



**PROGRAMMING 3A
MODULE MANUAL 2024
(First Edition: 2018)**

This manual enjoys copyright under the Berne Convention. In terms of the Copyright Act, no 98 of 1978, no part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any other information storage and retrieval system without permission in writing from the proprietor.



The Independent Institute of Education (Pty) Ltd is registered with the Department of Higher Education and Training as a private higher education institution under the Higher Education Act, 1997 (reg. no. 2007/HE07/002). Company registration number: 1987/004754/07.

DID YOU KNOW?

Student Portal

The full-service Student Portal provides you with access to your academic administrative information, including:

- an online calendar,
- timetable,
- academic results,
- module content,
- financial account, and so much more!

Module Guides or Module Manuals

When you log into the Student Portal, the 'Module Information' page displays the 'Module Purpose' and 'Textbook Information' including the online 'Module Guides or 'Module Manuals' and assignments for each module for which you are registered.

Supplementary Materials

For certain modules, electronic supplementary material is available to you via the 'Supplementary Module Material' link.

Module Discussion Forum

The 'Module Discussion Forum' may be used by your lecturer to discuss any topics with you related to any supplementary materials and activities such as ICE, etc.

To view, print and annotate these related PDF documents, download Adobe Reader at following link below:

www.adobe.com/products/reader.html

IIE Library Online Databases

The following Library Online Databases are available. These links will prompt you for a username and password. Use the same username and password as for student portal. Please contact your librarian if you are unable to access any of these. Here are links to some of the databases:

Library Website	This library website gives access to various online resources and study support guides [Link]
LibraryConnect (OPAC)	The Online Public Access Catalogue. Here you will be able to search for books that are available in all the IIE campus libraries. [Link]
EBSCOhost	This database contains full text online articles. [Link]
EBSCO eBook Collection	This database contains full text online eBooks. [Link]
SABINET	This database will provide you with books available in other libraries across South Africa. [Link]
DOAJ	DOAJ is an online directory that indexes and provides access to high quality, open access, peer-reviewed journals. [Link]
DOAB	Directory of open access books. [Link]
IIESPACE	The IIE open access research repository [Link]
Emerald	Emerald Insight [Link]
HeinOnline	Law database [Link]
JutaStat	Law database [Link]

Table of Contents

Introduction	6
Glossary	9
Learning Unit 1: Characteristics of Enterprise Software Systems.....	14
1 Introduction	14
2 Introduction to Enterprise Software Systems	15
3 Non-Functional Requirements	20
4 Challenges in Enterprise Software Systems	28
5 Concluding Remarks.....	31
6 Recommended Additional Reading	32
7 Revision Exercises.....	33
8 Solutions to Revision Exercises.....	34
Learning Unit 2: Design and Architecture Patterns	35
1 Introduction	36
2 Design Patterns	36
3 Architecture Patterns	44
4 Anti-Patterns	49
5 Recommended Additional Reading	53
6 Revision Exercises.....	54
7 Solutions to Revision Exercises.....	55
Learning Unit 3: Enterprise Software System Development	56
1 Introduction	56
2 Microsoft Enterprise Application Development Platform	57
3 Transaction Management	63
4 Messaging.....	75
5 Directory Services	81
6 Security.....	85
7 Services, Orchestration and Choreography.....	89
8 Portals.....	92
9 Application Hosting	93
10 Recommended Additional Reading	96
11 Revision Exercises.....	97
12 Solutions to Revision Exercises.....	98
Learning Unit 4: Optimising Application Performance.....	99
1 Introduction	100
2 C# Topics.....	101
3 Databases and Entity Framework.....	105
4 Service Design.....	111
5 Recommended Additional Reading	118
6 Revision Exercises.....	119
7 Solutions to Revision Exercises.....	120
Learning Unit 5: Methodologies and Architecture Frameworks	121
1 Introduction	121
2 Software Development Methodologies	122
3 DevOps	132
4 Enterprise Architecture	135

5	Recommended Additional Reading	140
6	Revision Exercises.....	141
7	Solutions to Revision Exercises.....	142
	Concluding Remarks.....	143
	Bibliography	144
	Intellectual Property	164

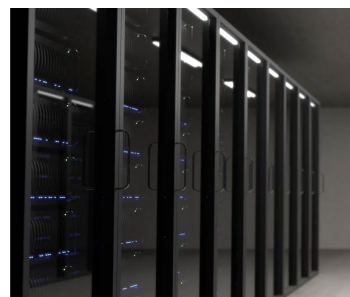
Introduction

Software is everywhere around us today. Maybe the most obvious example of using software is when we sit down in front of a computer to write a document or design a presentation. And in days past, desktop applications like these were most of the software applications in existence.

Today, technology has progressed far beyond that point. When we think of software, it ranges from embedded software that controls appliances and toys, to sophisticated virtual worlds created by game developers, to highly distributed software systems used by large enterprises. And the hardware that is needed to run this software varies just as much in scale and complexity.



Raspberry Pi



Data centre

Figure 1: Hardware can range from a Raspberry Pi to a large data centre

C# is a fantastically versatile language that allows us to write software for almost any purpose. Here are just a few of those:

- Writing software for Internet of Things devices on the Raspberry Pi (Sonnino, 2017);
- Developing games in Unity (Tuliper, 2014);
- Creating applications that can run on a variety of Windows platforms, including the Xbox and HoloLens (Wigley & Nixon, 2015);
- Designing mobile applications that run on several phone operating systems (Burns, et al., 2018); and
- Deploying ultra-scalable web applications to the Azure cloud (Lin, et al., 2018).



Have a look at each of the references for a guide to getting started with each one!

It is great news that we can use the language that we know already in such a vast number of different ways. However, each kind of software application also has its own unique challenges and associated technologies that we need to learn about.

In this module, we will focus on what makes developing an enterprise software system different from other software development. We will look at design patterns and architecture patterns. We will learn about the technologies that are frequently used in the .NET ecosystem. We will investigate how to optimise application performance. And we will look at processes that are followed to develop such systems.

This Module at a Glance

Figure 3 shows an overview of everything that will be covered in this module.

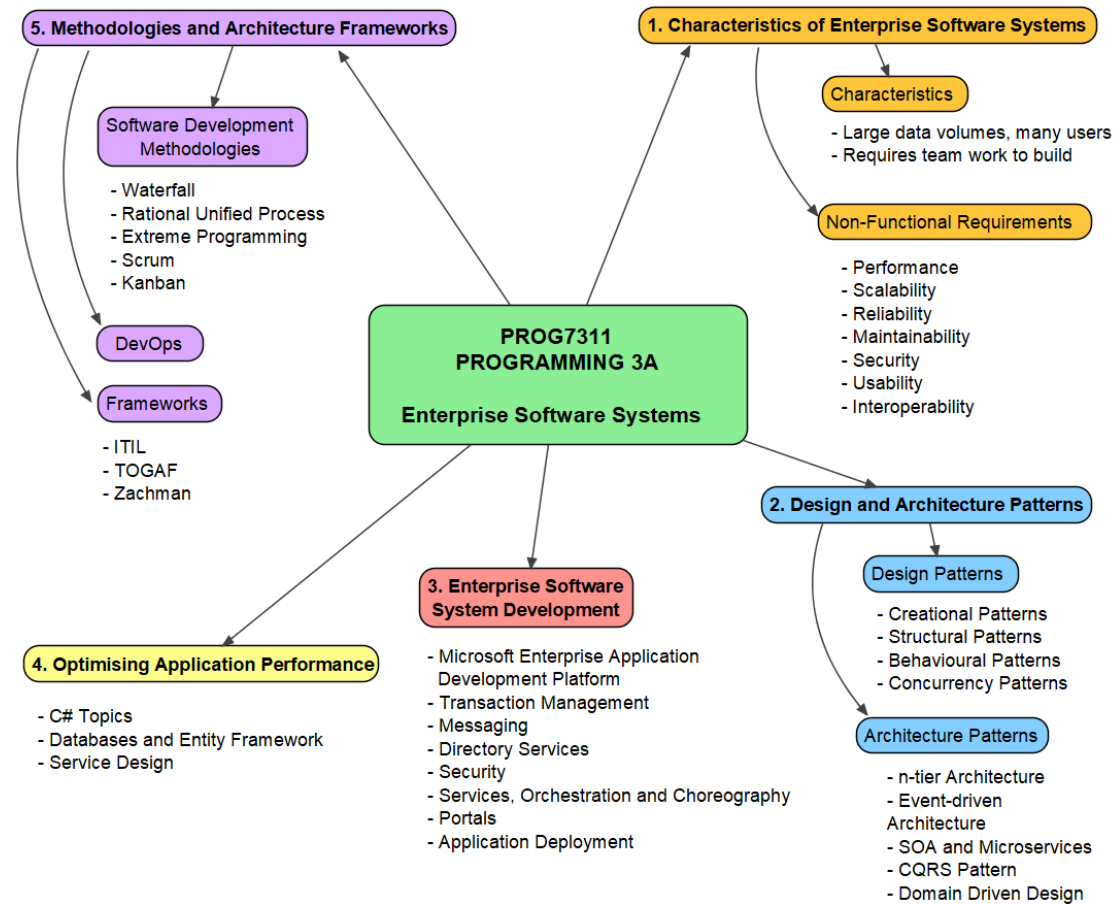





Figure 2: Overview of the Module

A Word from the Author

With just more than 12 years of experience in developing software, I have been everything from a Junior Software Developer to a Senior Software Architect. I have seen successes and failures. And I have learned a lot from them. But I do not know everything yet – no one person ever can. As my husband always says: “Put three software architects in a room, and you will have at least six different opinions on any topic”.

So, while this manual will provide you with a good foundation for developing enterprise software systems, there is always more to learn. .NET software development is one of the fields where there is a lot of information freely available online. I encourage you to make use of the internet to find more information about the topics presented here!

Conventions Used in this Manual

	Where this icon appears, there is a reference one or more freely available online resources such as e-books.
	Where you see this icon, there will be information about how the software development aspects of enterprise systems link up with other disciplines.
	Where you see this icon, there will be an opportunity to think about a real-world example that links to the theory.

