides who they will accept as collaborators
   6. When the answer is complete, press FINISH on Pivotal Tracker, and make sure you've made it available on GitHub for review
   7. Chore: Collaboration should be recorded in a Hangout (provide link in document)
3. Each question (save #39) must be peer-reviewed, and all reviewers must be listed both in pivotal tracker and in your question document
   1. You must make your answer available on GitHub for your reviewers
   2. Every question must have a minimum of 3 reviewers
   3. Everyone must review a minimum of 3 questions (not your own)
   4. You may request specific reviewers on Piazza (they don't have to agree)
   5. You may sign up to be a reviewer on someone else's question on Pivotal Tracker
   6. The potential reviewers must agree to meet physically or remotely in order to carry out their reviews
   7. Anyone who can 'attend' the meeting is permitted to be a reviewer, but their participation must be documented
   8. The review summary must be included in the question document and must include the responses of each reviewer: if the answer is endorsed, the reasons must be given, if the answer is endorsed with reservations, the proposed changes must be listed (e.g., format, spelling or grammar corrections), if the answer is rejected, the reasons why and proposed changes to make it acceptable must be listed (e.g., answer is vague, incomplete, imprecise, incorrect, rambling, etc. if criticism applies to only portion(s) of answer, particular portion(s) must be cited)
   9. If the answer is accepted, press ACCEPT, if accepted with reservations press ACCEPT after the owner has made the trivial corrections, if rejected, press REJECT.
   10. Chore: Reviews should be recorded in a Hangout (provide link in document)
4. If your answer is accepted, you will post a message on Piazza saying that it is complete, and that in 24 hours you will perform your final merge/upload to GItHub and set it to DELIVER (I realize this is a departure from the usual cycle, but it makes sense in this case). If your answer is rejected, reset your question status to START, and discuss what you must do with your collaborators: follow steps 2-4 again. Note that your document must be available on GitHub.
5. Each of you has permission to examine any answer and REFACTOR it if necessary, but you must give a clear reason for doing so, you must clearly mark your changes, and your modifications must serve a real purpose that makes a genuine and significant contribution to the answer, and does not serve as mere window-dressing on your part. Note that the refactored document must be made available in a timely manner on GitHub.
6. Once the status of the question is set to DELIVER, no other modifications can be made, and I will review and grade the answer.
7. Chore: Question #39 will be answered by all of you, without other collaboration, review or refactoring. You will have to communicate with each other to determine when the status of #39 is ready to be set to DELIVER. You must make sure that you don't stomp on anyone else's answer when uploading your answer for #39.
8. Chore: Once ALL the answers have been set to DELIVER, one or more of you should prepare a cover sheet and merge all the answers (final version) into a single Word document (do NOT delete or modify the other documents).

Grading

1. Each question will be graded on a 100 point scale, then the scores for all questions will be averaged:
   * Twenty points for spelling, grammar, format and general structure
   * Twenty points for correctness and precision
   * Twenty points for completeness
   * Twenty points for clarity
   * Twenty points for overall quality: your writing style should be persuasive and capture the interest of the reader, any accompanying drawings or illustrations should be properly rendered
   * Submit thoughtful and carefully reasoned answers, NOT canned, generic responses gleaned from google or Wikipedia!
2. Each student will be graded on a 100+ point scale
   * 30 points for answering two questions + #39 (10% of each question's score)
   * 20 points for two collaborations – include link to Hangout (10% of each question's score)
   * 30 points for 3 reviews – include link to Hangout (10% of each question's score)
   * 10 points for proper management of and communication on Pivotal Tracker and Piazza
   * 10 points for proper interactions with GitHub
   * +2.5 points for any chores performed (note #7 and #8 above, as well as Hangout setups: #2g and #3j) – Setting up the GitHub repo was the first chore completed by Ben. Please let me know if anyone else should get credit for this as well (i.e., if anyone else assisted)
   * +2.5 points for any significant refactors you perform on someone else's question (step 5 above)
   * -2.5 points for any significant refactors performed on YOUR question (step 5 above)
3. Your final grade will be the average of the Class Performance grade (#1) and your individual contribution grade (#2)

