1. Process Drivers (15 pts)
Darlene has been appointed to develop a software process for WidgetCo’s brand new 4-person software development effort, based on her successful tenure as the company’s system admin in the IT department. The company has a successful track record of building products such as turkey basters and meat thermometers for distribution through grocery stores; now they want to also distribute software for the Computerized Kitchen – user-visible functions like recipe planners, food databases, and product ordering from the stovetop (order when you realize the need!).

Her boss describes the products like this:

“Keep them simple and intuitive. For the first few years, let’s do apps that are modeled after existing kitchen capabilities in their functions and, wherever possible, the user interfaces. The entire line of WC KitchenApps is to have a similar style of user interface, displayed on small monochrome touchscreens or using voice I/O – so think ahead on the UI stuff. User interfaces are to be simple but must be foolproof – the liability possibilities are huge – and will have to work for users all across the country. Once we ship we have a devil of a time getting anything back, so take care that you know what works before it gets to my mother’s kitchen, OK?”

“We’re looking at applications with limited lifecycle implications: documentation delivered to customers is just the back-of-the-package stuff, there is no need for licensing, update/patch management, all that stuff you’ve bothered me with in the IT department. Time to market is not critical, but these apps have to be as reliable as the existing products we sell: turkey basters that last for years without failure. That’s our reputation and brand, and we don’t want some software stuff messing up the sales of our OneDrop Baster line!”

Darlene has been working at WC long enough to know that the grocery store distribution channel has some salient characteristics the company has adapted to: large volume, razor-thin margins on low prices, and fierce competition for shelf-space and customer notice (many sales are made based on only what the potential customer sees on the package.) Responding to this environment, WC has developed a culture based on fine-grained control of their costs, and reasonable adaptability to changing conditions.

After checking into the emerging Computerized Kitchen marketplace, Darlene realizes the computerized interfaces are all extremely well worked out and settled by appliance industry groups, and they will be stable for a long, long time (a stove lasts quite a while!)

1a - (10 pts) What software process should Darlene select as the best (choosing from the McConnell definitions)? Justify your answer as Darlene might have to!

<follow model in Q&A for lecture 1; my result is M.W. (GUI prototyping as the ront ed risk reduction modification). Calculations and discussion in separate paper.>

However, there are many interpretations to be made here and there isn’t really enough info – TA should score according to completeness of rationale, apparent understanding of lifecycle models.

Some elements to dig out:
It’s a new development team, with unsophisticated management (a Sys Admin is not a program manager or architect). User interface and reliability are the critical drivers. Moderate management visibility and re-directability are desired.

A good case can be made for spiral (for the risk reduction phases in the UI), but this has to be tempered by realizing that Darlene is unsophisticated … likely to get bogged down in risk analysis since she hasn't done this before.
After two years of work, the team has shipped successful products and has grown. But, it has been quite hectic, the team is complaining that not all the underlying architecture is as uniform as they’d like, and the boss wants a unified suite with advanced features instead of a variety of standalone tools. (This gold-plating and re-writing is common for a follow-on effort; so common, it has a name: the “second-system effect.” The term was first used by the estimable Fred Brooks in The Mythical Month.)

1b - (5 pts) Darlene estimates the new version will be about 40KSLOC best case, 60KSLOC most-likely, and 100KSLOC worst-case. She looks over her records and figures that each KSLOC costs about $10K to develop. At the same time, she’d like to get her life back, so she wants to send the team off for “process training” and hire a consultant to help improve the process maturity level. How much money can she justify investing in process upgrade if it has to pay for itself on the Version 2 effort alone?

We have 4 different size estimates we might consider: best-case (40K), most-likely (60K), worst-case (100K) and combined (40/6 + 2*(60/3) + 100/6 = 63K). We have only one metric for converting from size to effort ($10K/KSLOC), so we have 4 effort estimates ($400K, $600K, $630K and $1M). We know that you only go up one level at a time – so, whatever level they’re at now, they will only step up one level for Ver 2. From Lecture 2, slide 10 middle (average) curve we read off about a 6% to 7% effort reduction for each level of improvement in the 40K-100K range; let’s simplify and just use the conservative 6%, which would be an improvement of $24K, $36K, $38K or $70K depending on which estimate you’re using. This is the money that might be set aside for process upgrade.

Which estimate to use? This depends on the company culture... I find the combined estimate to be the most robust, so if I were in Darlene’s place I’d see what process upgrades I could buy for $38K.

All 5 pts for declaring “$38K is justifiable”

Lose 1 point for throwing away information: using “10%/level” heuristic instead of 7%, or for simply adopting one of the three estimates without using a combining function (other combining functions are OK, doesn’t have to be 1/6 – 2/3 – 1/6)

Comments on the Process Maturity Effect chart (Lecture 2 slide 10). You can dig a lot of numbers out of this chart … for example, if you’re really conservative, you might take the lower 95 percent curve, and use numbers from 3% to 4% instead of 6% to 7% (which is the average savings). Or if you’re shading to the optimistic, you might use the upper 95% curve, and plan on 9% to 10% savings! Exactly which curve of the three you picked doesn’t matter for this problem – but it might matter enormously in real life

2 EXTRA POINTS for recognizing that the money is paid up front but the savings are later.

2. Requirements (31 pts)
2.1 User types (9 pts)
For each of the following applications, briefly describe a favored user, a disfavored user, and an ignored user; explain (in one sentence) why each user is so categorized.