Glossary

Term	Definition
Anti-Pattern	An anti-pattern is a common way of doing things that will not work correctly.
Architecture Pattern	Architecture patterns are solutions to common problems at an application wide level.
Authentication	“Authentication is the process that confirms a user’s identity”. (L, 2018)
Authorisation	“Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features.” (Technopedia, 2018c)
Business Process	“A business process is a collection of linked tasks which find their end in the delivery of a service or product to a client.” (Appian, 2018)
Caching	“Caching is a common technique that aims to improve the performance and scalability of a system. It does this by temporarily copying frequently accessed data to fast storage that's located close to the application. If this fast data storage is located closer to the application than the original source, then caching can significantly improve response times for client applications by serving data more quickly.” (Narumoto, et al., 2017b)
Command Query Responsibility Segregation (CQRS)	In the Command Query Responsibility Segregation (CQRS) pattern, two different models are used: one for reading data and a different one for writing of data. (Fowler, 2011)
Computer Security	“Computer Security is the protection of computing systems and the data that they store or access.” (University of California Santa Cruz, 2015)
Continuous Delivery	“Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.” (Humble, 2017)

Term	Definition
Continuous Integration (CI)	“Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.” (ThoughtWorks, 2018)
Design Pattern	“In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.” (SourceMaking.com, 2018a)
DevOps	“DevOps is the blending of tasks performed by a company's application development and systems operations teams.” (TechTarget, 2017)
Directory Services	“Directory services are software systems that store, organize and provide access to directory information in order to unify network resources. Directory services map the network names of network resources to network addresses and define a naming structure for networks.” (Technopedia, 2018b)
Domain Driven Design (DDD)	“Capture the domain model in domain terms, embed the model in the code, and protect it from corruption.” (Lowe, 2018)
Emergent Behaviour	“Emergent behavior is behavior of a system that does not depend on its individual parts, but on their relationships to one another. Thus, emergent behavior cannot be predicted by examination of a system's individual parts. It can only be predicted, managed, or controlled by understanding the parts and their relationships.” (thwink.org, 2014)

Term	Definition
Enterprise Architecture (EA)	“Enterprise architecture (EA) is a discipline for proactively and holistically leading enterprise responses to disruptive forces by identifying and analyzing the execution of change toward desired business vision and outcomes. EA delivers value by presenting business and IT leaders with signature-ready recommendations for adjusting policies and projects to achieve target business outcomes that capitalize on relevant business disruptions.” (Gartner, 2018)
Enterprise Information Portal (EIP)	“An Enterprise Information Portal (EIP) is a class of applications that enables organizations to unlock internally and externally stored information and provide users a single gateway to the personalized information needed to make informed business decisions.” (Information Builders, 2018)
Enterprise Software System	“A large software system platform designed to operate in a corporate environment such as business or government.” (Techopedia, 2018)
Functional Requirement	A statement of what the software must do. (Eriksson, 2015)
Interoperability	“The definition of interoperability is the ability of a system, software or product to exchange and make use of information with other systems, software or products without special effort on the part of the user. For two systems to be interoperable, they must be able to exchange data and subsequently present that data such that it can be understood by a user.” (International TechneGroup Incorporated, 2018)
Maintainability	“The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.” (Crouch, 2018)
Microservices Architecture (MSA)	“[T]he microservice architectural style [1] is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.” (Lewis & Fowler, 2014)

Term	Definition
.NET Core	“.NET Core is a cross-platform version of .NET for building websites, services, and console apps.” (Microsoft Corporation, 2018)
.NET Framework	“.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.” (Microsoft Corporation, 2018)
Non-Functional Requirement (NFR)	A statement of how the software should perform. (Eriksson, 2015) Also called a Quality Attribute.
Performance	How fast a software application must be able to perform a specific task given a certain system load.
Portal Software	“A portal is a web-based platform that collects information from different sources into a single user interface and presents users with the most relevant information for their context.” (Liferay, 2021)
Reliability	“Ability of a computer program to perform its intended functions and operations in a system's environment, without experiencing failure (system crash).” (Pan, 1999)
Scalability	“In software engineering, scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be enlarged.” (Gerard, 2017)
Security Development Lifecycle (SDL)	Microsoft's process for developing more secure software systems.
Service Oriented Architecture	“Service Oriented Architecture is a software architecture where distinct components of the application provide services to other components via a communications protocol over a network.” (Despodovski, 2017)
Software Development Methodology	A software development methodology is “a framework that is used to structure, plan, and control the process of developing an information system.” (Vskills, 2018)

Term	Definition
Software Security	“The process of designing, building and testing software for security—identifies and expunges problems in the software itself. In this way, software security practitioners attempt to build software that can withstand attack proactively.” (McGraw, 2004)
Transaction	“A transaction is a single unit of work which means either ALL or NONE. If a transaction is successful, all of the data operations are committed and become a durable part of the database. If a transaction encounters errors/exceptions and must be cancelled or rolled back, then all of the data modifications/operations need to be removed.” (Mohapatra, 2016)
Usability	“Usability is a measure of how easy it is to use a product to perform prescribed tasks.” (Microsoft Corporation, 2000)

Learning Unit 1: Characteristics of Enterprise Software Systems

Learning Objectives:

- Explain the scope of enterprise software systems.
- Differentiate between functional and non-functional requirements.
- Explain how non-functional requirements impact software design.
- Discuss what emergent behaviour is.
- Discuss the challenges brought about by organisational growth in Enterprise Software Systems.
- Motivate the need for change management in large enterprises.

Material used for this learning unit:

- Internet access.

How to prepare for this learning unit:

- Before you attend the lectures, read all sections of the learning unit in this module manual.

My notes

1 Introduction

When studying enterprise software system development, the first question that may come to mind is “Why should I care?”. In this first learning unit, we will attempt to answer this question by studying what an enterprise software system is, and what some of the challenges are that need to be met to develop good enterprise software systems.

Take a moment to answer this question for yourself – there are no right or wrong answers here yet, it is about your perception in this moment:

“What is an enterprise software system?”

Got your answer? Great! Let us look at some possibilities that came to mind.

- “It is a big *huge scary system* that I have no idea where to even start with!” Well, you have come to the right place! At the end of this module, the unknowns will be far less, and you will have a good foundation for getting started in

the enterprise development world. Also remember that when you start out a career in enterprise software development, there will always be a team to support you!

- “It is a *boring business system* where you do the same thing day in and day out.” To some extent that may even be the mark of a good enterprise software system. Trust me, it is better to have a good old boring day than say one where you know that your adventurous coding caused a bank’s internet banking system to crash! However, enterprise software systems are not without fun problems to solve. The problems are just different from what you would face if you wrote software for the space shuttle. So, you will see that it is not as boring as you might think!

Regardless of your answer, write it down somewhere. At the end of this module, you will have the opportunity to reflect on your answer.

2 Introduction to Enterprise Software Systems

2.1 *The Nature of Enterprise Software Systems*

An enterprise software system is “a large software system platform designed to operate in a corporate environment such as business or government” (Techopedia, 2018). Other terms used for the same concept include Enterprise Application (EA) (Techopedia, 2018).

Stephen Watts adds extra information in his definition:

“In general, enterprise application software is large-scale software that is aimed to support or solve the problems of an entire organization. This large-scale software allows for several different user roles, and the roles define the actions a specific user can perform.” (Watts, 2017a)

This definition highlights some interesting points. An enterprise software system should meet a need in an enterprise – no enterprise would fund development if it did not. And to do that it will need to have properties that are like that of the enterprise itself: an enterprise is typically a large organisation with diverse people working together towards a common goal.

Martin Fowler, one of the most well-known people in enterprise software architecture circles, defines some common properties of enterprise software systems (Fowler, 2014):

- *Large volumes of data* are persisted by such applications.
- Many *concurrent* users need to be supported.
- *Many user interface screens* are required and often data needs to be displayed in different ways to users with different roles.
- Different systems need to be *integrated* with each other to exchange various kinds of data.
- *Complex business logic* is often required.

These properties already point out some technical challenges. How do I ensure that my web application can handle it when a hundred thousand users decide to access it all at once? Where do I store all the data for the large number of transactions that happen every day? How do I ensure that the business intelligence department get their statistics without causing interruption to other users?

Vangie Beal lists, amongst others, these common types of enterprise applications (Beal, 2010):

- Customer Relationship Management (CRM).
- Business Intelligence (BI).
- Enterprise Resource Planning (ERP).
- Automated billing system.
- Payment processing.

From this list we can see again that enterprise software systems are used to support business functions. Need to bill subscribers every month? There is a system for that. Need to find out where the largest part of the subscriber base lives? Yep, there is a system for that too.

We have now explored what the “enterprise” part of enterprise software systems is all about. But why call them “systems”? Let’s look at the definition of a system according to the Oxford dictionary:

“A set of things working together as parts of a mechanism or an interconnecting network; a complex whole.” (Oxford Dictionary, 2018)

As we have already seen, enterprise software systems usually need to be integrated with other software systems. Thus, the different systems are parts of the bigger information systems infrastructure of an enterprise.

2.2 Teamwork

When implementing large projects, it is essential that multiple people work together to make it possible. Imagine for a moment that an architect (the kind that designs buildings, not a software architect) comes up with an idea for a huge new skyscraper. The first step is for the architect to draw up the plans. Once that is done, how long do you think it would take the architect to build the whole building herself? We mean all by herself – excavating a huge hole, laying the foundations, putting in place the pillars that hold up the first floor and countless other tasks. It would likely take more than a lifetime!

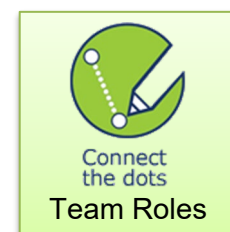
The same applies to large software projects. Although enterprise software systems are not by any means the only large projects out there, these systems almost always require at least a team of people to make it a reality. In fact, in large projects multiple teams of people may be working on the same project.

Fun Fact

According to Eric Lai, there were “25 ‘feature teams’ of about 100 employees each working on the upcoming replacement to Windows Vista.” (Lai, 2008)

2.2.1 Different Roles in a Software Development Team

Being software developers ourselves, it is tempting to think that the only role needed in a team is, well, the software developer. And while the software developers are the people that write the actual code, and without them there certainly is no team, there are multiple other roles in a typical software team that are also important.



There are two reasons why it is essential to have different roles in a project:

- *Specialisation.* There are so many different kinds of tasks to complete that it is impossible for one person to master them all. Do you think the architect in our example would be able to plaster walls as well as somebody that has been doing that all their lives? Unlikely. Or would a software developer be just as good at regression testing as an experienced tester? In my experience, just as unlikely. It is not about ability, but about experience and personal preference.
- *Level of detail.* In an enterprise software development project, it is important for at least one person to have an overview of how all the different parts of the system fit together. To guide all the development effort towards a common, consistent software architecture. But such systems are far too large for a single person to also be actively involved in all the technical details of every single part of the system.

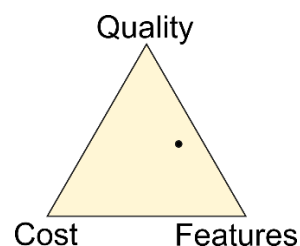


Figure 3: Project Forces

- *Balancing of project forces.* Figure 3 shows a representation of the different forces in a project. There are multiple similar models, but the basic idea behind all of them are the same. If you focus on one of the areas, you must compromise on the other two. Want to save money (or time)? Sure, but then reduce the scope of the project and/or be willing to accept a lower quality output. Want excellent quality? Can do, but it will cost more. The dot in the figure represents one of the many possible places where the project could be between the three points.

