**RQ1: What are the different teaching strategies used in software architecture education?**

|  |  |  |
| --- | --- | --- |
| Strategy | Explanation | Study ID |
| 1. Teach traditional classes in software architecture education | In this strategy, professors or tutors share their knowledge with students in person. | S29 |
| 2. Flipped classroom approach | Teachers or educators offer lectures or show videos to students before the face-to-face session. | S29 |
| 3. Problem-based learning | Students address architectural problems and instructors play minimum roles in this strategy S29, while S42 examines the application of problem-based learning to adult learners focusing on challenges and effectiveness of the strategy without any preset goals and expected outcomes. In S43, it is tailored for computing courses and defined as mapping real-world conceptual models directly to high level component abstraction. This is further refined to analyse their understanding using Neural Pathway Based Learning (NPL) strategies. *(It is a pedagogical technique to use a combination of teaching strategies to create deep understanding, knowledge, and critical evaluation by using techniques such as a multiple-choice quiz, animations, games and puzzle based learning, and repeated learning)*. | S29, S42, S43 |
| 4. Case-based learning | This strategy involves a real-life scenarios that allow students to analyse, apply taught concepts and mitigate potentials adjustments S29. In S42, courses were workshop-based with standard structures such as preset goals and expected outcomes. | S29, S42 |
| 5. Online learning | In this strategy, two fundamental modalities were used to deliver learning outcomes namely Massive Open Online Courses (MOOCs) and Small Private Online Courses (SPOCs). | S29 |
| 6. Intelligent Tutorial System | This strategy provides personalised learning pathways by tracing each student’s progress. | S29 |
| 7. Product-Based Learning (PBL) | Product-based learning strategy for programming assignments requires students to complete assignments individually, grade by a human, employ graphical user interface, have open-ended specifications and resembles a real-world product. | S30 |
| 8. Apprenticeship learning | This is an educational model in which an apprentice learns a skill or trade through observation and practical experience under guidance of an expert. | S30 |
| 9. Patterns-based approach | This strategy leverages patterns for modelling instructional design knowledge and to connect it with patterns in software architecture. | S31 |
| 10. Abstract modelling | The strategy focuses on abstraction modelling, design patterns and functional decomposition in software engineering which enhances student’s understanding of system design and prepares them for real-world software development opportunities S32. | S32 |
| 11. Abstraction and System Thinking | For this strategy, the study highlights the importance of teaching abstraction and systems thinking to improve proficiency in modelling software systems. | S33 |
| 12. Student Ownership of Learning (SOL) | This is a strategy that features a model of learning that blends desirable behaviours of student to increase achievements of students and level of learning which is built on their own experiences, knowledge and view of how the world works. | S34 |
| 13. Project-based learning  Mobile Project-based Learning [S39] | This strategy addresses challenges of teaching embedded software to students who have limited electronics background integrating open-source tools and hands-on projects S35, while S36 studied the student’s experience with the group-based project exams. With mobile project-based learning, it creates an interactive and collaborative learning environment S39. | S35, S36,  S39 |
| 14. Usability-Supporting Architecture Patterns (USAPs) | By adopting Usability-Supporting Architecture Patterns (USAPs approach, it helps to tackle the problem of usability flaws in software systems focusing on architecturally sensitive usability concerns early in the design process S37. | S37 |
| 15. Early Engagement Strategy | In this strategy, students are involved from the first days of the course period to prevent them dropping out. | S38 |
| 16. Adapted Learning Contract Strategy | With this strategy, course contents covered are reinforced by adapting to specific characteristics of each student to increase performance and chances of passing the course. | S38 |
| 17. Collaborative Decision-making | This strategy explores behaviour of software  engineering students as a beginner software architects in various roles and finds out how they make, challenge and capture a set of an architecture design decision and architectures in a collaborative way. The roles assumed by students is expected to enable and promote critical design thinking for decisions in a better quality and architectures. | S40 |
| 18. Scrum as a Teaching Strategy | As scrum is a popular empirical process control model designed to manage complex activities, it was found promising to be used as one of the teaching strategies to facilitate learning process in software architecture. | S41 |
| 19. Software Product Line Architecture | This is a methodology in which common features are shared with a family of products, addressing business, architecture, processes and organisational aspects. It has proven to be a methodology for the development of software products at lower costs, shorter time, and with higher quality. This is a proposed architecture to develop mobile-learning (m-learning) applications. | S44 |
| 20. Constructive Alignment | This strategy or principle combine constructivist learning theories and aligned curriculum to enhance the learning outcomes by encouraging students to adopt deep learning approaches. | S45 |
| 21. Simulation-based Teaching Strategy | This strategy uses many system-engineering concepts of modelling, simulation, visualization, and analysis with the goal to disciplined design, implementation, testing, and evaluation practices in students. | S46 |
| 22. VisAr3D: Virtual and augmented reality | This approach called Visar3D (Software Architecture Visualization in 3D) mobilises teachers and students in software architecture teaching/learning, using virtual reality (VR) and augmented reality (AR) technologies. This will allow an intuitive exploration and interaction by software architecture students using the resources and facilities such as the manipulation of models in a simulated learning environment using virtual elements. | S47 |