My answers are just examples of acceptable answers.
2.1.1 APP1 (3 pts): Adult video delivery to multimedia cell phone
2.1.1 a) FAVORED: adult (is sales target)
2.1.1 b) DISFAVORED: child (legal liability if grant access)
2.1.1 c) IGNORED: blind users (special needs expensive to support, but maybe some like to listen)

2.1.2 APP2 (3 pts): Operating System software company product information web site
2.1.2 a) FAVORED: product purchasers, potential investors (is sales target or investment source)
2.1.2 b) DISFAVORED: engineers seeking detailed product info (possible competitors)
2.1.2 c) IGNORED: children (not a sales target)

2.1.3 APP3 (3 pts): Software Engineer job data bank/resume matcher
2.1.3 a) FAVORED: Software engineers employers or skills-training advertisers (is sales target)
2.1.3 b) DISFAVORED: Automated search engines (pulls data for free into competitor DB) or job seekers not meeting software engineering background requirement (lowers database quality to target users)
2.1.3 c) IGNORED: College students in non-CS class sequence (not yet a hiring target, but maybe will be someday)

2.2 Requirement Types (12 pts)
(8 pt) The table below provides some possible Requirements for a graphical desktop software calculator utility. The calculator is to perform basic floating-point arithmetic functions (+, -, *, /), plot trigonometric functions. For each Requirement, indicate whether the Requirement is a Function, or Attribute; if a Function, indicate whether the Function should be Hidden or Evident; if an Attribute, identify the Function to which it applies and give a feasible range of values.

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>F/A</th>
<th>E/H or F/values</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The calculator shall have a GUI which mimics a traditional desktop calculator with a single display field, data entry and function-invocation buttons.</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>R2</td>
<td>The Calculator shall perform operations of addition, subtraction, division and multiplication.</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>R3</td>
<td>The Calculator shall operate on, and return, integer and floating point values.</td>
<td>A</td>
<td>R2/standard Java double</td>
</tr>
<tr>
<td>R4</td>
<td>GUI shall have adjustable button label font sizes.</td>
<td>A</td>
<td>R1/8 to 12 pt</td>
</tr>
</tbody>
</table>

(4 pt) A new requirement is levied: to solve the Traveling Salesman Problem for a graph written out in an input file. Write out a functional requirement for this, and any attributes you think it needs.

answer: (goal is to show ability to frame a functional/attribute req pair, not to build complete set)

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>F/A</th>
<th>E/H or F/values</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5</td>
<td>... find an optimal solution to the Travelling Salesman Problem</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>R6</td>
<td>... accept an input file describing a TSP</td>
<td>A</td>
<td>R5/ legal file name,</td>
</tr>
</tbody>
</table>

WORTH 8% of the class grade, so 80 points ESTIMATED TIME: 3 hours
2.3 Requirement Writing (10 pt)
Your friends are going to the grocery store, and you would like them to buy you some ice cream. Normally just tell your friend a specific brand and model; but, since aren’t sure what will be in stock, you describe requirements that would lead your friend to pick a flavor that you'd like.

Your friend can visually recognize common ice cream ingredients, such as nuts, chocolate chips, etc. Your answers should depend on the properties of the ice cream itself, and not the package. Assume that your friend will open and visually inspect the ice cream package if necessary (they’re a good friend, and it’s a very accommodating store!)

First, think of an ice cream flavor that you want. The rest of the questions will ask you to write requirements describing this ice cream flavor.

2.3a) What ice cream flavor did you pick?
ROCKY ROAD

2.3b) (2 pt) Give two requirements about the ice cream flavor that are overly precise. That is, the requirements are so detailed that it would either be impossible, or extremely tedious to test to see if the ice cream meets that requirement. Describe why each requirement is too precise.

R1: The ice cream shall have a chocolate component, where the chocolate was grown in the Peruvian mountains at an altitude between 2000 and 2500 meters. (Too precise because product origins seldom listed, so likely not testable)
R2: The ice cream shall have a crunchy component, where the crsush force of the crunchy elements ranges from 7 PSI to 13 PSI. (Too precise, because your friend can’t test this without destroying the sample)

2.3c) (2 pt) Give two requirements about the ice cream flavor that are ambiguous. That is, each requirement describes the ice cream flavor you have in mind, but also describes other kinds of ice cream. Describe why each requirement is ambiguous.

COMMENT: We need to discuss “ambiguous requirements” in class … I didn’t really read this question closely enough!

Generally, a requirement should not be so precise that all by itself it only applies to one outcome – that would simply be an alias for a preselected goal! Instead, the goal is for the union of a set of requirement to constrain a solution to something that the user desires. So if R1 is met by outcomes A, B, C and R2 is met by outcomes B, D, and e, then we have an acceptable set of requirments (resolving to outcome B).