It is hard for one person to not prefer one of the forces over the other. For example, software architects typically really care about quality and would choose that over cost any day.

And so ideally there should at least be three people on a project that represent the different forces, so that they can balance out each other's preferences.

Let us look briefly at some of the typical roles that one would expect to find in a software team. The exact makeup of every team depends on many factors, including which software development methodology the team uses. Typically, a team has the following roles (by no means an exhaustive list):

- *Software developer* – Responsible for writing the code, sometimes in a very specific area for example back-end developer;
- *Software tester* – Responsible for checking that the software works correctly. Can be either a manual tester that manually performs tests on a system, or automation tester that writes test scripts that tests the system;
- *Business analyst* – Responsible for analysing the requirements of the enterprise;
- *Software architect* – Responsible for the high-level architectural design of the system; and
- *Project manager* – Responsible for ensuring that the project is within budget and on time.

We will discuss the roles in more detail in Learning Unit 5, as they apply to the different software development methodologies.

2.3 Functional Requirements

All software applications have requirements. It might be a simple one liner hand-written on a napkin, or it might be a document with tens or even hundreds of pages. It might all be specified up front, or the requirements may get clarified as the project progresses. But fundamentally, software developers need to know what the application is supposed to do to build it.

When we talk about functional requirements, it is all about *what* the system is supposed to *do*. Here are some examples:

- The user shall be able to log into the system with a username and password.
- The system shall be able to request client details from the customer relationship management system.
- The customer shall be able to view the progress of his/her order throughout the order process.

In Learning Unit 5 we will look at how requirements are managed in different software development methodologies.

2.3.1 Good Requirements

Writing good requirements can be challenging. And the details of how to determine what the client really needs is beyond the scope of this module (read online about *Business Analysis* if you are interested in knowing more). Nevertheless, it is useful to know the properties of a good requirement. Ivy Hooks listed four properties of a good requirement (Hooks, 1993):

- Necessary
- Verifiable
- Attainable
- Clear

Consider this example of a requirement – how does it measure up to Hook’s properties?

“The app will be user friendly.”

It may be deemed necessary, but how would you be able to measure this? “User friendly” is very subjective and depends heavily on the user’s skills and previous experiences.

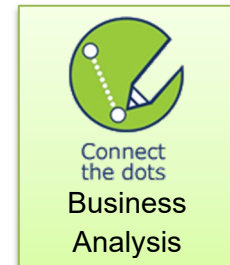
2.3.2 Priorities

Usually not all requirements have the same priority. For example, it might be more important from a business perspective that a user shall be able to place an order than for the user to be able to track the progress of that order. It does not mean that the software application will not eventually be able to do both. But the priority will indicate which functions should be implemented first and tested most thoroughly.

3 Non-Functional Requirements

According to Ulf Eriksson, “Non-functional requirements describe how the system works” (Eriksson, 2015). These requirements are also referred to as *quality attributes* (Dalbey, 1998).

Non-functional requirements (NFRs) include items such as performance, scalability, reliability, maintainability, security, usability, and interoperability.



Although this is by no means an exhaustive list of all the non-functional requirements, it serves to illustrate what a non-functional requirement is all about. A software system could comply with every functional requirement that was specified and still not be fit for use if the non-functional requirements are not met.

Consider an e-commerce website that allows a user to place an order. The website could allow a user to log in with a username and password and that would meet the functional requirement. But if we neglect to think about *security*, that user's password and even worse, credit card details, could be intercepted by a third party for malicious purposes. The website could allow a single user to search for a product to buy – even the most inefficient code could meet this functional requirement. But the enterprise would certainly not be happy with the *scalability* of the website if the highest number of users it could successfully handle is only 10.



In a perfect world, it would be great if we could meet every single non-functional requirement ever defined. But there is typically a limited amount of funding available to develop software. And thus, it will be necessary to prioritise non-functional requirements according to the needs of a specific project.

Let us look at some of the most common non-functional requirements. Each of these is a whole field of study in its own right, but here we will stick to just a brief discussion on each one.

3.1 Performance

Ulf Eriksson explains that performance can be measured in terms of system response times under specific circumstances (Eriksson, 2015). He also adds that it is worthwhile thinking about peak periods when the system will experience high load.

So how important is performance?

Fun Fact

According to Neil Patel, “40% of people abandon a website that takes more than 3 seconds to load.” (Patel, 2018)

When it comes to peak periods, Black Friday in the e-commerce world is an excellent example. Have you ever tried to buy something on Black Friday and felt the frustration of the website being ridiculously slow? That could have been avoided if peak time performance was planned for.

Of course, Black Friday is an extreme example. But most systems will have a peak usage period. For example, a billing system may need to bill all subscribers on the same day of the month. In this case, the end user will not really notice the difference in performance if he gets billed a few hours later than usual. But the enterprise will be unhappy if it takes a week to bill everybody.

3.2 Scalability

Nicolas Gerard describes scalability as follows:

“In software engineering, scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be enlarged.” (Gerard, 2017)

The demands on enterprise software systems tend to grow over time. In some cases, the growth is explosive – for example the rapid growth of number of Facebook users. But in most cases, the growth is slow and steady. Regardless, scalability is something that should be planned for early on since it will impact how the system is designed.

Gerard also points out that there are two ways to scale a system (Gerard, 2017):

- Vertical scaling – adding more memory or processors to a single server; and
- Horizontal scaling – adding more servers to the system.

Both these options require new hardware to be purchased if the enterprise software system is hosted on the premises of the enterprise. With the advent of cloud computing, scaling has become easier. Using Infrastructure as a Service (IaaS), adding a new application server now no longer requires purchasing new hardware. Even better, with Software as a Service (SaaS) the cloud will handle scaling without the enterprise having to do anything (except pay for the additional usage).

3.3 Reliability

Reliability has to do with “requirements about how often the software fails” (Dalbey, 1998).

ANSI defines software reliability as follows:

“[T]he probability of failure-free software operation for a specified period of time in a specified environment.” (Pan, 1999)

Although reliability is arguably more important to mission critical systems (such as aircraft avionics) than in enterprise software systems, system down time can certainly have a large financial impact on an enterprise. Just imagine how much money would be lost if the stock exchange systems would go down for even a morning.

In less severe cases, reliability might have an impact on the consumer perception of a business. For example, if your bank’s internet banking site keeps going down, will you not consider moving to a different bank with a more reliable site?

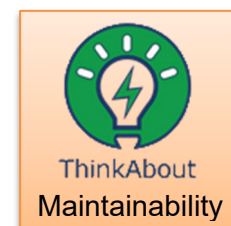
Requirements about reliability can also reveal useful information. For example, it may become clear that just about nobody cares whether the system is up early on a Monday morning, making that the perfect time for planned maintenance.

3.4 Maintainability

Maintainability is defined according to the IEEE Standard Glossary of Software Engineering Terminology as:

“The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.” (Crouch, 2018)

Imagine you start a new job on an enterprise software system that has been in development for five years. People have come and gone. And hundreds of thousands of lines of code have been written. And now *you* need to find that illusive bug that sometimes make this one table cell become non-editable for no apparent reason. Would you appreciate it at this point if there was a consistent folder structure and class naming convention?



The key to maintainability is consistency. If there is a project naming convention for database tables, for example, it will be a good step towards achieving maintainability since it will help people to find tables with the least amount of effort. But it is only useful if everybody consistently sticks to that naming convention all the time.

Steve Crouch makes the excellent point that you should design software for maintainability right from the start. (Crouch, 2018) Decide early on naming conventions, folder structures and even common design patterns, so that code follows a predictable pattern right from the start.

There is a lot that can be said about maintainability. More than we have time for in this module. But the last thing that I do want to mention is another of Crouch's points:

“Readable code is easy to understand (‘write programs for people’).” (Crouch, 2018)

With small, short-lived projects it is easy to get away with unreadable code – nobody will probably read your PROG6212 programs in future. But a large enterprise system tends to have a long lifetime by programming standards. Even if your code works perfectly today, it does not mean that years from now somebody else will not have to modify it. So be a good digital citizen and write readable code!

3.5 Security

Gary McGraw defines software security as follows:

“Software security – the process of designing, building and testing software for security – identifies and expunges problems in the software itself. In this way, software security practitioners attempt to build software that can withstand attack proactively.” (McGraw, 2004)

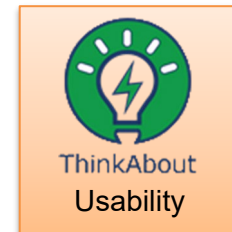
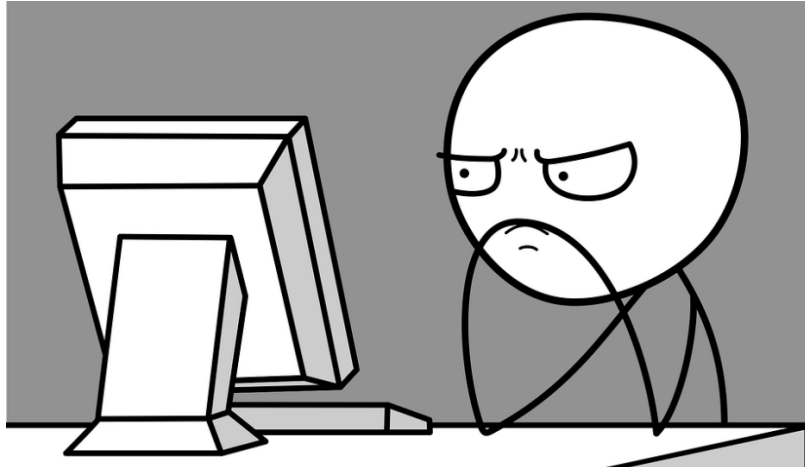
This is an excellent definition of security since it points out that security should be kept in mind throughout the life cycle of a software system. Security is not the responsibility of only the operations people that maintain the information technology systems of an enterprise. It should absolutely be a “multidisciplinary effort”. (McGraw, 2004)

We will discuss designing for security in more detail in Learning Unit 3.

3.6 Usability

Microsoft defines usability as follows:

“Usability is a measure of how easy it is to use a product to perform prescribed tasks.” (Microsoft Corporation, 2000)



Take a moment to think about this. What is the least usable software application that you have ever come across? Why did you find it difficult to use? What is the most usable software that you have ever used? How is it different from that least usable one?

Usability is more subjective than most of the non-functional requirements. And it does depend on the background of the user base as well. Something that is hard to use for you, might be easy for the next person because they have seen something similar before. So, when designing an application's user interface, design with the target audience in mind. Do not assume that the end users of your system have the same knowledge and experience as you!

There is a whole field of study called *User Experience (UX) Design* that deals with this topic. Read about it online to learn more.