QUESTIONS

1. Note that the ACM/IEEE Code of Ethics and Professional Practice is provided in Appendix A of this document. With respect to Therac-25, the Ariane 5 rocket and the FBI Virtual Case File
   1. Describe what happened.
   2. Describe the causes leading up to a.
   3. Describe how a. could have been prevented.
   4. Which A/I CEPP principles were violated in each of the above projects.
   5. How can we ensure that developers, managers and clients adhere to these principles, and that violators are disciplined? What, if any, possibilities might there be?
2. Describe the lifecycle of a fictitious project that is being developed by a distributed team via local and remote git interactions (e.g., GitHub) from initial set up through completion. Be thorough: describe and include ALL interactions that would normally occur.
3. Answer the following with respect to PivotalTracker:
   1. Describe the typical lifecycle of a fictitious user story from Start to Acceptance.
   2. Explain the meaning of each entry and describe its usage (including estimation, requestor, owner, activity, etc.).
   3. Discuss its role in project communication.
4. Answer the following with respect to Google Hangouts On Air (David: you can take this one if you want)
   1. Explain how to get started with Google Hangouts On Air.
   2. Explain how to set up a Google Hangout online meeting.
   3. What, if anything, can you do if you want to join a hangout but the maximum number of people have already joined?
5. Explain the following with respect to effort estimation
   1. Describe (in general: categorize) the techniques employed in a Plan and Document project.
   2. Describe how the objectives of Plan and Document estimation differ from those of Agile estimation.
   3. Describe how Planning Poker is used to estimate a user story.
   4. How is Planning Poker similar to/different from the way tasks are estimated on AgileVentures?
6. With respect to pair programming:
   1. What is it?
   2. What are its advantages?
   3. What are its disadvantages?
   4. What is promiscuous pairing? Should you do it?
   5. Discuss the following statement by Fred Brooks: "You cannot win if your team loses, and you cannot fail if your team wins".
7. Kent Beck, one of the original signers of the Agile Manifesto, is the founder of Extreme Programming, one of the major versions of Agile process development.
   1. Why is it called 'Extreme' programming: how does this relate to the best practices of Plan and Document developments?
   2. Describe the (current) thirteen practices of Extreme Programming.
   3. Do these practices adequately describe Agile process methodology?
8. Answer the following questions with respect to the Waterfall, Incremental, RAD, Spiral and Agile process models.  
   Discuss with respect to ALL aspects of a project that would be considered such as project size, type, duration and technical challenge, team characteristics such as skill level and experience, and client characteristics such as availability, flexibility (or not), sector (private, government, . . . ), technical astuteness, etc..
   1. What are the major features of the process model?
   2. When would you recommend using this process model?
   3. When would you NOT recommend using this process model?
9. The Software Engineering Institute's Capability Maturity Model is used to evaluate an organization's software development practices (i.e., process maturity). Illustrate the practices that would have to be in place to achieve each of the following levels. Hint: discuss with respect to Key Process Areas (KPAs).
   1. Initial
   2. Repeatable
   3. Defined
   4. Managed
   5. Optimizing
10. Give two examples of each of the following Software Engineering productivity mechanisms:
    1. Clarity via conciseness,
    2. Synthesis
    3. Reuse
    4. Automation via Tools
11. Answer the following:
    1. Define SaaS
    2. Define SOA
    3. Define Cloud Computing
    4. Describe the relationships among each of the above
12. For each of the following workstyles: Rational Extrovert, Rational Introvert, Intuitive Extrovert and Intuitive Introvert:
    1. Describe the typical characteristics of each
    2. Describe how they would typically interact/communicate with others
    3. Describe how you should interact/communicate with them in order to be effective
13. Answer the following with respect to scheduling:
    1. Describe what a PERT chart is, what it shows and what are its limitations.
    2. Describe what a Gantt chart is, its similarities to a PERT chart and its differences
    3. Discuss how scheduling is handled in an Agile project
14. With respect to the illustration found at time marker 5:50 of the ESaaS Ch. 2.4: 3-tier shared-nothing architecture & scaling lecture lecture in SPOC I of the typical physical hardware configuration for developer, medium scale deployment and large-scale deployment of a SaaS application, show how the these hardware configurations map to the various parts and layers of the logical architecture, as well as the implementation software. In addition to your explanatory text, draw and label a complete diagram depicting the hardware-logical-software layers, scan it, and paste it into your document.
    1. at the client-server layer.