RQ1.a. **Which teaching strategy aligns with industry expectations and requirements?**

|  |  |
| --- | --- |
| Strategy | Industry Alignment |
| 1. Teach traditional classes in software architecture education | Students design software systems that solve real-world problems, meet the requirements of stakeholders and share positive user experience S29. |
| 2. Flipped classroom approach | Students learn to implement concepts in the real-world scenarios S29. |
| 3. Problem-based learning | Students involve in sharing experiences and thinking of other students applied in real-life scenario S29, while S42 emphasises on practical skills and real enterprise application programming and integration. In S43, students focus on Service Oriented Architecture (SOA) and component-based software engineering (CBSE) with different problem scenarios. |
| 4. Case-based learning | Students learn to analyse and apply taught concepts and mitigate potential adjustments within a realistic context S29, while S42 emphasises on practical skills and real enterprise application programming and integration. |
| 5. Online learning: Delivered through two fundamental modalities | This strategy helps students to be taught with course contents and experiences in improving the quality and productivity of software professionals in the academia and industry S29. |
| 6. Intelligent Tutorial System | Architects are taught with necessary skills to enter the job market S29. |
| 7. Product-Based Learning (PBL) | Aligns with industry expectations by simulating real-life software product development S30. |
| 8. Apprenticeship learning | Development of skills which involve tangible processes that can be easily observed S30. |
| 9. Patterns-based approach | Aligns with industry expectations by leveraging software architecture patterns S31. |
| 10. Abstract modelling | The proposed teaching strategy align with industry requirements for software engineering as it provides step-by-step approach in gaining holistic view of a system while understanding what needs to be done S32. |
| 11. Abstraction and System Thinking | Emphasising modelling in terms of functions, structures, and behaviours aligns with industry practices, preparing students for real-world software engineering challenges. |
| 12. Student Ownership of Learning (SOL) | This model emphasise on problem-solving and critical thinking of skills which aligns with the expectations of the industry S34. |
| 13. Project-based learning  Mobile Project-based Learning [S39] | Students are involved practical skills to complete a project which aligns with the industry expectations S35, S36 while students are engaged in practical project-based tasks using mobile technologies ensuring technological adaptability and advancements S39. |
| 14. Usability-Supporting Architecture Patterns (USAPs) | Usability issues of the software architecture was tested in real-world settings S37. |
| 15. Early Engagement Strategy | The strategy used a gamified platform for individual practice that tracked specific-purpose programming exercises S38. |
| 16. Adapted Learning Contract Strategy | Students were assigned various weekly tasks with teacher’s comments and feedback to improve their skills S38. |
| 17. Collaborative Decision-making | Students are prepared for a real-world challenge in software architecture by mimicking industry roles in decision-making scenarios S40. |
| 18. Scrum as a Teaching Strategy | While implementing scrum as a teaching strategy, students were engaged in creating Sprint Goal, Sprint Backlog with plans for that particular sprint. This gave them skills in managing projects which mirrors real-world industry expectations and requirements S41. |
| 19. Software Product Line Architecture | This methodology proposes viable architecture to develop m-learning applications to teach programming. Therefore, it aligns with the industry expectations and requirements S44. |
| 20. Constructive Alignment | This strategy integrates formative feedback and aligns assessments with real-world tasks and the approach aims to develop skills that are directly applicable in the software industry S45. |
| 21. Simulation-based Teaching Strategy | Students are involved in a real project in testing the model with appropriate adjustment according to the tasks assigned. This helps them in preparing for the real-world industry requirement S46. |
| 22. VisAr3D: Virtual and augmented reality | With large-scale systems and advanced visualisation techniques, the study presents industry practices and tools to learn and teach software architecture S47. |