3.7 Interoperability

“The definition of interoperability is the ability of a system, software or product to exchange and make use of information with other systems, software or products without special effort on the part of the user. For two systems to be interoperable, they must be able to exchange data and subsequently present that data such that it can be understood by a user.” (International TechneGroup Incorporated, 2018)

Before we delve into the details of this definition, let us consider an everyday example of interoperability. Say you buy a laptop from one manufacturer and a monitor from another. If both specify that they have a High-Definition Multimedia Interface (HDMI) interface, would you expect them to just work together if you plugged in the screen? That is the point of having the HDMI standard, right? This illustrates the point that “without special effort on the part of the user” in the above definition.



Interoperability is far more complex than just does it work or not. The Levels of Conceptual Interoperability Model (LCIM) was originally proposed by the Virginia Modeling, Analysis and Simulation Center in 2003. (Tolk, et al., 2013). Figure 4 shows the different levels in the model. In this model, there are different levels of interoperability that systems can have with each other.

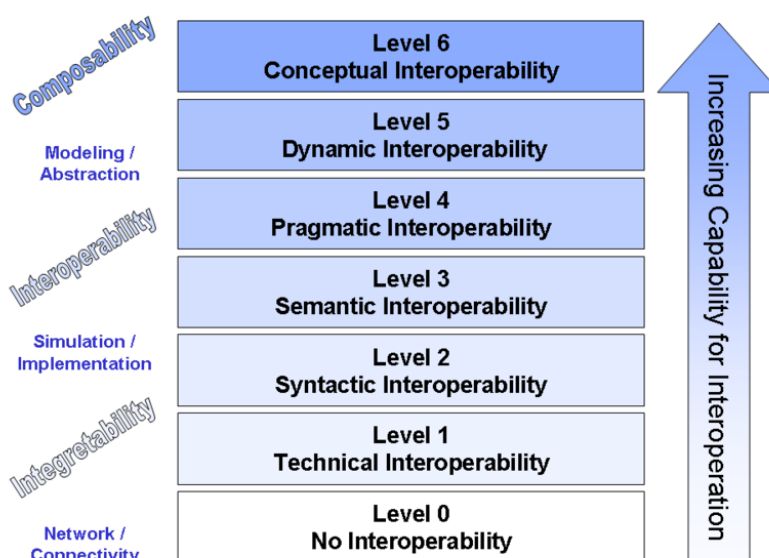


Figure 4: Levels of Conceptual Interoperability Model
from (Tolk, et al., 2013).

The different levels are defined as follows (Tolk, et al., 2013):

- At level 0, there is no interoperability, and we basically have *stand-alone systems* that can never interact with each other. This is a level that we are seeing less and less often in our ever more connected world. An example of this is single non-networked computer used for word processing.
- At level 1, it becomes possible for computers to exchange data over networks. This is achieved by implementing well defined *protocols* such as the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol stack that is widely used on the internet.
- At level 2, a “*common data format*” is defined, allowing systems to understand how the data that is exchanged is structured. An example of this would be to define that the interface between a client and server will use JavaScript Object Notation (JSON).
- At level 3, the “*meaning of the data*” is shared. For example, one of the fields in the JSON object exchanged between the client and server might be a date and time. At the semantic interoperability level, both systems would not only be able to parse the date, but also have a common understanding of the time zone that the time applies to.
- At level 4, the two systems understand the *context* of the data in the other system. Let us say an e-commerce store decides to have a half off everything promotion for two days. During normal operations, the stock management system will alert buyers if stock levels decrease at a certain rate due to buyers buying items from the e-commerce website systems. If the stock management system has the information about the context of the sudden influx of shoppers (the advertised promotion), then it can react differently than in normal circumstances.
- At level 5, the two systems understand how the data exchange affect the *state* of the other system *over time*. Let us consider a health care example. In a hospital, there is typically a system used by case managers and another system for billing. (Case managers are responsible for the administrative side of caring for a patient, keeping track of everything that happens during a hospital stay.) If the billing system understands how an event like moving a patient to intensive care affects the case management system’s state over time, then dynamic interoperability has been achieved.

- Level 6 is reached when the two systems have the same “*meaningful abstraction of reality*”.

It should be noted that the required level of interoperability for specific software systems can vary. For example, semantic interoperability may be enough in many cases.

This is by no means the only model of interoperability. Read more about this complex topic online.

4 Challenges in Enterprise Software Systems

4.1 Emergent Behaviour

Emergent behaviour is behaviour that is unforeseen, that happens because of complex interactions between systems – sometimes even seemingly unrelated systems.

Years ago, I was working on a system with a JavaEE back-end and a Java desktop application front-end. One morning we walked into the office to the complaints of the testers that the QA environment is refusing to run – the front-end was just not starting up. After some serious head scratching, I decided to try to run the application from my integrated development environment (IDE).

That revealed an error message about being unable to parse proxy settings. It turns out that somebody had changed the domain wide proxy settings the night before and had accidentally introduced a stray double quote in the exclusion list. Who could have guessed what effect that would have?

Fun Fact

Read the story about the *500-mile email* (Harris, 2002) for a fun example of emergent behaviour. It is too long to reproduce here, and I will not be able to tell it as well as the original author.

Here is a more detailed definition from thwink.org:

“Emergent behavior is behavior of a system that does not depend on its individual parts, but on their relationships to one another. Thus, emergent behavior cannot be predicted by examination of a system’s individual parts. It can only be predicted, managed, or controlled by understanding the parts and their relationships.” (thwink.org, 2014)

The larger the enterprise, the more complex and numerous its information systems. And the more likely it is that emergent behaviour will occur.

So, what kinds of behaviour are we talking about? Jeffrey C. Mogul of HP Labs researched what he calls “emergent misbehaviour”. He lists the following categories of misbehaviour (Mogul, 2005):

- *Thrashing* – when multiple processes try to access scarce resources, causing such huge overheads that none of them can function correctly. (Mogul, 2005) An example of this is when a computer’s virtual memory is so full that the operating system needs to constantly write and read pages to and from disk. This uses so much processing power that almost nothing else can happen. (Technopedia, 2021)
- *Unwanted synchronisation* – when processes that were not planned to run at the same time end up doing so inadvertently, thus using more resources than ever expected. (Mogul, 2005) Imagine a server being down, which causes the salary run to be delayed for a day. Which happens to then be on the same day as the monthly billing run, resulting in a flood of unplanned network traffic.
- *Unwanted oscillation* – when a system keeps on changing between states because of feedback between various processes. (Mogul, 2005)
- *Deadlock* – when multiple processes are all waiting for each other to do something, and none of them can make progress. (GeeksforGeeks, 2021) This can happen when multiple applications try to write to the same table in a database. (Slot, 2017)
- *Livelock* – when multiple processes keep changing state because of the interactions with one another, without making progress. Imagine the case where two people meet in a corridor and they struggle to move past each other. (Williams, 2021)
- *Chaotic behaviour* – when the behaviour of systems suddenly changes dramatically.

Knowing about these categories of potential problems is very useful when trying to find the cause of such a seemingly strange issue. But in practice, it often comes down to experience. The best advice I can give you when faced with emergent behaviour, is to get other people to look at the problem too. Different viewpoints on such a problem can be tremendously useful in resolving it!

4.2 Organisational Growth

When an enterprise grows, particularly when it grows at a fast rate, it can have all kinds of unexpected consequences. A large organisation is likely to have many different information systems. When the company has only ten employees, the founder can easily manage payroll in a simple software system on a single computer. By the time there are five hundred employees, there will be a whole payroll department that uses a networked system.

Enterprises can have different strategies when it comes to information systems. The most important distinction is between acquiring *custom software* built specifically for the enterprise (also called bespoke software) and buying *products off the shelf*. Enterprises tend to favour one or the other, but there may also be a mixture of the two.

For argument's sake, let us assume that the enterprise decides to go for custom software all the way. Imagine how many software development teams it now needs to develop and maintain all those software systems. Will the software development processes used by a small company be able to scale to accommodate this?

And this leads to our last topic: change management.

4.3 Change Management

Change management in an organisation is the process of ensuring that the roll out of software systems goes as smoothly as possible. Now that we have looked at emergent behaviour, it should be clear that if everybody does whatever they want whenever they want, the result is likely to be a chaotic one. So, some kind of process to help ensure that systems keep working well as new features or even whole new systems are introduced, is a good idea.

There are fundamentally two schools of thought around this concept:

- Strict governance must be put in place and any change must be approved by authorised people; and
- Software deployment should be automated and frequent.

The other way to put it, is that there is a divide between Change Management as it is traditionally implemented and DevOps. I love the way Ankush Seth puts it: “In other words, Change Management typically follows a waterfall approach, where DevOps follows agile.” (Seth, 2017)

We will look at this in a lot more detail in Learning Unit 5 when we discuss Waterfall, agile and DevOps. For now, I would like to leave you with this question: Do you think that large enterprises can still be agile?

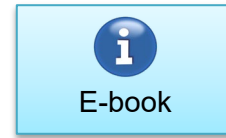
5 Concluding Remarks

No two enterprises are exactly alike in the way they do things. The same enterprise can even change dramatically over time, moving between different strategies as management change. So, there is no silver bullet when it comes to enterprise software system development. But this learning unit was intended to give you some insight into the environment in which enterprise software systems exist.

As we get into the more technical details in the next learning unit, I hope that having this background will help you to understand why things like design patterns are so important.

6 Recommended Additional Reading

In his free e-book entitled *The Five Pillars of a Great Business Analyst*, Yaaqub Mohamed lists five attributes that will make a business analyst stand out from the crowd. A lot of his advice is just as useful for software developers. Read more here:



- Mohamed, Y. *Patterns of Enterprise Application Architecture*. [ebook] Available at: <https://www.businessanalyststoolkit.com/free-business-analysis-ebooks-training/> [Accessed 27 October 2020].

For more information about what makes a good requirement, read the following articles online:

- *Writing Good Requirements* (Hooks, 1993); and
- *The Quest For Good Requirements* (Schedlbauer, 2011).

To learn more about developing maintainable software, read this online article by Steve Crouch:

- *Developing maintainable software* (Crouch, 2018)

For a detailed look at software security, read Gary's McGraw's excellent article:

- *Software security* (McGraw, 2004)

For excellent information about interoperability, read more on the IEEE Enterprise IT Body of Knowledge wiki:

- *Interoperability* (IEEE, 2017)

Read more about emergent behaviour by Jeffrey C. Mogul from HP Labs:

- *Emergent (Mis)behavior vs. Complex Software Systems* (Mogul, 2005)

Read more about databases and deadlocks:

- *Databases and Distributed Deadlocks: A FAQ* (Slot, 2017)

Read more about change management:

- *DevOps & Change Management In The Enterprise World* (Seth, 2017)

7 Revision Exercises

7.1 Revision Exercise 1

Motivate why it is necessary for there to be multiple different roles in a software development team.

7.2 Revision Exercise 2

Discuss three non-functional requirements, focusing on why each is important for enterprise software systems.

