CISE ENSE701

# Problem-based Scenario for the Course

## Background to this course

In this course, we will be practising a number of skills and gaining capability by developing software in teams for a client using specific software processes and practices (supported by tools commonly used in industry). The focus is on learning the theory (principles) and practical use of software development techniques and tools through doing software development and reflection on our experiences and improving by learning from mistakes. You should be understanding not just what and how, but also WHY certain techniques are good practice.

We are using a problem-based approach to learning, where the problem is to understand, develop and deploy a product for a client. The problem we will work on during the course will be the context and motivation for learning the practices, techniques and tools for collaborative software development. The product is NOT the main focus, but rather HOW and WHY we should develop a product in teams using good practice IS the focus. This is mainly what we will be learning and practising, and be evaluated on, not the product. It is expected that only part of the product will actually be developed in this course, since we have limited time and resources.

We will learn about, practise and be evaluated on your understanding and application of:

- What values and principles should guide your software development behaviour and practices and why
- How to interact with a client before and during the product development process and why
- How to develop a software product iteratively and incrementally in a small team and why
- How to capture user needs as user stories and acceptance criteria, and convert these into product features and designs and why
- How to decrease the risk of code integration problems when several developers code separately
- How to plan and monitor work and why
- How to manage change to requirements and why
- What to do to ensure code is ready to be deployed – it behaves as expected with no bugs – and why
- What characteristics of code make it good quality (or not) and why
- How to collaborate with others to code, including pair and mob programming, and why
- How to set up a development environment and tools that will allow good practice to be followed
- How to learn enough of a tech stack to make some progress in product development
- Learning how to learn from a number of sources commonly used in industry

The next section introduces the software development problem that will motivate our learning in these areas.
Please treat Peter the Product Owner as a “real” client and interact professionally with him.
Please make sure you understand the academic success criteria for each assessment item (HINT – it is NOT the product!)

Prepared by Peter The Product Owner

**Version History**

July 2021 V1. **Initial Ideas**

**Constraints**

1. The MERN technology stack must be used because I have developers with expertise in this stack who will continue development and maintenance of the product after you.
2. I will keep in touch with you using the MS Teams channel mainly and will visit face-to-face when I can.
3. Product name: Software Practice Empirical Evidence Database (SPEED)
4. The product should be available as a web app on any device
5. We cannot provide links to published articles – they are the copyright of the publisher
6. I first just want to test the idea out with some practitioners so only need limited functionality initially. I need to work out what the minimum product features will be and discuss this with you as the priority to work on first. I am NOT expecting all features to be finished by the end of your initial contract.

**Overview and Motivation**

I am passionate about Evidence-based software engineering and I want to support developers' decisions about the use of different practices *based on evidence and experience* rather than possibly unsubstantiated claims. There is a lot of evidence about claims that are documented in published academic research papers but these are unavailable to many commercial software engineers because: (a) they are behind a paywall, (b) they are written in unfamiliar academic language and (c) it is difficult to find the trends in evidence to make a decision without a lot of searching, filtering and reading. I want to make this easier for practitioners by doing most of this work and storing the results in a searchable database.

For example, there are many claims about the benefits of the SE practice "Test-Driven Development" (TDD). For instance, there are claims that using a TDD way of working will improve the quality of the code written by developers, and reduce the number of bugs that get through to a deployed product, compared to when developers leave writing tests until after the code has been written. But is there any evidence that these claims are true? Well, there are case studies and experiments that have been done to check these claims and have been published. Some of these published studies get results that agree with the claims, and some don't. I want to make this evidence available easily to practitioners, just by searching a particular claim for a particular SE practice, they will be shown the list of evidence (extracted from the published articles) that relate to this practice/claim with the different results shown. This will help them to decide if they should use this practice or not and believe the claims or not.

I call my software product the Software Practice Empirical Evidence Database (SPEED). Basically it will be a searchable database of evidence about different claims about different SE practices. These will be found and put into the database by SE experts I will hire in the Software Engineering Research Centre (SERC) at AUT. Any practitioner, researcher or student will be able to look for evidence related to claims about software engineering practices by searching this database and they will be shown a **summary** of the published relevant research evidence for and against the claims. For example they may be wondering if the claim is true that using TDD as a way of working improves code quality. They can open the SPEED app and select "TDD" from a list of SE practices which we have in our database, and then select "improvements to code quality" from a list of claims about TDD, and then be shown a list of empirical research articles titles and whether their results agree or disagree with that claim. (Note: To get the pdf of the full article a user will have to have access to the appropriate publisher's database or there may be pre-published versions available on databases like arXiv.org by themselves). We cannot provide the pdf in our database due to copyright restrictions).