**RQ2: How are students provided hands-on practical experiences related to software architecture teaching?**

|  |  |
| --- | --- |
| Strategy | Hands-on Practical Experiences |
| 1. Teach traditional classes in software architecture education | Students learn how to design software systems that solve real-world problems, meet the requirements of stakeholders and share positive user experience S29. |
| 2. Flipped classroom approach | Students organise face-to-face sessions, discussions and exercises. Students learn to implement concepts in the real-world scenarios S29. |
| 3. Problem-based learning | Students explore issues according to the learning goals share experiences and thinking of other students applied in real-life scenario S29. In the case of S42, learners are grouped into teams of 4-5 and given an open-ended problem with many exit solutions and they focus on the process of discovery, stimulate problem-solving, independent learning, and teamwork. In S43, it mentions that problem drives learning. Therefore, practical skills are based on different types of problems and problem scenarios where students are provided technology and subject content to choose from a number of solutions. Learners are challenged through visual instruction and exercises. |
| 4. Case-based learning | This strategy allows students to analyse and apply taught concepts and mitigate potential adjustments within a realistic context S29. In the case of S42, learners are grouped into teams of 4-5 and given them a case-study with some advance preparation to focus on the process of discovery, stimulate problem-solving, independent learning, and teamwork. |
| 5. Online learning: Delivered through two fundamental modalities | Helps students to value course contents that have life-long implications which contributes to improving the quality and productivity of software professionals in the academia and industry S29. |
| 6. Intelligent Tutorial System | Architects are trained through courses with various specific concepts and patterns for necessary skills to enter the job market S29. |
| 7. Product-Based Learning (PBL) | Provides hands-on experience through projects resembling real-world applications S30. |
| 8. Apprenticeship learning | Apprentice learns a skills or trade through practical experience and under guidance of an expert S30. The literature used live coding demonstration and providing demo codes through Git repository S30. |
| 9. Patterns-based approach | Provides practical examples through case studies, such as adult literacy programs S31. |
| 10. Abstract modelling | By using the modelling, teamwork, proper patterns and corresponding technologies, students can get hands on experience in system development while appreciating the theory of abstraction S32. |
| 11. Abstraction and System Thinking | Students engaged in constructing UML diagrams (activity, class, and sequence diagrams), providing practical experience in software modelling. |
| 12. Student Ownership of Learning (SOL) | This model provides hands-on practical experiences to students through interactive lab sessions and projects S34. |
| 13. Project-based learning  Mobile Project-based Learning [S39] | Students are provided with hands-on practical experiences through development of mobile robots using commercial sensors and actuators S35, while S36 didn’t specify any particular projects as a part of their study. In the case of mobile project-based learning, students take part in various practical exercises using mobile devices to access resources and to collaborate with other team members S39. |
| 14. Usability-Supporting Architecture Patterns (USAPs) | Participants were given a task to fix a usability problem in a software architecture design meeting the requirements of the usability scenario S37. |
| 15. Early Engagement Strategy | Students were assigned specific-purpose programming exercises with increasing difficulties with the use of a gamified platform. There were different number of exercises from various concepts, allowing students to try practicing repeatedly with automated tests, feedback, public mentoring and rewards and incentives in the form of badges, reputation, progress and percentages S38. |
| 16. Adapted Learning Contract Strategy | Students were offered several weekly tasks with flexibility and adaptation based on student’s preferences and needs S38. |
| 17. Collaborative Decision-making | Students engage in practical tasks using a unique decision-modelling tool to capture and analyse design decisions, providing hands-on experience in software architecture S40. |
| 18. Scrum as a Teaching Strategy | Students were involved in projects such as foundations for software architecture, RESTful APIs and Cloud during their semester which gave them hands-on practical experiences in various aspects of the software architecture field. These processes involved agile methodologies, fostering collaboration and iterative development S41. |
| 19. Software Product Line Architecture | The participants involved in this study were stakeholders such as software engineers, programming teachers, m-learning experts and mobile developers. This is not directly related to students’ hands-on practical experiences S44. |
| 20. Constructive Alignment | Students were strongly encouraged to attend both the lectures and laboratory classes during the 12-week course period. This gave them hands-on practical experiences in programming concepts and process of programming S45. |
| 21. Simulation-based Teaching Strategy | Students are involved in the simulation project with variety of commands to design, construct, and manipulate sensors, actuators, controllers, and communication buses concurrently in a flexible model-view-controller architecture. This gives them a hands-on practical experience S46. |
| 22. VisAr3D: Virtual and augmented reality | Offers students the opportunity to interact with and explore software architectures in a 3D environment, facilitating practical understanding through simulation and visualisation S47. |