8 Solutions to Revision Exercises

8.1 *Revision Exercise 1*

Read Section 2.2.1 for a description of why multiple roles are necessary.

8.2 *Revision Exercise 2*

Read Section 3 for a description of non-functional requirements.

Learning Unit 2: Design and Architecture Patterns

Learning Objectives:

- Explain the purpose of design patterns.
- Use creational design patterns to develop applications.
- Use structural design patterns to develop applications.
- Use behavioural design patterns to develop applications.
- Use concurrency design patterns to develop applications.
- Differentiate between design patterns and architecture patterns.
- Use n-tier architecture to develop software applications.
- Use event-driven architecture to develop software applications.
- Compare Service Oriented Architecture and Microservices Architecture.
- Use Command Query Responsibility Segregation pattern to develop software applications.
- Use Domain Driven Design to develop software applications.
- Differentiate between anti-patterns and code smells.

Material used for this learning unit:

- Microsoft Visual Studio 2019.
- *Fundamentals of Computer Programming with C#*, Chapters 20 and 21 (Nakov & Kolev, 2013).
- Internet access.

How to prepare for this learning unit:

- Read Chapters 20 and 21 of (Nakov & Kolev, 2013) to recap on object-oriented principles.

My notes

1 Introduction

“Learn from the mistakes of others. You can't live long enough to make them all yourself.” — Eleanor Roosevelt (Goodreads Inc, 2018)

In software development, as in any field of study, we build on existing knowledge. This is true in the context of research, where someone may for example study existing encryption algorithms and come up with a performance improvement. But it is also true when we apply what we have learned to develop software systems for an enterprise. This is probably most apparent when considering the use of patterns – generic designs that can be applied to common software development problems. We must first understand the pattern and then build on it to apply it to the specific case.

2 Design Patterns

2.1 *What are Design Patterns?*

“In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.” (SourceMaking.com, 2018a)

The concept of design patterns in software was first described in detail by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. (Shute, 2016) The authors later become known as the Gang of Four, and the twenty-three patterns that they described are collectively known as the Gang of Four patterns. (Carr, 2009) Their seminal book (Gamma, et al., 1995), first published in the 1990s, is still in print and even available as an e-book.

The Gang of Four wrote this about the patterns in their book:

“The design patterns in this book are descriptions of communicating objects and classes that are customized to solve a general design problem in a specific context.” (Gamma, et al., 1995)

According to the Gang of Four, every pattern has four essential elements:

1. *Pattern name* – a short but descriptive name that is used to identify the pattern.
2. *Problem* – a description of when the pattern is applicable.
3. *Solution* – the generic description of how to implement the pattern.
4. *Consequences* – “the results and trade-offs of applying the pattern.” (Gamma, et al., 1995)

It was also the Gang of Four that originally distinguished between the different purposes of a pattern – creational, structural, or behavioural. (Gamma, et al., 1995) (More about each purpose later in this learning unit.)

Figure 5 shows an overview of all the patterns that we will be studying. This subset of patterns was chosen to introduce you to design patterns. But these are by no means the only design patterns in existence. I encourage you to read more online about the rest of the patterns.

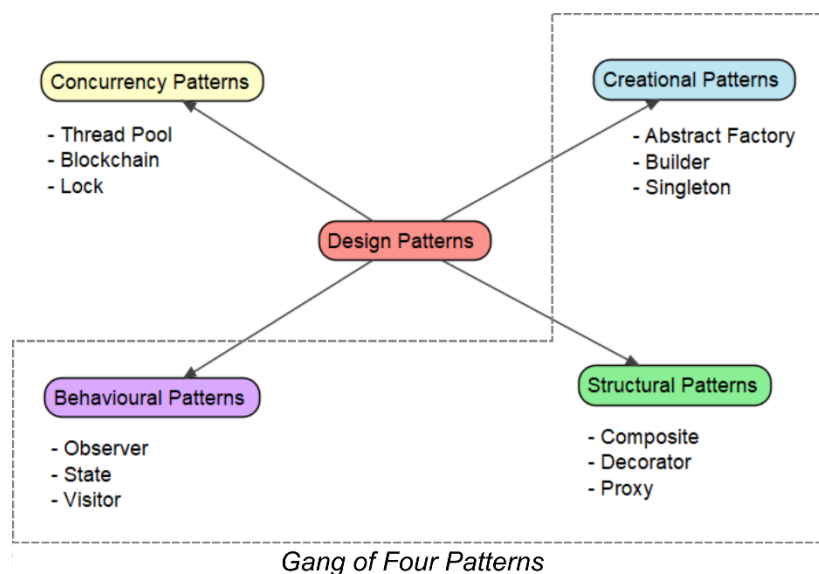


Figure 5: Design Patterns in this Learning Unit

You will notice that the concurrency patterns are not attributed to the Gang of Four – those were defined later.

Design patterns are language agnostic. This means that you could implement a design pattern in any object-oriented language.

So why do we want to use design patterns? By making use of a design pattern, we benefit from the knowledge of the original creator of the pattern. But we also gain common terminology that we can use to communicate with each other about our code.

2.2 *Creational Design Patterns*

Creational design patterns deal with ways to instantiate the objects that a system needs. The Gang of Four distinguishes between two types of creational design patterns (Gamma, et al., 1995):

- *Class creational patterns*: The class of object that is instantiated will be determined at runtime. The Factory Method (not covered in this module) is an example of a class creational pattern.
- *Object creational patterns*: The instantiation of objects is delegated to a specific object.

2.2.1 Abstract Factory Pattern

The abstract factory pattern is an object creational pattern. (Gamma, et al., 1995)

Read the following web page about the abstract factory pattern:

- Sourcemaking.com. 2018. *Abstract Factory Design Pattern*. [Online] Available at: https://sourcemaking.com/design_patterns/abstract_factory [Accessed 12 December 2022].

For more information as well as a C# code example of the abstract factory pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Abstract Factory*. [Online] Available at: <https://www.dofactory.com/net/abstract-factory-design-pattern> [Accessed 12 December 2022].

2.2.2 Builder Pattern

The builder pattern is an object creational pattern. (Gamma, et al., 1995)

Read the following web page about the abstract factory pattern:

- Sourcemaking.com. 2018. *Builder Design Pattern*. [Online] Available at:

https://sourcemaking.com/design_patterns/builder

[Accessed 12 December 2022].

For more information as well as a C# code example of the builder pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Builder*. [Online] Available at: <https://www.dofactory.com/net/builder-design-pattern> [Accessed 12 December 2022].

2.2.3 Singleton Pattern

The singleton pattern is an object creational pattern. (Gamma, et al., 1995)

Read the following web page about the singleton pattern:

- Sourcemaking.com. 2018. *Singleton Design Pattern*. [Online] Available at: https://sourcemaking.com/design_patterns/singleton [Accessed 27 October 2020].

For more information as well as a C# code example of the builder pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Singleton*. [Online] Available at: <https://www.dofactory.com/net/singleton-design-pattern> [Accessed 12 December 2022].

2.3 Structural Design Patterns

Structural design patterns describe the relationships between objects (Carr, 2009) and how to compose them to create complex structures at runtime (Gamma, et al., 1995).

2.3.1 Composite Pattern

Read the following web page about the composite pattern:

- Sourcemaking.com. 2018. *Composite Design Pattern*. [Online] Available at: https://sourcemaking.com/design_patterns/composite [Accessed 12 December 2022].

For more information as well as a C# code example of the composite pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Composite*. [Online] Available at: <https://www.dofactory.com/net/composite-design-pattern> [Accessed 12 December 2022].

2.3.2 Decorator Pattern

Read the following web page about the composite pattern:

- Sourcemaking.com. 2018. *Decorator Design Pattern*. [Online] Available at: https://sourcemaking.com/design_patterns/decorator [Accessed 12 December 2022].

For more information as well as a C# code example of the decorator pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Decorator*. [Online] Available at: <https://www.dofactory.com/net/decorator-design-pattern> [Accessed 12 December 2022].

2.3.3 Proxy Pattern

Read the following web page about the proxy pattern:

- Sourcemaking.com. 2018. *Proxy Design Pattern*. [Online] Available at: https://sourcemaking.com/design_patterns/proxy [Accessed 12 December 2022].

For more information as well as a C# code example of the proxy pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Proxy*. [Online] Available at: <https://www.dofactory.com/net/proxy-design-pattern> [Accessed 12 December 2022].

2.4 Behavioural Design Patterns

Behavioural design patterns describe how objects communicate with each other (Carr, 2009).

2.4.1 Observer Pattern

Read the following web page about the observer pattern:

- Sourcemaking.com. 2018. *Observer Design Pattern*. [Online] Available at:

https://sourcemaking.com/design_patterns/observer
[Accessed 12 December 2022].

For more information as well as a C# code example of the observer pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Observer*. [Online]
Available at: <https://www.dofactory.com/net/observer-design-pattern> [Accessed 12 December 2022].

2.4.2 State Pattern

Read the following web page about the state pattern:

- Sourcemaking.com. 2018. *State Design Pattern*. [Online]
Available at:
https://sourcemaking.com/design_patterns/state
[Accessed 12 December 2022].

For more information as well as a C# code example of the state pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *State*. [Online]
Available at: <https://www.dofactory.com/net/state-design-pattern> [Accessed 12 December 2022].

2.4.3 Visitor Pattern

Read the following web page about the visitor pattern:

- Sourcemaking.com, 2018. *Visitor Design Pattern*.
[Online] Available at:
https://sourcemaking.com/design_patterns/visitor
[Accessed 12 December 2022].

For more information as well as a C# code example of the visitor pattern, have a look at this web page:

- Data & Object Factory LLC. 2018. *Visitor*. [Online]
Available at: <https://www.dofactory.com/net/visitor-design-pattern> [Accessed 12 December 2022].

2.5 Concurrency Design Patterns

Concurrency patterns define ways to successfully perform multi-threaded tasks (Laker & Cenerelli, 2012).

Concurrency design patterns were not described by the Gang of Four, and as such, lack the detailed and standardised descriptions that we have seen so far in this learning unit.

2.5.1 Thread Pool

2.5.1.1 Problem

When many tasks need to be performed in parallel, it may not be feasible to have that many threads all at once. For each thread, memory needs to be allocated. (Varun, 2015) And operating systems have limits on the number of threads allowed per process.

2.5.1.2 Solution

Create a thread pool – a limited number of threads, each of which can process multiple tasks. Once a task is complete, a thread in the thread pool can request a new task to perform. (Laker & Jester, 2012b)

2.5.1.3 Consequences

Using a thread pool will perform better than using a single thread to perform tasks, without facing the overhead of creating a very large number of threads. (Laker & Jester, 2012b)

2.5.2 Blockchain

2.5.2.1 Problem

In a distributed communication system where there is no central controller, it is important to determine whether the data communicated from independent entities can be trusted. An example of such a system is the cryptocurrency Bitcoin. (Keyhole Software, 2018)

“In the most basic of terms, a blockchain is a distributed, append-only database ledger system.” (Keyhole Software, 2018)

2.5.2.2 *Solution*

A blockchain uses cryptographic concepts such as hashing, Merkle trees and a nonce to ensure that transaction data can only be appended. (Keyhole Software, 2018).

