Assessment Strategy to 'Future Proof' Students as Computing Practitioners

Helen Partou (h.partou3@herts.ac.uk) & Lindsay Smith (l.1.smith@herts.ac.uk)
University of Hertfordshire, UK

Development > Rollout > Evolution

- Motivation
  - Team-based software development is core module delivery in computing at Hertfordshire
    - Students need relevant software engineering experience(s)
  - Previous software development platform not 'fit for purpose' teaching resource
    - Not compatible/upgradeable/adaptable
    - Overly complicated for 'Zero to Hero' student assessment in a six week development cycle

2014-15	2015-16	2015-16 Semester A	2015-16 Semester B	2016-to-date
Research technical options & feasibility of platform	Build platform & supporting resources	Pilot platform as teaching tool with small L7 cohort	Larger scale rollout for L5 cohort(s) on-campus & distance learning	Multiple module adoption & evolution of delivery

- Timeline, Scope & Feasibility
  - Development
    - Estimated 500 + staff hours
  - Approximate take up to date
    - In 7 modules
    - Delivered to 1000+ students
    - Assessed equivalent of 200 student teams

Assessment Strategies to Scope Student-based Solutions

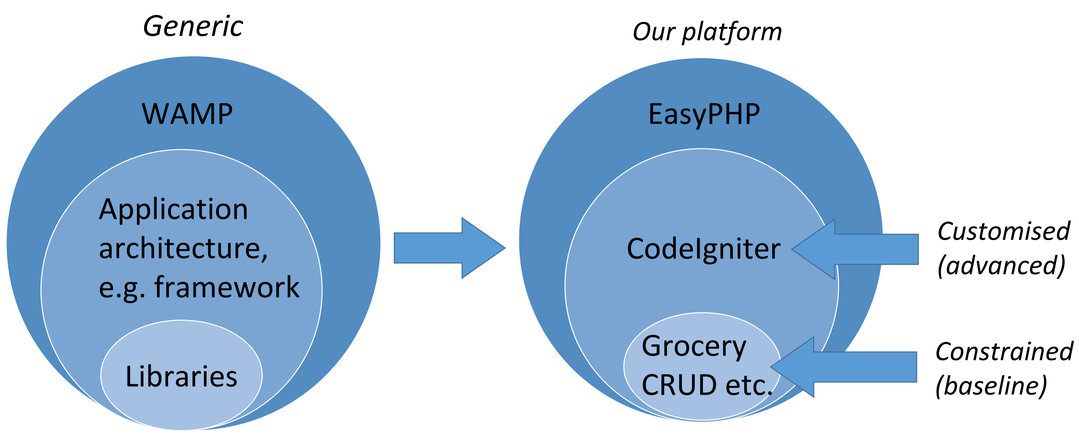
- Teaching resources are customised to support assessment
  - Demonstration videos, FAQs and supervision supports instructional scaffolding as students gradually increase technical expertise.
    - An example 'Orders' application provides opportunities for formative feedback and minimises the student-tutor 'expectation gap' [2] of assessment deliverables.
- Applications built in the platform are potentially scalable to any real-world scenario
  - Supports constructivism, e.g. cinema film showings
- Limitations for summative assessment include:
  - Managing trade-offs between case study complexity and platform functionality to define project scope
    - For example: matching deliverable technical competences with available assessment timeframe
- Summative assessment strategy has categorised marking criteria
  - ***Baseline*** = minimum engagement for a pass mark
  - ***Advanced*** = independent tasks gain higher marks
- Example documentation for software: User Acceptance Tests (UATs)
  - Staff simulate client role to check software is 'fit for purpose'
- UATs support delegation of tasks to team members
  - Promoting "T-Shaped" individuals (specialised generalists) [3]

