# Background [.75]

## extensive past work on autograding of code. Navrat 2014 even shows that performance on frequent but short autograded programming assignments can be used to predict students' final grades in programming-related courses with surprising accuracy.

## taxonomy: fully automated vs. human assist; test-based vs. static analysis

## about ESaaS; projects; use of Pivotal Tracker, CodeClimate, Travis for coverage

# Why Another Autograder? [.5]

## 17 autograding systems and over 60 papers about them, in 2006-2010 alone [survey]. why develop our own?

## many existing systems not readily availbale and/or tightly integrated to some LMS. we were using Coursera and later OpenEdX

## good Ruby tools => thought we could do better than for other languages

## cloud scale => new in 2011. cite enrollment #s from early courses.

## need for security/isolation from LMS

## for reasons of managing large cohorts of students w/diverse HW/SW, solutions that require extensive install/config on student machine (Mailing It In [cite], download & deploy grader yourself, etc) were rejected. Needed to be zero-config.

## we believe another challenge to adoption is hassle for teacher to deploy. We believe SaaS-based hosted LMSs like OpenEdX will help. We're running our autograder as a service with OpenEdX as a submission front-end, and welcome others to use it.

# *rag*, a Ruby-based autograder for ESaaS [1.25]

## Background: what's Rspec? what's Cucumber? what's Travis?

## our narrow Unix-like view of an autograder: gets submission & rubric, outputs a score and feedback. All other policy issues - resubmission policy,gradebook maintenance, LMS features - are out of scope. Likewise, we don't argue one way or the other whether autograders should completely replace human evaluation

## Student expderience. Submit files online, wait a bit, receive numerical score + feedback.

## RSpecGrader

## FeatureGrader

## CIGrader - like AWAT [cite]. can assess full-stack GUI apps interactively. Mentioned as Future Directions area in both Douce and Ihantola surveys.

## Summary table: grader types, types of assignments they grade, and how students submit work

# Cloud Grading Architecture WIth OpenEdX [.75]

## queue system and openEdX; can connect to other stuff as well

## autograder, rubric file

## CI workflow for ensuring autograders running

## Multiple layers of security: watchdog timers, sandboxed interpreter, threads. Hollingsworth 1960 observed that it was possible for students to submit programs that deliberately damage the autograder.

## Policy: resubmission, plagiarism detection, formative vs summative, avoding "autograder-driven development"

# Reflection/Lessons [.75]

## Value of tools ecosystem in Rails world. Majority of new autograders in 2010 survey were either for Java or language-neutral with output checking only. We had good reasons for using Ruby and still do. Student feedback in campus course bears this out.

## Getting unnecessary stuff out of the way: avoid wasting student time with admin, setup, etc.

### VM provides courseware; maybe soon C9.

### Heroku provides deployment.

### OpenEdX presents the HW explanations, videos, selfcheck questions along the way using RuQL (cite), inspired by ideas like OKgrader (cite)

## Challenge: tuning rubrics. Use campus course to debug.

## Challenge: stability. By and large edX works. Good separation of concerns between LMS & autograder authors.

## Challenge: test suite quality. THis si a general problem in SWE. Example: if multiple tests are effectively redundant, student scoring is positively distorted if all pass and negatively distorted if all fail.

# Future work .5

## As Douce et al. note (review), one flaw of many of the autograders in their survey (cite) is that "A program can be submitted that may be correct in its operation yet be pathological in its construction." (give example from Autostyle papers.) Opportuntiy: "rich because of scale" - automatic code feedback based on large # of submissions to same assignment. CodeWebs, others. pointers to AutoStyle papers.

## plagiarism detection: can do, but has generally been a non-goal for us. Woit & Mason found that not only is cheating rampant (in their own 5-year study and supported by earlier studies), demonstrated dramatically by students who got A's on required programming assignments but failed the exact same questions when they appeared on proctored exams, but also that students don't do optional exercises, and part of the value of an initial online assessment is to serve as a wake-up call to motivate students to practice,

## Others (cite) have instrumented every code checkin and/or instrumented the desktop IDE such as Eclipse (cite); Cloud-based IDEs may allow deeper instrumentation (cite Cryolite).

## Octobear integration (trigger autograders from GitHub push)

# Adoption by others .25

## Both surveys ask why existing autograders aren't reused more.

## SPOC instructors using it as part of overall ESaaS; hosted on edge.edx.org through arrangement with me; advise Sam interested

## Can also stand up your own OpenEdX and use our graders as a service. Contact Sam.

## Or take the code and create another adapter for it to plug into your favorite campus system (we'd love an LTI adapter). Github pointer.