The details of how blockchains work are beyond the scope of this manual. Read more in (Keyhole Software, 2018).

2.5.2.3 *Consequences*

Data from independent systems can be trusted using a blockchain.

2.5.3 **Lock**

2.5.3.1 *Problem*

More than one thread may want to access the same resource, for example a database. A lock is a mechanism that allows threads to synchronise, so that only one can access that resource at a time. (Amarasinghe, et al., 2015)

2.5.3.2 *Solution*

A lock has two operations: acquire (take ownership of the lock) and release (releasing ownership). Only one thread can own the lock at any point, so other threads must wait until the lock is released before acquiring the lock. (Amarasinghe, et al., 2015)

2.5.3.3 *Consequences*

Using a lock can prevent multiple threads accessing and modifying data concurrently, leaving the data in an inconsistent state.

If applied incorrectly, this could result in deadlock (Amarasinghe, et al., 2015) (a situation where no process can be made (GeeksforGeeks, 2018)).

3 Architecture Patterns

3.1 *What are architecture patterns?*

We have seen that design patterns are generic solutions to problems in software applications. The Gang of Four wrote this about where design patterns fit in:

“Design patterns are not about designs such as linked lists and hash tables that can be encoded in classes and reused as is. Nor are they complex, domain specific designs for an entire application or subsystem.” (Gamma, et al., 1995)

Design patterns are applicable to a part of an application. For example, a user interface window may be implemented using the Model View Controller pattern. In contrast, architecture patterns are solutions to common problems at an application wide level.

Architecture patterns are also sometimes referred to as architectural styles. (Microsoft Corporation, 2010) But there is disagreement about the use of this terminology. (Graca, 2017) In this module, we will use the term architecture pattern.

3.2 *n-tier architecture*

In n-tier architecture, the different components in an application are organised into several horizontal tiers of components that perform a specific role in the application. (Richards, 2015) There can be any number of tiers, but many applications have only three tiers. (Microsoft Corporation, 2010)

Before we dive into the details, a note on terminology. The term n-tier architecture is also sometimes called layered architecture. While the two terms are closely related, they are in fact not identical. Layers are a logical separation of related functionality that would typically be deployed on the same physical server. Tiers are designed in a way that each tier can be deployed on a different server. (Sharma, 2011)

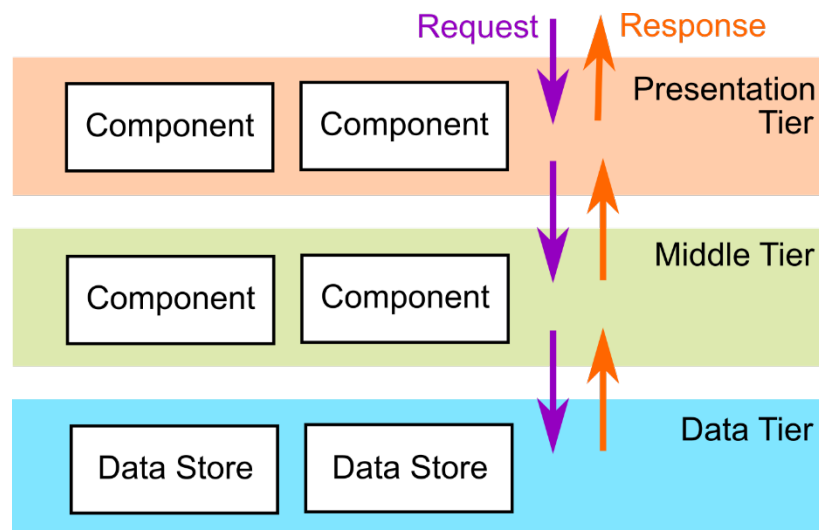


Figure 6: n-tier architecture

In an n-tier architecture, each tier interacts only with the tier directly above or directly below it. An example of a request flowing through multiple tiers is shown in Figure 6. Remember that each of the tiers will in practice reside on a different server. For example, the presentation tier may be hosted on a web server, the middle tier on a different application server and the data tier on a database cluster.

To improve scalability, communication between tiers is typically asynchronous. (Microsoft Corporation, 2010)

The benefits of using an n-tier architecture are:

1. **Maintainability:** Change to a single tier can be deployed without affecting the other tiers. (Microsoft Corporation, 2010)
2. **Scalability:** Tiers are designed to be separate from the get-go, making adding more resources to a specific tier easy. (Microsoft Corporation, 2010)
3. **Flexibility:** The different tiers can be managed independently. (Microsoft Corporation, 2010)
4. **Availability:** Because the architecture is modular by design, it is easy to configure failovers. (Microsoft Corporation, 2010)
5. **Security:** Each tier can be secured to the extent required, independently of other tiers. (Stringfellow, 2017)

3.3 Event-driven architecture

In an event-driven architecture, there are producers which create events, which are processed by event consumers. (Wasson & Bennage, 2018) The consumers and producers are highly decoupled, making this architecture very scalable. (Richards, 2015)

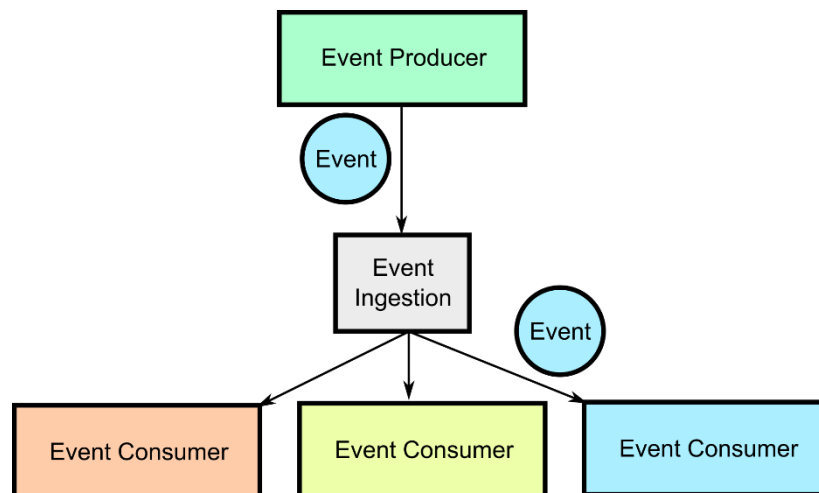


Figure 7: Event-driven architecture

Event-driven architecture can be implemented in two ways (Richards, 2015):

1. Using a central mediator that directs an event to the consumers that can process it.
2. Using a broken topology where the message flow is distributed instead of located within a single mediator.

The benefits of using an event-driven architecture are (Wasson & Bennage, 2018):

1. **Scalability:** Because of the loose coupling, it is very easy to add more consumers of a type if required.
2. **Flexibility:** It is easy to add new consumers because of the loose coupling of producers and consumers.
3. **Performance:** A consumer can respond as soon as it receives the event.

3.4 Service Oriented Architecture and Microservices

Service Oriented Architecture (SOA) and Microservices Architecture (MSA) have a lot in common, but there are also some important differences. Let us start by looking at SOA, since it is the older architecture (the name was first used in the 1990s (Despodovski, 2017)).

“Service Oriented Architecture is a software architecture where distinct components of the application provide services to other components via a communications protocol over a network.” (Despodovski, 2017)

This definition, unsurprisingly, mentions that SOA makes use of services. SOA can be implemented using different styles of services – from web-based services where data is exchanged in a text format to services that transmit their data in binary formats. (IBM, 2016)

A component in SOA can either be a service provider or a service consumer. Providers and consumers often communicate via an Enterprise Service Bus (ESB). (Miri, 2017) SOA focuses a lot on how to integrate distributed components into a larger application, rather than on how to split an application into different re-usable modules. (Despodovski, 2017)

Making use of services does not necessarily mean that the architecture of the system is SOA, however – there is more to it than that. David Sprott and Lawrence Wilkes explain it like this:

“SOA is not just an architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the right services are provided and consumed.” (Sprott & Wilkes, 2004)

MSA, a term first used in 2011, is an architecture where the focus is on developing completely independent services, each with a very specific function. (Despodovski, 2017) James Lewis and Martin Fowler defines MSA as follows:

“[T]he microservice architectural style [1] is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.” (Lewis & Fowler, 2014)

So, what is the fundamental difference between SOA and MSA? In SOA, the idea is to share as much as possible, while with MSA it is all about the services being as independent of each other as possible. Because of this, it is important to have strict corporate governance in place in SOA, because breaking one service can have a large impact on the overall system. With MSA, since the services are far more independent, governance can be relaxed, and this makes it easier to implement DevOps. (Watts, 2017b)

3.5 Command Query Responsibility Segregation pattern

In the Command Query Responsibility Segregation (CQRS) pattern, two different models are used: one for reading data and a different one for writing data. (Fowler, 2011)

Traditional models where all the create, read, update, and delete (CRUD) operations are done on the same set of entities, have some drawbacks. For example, there may be columns in a table that need to be updated regardless of whether they are applicable for a specific operation. And conflicts may occur if multiple users try to update data in parallel. (Narumoto, et al., 2017)

To solve these kinds of issues, a model is defined for writing data, that is specifically designed for this purpose and has its own security and permissions. A separate model is created for reading data, designed to only get the data that is relevant for display and with possibly less stringent security requirements. (Narumoto, et al., 2017)

Be aware that using this pattern adds complexity to the code, that may sometimes outweigh the benefits of this pattern. (Fowler, 2011)

3.6 Domain Driven Design

According to Steven A. Lowe, Domain Driven Design (DDD) can be summed up as “capture the domain model in domain terms, embed the model in the code, and protect it from corruption.” (Lowe, 2018)

To understand this definition, we first need to understand the concept of a domain as it is used here. The domain is the discipline where the software system will be used. (Powell-Morse, 2017) Will the software be used in an insurance company? Then the domain is insurance.

The focus in DDD is to capture what the experts in the domain know, in terms that are familiar to them. Then you consistently use the terminology in your designs and code. This results in ubiquitous language – the technical and non-technical stakeholders use the same terminology, thus improving communication between people. (Lowe, 2018)

4 Anti-Patterns

4.1 Anti-Patterns and Code Smells

An anti-pattern is almost the opposite of a design pattern. A design pattern, if implemented well and in the correct context, will ensure that the software does what it was intended to do. So, using a design pattern helps to create good software.

