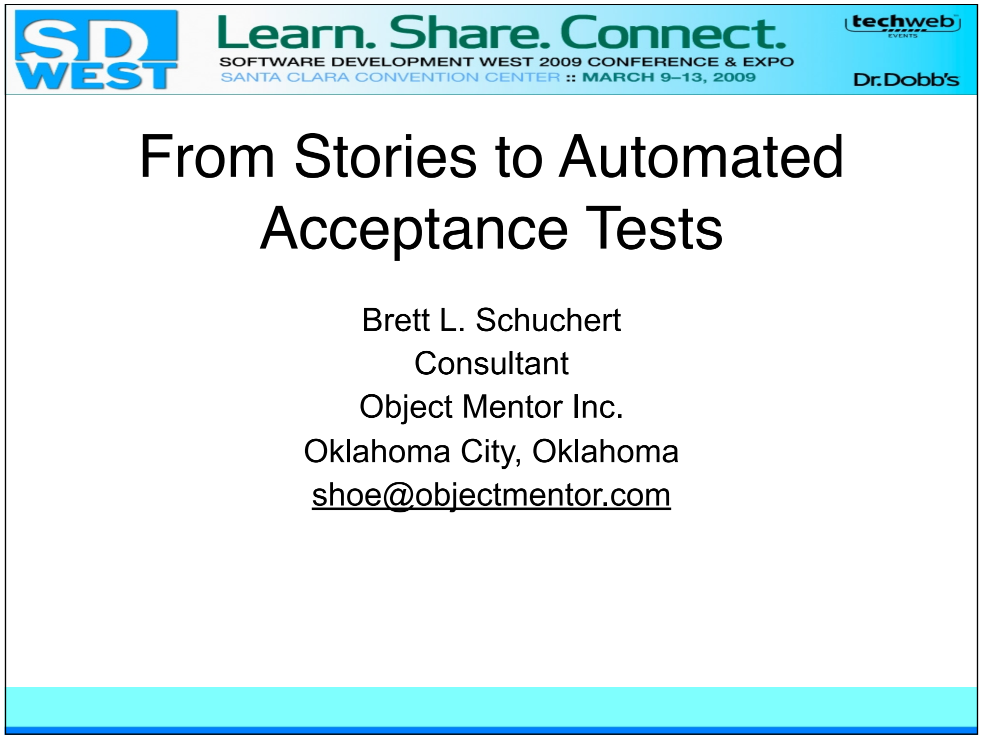


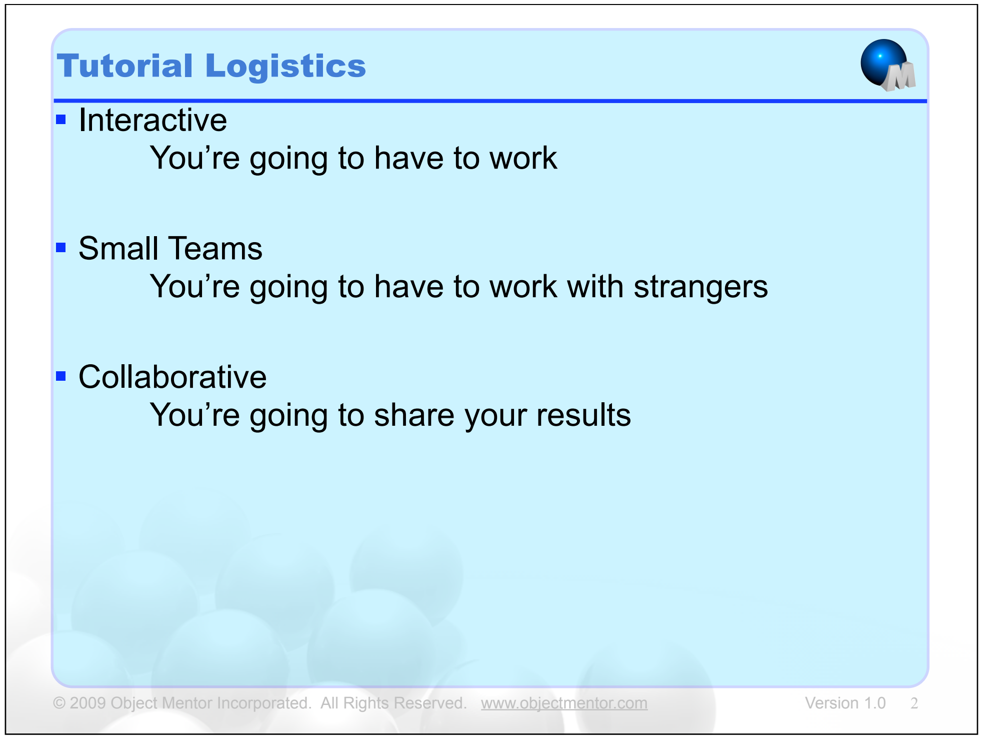
SD WEST Learn. Share. Connect.
SOFTWARE DEVELOPMENT WEST 2009 CONFERENCE & EXPO
SANTA CLARA CONVENTION CENTER :: MARCH 9–13, 2009
techweb EVENTS
Dr.Dobb's

From Stories to Automated Acceptance Tests

Brett L. Schuchert
Consultant
Object Mentor Inc.
Oklahoma City, Oklahoma
shoe@objectmentor.com

Timing: 2 minutes - not going to introduce myself much.

OK, I have no idea how many people to expect. Because of that, I'm not sure how this will run. If it's small (< 30) people, then I'll have time for all groups to present answers to their exercises. If it's much larger, then I won't have that luxury and I'll instead start with volunteers. After that, I'll call on groups.

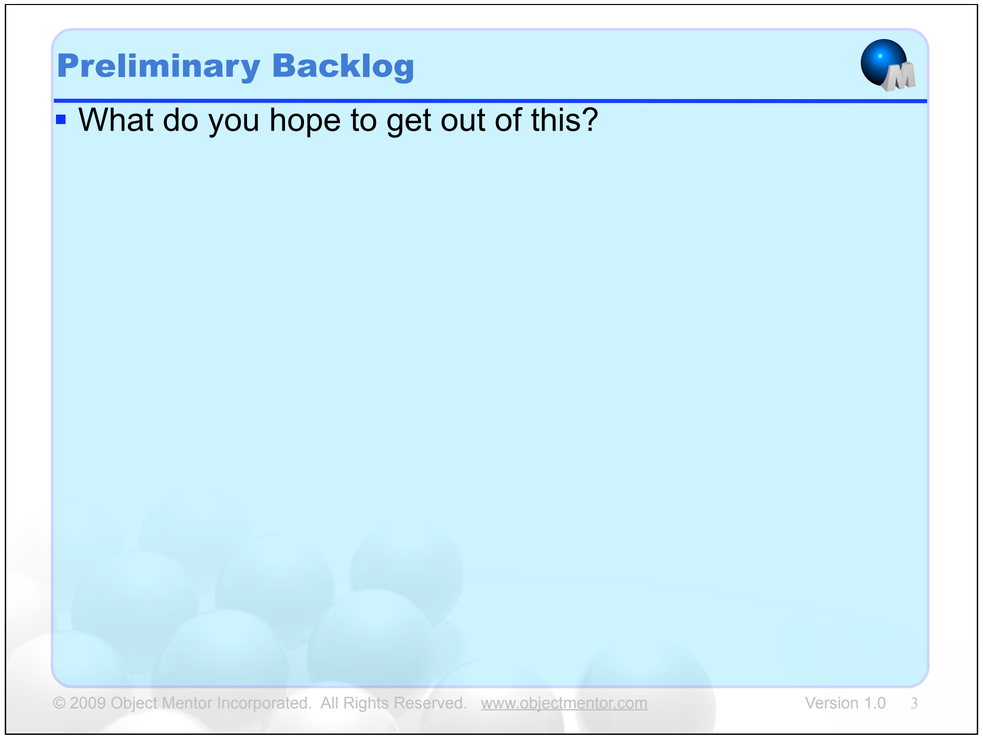


Tutorial Logistics

- Interactive
  - You're going to have to work
- Small Teams
  - You're going to have to work with strangers
- Collaborative
  - You're going to share your results

Timing: 4 minutes, 1 to describe, 3 to wait for people to form into small groups.

This tutorial is designed to involved a lot of group interactions. Rather than the presenter being the primary source of info, it is designed so that the presenter is the coordinator. It is meant to be much more social.