I do not hold a requirement ambiguous because it can have multiple outcomes (for example, is satisfied by multiple flavors of ice cream); instead, I hold an requirement to be ambiguous if the requirement itself is open to multiple interpretations … sy, if it used words or phrases that can have multiple eans according to what dictionary meaning you select for the words, or how you apply emphasis.
Still, we have to grade the question as asked … so, we are simply looking here for requirements that can be shown to apply to multiple flavors of ice cream.

R1: The ice cream shall be sweet. (Many flavors are sweet. Or maybe it means “pleasing to the ear.” Or not salted. See the dictionary meanings list for “sweet!” This one is ambiguous :)

R2: The ice cream flavor shall complement tonight’s wine (a Merlot). (“Complement” is a personal judgement without testable values. And multiple flavors would fit.)

2.3d) (6 pt) Give two requirements about the ice cream that are "just right," not ambiguous, not overly precise. Describe why each requirement has hit the "sweet spot" between ambiguity and over-precision, and is easily validatable.

COMMENT: Following the pattern, these would be requirements that could only be met by the single selected ice cream.

R1. The ice cream shall have chocolate, marshmallows and nuts as components. (Easily seen or read from label, met only by RR)

R2. The ice cream shall be named the same as an Annabelle candy bar. (No common ice creams named Look!, U-NO, Abba-Zaba, or Big Hunk.) The Rocky Road candy bar is generally sold in the same stores as the Rocky Road ice cream, so my friend can check the candy counter to test this.

But, FWIW, I would personally prefer requirements that usefully express a user preference or need independent of the precise implementation … so that I would probably get RR, but maybe I get something just as good (but lower in fat 😊) such as:

R1. The ice cream shall have a chocolate component. (Constrains to a small set of flavors, but likely to be available and validatable; validate by inspection of the label)

R2: The ice cream shall have a milk-fat component between 2% and 8% (Orthogonal to “chocolate” requirement, constrains to a small set of products (no sherberts, no low-fat), but likely to be available and validatable by inspection of the label)
3. Development Methodology (12 pts)
Three software applications are described below. For each:

(1 pt) choose the best development method (OO/UseCase, Function/Transformational, or Formal)
(1 pt) choose the best fundamental Process model (plan-driven or agile)
(1 pt) Considering both effect and exposure briefly describe a significant program risk
(1 pt) Describe briefly how you might detect the occurrence of the risk condition/event.

3.1: New interface for Automatic Teller Machine connecting to an existing banking system using a well-defined protocol.

<table>
<thead>
<tr>
<th>3.1a Dev Method</th>
<th>OO/UC to capture user needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1b Process Model</td>
<td>Agile</td>
</tr>
<tr>
<td>3.1c Risk</td>
<td>&lt;Check for sanity&gt; Ideal answer cites an overall-risk table as in Lecture 3 slide 26, but any fairly-likely risk of something fairly-bad is OK. Risk of UI error causing user data entry error is a good overall risk.</td>
</tr>
<tr>
<td>3.1d Detection</td>
<td>&lt;Check for sanity&gt; Detection could be through UI testing, or through field feedback monitoring of error rates (compare envelope contents to user entered data, for example)</td>
</tr>
</tbody>
</table>

3.2: Replacement banking system (multi-branch support, does account maintenance, database maintenance, batch reports.) Uses same (simple) job submission language and interfaces as existing system, but is being implemented on new hardware/software platform.

<table>
<thead>
<tr>
<th>3.2a Dev Method</th>
<th>Functional/Transformational (spec-heavy, frozen interfaces, easy mapping functions to functions for regression testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2b Process Model</td>
<td>Plan-driven</td>
</tr>
<tr>
<td>3.2c Risk</td>
<td>&lt;Check for sanity&gt;</td>
</tr>
<tr>
<td>3.2d Detection</td>
<td>&lt;Check for sanity&gt;</td>
</tr>
</tbody>
</table>

3.3: X-Ray dosage calculator for human diagnostic system

<table>
<thead>
<tr>
<th>3.3a Dev Method</th>
<th>FM (for safety concerns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3b Process Model</td>
<td>Plan-driven</td>
</tr>
<tr>
<td>3.3c Risk</td>
<td>&lt;Check for sanity&gt;</td>
</tr>
<tr>
<td>3.3d Detection</td>
<td>&lt;Check for sanity&gt;</td>
</tr>
</tbody>
</table>
Misc (13 pt)
(5 pts) Increasing the number of people often makes a late software project more late. Give one reason why.

New personnel startup times impact ongoing work, then increased communication demands slow down everyone.

(5 pts) What fundamental premise shapes the difference between plan-driven and agile methodologies? Express graphically.

<sketch of cost (Y axis) vs program phase (X axis) showing plan-driven being exponential increase, while agile is something like a log(x) increase; see Lecture 2 Slide 24>

Three almost-free points!
(1 pt) How do you get a development team to optimize for a particular characteristic?

Measure that characteristic and publish the results often.

(1 pt) What question took you the longest to answer, and how long was it?

(1 pt) How long did the entire test take you?