Смешанное проектное обучение программной инженерии

Т.Берленко, М.Заславский, К.Кринкин

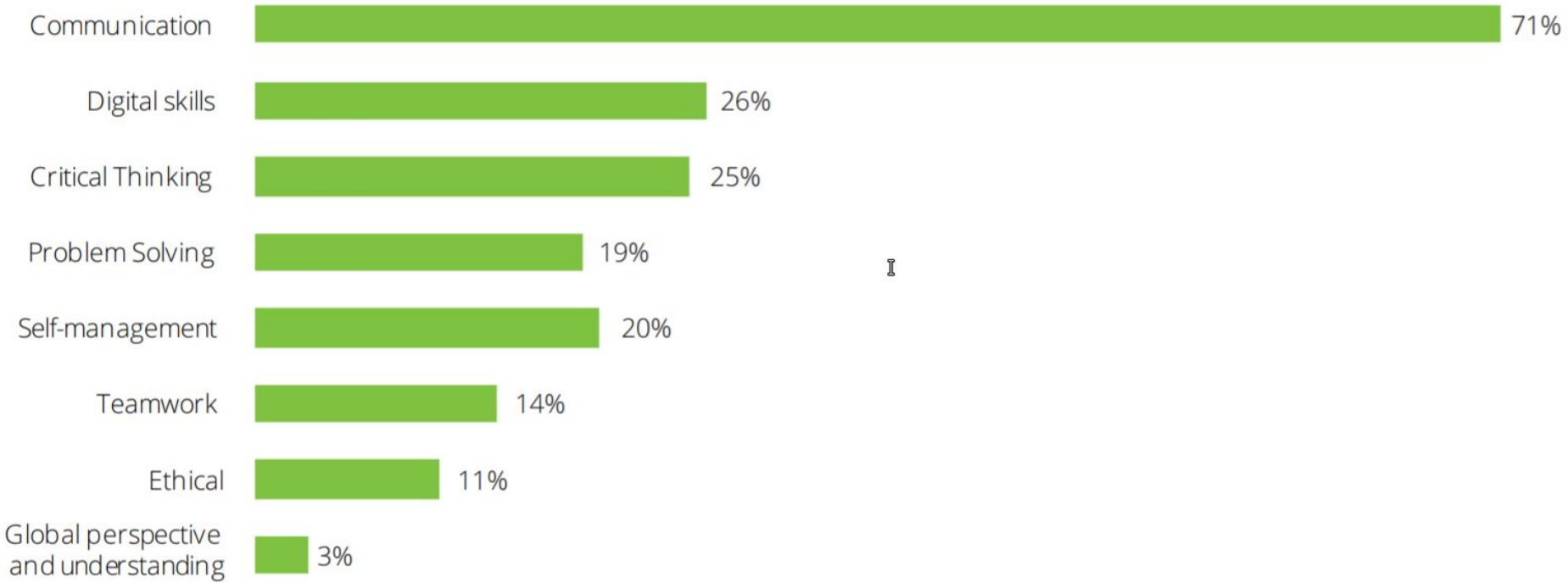
23 октября 2020

Ситуация

- Спрос на IT специалистов растет вместе с индустрией.
- Образование не успевает вслед за индустрией:
  - базовые знания требуют много времени на освоение,
  - учебные задачи часто оторваны от действительности,
  - новые технологии и подходы,
  - новые технические навыки,
- Рост требований к программному инженеру:
  - Исследовательские навыки,
  - **Soft Skills,**
  - Необходимо понимать весь процесс разработки (быть и менеджером, и исполнителем).

Как вырастить программного инженера в рамках ВУЗовской подготовки?

Chart 5.1: Demand for soft skills, Workible



Source: Workible

https://www2.deloitte.com/content/dam/Deloitte/au/Documents/Economics/deloitte-au-economics-deakin-soft-skills-business-success-170517.pdf

Существующие подходы к обучению ПИ

Подходы:

- Зубрежка теории (SWEBOK, Agile-манифесты, Паттерны)
- Зубрежка технологий (Языки программирования, СУБД ...)
- “Классическое” проектное обучение

Недостатки:

- Перекос в сторону технологий, а не пользователей.
- Нет возврата ответственности студентам.
- Сложно научить процессу и Soft Skills.
- Тепличные условия.
- Неравномерная нагрузка.
- Непонимание, как применять полученные знания в реальных условиях.