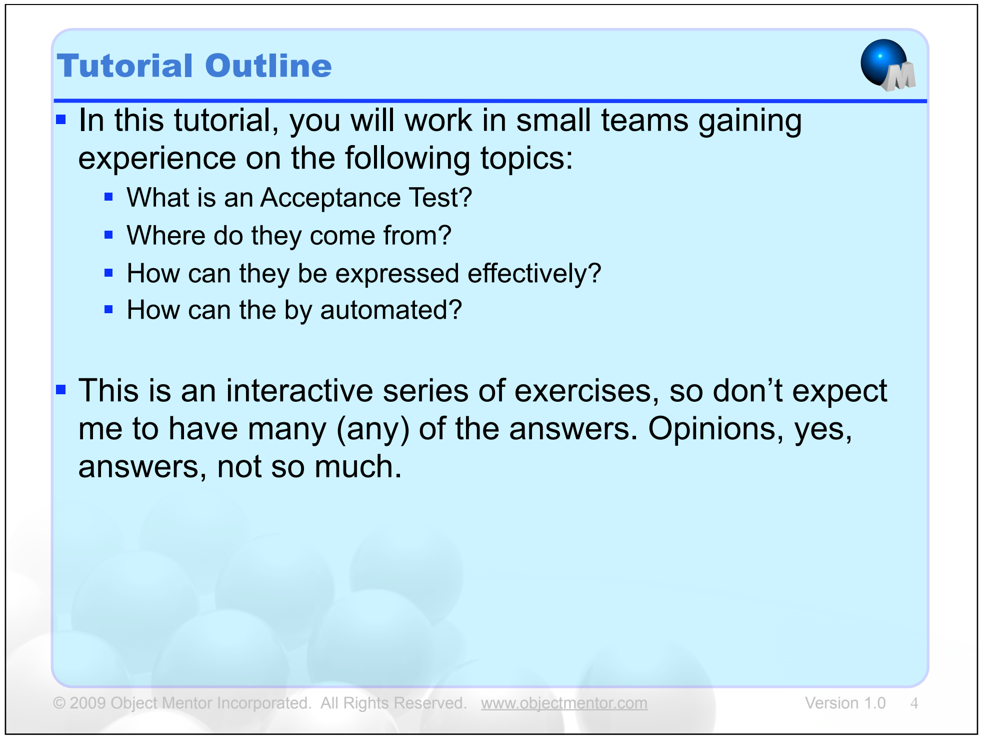


Preliminary Backlog

- What do you hope to get out of this?

Timing: 5 minutes

Create a backlog in a visible location (editor or flip chart). This is for guidance during and review after. And to also direct people to other tutorials if this will not have what they want.

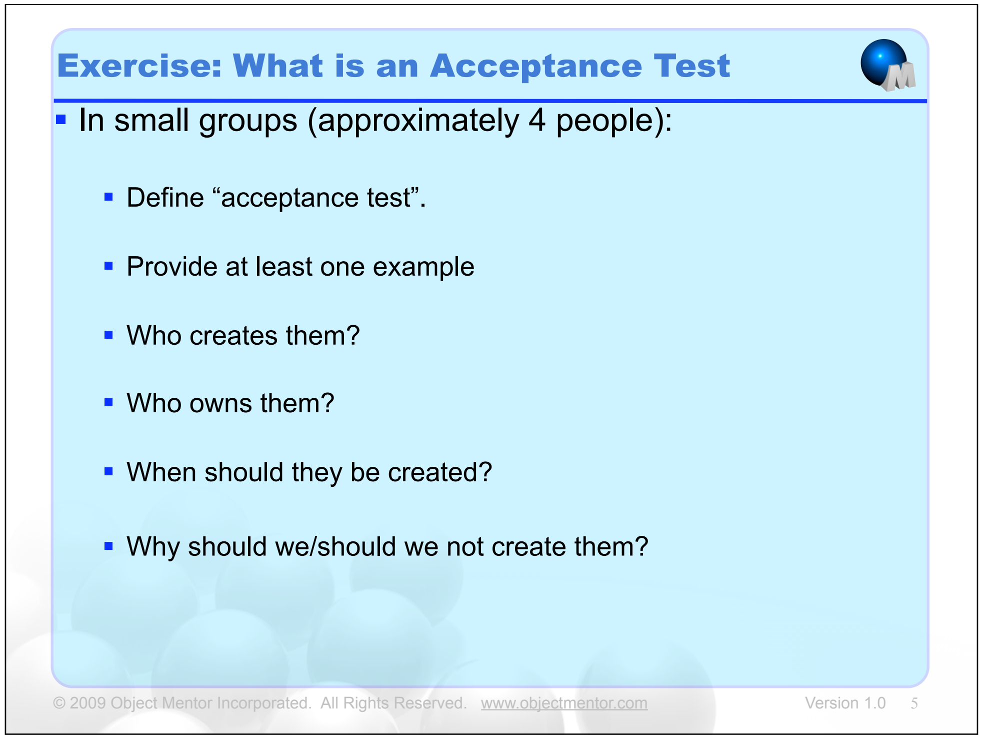


Tutorial Outline

- In this tutorial, you will work in small teams gaining experience on the following topics:
  - What is an Acceptance Test?
  - Where do they come from?
  - How can they be expressed effectively?
  - How can the by automated?

- This is an interactive series of exercises, so don't expect me to have many (any) of the answers. Opinions, yes, answers, not so much.

Timing 3 minutes to give overview.



Exercise: What is an Acceptance Test

- In small groups (approximately 4 people):
  - Define “acceptance test”.
  - Provide at least one example
  - Who creates them?
  - Who owns them?
  - When should they be created?
  - Why should we/should we not create them?

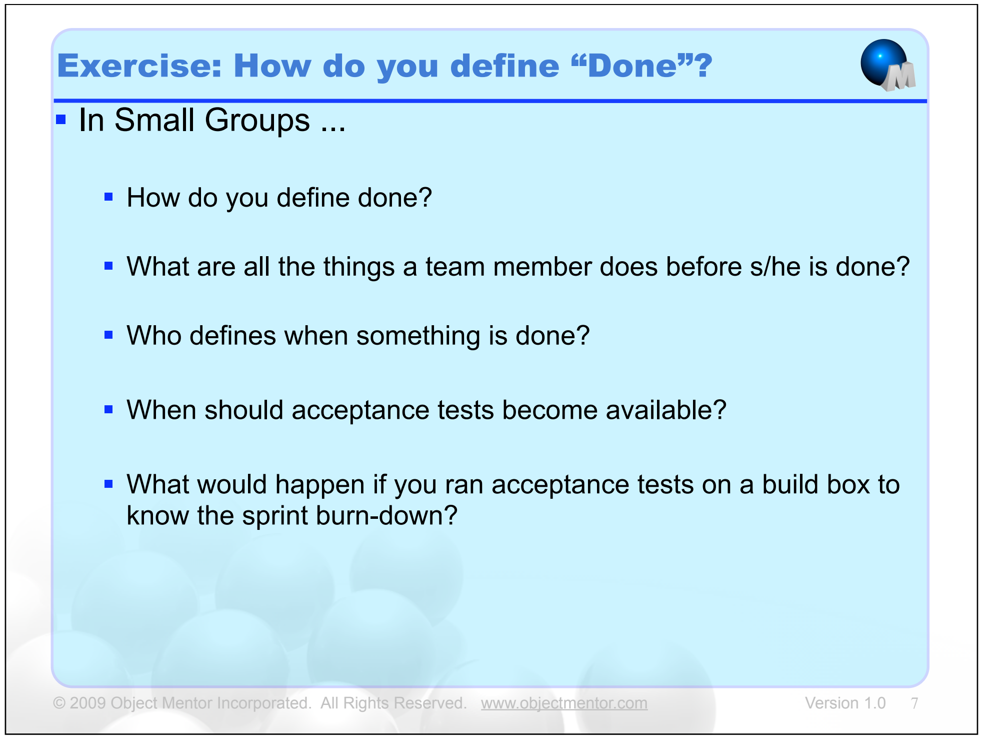
Timing: 9 minutes (23 minutes so far)

After this exercise, collect a few definitions. Could do some affinity clustering, followed by naming. In a 3-hour setting, might or might not. Won't know until I actually run it what I'm feeling like at that time.

So, What is an “Acceptance Test”?

Timing: 7 minutes, 30 so far

Collect a working definition. Pick out some key aspects. Put on a flip chart. Capture and report back? It'll be fun to go back and look at this once I've presented this 3 or 4 times (and thereafter).