**RQ3: What are the various assessment methods used to assess their students?**

|  |  |
| --- | --- |
| Strategy | Assessment Methods |
| 1. Teach traditional classes in software architecture education | The most common validation methods used for software architecture courses were student surveys, metrics to analyse student evaluations, and metrics to assess the quality of artifacts produced S29. |
| 2. Flipped classroom approach |  |
| 3. Problem-based learning | In S29, there isn’t any specific assessment methods mentioned other than student surveys, metrics to analyse student evaluations and assess the quality of the artifacts products. In S42, learners were assessed based on their written deliverables to the workshops, observation on their workshop participation to check whether learners has gained sufficient capabilities to practically apply the concepts and methods in their workplace. In S43, Likert Scale and open-ended questions were used to assess and obtain participant’s preferences or degree or agreement to given statements. |
| 4. Case-based learning | In S42, each group were given some readings and articles on some project scenario and they were required to solve the given questions within the scenario. |
| 5. Online learning: Delivered through two fundamental modalities |  |
| 6. Intelligent Tutorial System |  |
| 7. Product-Based Learning (PBL) [S30] | Graphical-based product-based assignments, human-grading for assignments, automated tools for plagiarism detection, and self-guided supplementary exercises with solutions methods were used to assess the teaching strategy S30. |
| 8. Apprenticeship learning [S30] |  |
| 9. Patterns-based approach | The literature followed a two-pronged approach for evaluation: (i) implementation of a platform for modelling instructional design variations, and (ii) use of the platform for modelling instructional design variations for 8 Indian languages S31. |
| 10. Abstract modelling | Team artifacts are considered for team evaluation in pre-class patterns and in-class patterns.  In high-stake patterns, students are assessed through the final team artifact that they are required to produce.  The literature considered and adopted an additional pattern that provides a generic solution for assessing teamwork called “Teamwork Grade Assignment” |
| 11. Abstraction and System Thinking | Overall assessment of the learners’ software modelling proficiency in terms of capturing the functional, structural, and behavioural aspects of information systems. |
| 12. Student Ownership of Learning (SOL) | This model uses a combination of formative and summative assessments including projects, quizzes and exams S40. |
| 13. Project-based learning  Mobile Project-based Learning [S39] | Outcome of the teaching strategy is assessed through peer-competition and evaluation of the projects students have taken S35 while S36 used oral group-based project based exams to assess the student’s projects.  To assess the effectiveness of Mobile-PBL, students were divided into 10 groups with 3 members each and they were assigned a total of 5 projects each over the course of their semester. Five groups were allowed with use mobile devices for their projects (Mobile-PBL), while other five groups implemented using PBL S39. |
| 14. Usability-Supporting Architecture Patterns (USAPs) | Used Technology Assessment Model (TAM) to assess the perceptions of software architects as to whether the redesigned A-Plus Architect tools would be useful and usable while also measuring USAPs’ and A-PLUS architect's usefulness and usability for curriculum designers and educators S37. |
| 15. Early Engagement Strategy | Students were assessed from the performances based on various categories specified: Continuous assessments (60%), lab assignments (40%), supervised projects (20%), objective test (40%) S38. |
| 16. Adapted Learning Contract Strategy | Same as #15 S38. |
| 17. Collaborative Decision-making | The study assesses students through their participation in role-based decision-making tasks and the quality of their contributions to the decision-modelling tool S40. |
| 18. Scrum as a Teaching Strategy | The learning outcomes of the course was assessed by comparing overall grade of each student for the last years with evaluation of satisfaction of courses at the end of course term with questionnaire. The supervisor also collected oral feedback from students on the course S41. |
| 19. Software Product Line (SPL) Architecture | No assessment methods for students were mentioned other than the SPL architecture model S44. |
| 20. Constructive Alignment | Two forms of assessments were used: formative feedback and summative grading that required a number of conditions to be met. The students were assessed using a portfolio-based approach using formative assessment in which students received feedback on their weekly tasks. A criterion-referenced assessment of portfolio of students determined individual scores or grades S45. |
| 21. Simulation-based Teaching Strategy | The strategy was assessed through objective and subjective measures, anecdotal observations, individual contributions from a background survey, 11 assignments, 10 anonymous weekly assessments, 18 project reports that included reflection, team and self-evaluation, and a course evaluation S46. |
| 22. VisAr3D: Virtual and augmented reality | No assessment methods featured S47. |