    2. at the three-tier shared nothing layer.
    3. at the model-view-controller layer within the application tier.
    4. at the software implementation layer within the model-view-controller layer.
15. Pipes-and-Filters, Client-Server, Peer-to-Peer and Layered are some of the major architectural styles used today.
    1. Define each.
    2. Provide an example of where/how each would typically be used.
    3. Discuss the advantages and disadvantages of each.
16. Answer the following:
    1. Define and describe the protocols on which HTTP is dependent.
    2. What is the significance of the IP address 127.0.0.1?
    3. What is a port and how/why is it used? What port is used by Web servers?
    4. Explain what a URI is and describe its basic structure.
    5. Define and describe DNS. Would it be possible to make an internet connection without it?
    6. Explain how HTTP is a 'stateless protocol'. If this is so, how can a server know that two different requests are actually from the same user?
17. Answer the following:
    1. Describe each of the four different HTTP methods and explain when/how they are used.
    2. What is REST and how is it implemented?
    3. Why is REST such an important organizing principle in Web applications?
    4. How are the HTML tag attributes id and class used?
    5. How are CSS selectors used?
18. The three pillars of Ruby are: Everything is an object, every operation is a method call . . . , and all programming is metaprogramming.
    1. Define and explain each of these.
    2. Provide two examples of each.
    3. Define and discuss 'duck typing' and its relationship to mix-ins.
19. Answer the following with respect to the Ruby language:
    1. Do objects have types? Do the variables that point to objects have types?
    2. What is the difference between the variable @name and the variable @@name?
    3. What happens when a method is invoked on an object, but the method cannot be found in the receiver's class?
    4. What iterator method is available to all collection-like Ruby classes?
    5. Define and discuss the scoping rules for local variables.
    6. Define and discuss 'poetry' mode.
    7. Define and discuss closures.
    8. Explain the purpose of the yield statement.
20. Answer the following:
    1. Define framework and explain its benefits.
    2. Is a framework the same or different from a toolkit? Explain.
    3. How is an architecture or design pattern related to a framework?
    4. Discuss convention over configuration with respect to the Rails framework.
21. Answer the following with respect to the Rails framework:
    1. When creating a form, which two things must be specified in a form\_tag in order for the correct controller action to receive it?
    2. How does a redirect work, and when should it be used?
    3. Describe the 'flash', and how it can be used in the situation described in b.
    4. Describe the session hash, as well as how and when it is used.
22. Answer the following:
    1. Define and describe the SMART guidelines for user stories.
    2. Show and explain the proper format for a user story as developed by Connextra.
    3. What is a storyboard and what is its relationship to a Lo-Fi User Interface sketch?
    4. Explain what is meant by 'Code Smells'.
    5. Describe and explain the SOFA guidelines for creating a well-written method.
23. Appendix B contains the contents of a Cucumber feature file for the RottenPotatoes application.
    1. Show the available regular expression steps from web\_steps.rb that could be used to match the steps in the feature, and explain how they match the feature step (i.e., explain how the regular expression is constructed).
    2. Show the additional regular expression steps in ruby that would have to be added, and explain how they would match.
24. Answer the following:
    1. What is BDD and what is its purpose?
    2. What tool do we use to transform user stories into acceptance and integration tests?
    3. What tool do we use to simulate the user interacting with the browser to perform steps of a scenario?
    4. Define and explain the difference between validation and verification
    5. Explain what is meant by regression testing and its relationship to cycling.
    6. Define and explain the use of stubs and drivers for testing in the context of Plan and Document testing methods.
    7. Define and describe the following types of Plan and Document integration testing methods: Modified bottom-up, Modified Top-down, Modified Sandwich and Big Bang (Make sure you discuss it with respect to stubs and drivers).
25. Answer the following:
    1. Define and explain the meaning of FIRST.
    2. Describe and explain what is meant by TDD aka Red-Green-Refactor?
    3. Define and describe RSpec and explain what it does.
    4. Explain what code coverage means and discuss the various ways in which it can be measured.
    5. Explain what is meant by the following Plan and Document testing methods: black box, white, grey box, mutation and fuzz.
    6. Define exhaustive testing and explain how it can be implemented (or not).
26. Answer the following with respect to RSpec:
    1. What is an expectation and how is it implemented?
    2. What is a matcher? Give two examples of built in matchers.