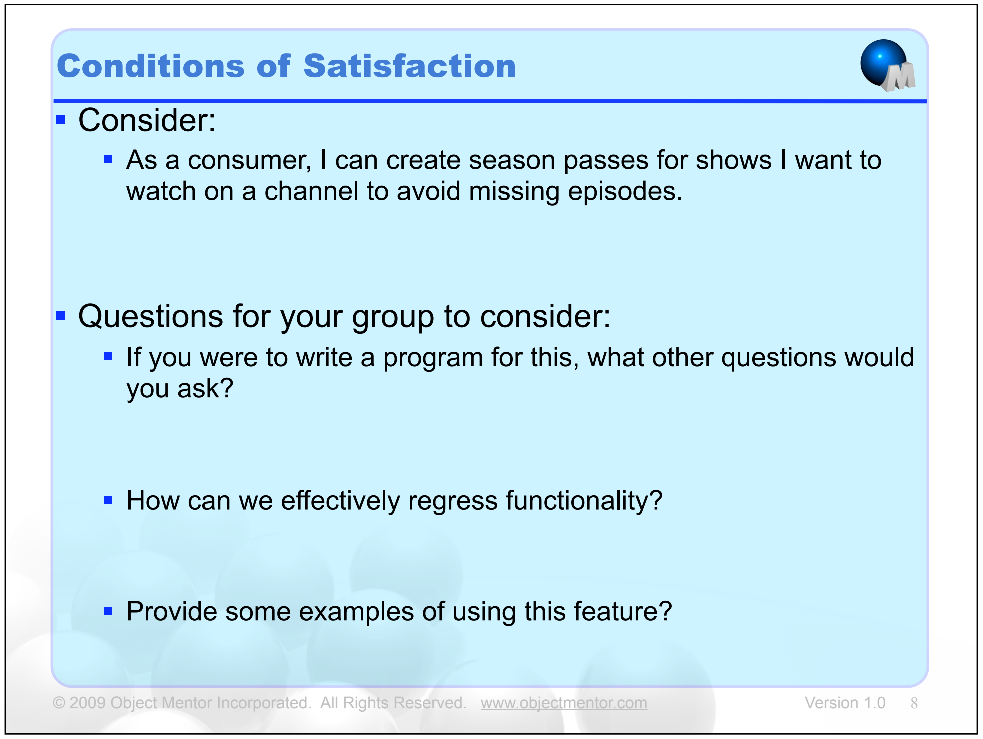
Exercise: How do you define "Done"?

- In Small Groups ...
  - How do you define done?
  - What are all the things a team member does before s/he is done?
  - Who defines when something is done?
  - When should acceptance tests become available?
  - What would happen if you ran acceptance tests on a build box to know the sprint burn-down?

Timing: 6 minutes (36 so far)

Part of what it takes to make this work is to get a team agreeing on the definition of done. At a minimum: checked in, merged, unit and acceptance tests executing and passing.

This also gets to the question of timing. Who should write these? When should they write them? Ideally, these need to be well defined during the sprint planning meeting. In practice, some at the beginning (pipelining) followed by the rest before the half-way mark can work.

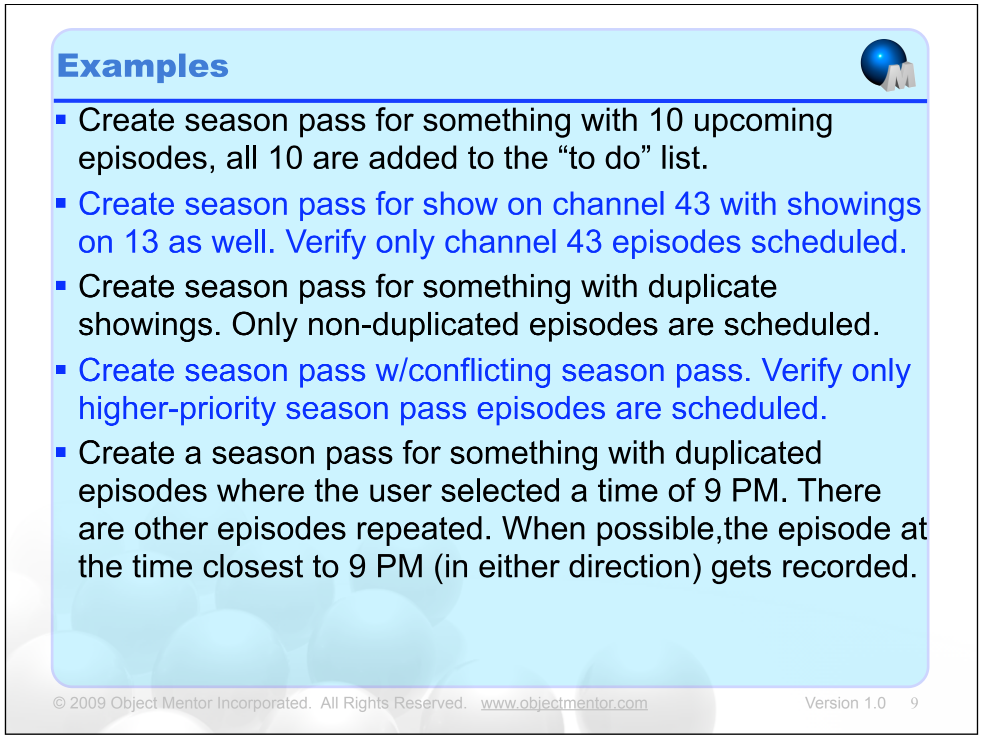


Conditions of Satisfaction

- Consider:
  - As a consumer, I can create season passes for shows I want to watch on a channel to avoid missing episodes.
- Questions for your group to consider:
  - If you were to write a program for this, what other questions would you ask?
  - How can we effectively regress functionality?
  - Provide some examples of using this feature?

Timing: 6 minutes, 3 minutes group work, 3 minutes summary. (42 so far)

I'm still using this term from Cohen, Conditions of satisfaction. Not sure if I should just remove it. I've moved to the "example" camp. In any case, this is an attempt for small groups to try and create some first-cut examples. What I'm expecting is that people will go abstract rather than concrete. What I want them to get out of this is that concrete is the way to go at first.

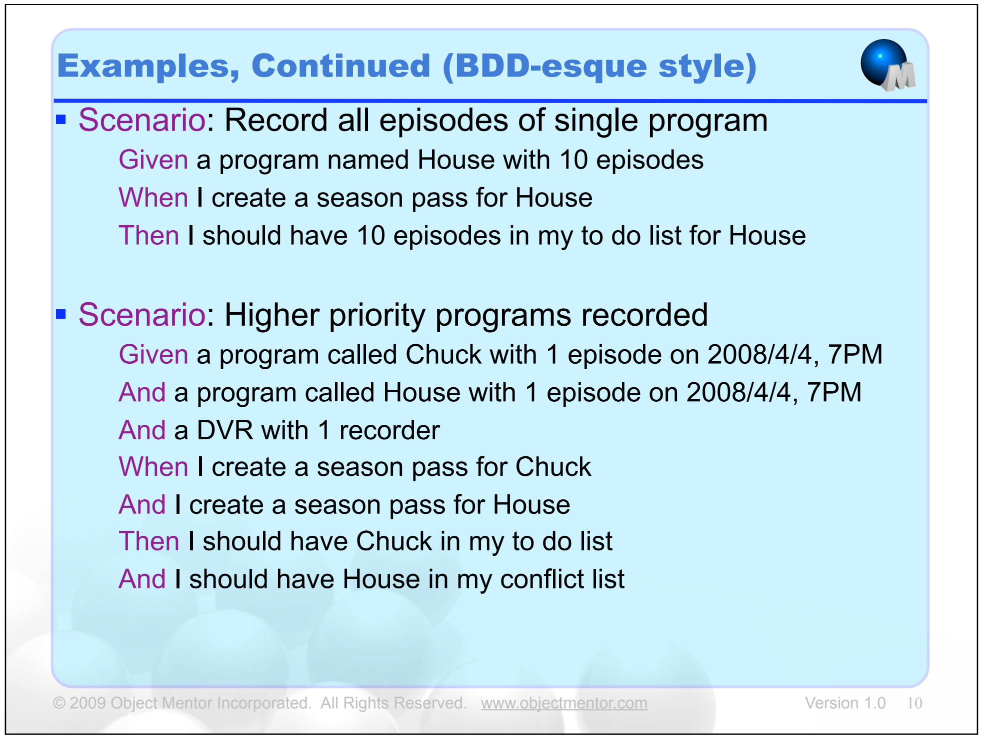


Examples

- Create season pass for something with 10 upcoming episodes, all 10 are added to the “to do” list.
- Create season pass for show on channel 43 with showings on 13 as well. Verify only channel 43 episodes scheduled.
- Create season pass for something with duplicate showings. Only non-duplicated episodes are scheduled.
- Create season pass w/conflicting season pass. Verify only higher-priority season pass episodes are scheduled.
- Create a season pass for something with duplicated episodes where the user selected a time of 9 PM. There are other episodes repeated. When possible,the episode at the time closest to 9 PM (in either direction) gets recorded.