1	**Smart Counties R Us Management System: additions, editing, retrieval/display (via filter), and deletion of data by using the system interface. *N.B. This part can all be achieved with the default framework implementation***	**Your mark**	**Max mark**
a)	i) Add new product [1 mark], ii) Add new SME [1 mark], iii) Associate a product with an SME [2 marks]		/4
b)	i) Add new area [1 mark], ii) Add new resident [1 mark], iii) Associate a resident with the area they live in [2 marks].		/4
c)	…		…
	Section 1 subtotal	**0**	**/20**
2	**Smart Counties R Us Management System interface: usability criteria**	**Your mark**	**Mark**
a)	User experience: colour scheme, page design and feedback, e.g. confirmation messages after user actions		/10
b)	Error/validation messages are meaningful and helpful		/6
c)	…		…
	Section 2 subtotal	**0**	**/30**
3	**Smart Counties R Us Management System: advanced features. *Weighting of marks in this part reflect task difficulty. N.B. These all add up to MORE than 25 marks, so choose features to implement wisely as 25 marks is the maximum.***	**Your mark**	**Mark**
a)	Allows deletion and/or disabling of data (ability to reactivate data)		/10
b)	Prevents registering duplicate data, e.g. the same resident twice		/5
c)	…		…
	Section 3 subtotal	**0**	**/25**

Purpose-built Platform as a Teaching Tool

- **Web-based**
  - We built an open-source development stack with an example 'Orders' system, utilising the Model-View-Controller (MVC) architecture for students to undertake data-driven web programming.

- **Portable & robust**
  - The tool is 'plug-and-play' and can be integrated with cloud-based tools.
  - Lightweight, compatible with multiple environments, re-usable and 100% reliable to-date .
  - Students can experiment with impunity.



Are Soft Skills Harder than Hard Skills in Software Development Projects?

- Problems
  - A STEM educational challenge is inherent complexity in delivering software development skills
    - Preparing students for employment in the computing industry
      - Employers cannot put graduates 'in front of a client' [1]
  - Teaching 'hard' skills focuses on technological constraints
    - Keeping up with technological change and advances
  - Teaching 'soft' skills focuses on team work
    - Student participation: **passengers** (lack of interest, engagement and/or feeling of inferiority) vs. **diligent isolation** (poor delegation, perfectionism and/or presence of passengers)
- Solutions
  - Reduction in technical complexity, e.g. robustness of platform enables 'Zero to Hero' solutions
    - Agile approach, staff development and staff-student feedback
  - Optimising teaching staff engagement with student teams
    - Managing student team autonomy
    - Student and staff teams collaboration
      - Team clinics, tutorial triage

Industry 4.0 and Future Developments

Current developments

- Technical
  - Exploring integration of the platform with Git-based systems, e.g. Azure DevOps, which facilitates sophisticated version control in the cloud.
- Compassion-focused pedagogy (CfP) [4]
  - Supporting student team dynamics and task management.

Future developments

- Feasibility of adapting this approach to fast-moving technological change.
  - How the approach and/or platform integrates with, or could transfer to, other fields and technologies
    - Such as Internet of Things (IoT) e.g. 'smart'/cognitive technologies/digitalisation.

References

[1] Matthews, D. (2016) 'What should computer science degree students learn?'. *Times Higher Education*. 10 March 2016. Available at: https://www.timeshighereducation.com/news/what-should-computer-science-degree-students-learn

[2] Christenson, S., Reschly, A. & Wylie, C. (2012) *Handbook of Research on Student Engagement*. New York: Springer

[3] Rubin, K. (2012) 'T-shaped Skills and Swarming Make for Flexible Scrum and Agile Teams'. Available at: http://www.scrumexpert.com/knowledge/t-shaped-skills-and-swarming-make-for-flexible-scrum-and-agile-teams/

[4] Gilbert, T. (2017) 'When Looking Is Allowed: What Compassionate Group Work Looks Like in a UK University'. In Gibbs, P. (eds.) *The Pedagogy of Compassion at the Heart of Higher Education*. Cham, Switzerland: Springer. pp. 189-202.

Poster presented at Advance HE STEM 2019