Title:	Software Architecture & Design APPROVED
Long Title:	Software Architecture & Design

Module Code:	COMP9054
Credits:	5
NFQ Level:	Expert
Field of Study:	Computer Science
Valid From:	Semester 1 - 2017/18 (September 2017)
Module Delivered in	1 programme(s)
Module Coordinator:	Donna OShea
Module Author:	Donna OShea
Module Description:	Software architects are typically responsible for the definition of software “systems by applying abstract knowledge and proven methods to a set of technologies with the goal of creating an extensible and maintainable solution” [Hofstader, 2009]. In this module, students will learn about the principles of software architecture and design and how and when to apply software design and architecture patterns to solve common problems when designing and developing software with the goal of creating an extensible and maintainable software solution.

Learning Outcomes	
On successful completion of this module the learner will be able to:	
LO1	Discuss the evolving role of a software architect in the digital age.
LO2	Appraise the different approaches to software design and patterns that can be applied to solve a problem in a variety of contexts.
LO3	Examine how architecture design and patterns capture structural and behavioural information of a system.
LO4	Design a software architecture solution from a presented case study.
LO5	Evaluate how Architecture as a Service (AaaS), serverless architecture systems and other emerging trends are impacting the field of software design and architecture.

Pre-requisite learning
Module Recommendations *This is prior learning (or a practical skill) that is strongly recommended before enrolment in this module. You may enrol in this module if you have not acquired the recommended learning but you will have considerable difficulty in passing (i.e. achieving the learning outcomes of) the module. While the prior learning is expressed as named CIT module(s) it also allows for learning (in another module or modules) which is equivalent to the learning specified in the named module(s).*
No recommendations listed
Incompatible Modules *These are modules which have learning outcomes that are too similar to the learning outcomes of this module. You may not earn additional credit for the same learning and therefore you may not enrol in this module if you have successfully completed any modules in the incompatible list.*
No incompatible modules listed
Co-requisite Modules
No Co-requisite modules listed
Requirements *This is prior learning (or a practical skill) that is mandatory before enrolment in this module is allowed. You may not enrol on this module if you have not acquired the learning specified in this section.*
No requirements listed

Co-requisites
No Co Requisites listed

Module Content & Assessment

Indicative Content
Software Architect The role of the software architect. Evolving role and responsibilities. Architecture without architects. Dealing with customers, developers, testers, managers.
Software Design Evolutionary versus planned design – opportunities and challenges. Software design process. Spectrum of design approaches – waterfall, planned design or design up front, emergent design, agile, cowboy hacking. Software entropy, complexity and technical debt. Essential versus accidental complexity. Accidental complexity i.e. Code Duplication, irreversibility. Rampant genericness. Design principles – SOLID, Separation of Concerns (SoC), Don't Repeat Yourself (DRY), Shy, Last responsible moment (LRM), Big Design Up Front (BDUF) versus Just Enough Design Initially (JEDI), Domain Driven Design (DDD). Emergent Design.
Software Architecture Definition. Business implications of architecture solution - Licencing, deployment and billing. Defining architecture. Different types of architecture – application architecture, enterprise architecture. Evolutionary Architecture. Architecture design, design process and the pattern decision relationship. Architecture principles. Internet Architecture and its impact.
Software Design Patterns Purpose. Creational, Structural, Behavioural, Concurrency, J2EE. Dependency Injection.
Software Architecture Patterns The purpose of architecture patterns. Using patterns practical considerations. Benefits of pattern use. Limitation of patterns. Different types of architecture patterns - Layered architecture, Event Driven Architecture, Microkernel, Microservices, Blackboard, NanoServices, Space based architecture, Model View Controller. Pattern analysis – agility, deployment, testability, performance, scalability, ease of deployment. Considerations when choosing an architectural pattern.
Architecture as a Service Serverless architectures. What is serverless? Serverless benefits and drawbacks. Future of Serverless. Examples of serverless architecture systems. Reference architecture – mobile backend, real time processing, web applications, IoT backend, real time stream processing. Functions as a service (FaaS) and Platform as a Service (PaaS) – comparison and opportunities. FaaS – state, execution duration, startup latency, gateway. FaaS providers – IBM OpenWhisk, AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, Iron.io and Webtask. Frameworks – Serverless, Nanoservices.
Future trends Future trends in software design and architecture. Team management and practices. Containers. Code generation. Security.