Timing: 1 minutes to cover, 2 minutes for questions (45 so far)

Here are some examples. A few better and more concrete than others. In some cases, having too much detail is not really important. For example, why the number 10 in the first one? Or why channel 43. This is where it helps to allow some things to be general, e.g. Channel, as opposed to the specific 43. I don't expect this to come up in discussion but who knows.

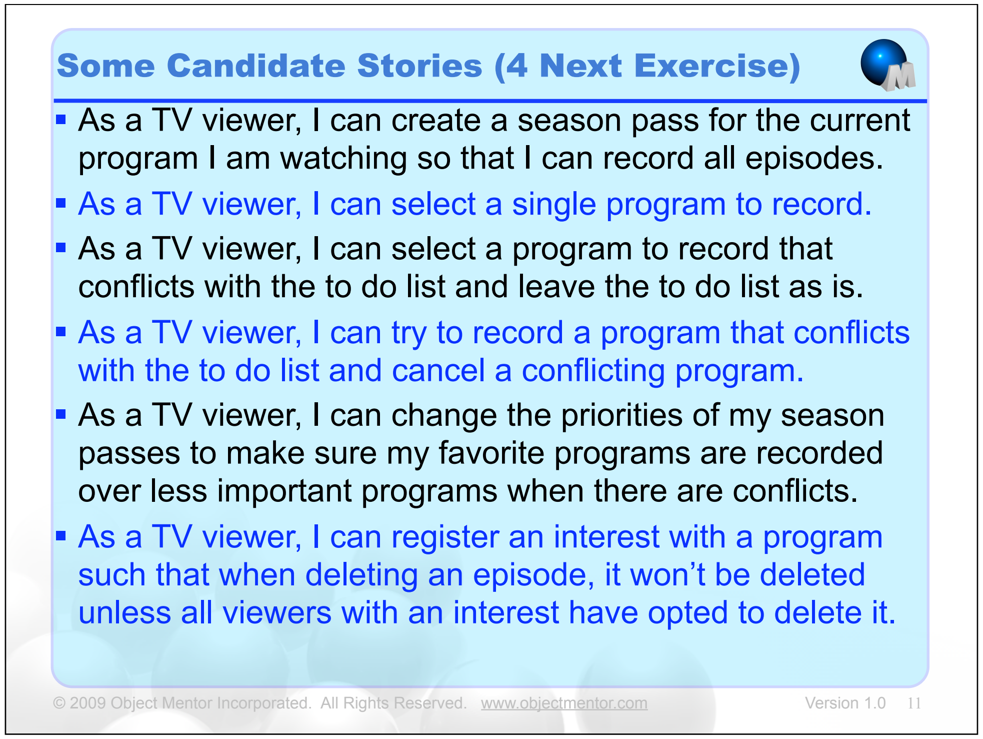


Examples, Continued (BDD-esque style)

- Scenario: Record all episodes of single program
  - Given a program named House with 10 episodes
  - When I create a season pass for House
  - Then I should have 10 episodes in my to do list for House
- Scenario: Higher priority programs recorded
  - Given a program called Chuck with 1 episode on 2008/4/4, 7PM
  - And a program called House with 1 episode on 2008/4/4, 7PM
  - And a DVR with 1 recorder
  - When I create a season pass for Chuck
  - And I create a season pass for House
  - Then I should have Chuck in my to do list
  - And I should have House in my conflict list

Timing: 5 Minutes to discuss and questions (50 so far)

I've recently been working (playing) with Cucumber because I'm reviewing the RSpec book. I am of two minds about story (feature) test runners. On the one hand they can be easy to write and read. On the other, there's a lot of duplication in the handling of their execution (a regular expression to handle the step). Even so, there's often duplication in tests written in FitNesse. Also, I want this to give ideas, not "the" way of doing things.

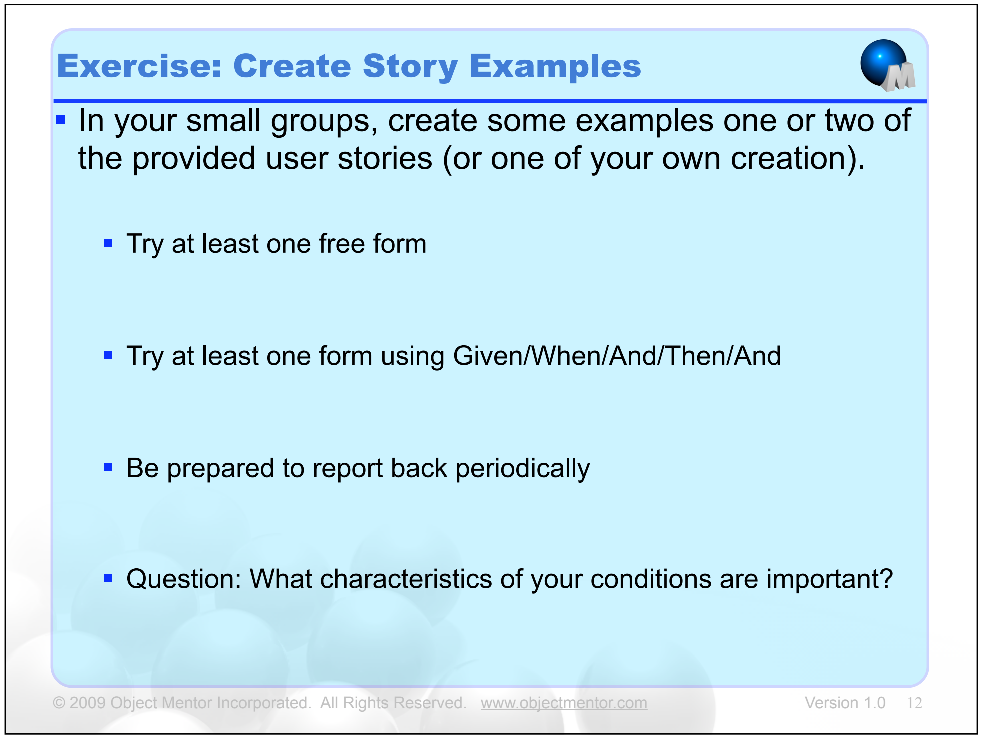


Some Candidate Stories (4 Next Exercise)

- As a TV viewer, I can create a season pass for the current program I am watching so that I can record all episodes.
- As a TV viewer, I can select a single program to record.
- As a TV viewer, I can select a program to record that conflicts with the to do list and leave the to do list as is.
- As a TV viewer, I can try to record a program that conflicts with the to do list and cancel a conflicting program.
- As a TV viewer, I can change the priorities of my season passes to make sure my favorite programs are recorded over less important programs when there are conflicts.
- As a TV viewer, I can register an interest with a program such that when deleting an episode, it won't be deleted unless all viewers with an interest have opted to delete it.

Timing 3 minutes: questions, 53 so far

This is another dense slide. This is meant as a reference for the next exercise.