За рамками технологий

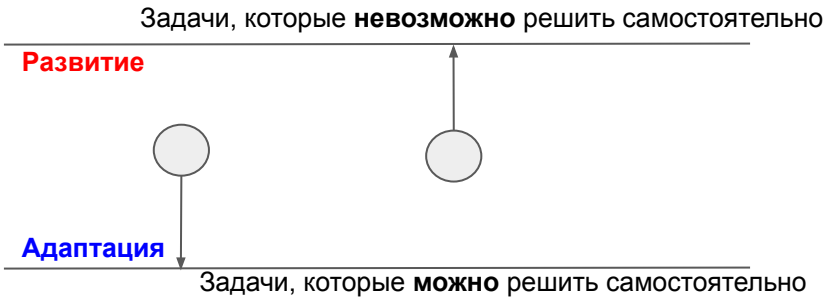
Полезные идеи, “интеграции в подсознание” студентов:

- Программы пишутся для людей.
- Ты лично отвечаешь за проект.
- Нужно проявлять инициативу.
- Процесс разработки продукта является частью продукта.
- Продукт всегда является образом создавшего его процесса.

“Кривая обучения”

Эффект Даннига-Крюгера

Зона ближайшего развития



Лев Выготский

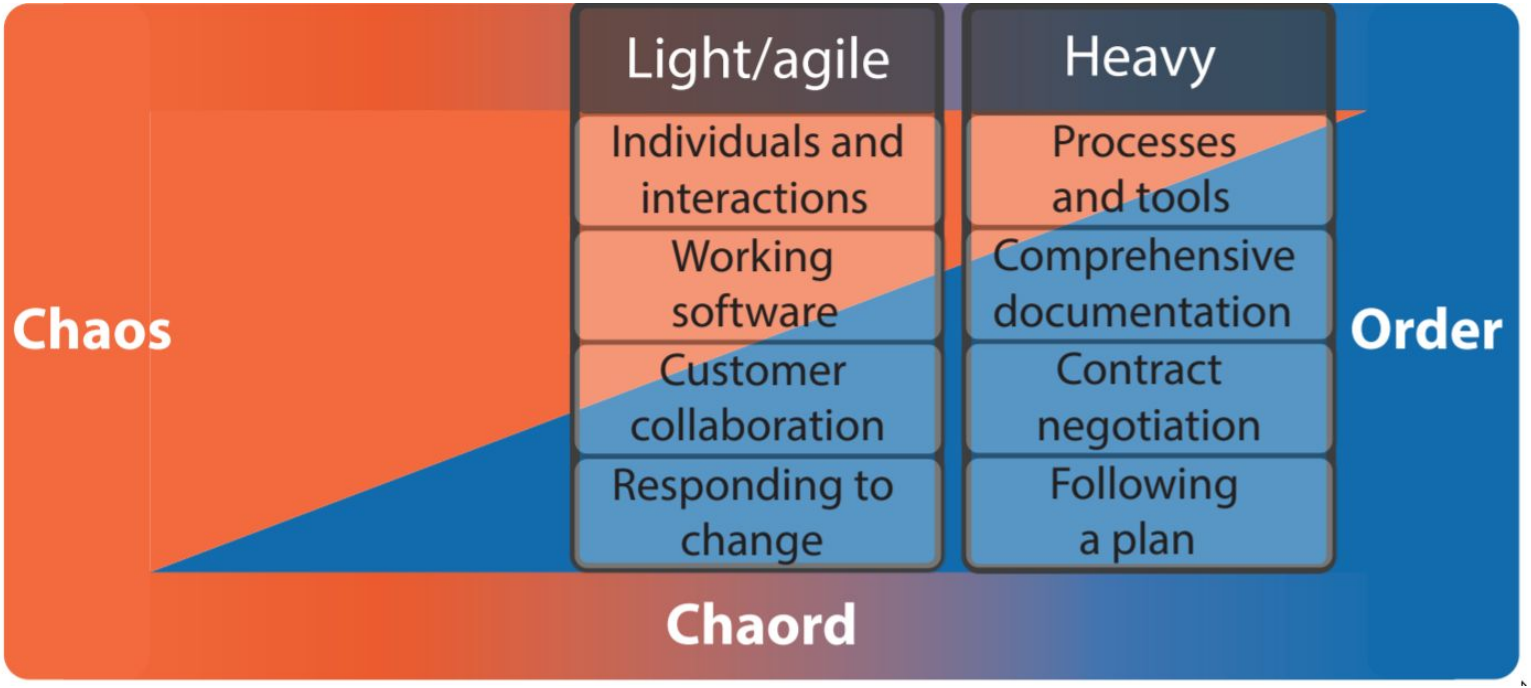
- Ученик активен: (выполняет самостоятельное действие)
- Учитель активен: ~~“делай, как я говорю”~~ “делай как я”
- Среда между учителем и учеником активна