    3. Define and describe seam and explain how it is used.
    4. Define and describe mock and explain how it is used.
    5. Define and describe stub and explain how it is used.
    6. Why is it so important to completely isolate the unit under test?
    7. How do we insure that the unit under test is completely isolated?
27. Some Plan and Document process models employ Formal Methods. The Cleanroom process model, described in Appendix C is one of these.
    1. What would be the advantages or disadvantages of replacing unit testing with inspection and formal verification (is it practical? would it be sufficiently accurate?)
    2. Discuss the following: According to the Cleanroom philosophy, using the operational profile would concentrate testing on the most common modes and usages, and therefore should theoretically detect the most integration/system faults. However, a different way of looking at this is that transition situations and seldom used options/paths are often the most complex and failure prone, and that testing should focus on these cases instead.
    3. Although Cleanroom may seem to be the antithesis of Agile, are there any similarities between its practices and any Agile practices (Are there any 'best practices' they hold in common)?
28. Appendix D consists of a blend of the pseudocode and actual ruby code for a method.
    1. Draw and label a flow graph for the given method.
    2. Determine the following with respect to the flow graph: number of regions, number of predicates, number of nodes and number of edges.
    3. Compute the Cyclomatic Complexity for the given method and show how it is derived three separate ways.
    4. Derive and list the basis paths set for the given method.
    5. Explain how this information can be used when designing unit tests for the given method.
29. Answer the following:
    1. Define and describe the DOM
    2. When user interaction generates a browser event, what can JavaScript code do with respect to the DOM?
    3. Define and describe AJAX.
    4. What tool do we use to test JavaScript and AJAX?
    5. What is the advantage of using AJAX to make HTTP requests to a Web server?
    6. Define and describe 'graceful degradation'.
30. Answer the following:
    1. Describe and discuss four language features that JavaScript shares with Ruby.
    2. Does JavaScript have classes? Explain.
    3. How should JavaScript code be related to HTML page markup?
    4. Define and describe JSAPI.
    5. Define and describe jQuery, and explain how it is used.
31. Scrum, like Extreme Programming, is another 'flavor' of Agile.
    1. Describe the activity for which this methodology is named that is performed daily by Scrum team members
    2. Define and describe a sprint. What does is correspond to in the Extreme Programming process model?
    3. Define and describe Scrum master.
    4. Define and describe Product Owner.
    5. Define and describe a Two-Pizza team, and discuss the pros and cons of its suggested size.
32. Answer the following with respect to version control:
    1. Define and describe the fork-and-pull management model.
    2. Should this scheme be used for public repositories? for remote collaboration?
    3. Define and describe a branch and explain what it is used for.
    4. Define and describe a crisscross merge.
    5. If you attempt a push and it fails with an error message, does this always mean that there is a merge conflict between your repo's version of some file and the origin repo's version of it?
33. Answer the following:
    1. Define and describe the five R's of bug fixing.
    2. Define and describe Intel's policy of 'constructive confrontation'.
    3. Explain what is meant by 'disagree and commit'.
    4. Describe the goals and characteristics of a design review.
    5. Describe the goals and characteristics of a code review.
    6. What is the purpose of collecting project metrics? Name at least two commonly gathered metrics.
    7. Describe the goals and characteristics of a project post mortem and explain its importance.
34. Answer the following:
    1. Define and describe a partial, and explain how to use it.
    2. Define and describe advice and provide an example of it.
    3. Can advice change the flow of execution in a Rails application? Give examples.
    4. Define and describe a join point
    5. Define and describe pointcut, and explain how it is related to join points.
    6. Define and describe AOP and explain what makes up an aspect.
    7. What are the advantages and disadvantages of using AOP?
35. Answer the following:
    1. Explain the difference between authentication and authorization.
    2. What is the relationship between third-part authentication and single sign-on?
    3. Give an example of an authorization provider.
    4. What would you use to cleanly indicate that a user has been authenticated in a Rails app?
    5. How can you identify model attributes that should be excluded from mass assignment?
    6. What is an association, and what kinds of associations are possible among application entities?
    7. Identify and describe the mechanism provided by relational databases to model associations.
36. Answer the following:
    1. What is the type of association in Rails that handles indirect relationships?
    2. Given:

Class Moviegoer

has\_many:reviews

How could we establish an association that could give us the movies that a moviegoer has reviewed.

* 1. What is the RESTful way to create routes for associations?
  2. Show the RESTful way to create a route for movies and its associated item reviews
  3. What should be a concern if a movie is then deleted?
  4. What is a named scope, and how is it used?
  5. Does using a named scope incur performance penalties? Why or why not?

1. Answer the following with respect to Design Patterns:
   1. Define and describe SOLID as modified for Ruby on Rails applications.
   2. In contrast, describe and discuss four problems that thwart productivity.
   3. Model-View-Controller is considered to be a meta-pattern (aka Aggregate Design Pattern) that implements (at a minimum) the Observer (describing the relationship between the view and model), the Composite (by creating a 'tree' of views) and the Strategy (describing the relationship between the controllers and the views) design patterns. Describe and discuss its use in Ruby on Rails applications, and note in which tier it is found.
   4. Define and describe one creational pattern, one structural pattern and one behavioral pattern.
2. Answer the following:
   1. What is the main performance challenge for a SaaS app, and what measures can be taken to meet it.
   2. Define and describe PaaS, and explain how it is used.
   3. What would cause a project to abandon PaaS, and what can be done to circumvent or delay this.
   4. Why are releases more challenging for SaaS then for shrinkwrapped software?
   5. Define and describe the principles of least privilege, and explain how it relates to security.
   6. Define and describe the principle of psychological acceptability, and its relationship to security as well as the user experience.
3. Lessons Learned (Everyone Answers!): Most organizations carry out a postmortem (i.e., after delivery and acceptance) analysis of a project. In it, they assess all aspects of the project including products, process and resources with the intent of identifying both what well in the project and what areas should be improved upon in future projects. Before a formal postmortem, each team member does a self-assessment, called a Lessons Learned document. Write your own sections of our Lessons Learned document for the class by summarizing:
   1. What were the most important (to you, personally) concepts, principles, skills, tools, etc. that you learned/mastered throughout the semester.
   2. Note any concepts, principles, skills, tools, etc. that you would have liked to learn, but that we didn't cover.
   3. List all everything that you feel you did especially well this semester (master specific tools, languages, skills, etc.; contribute to the project in a specific way, etc., learn and implement specific things quickly, etc.).
   4. List anything that that you feel that you didn't do as well, or that didn't go as well as you hoped (getting a list of these items will help me to better schedule material for future semesters).
   5. THANK YOU!
4. Extra Credit (+10 added to your personal score if correct) If you are interested in learning this, I will be happy to review it with you on Monday: An important skill for a software engineer to have is to be able to learn and master new tools quickly. A common tool used by many organizations is a trade study. See Appendix E for an explanation and example. The following is a Trade Study problem:

Our customer wants an inventory system for her furniture warehouse and sales division

SLOC has been estimated at 1200, and our team's productivity rate is 100 lines per labor month per person, and development people cost $1000 per labor month. Other non-coding tasks have been estimated at 3 labor months.