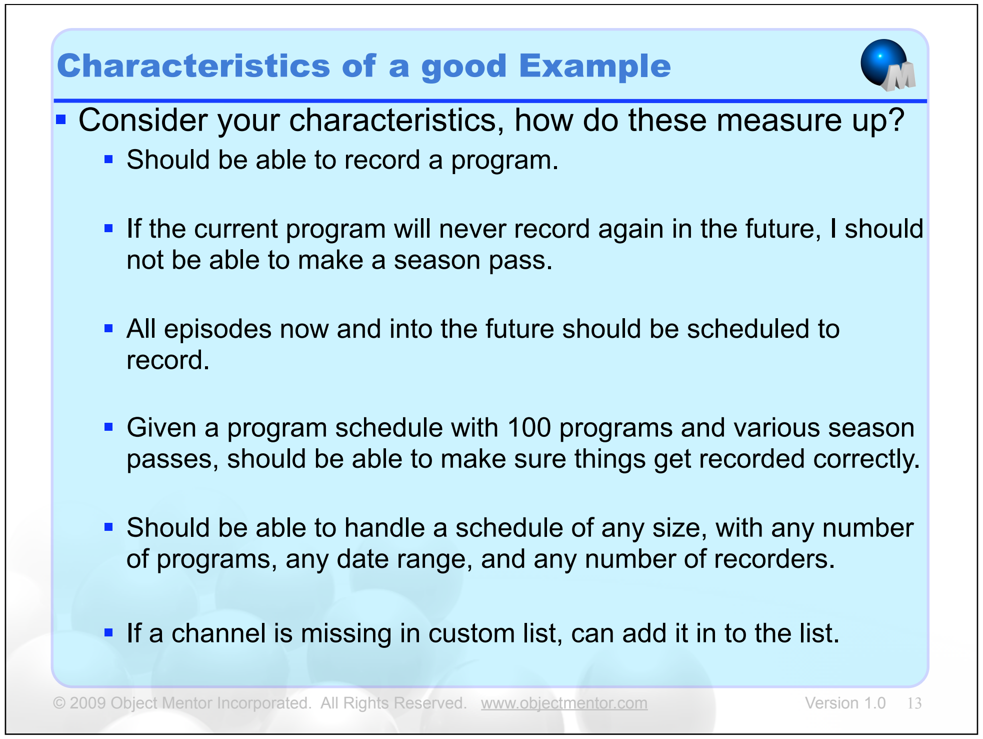
Exercise: Create Story Examples

- In your small groups, create some examples one or two of the provided user stories (or one of your own creation).
  - Try at least one free form
  - Try at least one form using Given/When/And/Then/And
  - Be prepared to report back periodically
  - Question: What characteristics of your conditions are important?

21 minutes: (5 minutes, 2 minutes) 3x (74 minutes)

Want them to try each form at least once. Will make things much more concrete. Will have some present their results.

Characteristics: (Rather than trying to reinvent the wheel) S.M.A.R.T
Specific, Measurable, Attainable, Relevant, Time-bound



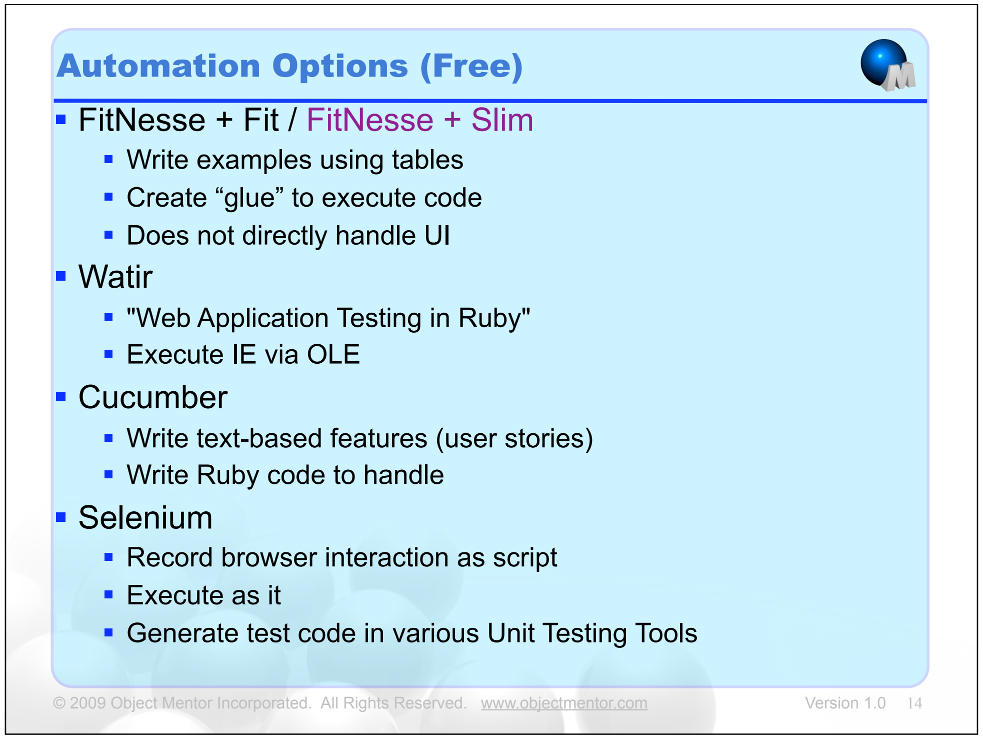
Characteristics of a good Example

- Consider your characteristics, how do these measure up?
  - Should be able to record a program.
  - If the current program will never record again in the future, I should not be able to make a season pass.
  - All episodes now and into the future should be scheduled to record.
  - Given a program schedule with 100 programs and various season passes, should be able to make sure things get recorded correctly.
  - Should be able to handle a schedule of any size, with any number of programs, any date range, and any number of recorders.
  - If a channel is missing in custom list, can add it in to the list.

Timing: 5 minutes, 79 total

These are each meant to violate one or more of the SMART characteristics. People learn from critiquing other work. This is safe to critique and it will reinforce the learnings. I picked this up from David Nunn. He and I have a talk proposed at Agile 2009 in Chicago on this very subject.

1: Not really specific
2. Not time-bound
3: Not Attainable
4. Not specific, not measurable
5: Probably not attainable
6: Not relevant

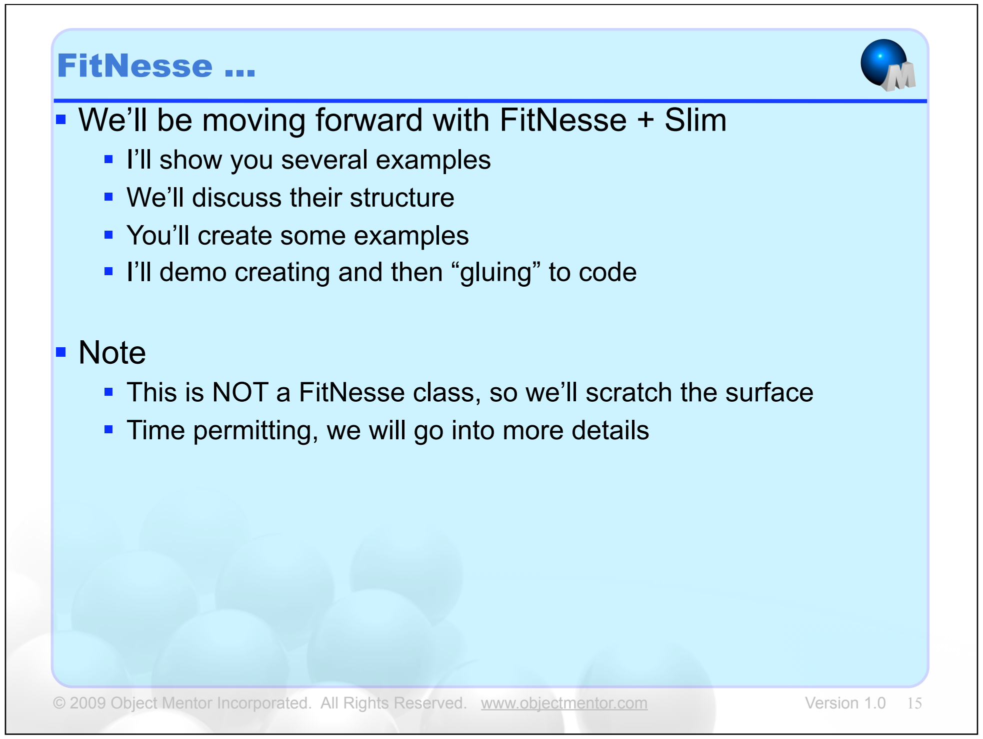


Automation Options (Free)

- FitNesse + Fit / FitNesse + Slim
  - Write examples using tables
  - Create “glue” to execute code
  - Does not directly handle UI
- Watir
  - "Web Application Testing in Ruby"
  - Execute IE via OLE
- Cucumber
  - Write text-based features (user stories)
  - Write Ruby code to handle
- Selenium
  - Record browser interaction as script
  - Execute as it
  - Generate test code in various Unit Testing Tools