The opposite is an anti-pattern – common way of doing things that has been proven to *not* work correctly. This means that anti-patterns should *never* be used. (Ritchie, 2010)

The term anti-pattern was first used by Andrew Koenig in 1995, and his definition was as follows:

“An antipattern is just like a pattern, except that instead of a solution it gives something that looks superficially like a solution, but isn’t one.” (Agile Alliance, 2018)

A code smell is something that indicates that there may be a problem. It is not a certainty, but rather an indication, that there is an issue. (Ritchie, 2010) Martin Fowler puts it this way:

“A code smell is a surface indication that usually corresponds to a deeper problem in the system.” (Fowler, 2006)

So, why learn about anti-patterns and code smells if they are undesirable? To enable you to avoid making the same mistakes others have already made.

4.2 Common Anti-Patterns

Although there are also anti-patterns in software architecture and project management (Sourcemaking.com, 2018b), we will focus on the software development anti-patterns. Here are some of the most common code-related anti-patterns:

1. *Spaghetti code*: Software that contains very little structure, making it hard to understand even for the original developer. (Sourcemaking.com, 2018d)
2. *The blob*: Also known as the *god class*. A single class that contains most of the software's functionality and logic. (Sourcemaking.com, 2018e)
3. *Exception handling*: Also known as *coding by exception*. Using exceptions to handle cases which are expected to happen and are not really exceptions. (Dejaeger, 2010)

There are also software development anti-patterns that have to do with the process of developing software rather than the code itself:

1. *Continuous obsolescence*: Depending on one or more technologies that change so often that developers cannot keep up with the changes, let alone develop meaningful new software. (Sourcemaking.com, 2018f)
2. *Reinventing the wheel*: Not making use of existing frameworks or libraries but writing everything from scratch. (Dejaeger, 2010)
3. *Golden hammer*: Using a known and trusted solution just because it is known, regardless of whether it is the best solution to a problem. (Sourcemaking.com, 2018g)

These are only a few examples. But they all illustrate one point – when an anti-pattern is present, it is guaranteed to be a problem.

4.3 Common Code Smells

Sourcemaking.com lists five categories of code smells (Sourcemaking.com, 2018c):

1. *Bloaters*: As a software system grows, methods, classes and parameter lists tend to increase in size over time.
2. *Object-Orientation Abusers*: Incorrectly using object-oriented principles.
3. *Change Preventers*: A class of problem where you have to make a large number of changes to your code if you change something in one spot.
4. *Dispensables*: Things that are unneeded such as dead code, that should have been removed.
5. *Couplers*: Things that create too much or too little coupling between classes.

Jeff Atwood distinguishes between code smells within classes (such as large classes) and code smells that have to do with the interaction between classes (for example classes that expose their internals unnecessarily) (Atwood, 2006).

Table 1: Common Code Smells Matrix

Derived from (Sourcemaking.com, 2018c), (Atwood, 2006) and (DevIQ, 2018)

	Code Smells Within Classes	Code Smells Between Classes
Bloaters	<ul style="list-style-type: none"> • Long Method • Large Class • Long Parameter List • Oddball Solution • Combinatorial Explosion • Class Does Not Do Much 	<ul style="list-style-type: none"> • Primitive Obsession • Data Clumps • Solution Sprawl • Required Setup/Teardown Code
Object-Oriented Abusers	<ul style="list-style-type: none"> • Switch Statements • Temporary Field 	<ul style="list-style-type: none"> • Refused Bequest • Alternative Classes with Different Interfaces • Static Cling • Class Depends on Subclass
Change Preventers	<ul style="list-style-type: none"> • Type Embedded in Name • Conditional Complexity 	<ul style="list-style-type: none"> • Divergent Change • Shotgun Surgery • Parallel Inheritance Hierarchies
Dispensables	<ul style="list-style-type: none"> • Comments • Duplicate Code • Dead Code • Speculative Generality 	<ul style="list-style-type: none"> • Lazy Class • Data Class
Couplers	-	<ul style="list-style-type: none"> • Feature Envy • Inappropriate Intimacy • Message Chains • Middle Man • Indecent Exposure • Incomplete Library Class
Obfuscators	<ul style="list-style-type: none"> • Uncommunicative Name • Inconsistent Names • Obscured Intent 	-

Read more about each of the code smells in (Sourcemaking.com, 2018c), (Atwood, 2006) and (DevIQ, 2018).

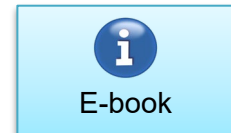
5 Recommended Additional Reading

For more about design patterns, read these books:

- Sarcar, V. 2018. *Design Patterns in C#: A Hands-on Guide with Real-World Examples*. Bangalore: Apress.
- Gamma, E. et al. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Read the free e-book by Mark Richards on enterprise architecture patterns:

- Richards, M. 2015. *Software Architecture Patterns*. O'Reilly Media, Inc. Available at: <https://www.oreilly.com/programming/free/software-architecture-patterns.csp> [Accessed 12 December 2022].



For more about enterprise architecture patterns, read this book by Martin Fowler:

- Fowler, M. 2002. *Patterns of Enterprise Application Architecture*. Pearson.

For more about blockchains, read this Keyhole Software white paper:

- Keyhole Software, 2018. *Blockchain For the Enterprise*. [Online] Available at: <https://keyholesoftware.com/wp-content/uploads/Blockchain-For-The-Enterprise-Keyhole-White-Paper.pdf> [Accessed 12 December 2022].

Also read the paper by Satoshi Nakamoto, the original creator of blockchain:

- Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online] Available at: <https://bitcoin.org/bitcoin.pdf> [Accessed 12 December 2022].

6 Revision Exercises

6.1 *Revision Exercise 1*

What is the difference between a design pattern and an architecture pattern? When is it applicable to use each one?

6.2 *Revision Exercise 2*

What is the difference between an anti-pattern and a code smell? Explain by mentioning at least one example of each one.

7 Solutions to Revision Exercises

7.1 Revision Exercise 1

Read Sections 2.1 and 3.1 for more information.

7.2 Revision Exercise 2

Read Section 4 for information about anti-patterns and code smells.

Learning Unit 3: Enterprise Software System Development

Learning Objectives:

- Discuss the building blocks of the Microsoft Enterprise Application Development Platform and their relationships to each other.
- Design software applications using end-to-end transaction management to ensure data integrity.
- Use enterprise software messaging standards to develop software applications to achieve efficient communication.
- Use the Lightweight Directory Access Protocol (LDAP) to develop applications that integrate with enterprise identity-related services.
- Discuss the principles of enterprise system security.
- Compare service orchestration and choreography.
- Use service orchestration to develop software applications.
- Use service choreography to develop software applications.
- Explain the role of an enterprise portal.
- Compare the various possible ways in which an enterprise software system can be deployed and managed.

Material used for this learning unit:

- Microsoft Visual Studio 2019.
- Internet access.
- Free account at Microsoft Learn.

How to prepare for this learning unit:

- Before you attend the lectures, read all sections of the learning unit in this module manual.

My notes

1 Introduction

In Learning Unit 1, we learned about the requirements and challenges of large enterprise software systems. Microsoft and other software and infrastructure providers also understand these requirements and they provide the technologies to make solving enterprise business problems easier. In this learning unit, we will encounter some of the building blocks that are available to you as an enterprise software developer.

Before continuing with the learning unit, sign up for a free account at Microsoft Learn at this address: <https://docs.microsoft.com/en-us/learn/> While the online learning resources developed by Microsoft can be accessed without an account, with an account you can track your progress through the learning paths and earn badges.

2 Microsoft Enterprise Application Development Platform

2.1 Core Layers

In 2005, Microsoft used the term Enterprise Application Development Platform to describe all the technologies available to develop connected enterprise software systems on the .NET framework. (Microsoft Corporation, 2005) Although this is not a term that is widely used anymore, the concept is still a valid and interesting one.

Have a look at Figure 1 of (Microsoft Corporation, 2005) – it is online available here: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973223\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973223(v=msdn.10)?redirectedfrom=MSDN) [Accessed 12 December 2022].

The intent with the platform was to use open standards such as web services, in a Service Oriented Architecture (SOA).

The platform was designed from the start to be layered. The defined layers are (Microsoft Corporation, 2005):

- .NET Framework;
- Data layer, including databases and portals and host-based integration;
- Application layer, containing packed applications;
- Business service layer, where enterprise application code typically resides;
- Business process layer that can handle orchestration and communication between organisations; and
- Client layer, including smart devices, smart clients and web clients.

All the layers are supported by tools, management systems, and patterns and best practices (Microsoft Corporation, 2005).

While some of the layers may be familiar to you after studying n-tier architecture in Learning Unit 2, you may find that some of the layers listed above feel out of place. That is because the layers as defined here focus more on the technologies that are available to implement enterprise systems, than the architecture of the systems themselves.

2.2 Core Technologies and Products

Let us look at the technologies and products in use the Microsoft Enterprise Application Development Platform when it was first designed. Many of these are still in use today, but I am sure that you will spot some that are outdated.

2.2.1 .NET Framework

The Common Language Runtime (CLR) and .NET Framework classes are the main components of the .NET Framework. (Microsoft Corporation, 2005) You learned all about these in PROG6212, at least as far as the C# implementation is concerned. There are numerous other programming languages that have Common Language Infrastructure (CLI) compilers, for example Eiffel (Eiffel.org, 2018), F# (Carter, et al., 2021) and Visual Basic .NET (Petrusha, et al., 2018).

2.2.2 Data layer

On the data layer, we find the following (Microsoft Corporation, 2005):

- Microsoft SharePoint Server – portal software;
- Microsoft SQL Server – relational database management system (RDBMS); and
- Microsoft Host Integration Server (HIS) – software that provides connectivity with IBM mainframe systems (Larsen & Ohlinger, 2016).

2.2.3 Application layer

On the application layer, perhaps the most unexpectedly named layer, we find Windows Server Systems (WSS). This layer is named application layer, since it refers to the applications that come with the server operating system. Application services include (Microsoft Corporation, 2005):

- Web Applications (ASP.NET);
- Data Access (ADO.NET);
- Active Directory (Lightweight Directory Access Protocol (LDAP));
- Security (Code Access Security (CAS));
- Transaction Management (Distributed Transaction Coordinator (DTC)); and
- Messaging (Microsoft Message Queuing (MSMQ)).

2.2.4 Business Process Layer

Microsoft BizTalk Server provides the capability to integrate services and applications (Microsoft Corporation, 2005). Here is Microsoft's definition of BizTalk Server:

"BizTalk Server is a publish and subscribe architecture that uses adapters to receive and send messages, implements business processes through orchestration, and includes management and tracking of these different parts. BizTalk Server also includes trading partner management for business-to-business messaging, high availability to maximize uptime, a development platform to create your own components, an administration console to manage your artifacts, and business activity monitoring to manage aggregations, alerts, and profiles." (Ohlinger, et al., 2017)

2.2.5 Clients

Three types of clients are identified (Microsoft Corporation, 2005):

- Smart devices – smartphones and personal digital assistants (PDAs) that run apps;
- Smart clients – Windows Forms applications and Microsoft Office applications;
- Web applications – user interfaces implemented using web technologies.