Chaordic

"By chaord, I mean any self-organizing, adaptive, non-linear complex system, whether physical, biological, or social, the behavior of which exhibits characteristics of both order and chaos or loosely translated to business terminology, cooperation and competition."

© D. Hock, "The chaordic organization: Out of control and into order," World Business Academy Perspectives, vol. 9, no. 1, 1995.

Chaordic в программной инженерии



© S. Krusche, B. Bruegge, I. Camilleri, K. Krinkin, A. Seitz C. Wöbker. Chaordic learning: a case study. In Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track. IEEE Press, Piscataway, NJ, US

Концепция

Как быть?

- Студенту необходимо побыть с обеих сторон баррикады.
- Проектно-проблемно ориентированное обучение.
  - **Максимум работы делают студенты.**
  - Условия работы диктуются задачей.
  - Необходимо пройти максимальное количество этапов жизненного цикла.
  - **Необходимо столкновение с реальностью** (небольшое, но запоминающееся).
- Необходимо соответствовать организационным ограничениям ВУЗа:
  - Понятный алгоритм получения оценки.
  - Соблюдение сроков обучения.
  - Синхронизация с учебным планом.

Идея организации дисциплины

- Цель - успех проекта.
- Обучение на практике (но лекции также присутствуют).
- Реальные проекты:
  - Нет синтетических задач.
  - Заказчики.
  - Приемочное тестирование.
  - Требования к процессу.
- Двухэтапное обучение:
  - 5й семестр - как разработчик,
  - 9й семестр - как лидер проекта и куратор группы проектов.
- Опора на лучшие практики: SWEBOK и Agile
- Широкий пул технологий (в том числе и неизвестные студентам).

SWEBOK

- **Software requirements** – программные требования
- **Software design** – дизайн (архитектура)
- **Software construction** – конструирование программного обеспечения
- **(включаем в требования)Software testing** - тестирование
- **(включаем в требования)Software maintenance** – эксплуатация (поддержка) программного обеспечения
- **Software engineering management** – управление в программной инженерии

Что мы берем из Agile

- Система итераций.
- Гибкий подход к планированию.
- Частые синхронизации с заказчиком.
- Частые релизы.
- Работоспособность проекта - основной приоритет.

Реализация

Требования к проектам

- У проекта есть заказчик: преподаватель/индустриальный партнер, с которым может взаимодействовать команда.
- Используется открытый стек технологий.
- Проект может быть приведен к завершенному виду за семестр.
- Для проекта есть единообразное описание:
  - Цель,
  - Задачи,
  - Стек технологий.

Иерархия студентов и команд

Команда: 1 студент первого года **магистратуры** + 2-4 студента третьего курса **бакалавриата**. Студент-магистрант является **лидером** проекта.

Все команды поделены на группы по 3-4, каждой группой руководит **магистрант-куратор**.

Распределение студентов по проектам

Принципы:

- Кто первый встал, того и проект.
- Осознанный выбор.

Последовательность:

- Преподаватель озвучивает дату и время открытия списка проектов.
- Студенты выбирают максимум три проекта, которые им интересны.
- Преподаватель синхронизирует пожелания с реальностью.

Работа в течение семестра

Семестр поделен на 4 итерации, продолжительность каждой - 1 месяц.

Первая итерация: студенты разбиваются на команды, знакомятся, выявляют требования к проекту. Студентам предоставляются онлайн курсы, которые призваны уравнять знания.

По завершении первой итерации:

- Создан первоначальный макет UI, согласован с преподавателем.
- В репозитории проекта созданы задачи на следующую итерацию.
- Приложение собирается у куратора и показывает нужный UI с заглушками/есть видео его работы.
- Студенты прошли необходимые онлайн-курсы.