Timing: 6 minutes, 85 total

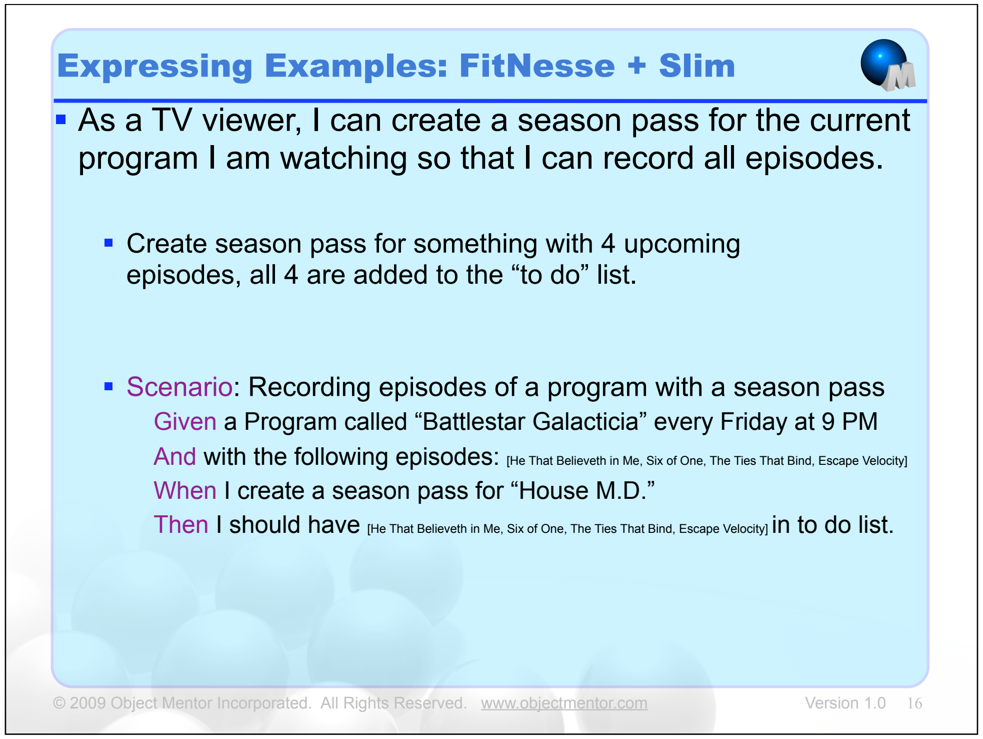
Quick listing more to give an idea of what’s out there. I expect (hope) that this will lead to questions about when to use which kind of tool. One point I want to really reinforce is the idea that it’s not either-or. It’s yes, and!



FitNesse …

- We'll be moving forward with FitNesse + Slim
  - I'll show you several examples
  - We'll discuss their structure
  - You'll create some examples
  - I'll demo creating and then "gluing" to code
- Note
  - This is NOT a FitNesse class, so we'll scratch the surface
  - Time permitting, we will go into more details

Timing: 3 minutes, 88 total

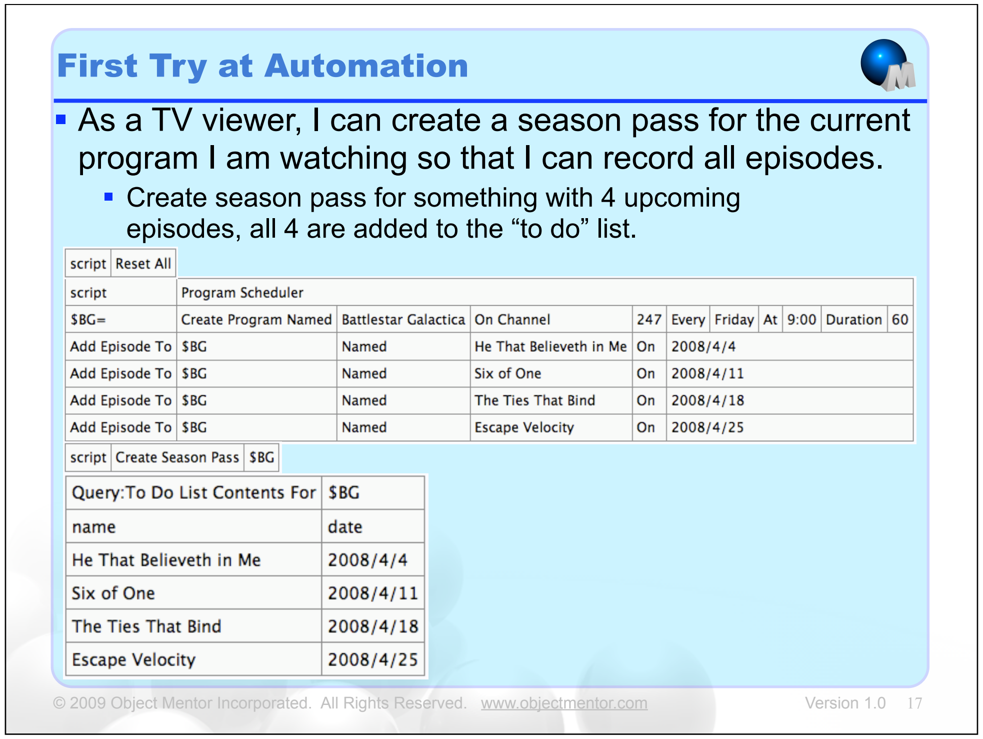


Expressing Examples: FitNesse + Slim

- As a TV viewer, I can create a season pass for the current program I am watching so that I can record all episodes.
  - Create season pass for something with 4 upcoming episodes, all 4 are added to the "to do" list.
  - Scenario: Recording episodes of a program with a season pass
    Given a Program called "Battlestar Galacticia" every Friday at 9 PM
    And with the following episodes: [He That Believeth in Me, Six of One, The Ties That Bind, Escape Velocity]
    When I create a season pass for "House M.D."
    Then I should have [He That Believeth in Me, Six of One, The Ties That Bind, Escape Velocity] in to do list.

Timing: 3 minutes, 91 total

Here are a couple representations of the same example. This is one way of many to express it. I wanted it to look reasonable and this is the best I could come up with before I finally put the slides to rest.



First Try at Automation

- As a TV viewer, I can create a season pass for the current program I am watching so that I can record all episodes.
  - Create season pass for something with 4 upcoming episodes, all 4 are added to the “to do” list.

script	Reset All

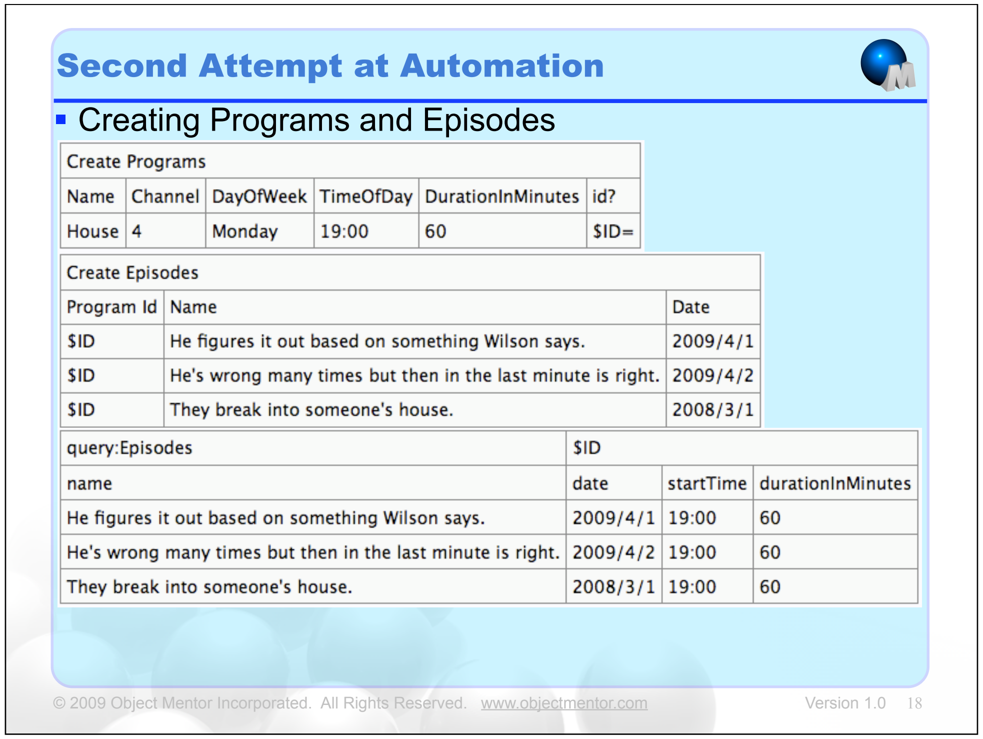
script	Program Scheduler									
$BG=	Create Program Named	Battlestar Galactica	On Channel	247	Every	Friday	At	9:00	Duration	60
Add Episode To	$BG	Named	He That Believeth in Me	On	2008/4/4					
Add Episode To	$BG	Named	Six of One	On	2008/4/11					
Add Episode To	$BG	Named	The Ties That Bind	On	2008/4/18					
Add Episode To	$BG	Named	Escape Velocity	On	2008/4/25					

script	Create Season Pass	$BG

Query:To Do List Contents For	$BG
name	date
He That Believeth in Me	2008/4/4
Six of One	2008/4/11
The Ties That Bind	2008/4/18
Escape Velocity	2008/4/25

Timing: 12 minutes, 103 total

This really is the first way I though of the problem. I had a zero to many relationship, Program has zero or more episodes. When I see this, I used to default to using do fixtures (or two column fixtures). So when I started working on this in Slim I immediately used a script table. In retrospect, this was probably overkill.



