ENGINEERING SOFTWARE AS A SERVICE

An Agile Approach Using Cloud Computing

Armando Fox & David Patterson • http://saasbook.info

Part I covers Software as a Service: SaaS Architecture, Ruby on Rails, and JavaScript

Part II covers Agile development: Behavior-Driven Design, Test-Driven Development, Design Patterns, Scrum, Velocity for both greenfield and legacy SaaS apps.

- Fulfills 2013 ACM/IEEE Computer Science curriculum standard for software engineering.
- Used in the free Massive Open Online Courses CS169.1x and CS169.2x from UC Berkeley and EdX, from which 10,000 people earned certificates in 2012.
- Instructor support includes instructor's manual, online videos, **automatic grading of programming assignments and quizzes**, an instructors' forum, and a ready-to-run virtual machine with all necessary software pre-installed.
- Visit **http://beta.saasbook.info/beta-program** if interested in joining beta

Comparative Analysis (as of May 2013)

Book	Fox & Patterson, *Engineering Software as a Service*	Pressman, *Software Engineering*	Sommerville, *Software Engineering*
Amazon Rating	★★★★½	★★☆☆☆	★★★☆☆
List price (new)	**$29.99**	$199.00	$156.20
Amazon price (new)	**$25.92**	$127.99	$121.49
Used price (Amazon)	**—**	$109.43	$103.86
Kindle price	**$9.99**	$103.98	$103.41
# of pages	**459**	895	792
# of chapters	**12**	32	28
Current Edition/Year	**2nd Beta, 2013**	7th, 2009	9th, 2010
First published	**2012**	1982	1982
Publisher	(Self published)	McGraw-Hill	Addison-Wesley

"It is a pleasure to see a student text that emphasizes the production of real useful software."
—Frederick P. Brooks, Jr., author of *The Mythical Man-Month*

"I'd be far more likely to prefer graduates of this program than any other I've seen."
—Brad Green, Engineering Manager, Google Inc

"I recommend this unique book and course to anyone who wants to develop or improve their SaaS programming skills." —Thomas M. Siebel, founder of Siebel CRM Systems

The full answer is in the *Communications of the ACM* article "Viewpoint: Crossing the Software Education Chasm", May 2012, pp. 17-22. (Link available on website http://saasbook.info)

Cloud computing and the shift in the software industry toward Software as a Service using Agile development has led to tools and techniques that are a much better match to the classroom than earlier software development methods. We leverage the productivity of modern programming frameworks like Ruby on Rails to allow students to experience the whole software life cycle repeatedly within a single college course, which addresses many criticisms from industry about software education. By using free-trial online services, students can develop and deploy their SaaS apps in the cloud without using (overloaded) campus resources. For each software engineering topic, we describe both the Agile and the "plan-and-document" methodologies: Waterfall, Spiral, and RUP. This contrast allows students to decide for themselves when each methodology is appropriate for SaaS and non-SaaS applications.

The experience of many instructors (including ourselves) is that students enjoy learning and using Agile in projects. Its iteration-based, short-planning-cycle approach is a great fit for the reality of crowded undergraduate schedules and fast-paced courses. Busy students will by nature procrastinate and then pull several all-nighters to get a demo cobbled together and working by the project deadline; Agile not only thwarts this tactic (since students are evaluated on progress being made each iteration) but in our experience actually leads to real progress using responsible practices on a more regular basis. We even show how to use Agile techniques on legacy code that wasn't developed that way to begin with; that is, Agile is good for more than just writing new code from scratch. Students are much more likely to actually follow the Agile methodology because the Ruby on Rails tools make it easy and because the advice is genuinely helpful for their projects. We believe Agile teaches skills that transfer to non-agile projects, should need arise.

Body of Knowledge coverage

KA	Knowledge Unit	Topics Covered	Hours
SE	Software Processes	Software lifecycle Waterfall, Spiral, RUP Agile Software quality Outcome: Core-Tier1 #1, #2, #3, #4, #5 Outcome: Core-Tier2 #6, #7 Outcome: Elective #8, #12, #13, #14	3
SE	Software Project Management	Pair Programming Scrum Conflict Resolution Meetings and Agendas Software Development Estimation Software risks and risk reduction Outcome: Core-Tier2 #1, #2, #3, #4, #5, #6, #7, #8, #9 Outcome: Elective #10, #11, #12, #13, #14, #15, #16, #17, #18, #19, #20, #21, #22, #24, #25	4
SE	Tools and Environments	Configuration management Version control systems Tool selection and use Outcome: Core-Tier2 #1, #2, #3, #4	4
SE	Requirements Engineering	Requiremente Elicitation Use cases and User Stories Lo-Fi UI Functional vs. Non-functional requirements Forward and Backward Tracing Risk mitigation via prototypes Outcome: Core-Tier1 #1, #2, #3 Outcome: Core-Tier2 #4, #5, #6 Outcome: Elective #7, #8, #10, #11	4