**RQ4: What are the outcomes of software architecture teaching strategies?**

|  |  |
| --- | --- |
| Strategy | Outcomes |
| 1. Teach traditional classes in software architecture education | No outcomes specified in S29. |
| 2. Flipped classroom approach |  |
| 3. Problem-based learning | In S42, the study reported improvements in learners’ understanding and comprehension of various software architecture concepts and priciples through use of case studies, analogies, demonstrations and discussions with varying effectiveness between PBL and CBL in #4. However, they recommended to apply this approach to less experienced learners. One of the main outcomes of S43 was an intuitive prototype knowledge based learning environment on virtual learning environment for software components-based design and implementation techniques. It provided excellent learning experience for students to learn advanced topics such as software components and service computing. |
| 4. Case-based learning | As stated above in #3. |
| 5. Online learning: Delivered through two fundamental modalities |  |
| 6. Intelligent Tutorial System |  |
| 7. Product-Based Learning (PBL) | The outcome from the teaching strategy were student spent for time in coding, nearly 6 hours per week, it mitigated issues related to plagiarism and improved learning rates for programming assignments S30. |
| 8. Apprenticeship learning |  |
| 9. Patterns-based approach | Demonstrates improved scalability and adaptability of educational technologies. The literature presented an architecture that integrates Pattern-Oriented Instructional Design that is driven by instructional design methodologies and Pattern-Oriented Software Architecture that drives the design of educational technologies. It also demonstrated the application of approach to model patterns in adult literacy case study in India. The literature also provided an implementation of the approach that generates instructional design authoring tools based on patterns. The literature claimed that it demonstrated a first step that addresses some challenges during design of educational technologies through solutions in software engineering S31. |
| 10. Abstract modelling | The integration of abstraction and modeling aims to equip students with foundational knowledge to navigate enterprise software development effectively S32. |
| 11. Abstraction and System Thinking | Students demonstrated proficiency in modelling functions but showed lower proficiency in modelling structures and behaviours, indicating areas needing instructional improvement S33. |
| 12. Student Ownership of Learning (SOL) | This model demonstrates an improved student engagement and learning outcomes S34. |
| 13. Project-based learning  Mobile Project-based Learning | The teaching strategy demonstrated an improved understanding of embedded systems and critical thinking among students S35, while S36 saw majority of the students preferred group project exams as their assessment methods along with positive student attitudes and improved collaboration skills.  The project completion rate for the group which used mobile devices were higher than those who were not allowed to use mobile during the project task period. Students reported improved practical experience, and enhanced motivation and engagement through the Mobile-PBL approach S39. |
| 14. Usability-Supporting Architecture Patterns (USAPs) | Using a paper-based version of USAPs, the study found out that software engineering graduate students achieved significantly better results in a software architecture design task than they did using usability scenario S37. |
| 15. Early Engagement Strategy | Students achieved a high percentage of the maximum score corresponding to the tasks in the adapted learning contract S38.  The adapted learning contract promotes early engagement, and that the order in which tasks are offered is very relevant.  Beyond alleviating the possible negative effect of group size increase on student performance, either directly or through the impact of group size on attendance and participation, the early implication and formative contract seem to have resulted in a sizeable improvement in student performance S38. |
| 16. Adapted Learning Contract Strategy |
| 17. Collaborative Decision-making | The approach led to improved understanding of decision-making processes, enhanced collaboration skills, and better preparedness for industry roles among students S40. |
| 18. Scrum as a Teaching Strategy | The implementation of scrum as teaching strategies showed that academic performance of students were better than the previous years which indicated it had better learning outcomes S41. |
| 19. Software Product Line (SPL) Architecture | The study shared only the outcome of their findings: The choices and approaches adopted can support the project and design decisions for the conception of SPLs in other domains, and the architecture model was considered feasible S44. |
| 20. Constructive Alignment | The study found out that students prefer formative feedback (without marks) in the beginning to help develop their understanding of unit concepts and encourage them to achieve learning outcomes. Students can improve their understanding from feedback. The study recommends summative feedback (grading) to be applied in final assessment S45. |
| 21. Simulation-based Teaching Strategy | Based on the framework of modelling, simulation, visualization, and analysis, this approach provides a disciplined strategy for creating, executing, presenting, and analysing meaningful test cases as a part of formal test plans which are based on sound methodology of controlled experiments using scientific method. The study also concludes that extensive outcomes from classroom deployments proved students benefit from this approach S46. |
| 22. VisAr3D: Virtual and augmented reality | Aims to improve comprehension of complex software architectures and enhance student engagement through immersive learning experiences S47.  The system is not fully implemented and only the prototype is being tested. |