Работа в течение семестра

Остальные три итерации направлены на разработку функциональности приложения, начинает наблюдаться рост Product Value.

В конце каждой из трех последних итераций:

- В репозитории проекта созданы задачи на следующую итерацию.
- В репозитории проекта находится презентация команды.
- Есть тесты для разработанной функциональности.
- Приложение собирается у куратора (используется инструкция от команды) и показывает нужный UI со всеми требуемыми задачами.

В завершении каждой из итераций проводится показ проекта, в котором участвуют команды и заказчики.

Организация взаимодействия в процессе итераций

Бакалавры-разработчики:

- Выполнение задач: программирование/тестирование

Лидеры проекта:

- Управление задачами/командой/проверка кода в репозитории

Кураторы:

- Проведение встреч с командами
- Согласование архитектуры проекта с командой и с руководителем курса
- Предоставление результатов работы за итерацию руководителю курса
- Проверка задач, проектов и оценивание участников

Промежуточные метрики процесса

Команда проекта:

- **Product value** (пропорционально use cases).
- Своевременность и выполнение требований итераций.
- Состояние репозитория.

Куратор:

- Эскалация проблем.
- Качество взаимодействия.

Оценивание по результатам итераций

Критерии оценивания бакалавров:

- Задачи, поставленные лидерами команды/кураторами, были выполнены в полном объеме и вовремя.

Критерии оценивания лидеров:

- Корректное и полное описание задач на итерацию.
- Выполнение задач в полном объеме и вовремя.
- Качество демонстрационных материалов.
- Пригодность результатов к дальнейшему использованию.

Результаты

Статистика

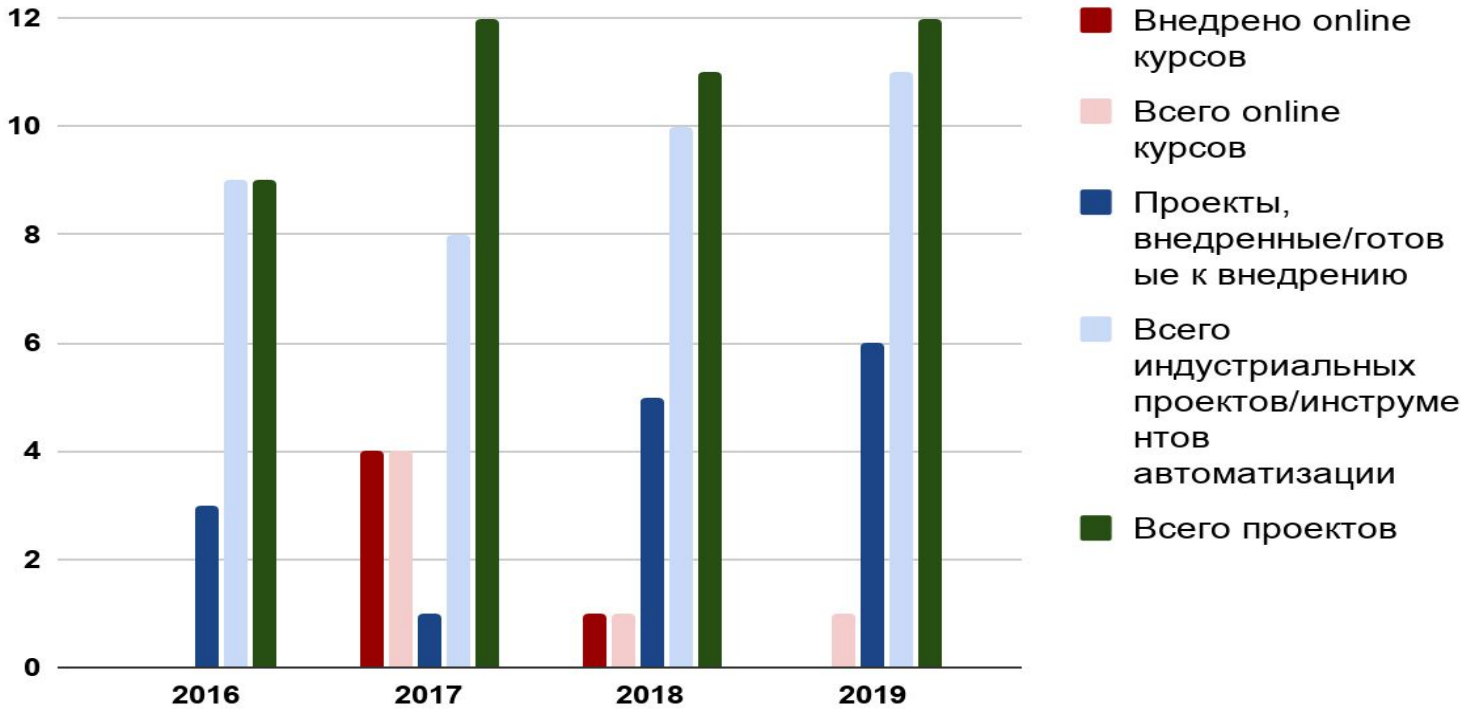
За **4** года запуска курса суммарно в развитии **44** проектов принимало участие **149** студентов.