Second Attempt at Automation

- Creating Programs and Episodes

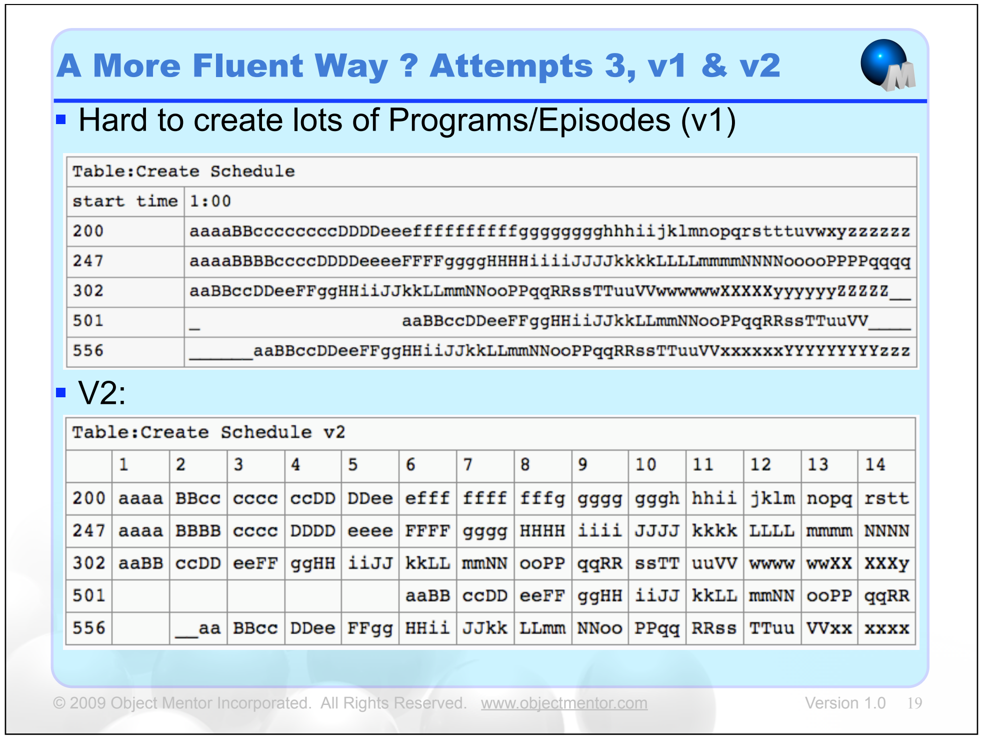
Create Programs					
Name	Channel	DayOfWeek	TimeOfDay	DurationInMinutes	id?
House	4	Monday	19:00	60	$ID=

Create Episodes		
Program Id	Name	Date
$ID	He figures it out based on something Wilson says.	2009/4/1
$ID	He's wrong many times but then in the last minute is right.	2009/4/2
$ID	They break into someone's house.	2008/3/1

query:Episodes	$ID		
name	date	startTime	durationInMinutes
He figures it out based on something Wilson says.	2009/4/1	19:00	60
He's wrong many times but then in the last minute is right.	2009/4/2	19:00	60
They break into someone's house.	2008/3/1	19:00	60

Timing: 2, 3, 2, 2 (9 total), 112 total

My next attempt at creating programs and episodes involved using Decision Tables (what fit would call Column Fixtures)



A More Fluent Way ? Attempts 3, v1 & v2

- Hard to create lots of Programs/Episodes (v1)

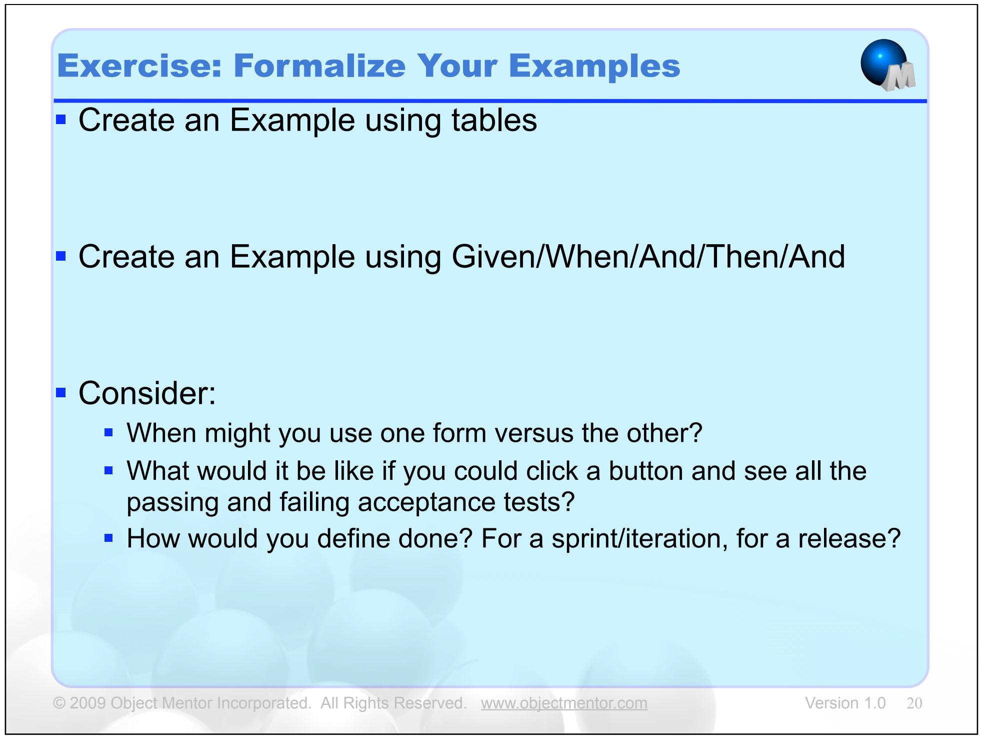
Table:Create Schedule	
start time	1:00
200	aaaaBBccccccccDDDDeeefffffffffggggggggghhhiijklmnopqrstttuvwxyzzzzzz
247	aaaaBBBBccccDDDDeeeeFFFFggggHHHHiiiiJJJJkkkkLLLLmmmmNNNNooooPPPPqqqq
302	aaBBccDDeeFFggHHiiJJkkLLmmNNooPPqqRRssTTuuVVwwwwwwXXXXyyyyyyZZZZ__
501	_ aaBBccDDeeFFggHHiiJJkkLLmmNNooPPqqRRssTTuuVV____
556	______aaBBccDDeeFFggHHiiJJkkLLmmNNooPPqqRRssTTuuVVxxxxxxYYYYYYYYYzzz

- V2:

Table:Create Schedule v2														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
200	aaaa	BBcc	cccc	ccDD	DDee	efff	ffff	fffg	gggg	gggh	hhii	jklm	nopq	rstt
247	aaaa	BBBB	cccc	DDDD	eeee	FFFF	gggg	HHHH	iiii	JJJJ	kkkk	LLLL	mmmm	NNNN
302	aaBB	ccDD	eeFF	ggHH	iiJJ	kkLL	mmNN	ooPP	qqRR	ssTT	uuVV	wwww	wwXX	XXXy
501						aaBB	ccDD	eeFF	ggHH	iiJJ	kkLL	mmNN	ooPP	qqRR
556		__aa	BBcc	DDee	FFgg	HHii	JJkk	LLmm	NNoo	PPqq	RRss	TTuu	VVxx	xxxx

Timing: 6 minutes, 118 total

Finally, I needed to create lots of programs and episodes and then make sure that for various numbers of season passes I would record thing based on priority. I was having problems creating complex examples until I realized I was thinking about the program guide. So I created V1 then recently, based on a complaint by David Nunn, created V2.