Our customer has heard of Half-Baked inventory software. The cost is $9000, but it does not have sales support capability, and will take 1 person one labor month for installation and training. Another program (HiPricedSpread) costs $28,000, has sales support; and will take one person 2 labor months for installation and training. She has asked us to perform a trade study to determine whether it would be more cost effective to have us develop the entire system in-house, or to buy the commercial-Off-the-Shelf Software and have us do the installation and training. The criteria for selection will be cost and sales support. Sales support capability is twice as important as cost.

Do a Trade Study comparing in-house development with purchase of the software.

(Hint: make sure that you tally the code estimates and then derive and include any additional costs items given in labor months)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Trade Study Name:** Choose Commercial-Off-the-Shelf Software or Develop In-House | | | | | | | | | | | |
| **Problem Statement:** Select best value based on cost and built-in sales support capability | | | | | | | | | | | |
| **Candidates 🡺** | | | **Half-Baked** | | | **HiPricedSpread** | | | **In-House** | | |
| **Criterion** | **Wgt** | **Model** | **Value** | **Score** | **Wgt\*Score** | **Value** | **Score** | **Wgt\*Score** | **Value** | **Score** | **Wgt\*Score** |
| **Cost** | **1** | **Less is More** | $ |  |  | $ |  |  | $ |  |  |
| **Sales Support** | **2** | **0/1** |  |  |  |  |  |  |  |  |  |
| **Rating** |  |  |  |  |  |  |  |  |  |  |  |

What is your decision? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Appendix A**

ACM/IEEE SE Code of Ethics and Professional Practice

* Software engineers shall act consistently with the public interest
* Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest
* Software engineers shall ensure that their products and related modifications meet the highest professional standards possible
* Software engineers shall maintain integrity and independence in their professional judgment
* Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development maintenance
* Software engineers shall advance the integrity and reputation of the profession, consistent with the public interest
* Software engineers shall be fair to and supportive of their colleagues
* Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession

**Appendix B**

Background: movies have been added to database

Given the following movies exist:

| title | rating | release\_date |

| Aladdin | G | 25-Nov-1992 |

| The Terminator | R | 26-Oct-1984 |

| When Harry Met Sally | R | 21-Jul-1989 |

| The Help | PG-13 | 10-Aug-2011 |

| Chocolat | PG-13 | 5-Jan-2001 |

| Amelie | R | 25-Apr-2001 |

| 2001: A Space Odyssey | G | 6-Apr-1968 |

| The Incredibles | PG | 5-Nov-2004 |

| Raiders of the Lost Ark | PG | 12-Jun-1981 |

| Chicken Run | G | 21-Jun-2000 |

And I am on the RottenPotatoes home page

And I check the following ratings: G PG PG-13 R

And I press "Refresh"

Scenario: sort movies alphabetically

When I follow "Movie Title"

Then I should see all movies sorted by title in increasing order

Scenario: sort movies in increasing order of release date

When I follow "Release Date"

Then I should see all movies sorted by release date in increasing order

**Appendix C**

The Cleanroom Process Model

* Process model that supposedly precludes the introduction of defects
* Uses formal methods to develop each program unit by refining increasingly detailed 'box' structures:

1. Black box (will generate result R after applying function f to input S)
2. State box (given current state, system will transition to new state and generate result R after applying function f to input S)
3. Clear box: Define algorithm that will satisfy requirements of black/state box

* Code is then generated for each program unit
* Unit is inspected: Together, every member of the team inspects each line of code, and then verified using formal proofs instead of unit testing.
* Statistical Usage tests are developed according to usage probability (i.e., the more a section of code will be 'visited' by a user, the more tests are written for that pathway)