Их усилиями было написано более миллиона (ровно **1 687 375**) строк кода и сделано **3253** коммита в рабочие репозитории.

Проблемы

- Не все студенты достаточно мотивированы участием в проектной деятельности на протяжении всего семестра: некоторые студенты привыкли делать всё в последний момент.
- Студенты в любой момент могут отчислиться.
- После получения оценки по дисциплине могут не участвовать в процессе развертывания приложения.
- Не все проекты удается довести до финала за семестр, не всегда получается продолжить начатое другим студентам.

Статистика внедрения проектов



Разработка онлайн-курсов

Студенты принимали участие в разработке онлайн-курсов на Stepik:

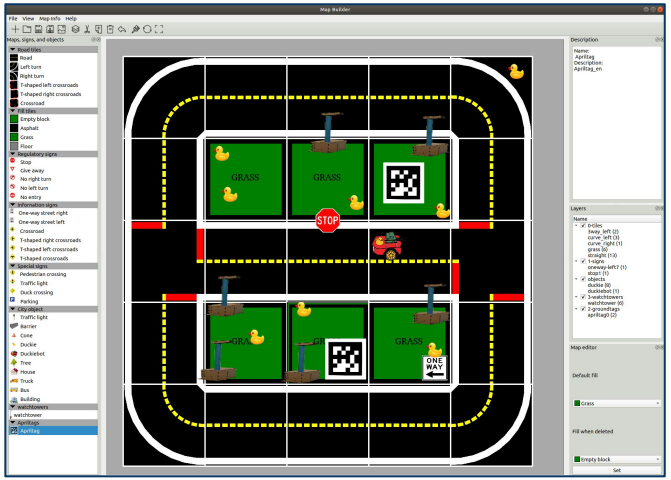
- Введение в нереляционные базы данных: https://stepik.org/course/2586
- Информатика ЛЭТИ (закрытый курс)
- Программирование ЛЭТИ (закрытый курс)
- Отладка и профилирование в Linux (в данный момент дорабатывается)

А также сами с нуля создали курсы:

- Тренажер решения задач на языке С: https://stepik.org/course/44687
- SSH Трюки (в данный момент дорабатывается)
- Работа с гугл-таблицами (в данный момент дорабатывается)

Успешные внедрения

В 2019 году студентами курса был разработан проект “Редактор карт симулятора Duckietown”. С помощью студентов в рамках летней стажировки в JetBrains Research проект был переведен на английский язык и получил новые функции.



В 2020 году проект был интегрирован в репозиторий проекта Duckietown: https://github.com/duckietown/dt-gui-tools/pull/6

Примеры тем проектов

- Автоматическая проверка дипломных презентаций
- Инструмент для генерации датасетов для фотограмметрии
- Инструмент для генерации отчетов
- Сбор статистики курса "Основы программирования для Linux"
- Генератор карт для симулятора Duckietown
- Расширение Chrome для улучшения UI github
- Инструмент для проверки человека на присутствие в БД
- Тренажер публичных выступлений
- Инструмент автоматического удаления пауз из видео

Ссылки

- S. Krusche, B. Bruegge, I. Camilleri, K. Krinkin, A. Seitz C. Wöbker. Chaordic learning: a case study. In Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track (ICSE-SEET '17). IEEE Press, Piscataway, NJ, US
- Концепция курса: http://se.moevm.info/doku.php/courses:mse:concept
- Архив 2016 - 2019 года: http://se.moevm.info/doku.php/courses:mse:archive

Контакты

Кирилл Кринкин
kirill@krinkin.com