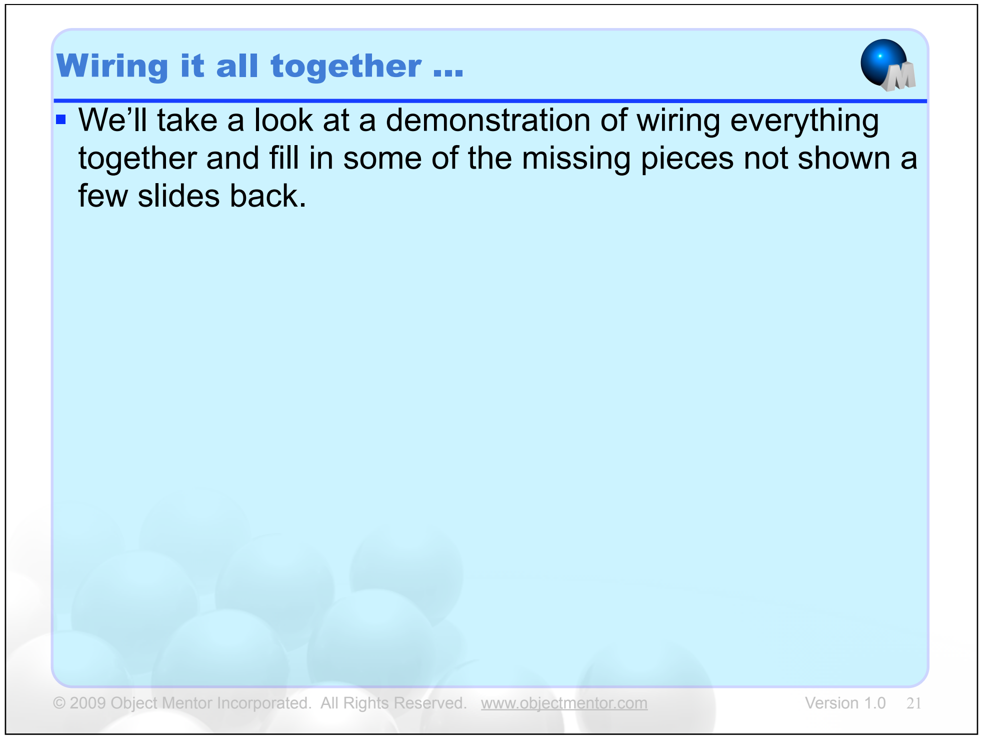


Exercise: Formalize Your Examples

- Create an Example using tables
- Create an Example using Given/When/And/Then/And
- Consider:
  - When might you use one form versus the other?
  - What would it be like if you could click a button and see all the passing and failing acceptance tests?
  - How would you define done? For a sprint/iteration, for a release?

Timing: 21 (5,2 3x) minutes, 139 total

Now it is their turn. If I get any decent examples, I'll code them up and get them at least running and red.

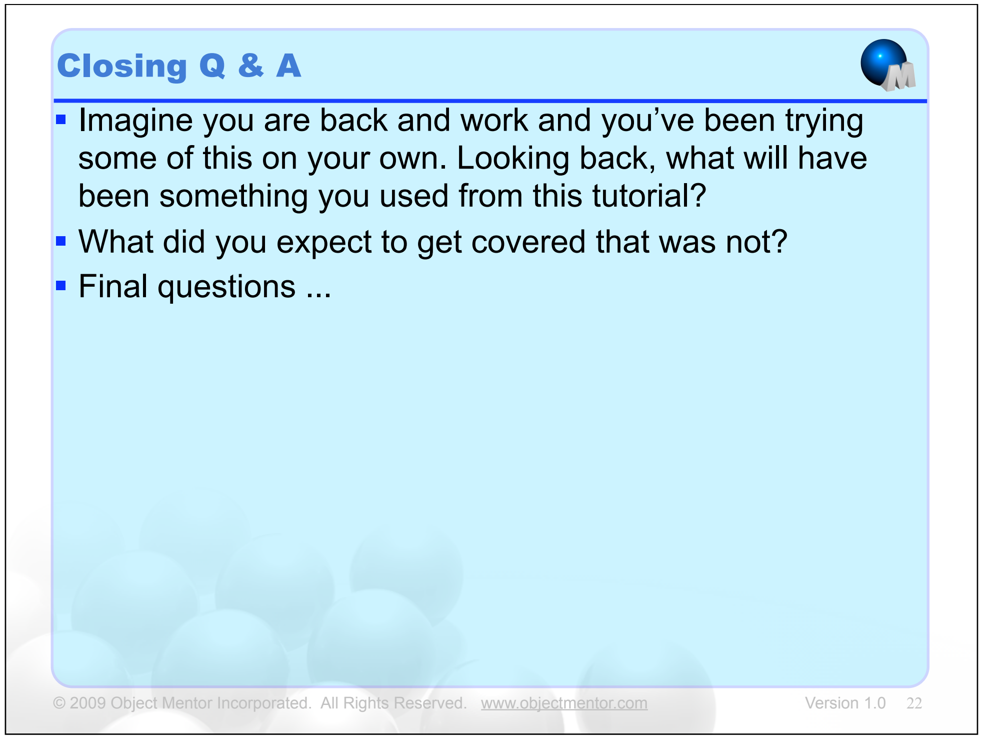


Wiring it all together ...

- We'll take a look at a demonstration of wiring everything together and fill in some of the missing pieces not shown a few slides back.

Timing: 3 minutes, basic nav & execution, 4 minutes, creating, 10 from scratch example, 17 total, 156 total

Switch to demo mode.

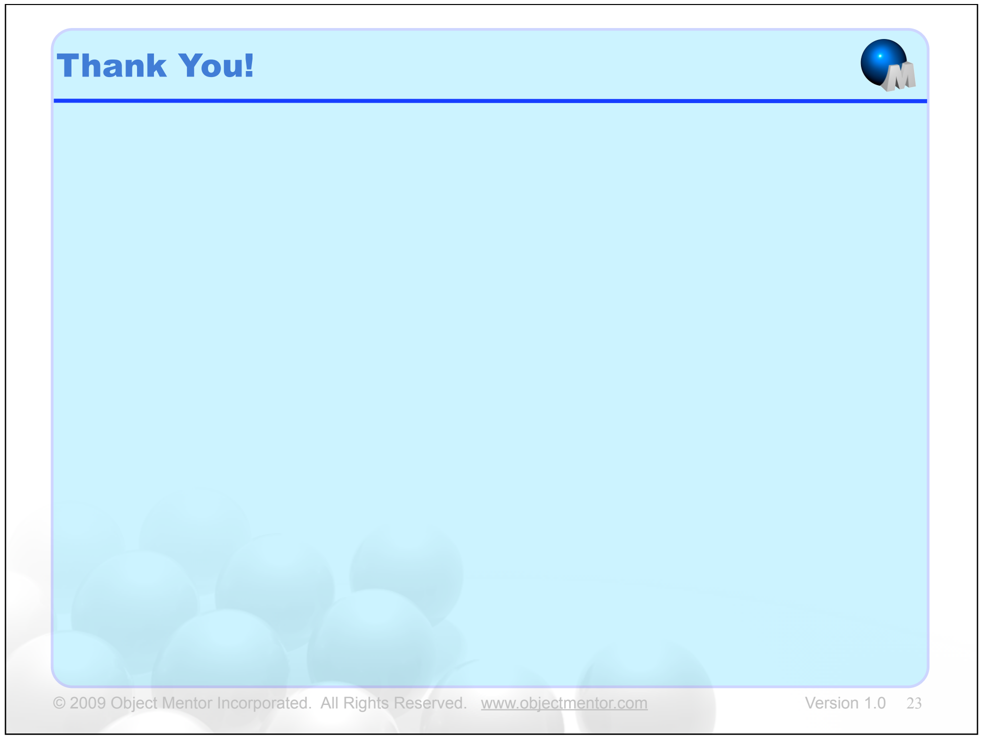


Closing Q & A

- Imagine you are back and work and you’ve been trying some of this on your own. Looking back, what will have been something you used from this tutorial?
- What did you expect to get covered that was not?
- Final questions ...

Timing: 10 minutes, 166 total

First question is future-perfect thinking, I want them to think about this and respond. The second question is a context-free meta question from Gauss and Weinberg “Exploring Requirements”.



Thank You!

Timing: 1, 167 total

Small and simple thank you. Since I'm looking forward to giving this talk, I appreciate the opportunity and I don't think I need to write a huge thank you.