**Appendix D**

def index

@all\_ratings = Movie.all\_ratings

IF params[:ratings] != {}

THEN

@selected\_ratings = params[:ratings]

ELSE

IF session[:ratings] != {}

THEN

@selected\_ratings = session[:ratings]

ELSE

@selected\_ratings = {}

ENDIF

ENDIF

if @selected\_ratings == {}

THEN

@selected\_ratings =

Hash[@all\_ratings.map{|rating| [rating, rating]}]

ENDIF

sort = params[:sort] || session[:sort]

if sort == 'title'

THEN

order = {:order => :title}

@title\_hdr = 'hilite'

#@movies = Movie.all(order)

ELSE

IF sort == 'release\_date'

THEN

order = {:order => :release\_date}

@release\_date\_hdr = 'hilite'

#@movies = Movie.all(order)

ENDIF

ENDIF

if params[:ratings]!=session[:ratings] OR

params[:sort]!=session[:sort]

THEN

session[:ratings] = @selected\_ratings

session[:sort] = sort

flash.keep

redirect\_to :ratings => @selected\_ratings, :sort => sort

ELSE

@movies =

Movie.find\_all\_by\_rating(@selected\_ratings.keys, order)

ENDIF

END index

**Appendix E**

Trade Studies

* Process used to make preferred technical decision
  + Considers a variety of alternatives
  + Examines both weaknesses and strengths associated with a particular decision
  + Can satisfy multiple objectives
* Start by determining what decision must be made
* Find and evaluate possible candidates
* Identify/finalize candidates to be used for study – each must represent a viable alternative
* Formulate tradeoffs: express each criterion using a formula and/or range and a weight
  + Within a range/outside a range
  + More is better/less is better
  + Present/not present
  + The weight assigned to a criterion indicates its importance relative to other criteria
  + May use Delphi process to identify criteria and assign formulas and weights
    - Have set of expert stakeholders determine criteria
    - Have each assign weight to each criteria anonymously
    - Discuss if wide variance
    - Repeat last two steps until convergence
* Calculate weighted values for each criterion for each candidate
* Convert each set of values to a utility score
  + Can be normalized, but not necessary
* Make selection
* Evaluate certainty/uncertainty/risk relative to selection

Worksheet:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Trade Study Name:** Choose Best Tomato Variety to Plant in Garden This Summer | | | | | | | | | | | |
| **Problem Statement:** Select best value red paste tomato | | | | | | | | | | | |
| **Candidates -->** | | | Red Diamond | | | Red Ruby | | | Red Garnet | | |
| **Criterion** | **Weight** | **Model** | **Value** | **Score** | **Wgt\*Score** | **Value** | **Score** | **Wgt\*Score** | **Value** | **Score** | **Wgt\*Score** |
| Cost/gram | 1 | Less is More | $8.85 |  |  | $7.95 |  |  | $6.85 |  |  |
| Flavor Rating | 2 | More is More | 4 |  |  | 3 |  |  | 2 |  |  |
| Resistance to Late Blight | 1 | 0/1 | 0 |  |  | 1 |  |  | 0 |  |  |
| Rating |  |  |  |  |  |  |  |  |  |  |  |

Compute weighted score for cost/gram ----

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trade Study Name: Choose Best Tomato Variety to Plant in Garden This Summer | | | | | | | | | | | |
| Problem Statement: Select best value red paste tomato | | | | | | | | | | | |
| Candidates --> | | | Red Diamond | | | Red Ruby | | | Red Garnet | | |
| Criterion | Weight | Model | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc |
| Cost/gram | 1 | Less is More | $8.85 | (8.85-8.85) /8.85 = 0 | 0 \* 1 = 0 | $7.95 | (8.85-7.95) /8.85 = .1 | .1\*1 = .1 | $6.85 | (8.85-6.85)/8.85 = .23 | .23\*1 = .23 |
| Flavor Rating | 2 | More is More | 4 |  |  | 3 |  |  | 2 |  |  |
| Resistance to Late Blight | 1 | 0/1 | 0 |  |  | 1 |  |  | 0 |  |  |
| Rating |  |  |  |  |  |  |  |  |  |  |  |