Assessment Breakdown	%
Course Work	100.00%

Course Work

Assessment Type	*Assessment Description*	*Outcome addressed*	*% of total*	*Assessment Date*
Project	In this assessment the student may be expected to evaluate software code. As part of their evaluation they will be expected to apply best practices in software design and patterns in the code.	1,2	30.0	Week 7
Project	From a presented case study the student may be expected to design an architectural solution for a give system. Their role as architect will be critically evaluated as part of this project. In addition, the student will be expected to evaluate how emerging technological trends has and will have an impact on the design of the solution.	1,3,4,5	70.0	Sem End

No End of Module Formal Examination

Reassessment Requirement
Coursework Only *This module is reassessed solely on the basis of re-submitted coursework. There is no repeat written examination.*

The institute reserves the right to alter the nature and timings of assessment

Module Workload

Workload: Full Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture delivering theory underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Lab supporting learning outcomes.	1.0	Every Week	1.00
Independent & Directed Learning (Non-contact)	Independent & directed learning	4.0	Every Week	4.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	3.00

Workload: Part Time				
Workload Type	*Workload Description*	*Hours*	*Frequency*	*Average Weekly Learner Workload*
Lecture	Lecture delivering theory underpinning learning outcomes.	2.0	Every Week	2.00
Lab	Lab supporting learning outcomes.	1.0	Every Week	1.00
Independent & Directed Learning (Non-contact)	Independent & directed learning	4.0	Every Week	4.00
			Total Hours	7.00
			Total Weekly Learner Workload	7.00
			Total Weekly Contact Hours	3.00

Module Resources

Recommended Book Resources

- Mark Richards 2015, ***Software Architecture Patterns***, O' Reilly [ISBN: 9781491924242]
- R. N. Taylor, N. Medvidovic and E. M. Dashofy 2009, ***Software Architecture: Foundations, Theory, and Practice***, Wiley [ISBN: 9780470167748]
- Len Bass 2012, ***Software Architecture in Practice***, 3rd Ed., Addison-Wesley Professional [ISBN: 9780321815736]
- Gregor Hohpe 2016, ***37 Things One Architect Knows About IT Transformation: A Chief Architect's Journey***, CreateSpace Independent Publishing Platform [ISBN: 9781537082981]

Supplementary Book Resources

- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides 1994, ***Design Patterns: Elements of Reusable Object-Oriented Software***, 1st Ed., Addison-Wesley Professional [ISBN: 9780201633610]

Recommended Article/Paper Resources

- Sherman, S. and Hadar, I. 2015, ***Toward defining the role of the software architect: An examination of the soft aspects of this role***, In Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering
- G. Hohpe, I. Ozkaya, U. Zhun and O. Zimmermann 2016, ***The Software Architect's Role in the Digital Age***, IEEE Software, 33
- N. B. Harrison, P. Avgeriou and U. Zdun ***Using Patterns to Capture Architectural Decisions***, IEEE Software, 24(4)
- P. Kruchten, H. Obbink and J. Stafford 2006, ***The Past, Present, and Future for Software Architecture***, IEEE Software, 23(2)
- Shaw, M. and Clements, P. 2006, ***The golden age of software architecture.***, IEEE Software, 23(2)
- G. Booch 2011, ***Draw Me a Picture***, IEEE Software,, 28(1)
- Hofstader, J. 2009, ***We Don't Need No Architects!***, The Architecture Journal, 15 http://www.iasa.se/wp-content/uploads/2009/08/TAJ15.pdf

Other Resources

- Website: ***Serverless*** https://serverless.com/
- Website: ***Serverless Google Cloud Platform*** https://cloud.google.com/functions/
- Website: ***Serverless Reference Architectures with AWS Lambda*** http://www.allthingsdistributed.com/2016 /06/aws-lambda-serverless-reference-arch itectures.html
- Website: ***Software Architecture Conference*** https://conferences.oreilly.com/software -architecture/sa-ny
- Website: ***Serverless Conference*** http://serverlessconf.io/
- Website: ***The Open Group Architecture Framework*** http://www.opengroup.org/subjectareas/en terprise/togaf
- Website: ***Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*** https://www.iso.org/standard/35733.html
- Website: ***arc42*** http://arc42.org/
- Website: ***Software Engineering Institute*** http://www.sei.cmu.edu/

Module Delivered in

Programme Code	Programme	Semester	Delivery
CR_KSADE_9	Master of Science in Software Architecture & Design	1	Mandatory