**Workflow and roles**

Three main activities that the SPEED product needs to support are: finding suitable articles, extracting the appropriate information from them, and entering them into the SPEED database. My plan is to crowd source the articles. Anyone from the public can propose an article to include in SPEED by submitting a link to it. Staff from

SERC (Moderators) will then moderate the article for quality and non-duplication. If it passes the moderation process the article is then read and analysed by staff at SERC (Analysts) and the relevant information extracted from the article and entered into the SPEED database.

**Finding articles**

Anyone can suggest an article to include in the SPEED database. A Submitter will submit the bibliographic details only (NOT the pdf) of a published study they want to suggest should be included in SPEED (e.g. Title, authors, journal name, year of publication, volume, number, pages, DOI). It needs to be enough information so the Moderator and Analyst can use AUT library to find the original article. The SPEED app will have a submission form for a Submitter to fill in. The bibliographic details can be uploaded into the SPEED form in a standard-format file (eg Bibtex format) or the information can be typed in manually, or a combination of these. The pdf version of the article can NOT be included, for copyright reasons. There can be no link to the article online either, apart from the DOI (Document Object Identifier – a URI) of the article. Users may have to pay the publisher to get the full article or pay the fees to join a commercial online database such as ACM or IEEEXplore. This is outside the scope of SPEED, however.

**Moderation of Articles**

Submitted articles will go into a queue in SPEED for the SERC moderator to do a quality check. The moderator should be notified by SPEED if there any articles waiting in the queue (emailed report?). The moderator will then retrieve the list of articles in the queue awaiting moderation. They can then manually check that each article in the submitted queue is not already in SPEED or has not been rejected previously (can this be automated?). They will also check that each article is relevant to SE and the empirical evaluation of SE practices and methods and has been published in a peer reviewed journal or conference.

**Analysis of Articles**

Articles that get passed by the Moderator will go into another queue ready for the SERC Analyst to work on. Rejected papers go into a rejected papers database. The analyst is notified if there are any papers in the queue ready for analysis (emailed report?). The submitter is notified of the outcome of the moderation (accept or reject).

The SERC analyst will analyse each of the papers in the analysis queue and extract the appropriate information from the articles and enter it into SPEED. The bibliographic information should be automatically transferred to SPEED from the information the submitter submitted. The input screens should be quick and easy to enter the information, with default values and drop down lists where appropriate.

As a stretch goal, in the future it would be desirable to have SPEED automatically extract some of the information from articles to be entered into SPEED for checking by an analyst. It would also be great in the future to have SPEED searching online databases for suitable articles for inclusion in the repository, and recommending these, using AI techniques.

**Searching SPEED**

Anyone can search the SPEED database - students, researchers, or practitioners. There is no restriction on who can search. Users of SPEED will be able to search for a specific SE method from a fixed list of methods in our database (which will grow over time). Having selected the SE method of interest, the user will be shown a fixed list of claims that relate to the SE method chosen. The user can also select a range of publication years to show. Any query done by a user can be saved and the user can run those queries at a later date to get the latest results.

These users can also submit a rating (1-5 stars) for any article and the average rating is displayed as “user article rating”.

As a stretch goal, in the future, a ChatBot could be used to assist with creating a query and returning the results.

### Viewing results

Initially, the results can be displayed in a table with each article source with evidence result for the claim selected, shown in a different row. The columns to be displayed can include title, authors, year of publication, journal/Conference name, SE practice, claim, result of evidence (e.g. agree or disagree), type of research (e.g. case study, experiment), type of participant (eg. Student, practitioner). It should be possible for Searchers to sort by any column (at least authors, publication year, and claim, and evidence result). It should also be possible for users to select which columns to display (and which to hide) so they can see the info that is important to them on a single page width.

Other ways of displaying (e.g. cards) and filtering the results will be considered in future.

### Administration

The Administrator of SPEED should be able to modify data and configure SPEED.