Compute weighted score for flavor rating ----

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trade Study Name: Choose Best Tomato Variety to Plant in Garden This Summer | | | | | | | | | | | |
| Problem Statement: Select best value red paste tomato | | | | | | | | | | | |
| Candidates --> | | | Red Diamond | | | Red Ruby | | | Red Garnet | | |
| Criterion | Weight | Model | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc |
| Cost/gram | 1 | Less is More | $8.85 | (8.85-8.85) /8.85 = 0 | 0 \* 1 = 0 | $7.95 | (8.85-7.95) /8.85 = .1 | .1\*1 = .1 | $6.85 | (8.85-6.85)/8.85 = .23 | .23\*1 = .23 |
| Flavor Rating | 2 | More is More | 4 | 4/5 = .8 | .8 \* 2 = 1.6 | 3 | 3/5 = .6 | .6 \* 2 = 1.2 | 2 | 2/5 = .4 | .4 \* 2= .8 |
| Resistance to Late Blight | 1 | 0/1 | 0 |  |  | 1 |  |  | 0 |  |  |
| Rating |  |  |  |  |  |  |  |  |  |  |  |

Compute weighted score for resistance ---

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trade Study Name: Choose Best Tomato Variety to Plant in Garden This Summer | | | | | | | | | | | |
| Problem Statement: Select best value red paste tomato | | | | | | | | | | | |
| Candidates --> | | | Red Diamond | | | Red Ruby | | | Red Garnet | | |
| Criterion | Weight | Model | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc |
| Cost/gram | 1 | Less is More | $8.85 | (8.85-8.85) /8.85 = 0 | 0 \* 1 = 0 | $7.95 | (8.85-7.95) /8.85 = .1 | .1\*1 = .1 | $6.85 | (8.85-6.85)/8.85 = .23 | .23\*1 = .23 |
| Flavor Rating | 2 | More is More | 4 | 4/5 = .8 | .8 \* 2 = 1.6 | 3 | 3/5 = .6 | .6 \* 2 = 1.2 | 2 | 2/5 = .4 | .4 \* 2 = .8 |
| Resistance to Late Blight | 1 | 0/1 | 0 | 0 | 0 \* 1 = 0 | 1 | 1 | 1 \* 1 = 1 | 0 | 0 | 0 \* 1 = 0 |
| Rating |  |  |  |  |  |  |  |  |  |  |  |

Sum the scores and select the highest one!

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trade Study Name: Choose Best Tomato Variety to Plant in Garden This Summer | | | | | | | | | | | |
| Problem Statement: Select best value red paste tomato | | | | | | | | | | | |
| Candidates --> | | | Red Diamond | | | Red Ruby | | | Red Garnet | | |
| Criterion | Weight | Model | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc | Value | Score | Wgt\*Sc |
| Cost/gram | 1 | Less is More | $8.85 | (8.85-8.85) /8.85 = 0 | 0 \* 1 = 0 | $7.95 | (8.85-7.95) /8.85 = .1 | .1\*1 = .1 | $6.85 | (8.85-6.85)/8.85 = .23 | .23\*1 = .23 |
| Flavor Rating | 2 | More is More | 4 | 4/5 = .8 | .8 \* 2 = 1.6 | 3 | 3/5 = .6 | .6 \* 2 = 1.2 | 2 | 2/5 = .4 | .4 \* 2 = .8 |
| Resistance to Late Blight | 1 | 0/1 | 0 | 0 | 0 \* 1 = 1 | 1 | 1 | 1 \* 1 = 1 | 0 | 0 | 0 \* 1 = 0 |
| Rating |  |  |  |  | 1.6 |  |  | 2.3 |  |  | 1.03 |

Note that you will get completely different results depending on how you weight each factor!