**RQ5: What challenges do students and educators face with the software architecture teaching approach?**

|  |  |
| --- | --- |
| Strategy | Challenges |
| 1. Teach traditional classes in software architecture education | The following were a list of challenges reported by the literature:   * Lack of real-world experience among instructors. * Abstract and ambiguous nature of software architecture. * Difficulty involving clients and industry experts * Teaching software architecture requires going beyond traditional teaching methods. * Small and simple projects are insufficient for providing students with practical experience. * Lack of involvement in open-source projects. * Personalized monitoring, modern methods developed in the industry may take time to reach academia * A need for courses to adapt to changes * Significant gap between students’ skills and the expectations of industry managers or hiring personnel. * Completed project scenarios are helpful in training software architecture, but they can also intimidate students S29. |
| 2. Flipped classroom approach |  |
| 3. Problem-based learning | In S42, the identified challenges for both PBL and CBL were to impart practical skills in a way that adult learners can apply immediately, to design a relevant course to the learner’s environment and to make abstract concepts easy to comprehend. In S43, one of the challenges mentioned was Component Based Software Engineering (CBSE) has been taught in similar ways to other programming languages with smaller abstraction knowns as functions and procedures. There wasn’t technique to teach high-level CBSE modelling and concepts. |
| 4. Case-based learning | As described in #3 above. |
| 5. Online learning: Delivered through two fundamental modalities |  |
| 6. Intelligent Tutorial System |  |
| 7. Product-Based Learning (PBL) | The challenges from this strategy included discrepancy between laboratory and midterm grades, and high plagiarism rates S30. |
| 8. Apprenticeship learning | As featured in #7 S30 |
| 9. Patterns-based approach | The literature reported challenges in integrating diverse instructional designs and software patterns S31. |
| 10. Abstract modelling | Some traditional patterns of software architecture have broader context and lack details for implementation, difficult to thoroughly understand the big picture of the system which makes challenging for the students to implement the systems S32. |
| 11. Abstraction and System Thinking | Students faced difficulties in abstract thinking and systems thinking, particularly in modelling structural and behavioural aspects of software systems. |
| 12. Student Ownership of Learning (SOL) | There were challenges such as varying levels of student preparedness and adapting teaching methods to diverse learning styles S34. |
| 13. Project-based learning  Mobile Project-based Learning | There were challenges such as difficulties in conveying abstract knowledge on embedded software in tangible form and bridging the gap between limited students’ background and curriculum requirements S35, while S36 highlighted differences in attitudes between engineering programs and the impact of the changes in exam format as their main challenges.  In mobile project-based learning, some challenges faced were potential distractions from social media, a need for both educators and students to develop necessary technological skills, a large amount of work necessary, and lack of defined norms and standards S39. |
| 14. Usability-Supporting Architecture Patterns (USAPs) | The study reported some persistent challenge of integrating usability effectively into software architecture design, to evaluate impact of the method on student outcomes, and to achieve collaboration between appropriate education experts to develop easier educational patterns which is understandable for users in various educational domains S37. |
| 15. Early Engagement Strategy | These study highlighted managing large class sizes and ensuring personalized attention as challenges during the implementation of these strategies S38. |
| 16. Adapted Learning Contract Strategy |
| 17. Collaborative Decision-making | Challenges included managing the complexity of role-based interactions and ensuring effective use of the decision-modelling tool S40. |
| 18. Scrum as a Teaching Strategy | Some of the challenges mentioned in the study were stimulating and nurturing learners’ interests in the topics of the course, ineffective group processes, short timeframe for classes or attention and time with other courses S41. |
| 19. Software Product Line (SPL) Architecture | The challenges related with the study were: Adaptation of interfaces, functionalities and strategies for effective adoption of mobile devices in the teaching of programming S44. |
| 20. Constructive Alignment | The workload for both staff and student increased. Students found hard to keep up with weekly tasks while teaching staff had huge weekly workload of providing feedback on students’ progress S45. |
| 21. Simulation-based Teaching Strategy | Some of the challenges identified in the study were the complexity of the simulation system, the learning curve associated with understanding the quasi-network-based architecture, and the need for adequate resources to support the simulation environment S46. |
| 22. VisAr3D: Virtual and augmented reality | Potential challenges include the need for access to virtual and augmented reality technologies, the learning curve associated with these tools, and the integration of such technologies into existing curricula S47. |