2.2.6 Supporting Tools

Visual Studio, in all its different editions, is the main tool that is used to support development of software systems.

2.3 Technologies Today

Which technologies did you identify that are outdated? How about Windows Forms and PDAs?

A great many new technologies were added since 2005, including cloud computing. In the rest of this learning unit, we will explore several technologies introduced here, followed by a brief discussion of the different options available today for deploying applications.

2.4 .NET Framework vs .NET Core vs .NET

If you created a new project in Visual Studio 2017, you may have noticed that there was a distinction between .NET Framework, .NET Core and .NET Standard projects (see Figure 8).

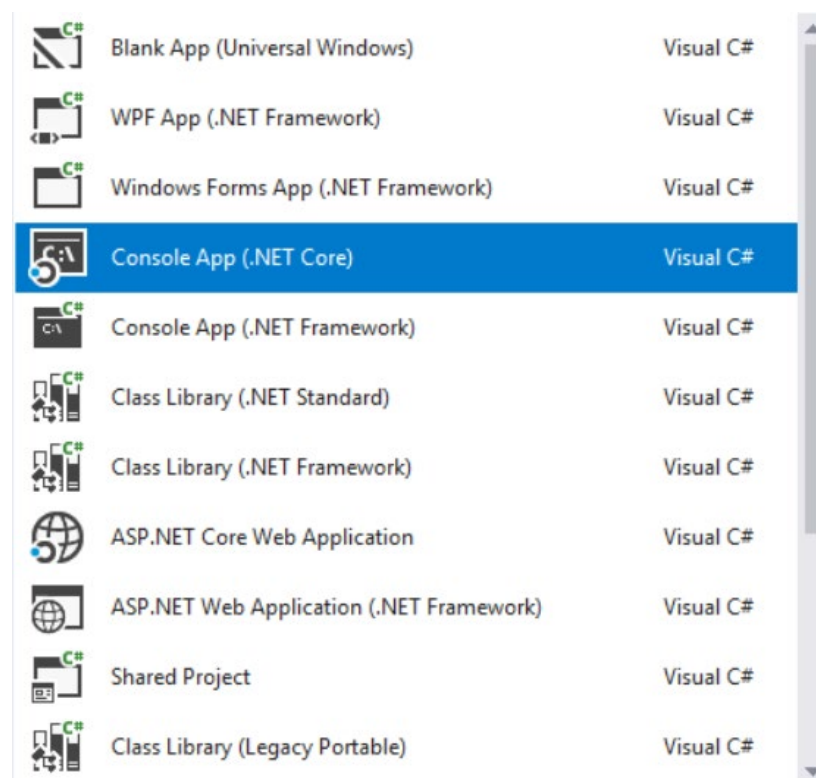


Figure 8: Visual Studio 2017 Projects

In Visual Studio 2019, the distinction is still there but less obvious (see Figure 9). Now there are projects for .NET Framework, and the others with nothing in the name. So, what is the difference?

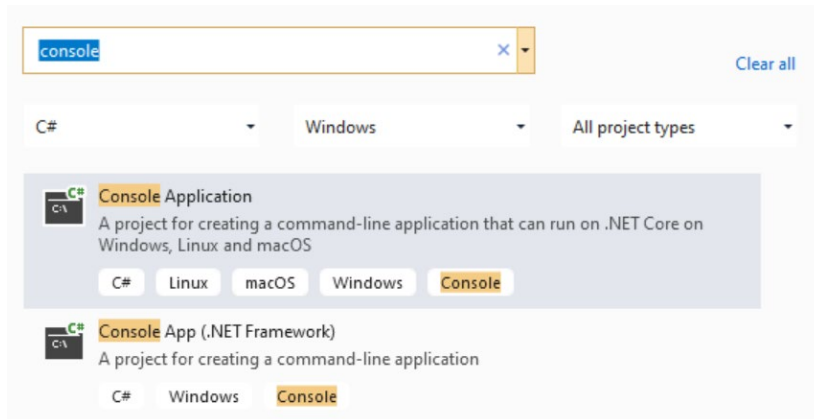


Figure 9: Visual Studio 2019 Projects

To understand the full story, we need to talk about the history of .NET. The .NET Framework was the very first implementation of .NET that Microsoft released all the way back in 2002 (Luijbregts, 2018), and this was the definition from the Microsoft website:

“.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.” (Microsoft Corporation, 2018)

Much later, in 2016, .NET Core was released. (Luijbregts, 2018) Here is the definition from Microsoft again:

“.NET Core is a cross-platform version of .NET for building websites, services, and console apps.” (Microsoft Corporation, 2018)

Read (Luijbregts, 2018) for more information about the .NET Ecosystem, up to 2018.

The important thing to note here is that .NET Framework and .NET Core were not different versions of the same thing. These two separate runtimes still exist to this day and are used for different purposes. (Luijbregts, 2018) The latest version of .NET Framework is 4.8. And the last version of .NET Core that still carried that name was .NET Core .3.1.

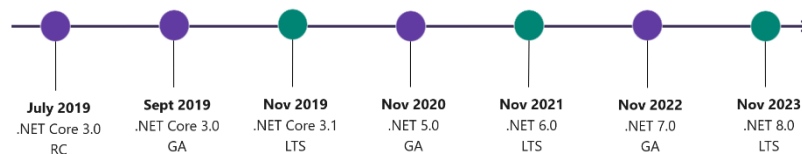
In May 2019, Microsoft announced its plans for the future, and it was simply called .NET 5. (Viswav, 2019) Going forward, the plan is to have just one .NET that can be used to target any

platform. (Lander, 2019) .NET 5 is actually the next version of .NET Core. But to avoid confusion with the .NET Framework 4.x numbers, it was decided to skip ahead to version 5. (Pine, et al., 2020)

The fact that .NET 5 now is considered the main implementation of .NET, explains why the projects in Visual Studio 2019 don't specify the name as .NET Core anymore. It is now simply .NET.

.NET Framework 4.x is still supported going forward though. But the main focus from Microsoft is now on .NET 5. (Pine, et al., 2020)

.NET Schedule



- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

Figure 10. .NET Release Schedule (from (Lander, 2019))

.NET 5 was released on schedule in November 2020. And at the time of writing of this manual, .NET 6 was right on schedule to release in November 2021. .NET 6 is a long-term support (LTS) release, which means it is a version that can safely be used in large software applications.

The question does remain: when to use .NET Framework, and when to use the shiny net .NET? .NET is perfect for cross-platform apps (not just Windows), and for creating microservices and hosting apps in Docker containers. .NET Framework should be used for existing apps that already use .NET Framework, or in cases where an app uses some technology that can only run on the .NET Framework. (Carter, et al., 2021)

Read (Carter, et al., 2021) for a more detailed description of when to choose which runtime.

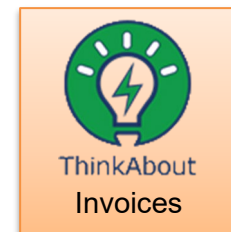
3 Transaction Management

3.1 Definition

“A transaction is a single unit of work which means either ALL or NONE. If a transaction is successful, all of the data operations are committed and become a durable part of the database. If a transaction encounters errors/exceptions and must be cancelled or rolled back, then all of the data modifications/operations need to be removed.” (Mohapatra, 2016)

The single unit of work described in this definition is made up of multiple tasks that need to be performed together. (Wenzel, et al., 2017) If it was a single task, there would be no real need for a transaction.

Consider this database example. When a user captures a new invoice, the data needs to be stored in a database. An invoice has a header containing information such as the invoice number and who the invoice was made out to, and multiple lines containing the details of the line items on the invoice. So typically, there are two tables involved – let us call them Invoice and InvoiceLine. Let us assume that the Invoice entry gets inserted without issues. But when we get to the third InvoiceLine, there is a constraint violation that keeps the line from getting inserted into the database. Without a transaction, you would end up with half of an invoice in the database. But if you used a transaction, you could roll back everything so that there is no inconsistency in the data.



Transactions have four key properties, with the acronym ACID (IBM, 2018):

- **Atomicity:** all the changes to data are performed or none of them;
- **Consistency:** the data will be in a consistent state when the transaction is committed;
- **Isolation:** while the transaction is being processed, other transactions cannot see the state of the data that will not yet be consistent; and
- **Durability:** when the transaction is committed, the data is stored permanently even if the system fails right after the commit completes.

Transactions can be distributed or non-distributed (also called local). A non-distributed transaction uses only one database, while a distributed transaction makes use of more than one database. (Pal, 2018)

In .NET applications, transactions can be managed in all kinds of places:

- SQL Server (stored procedures);
- ADO.NET;
- Entity Framework (EF);
- Windows Communication Foundation (WCF);
- Ambient .NET transactions using `TransactionScope`.

The main differences between these are where the transaction boundary is – where the transaction starts and ends. Let us look at each of them.

3.2 SQL Server Transactions

Commands can be executed on a SQL Server database using Transact-SQL – Structured Query Language (SQL) with some SQL Server specific additions. (Rouse, et al., 2017)

When using SQL to interact directly with the database, there are two kinds of transactions – implicit and explicit. An implicit transaction is created automatically when you call any of the following commands: CREATE, ALTER, DROP, TRUNCATE, INSERT, UPDATE and DELETE.

If an error occurs within one of these statements, the complete statement will be rolled back. (Mohapatra, 2016)

Explicit transactions are more interesting. When a developer decides that several statements should be executed as one unit of work, a transaction can explicitly be created. If the transaction is rolled back, then all the changes up to that point will be rolled back. (Mohapatra, 2016)

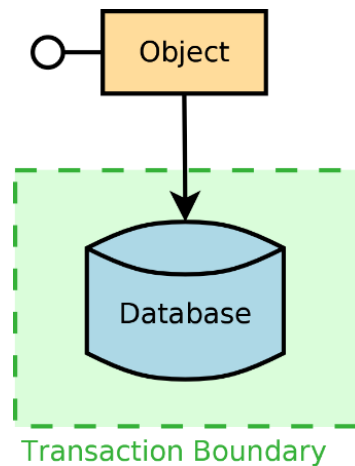


Figure 11: SQL Server transaction – non-distributed

Figure 11 shows a non-distributed SQL Server transaction. Notice that the transaction only extends as far as the database. Let us look at an example of this in SQL:

```
// Example of a committed transaction  
BEGIN TRANSACTION;  
DELETE FROM AppUser WHERE UserID = 9;  
COMMIT;
```

```
// Example of a rolled back transaction  
BEGIN TRANSACTION;  
DELETE FROM AppUser;  
ROLLBACK;
```

It is possible to name transactions and to nest transactions. Read more in (Laudenschlager, et al., 2016).

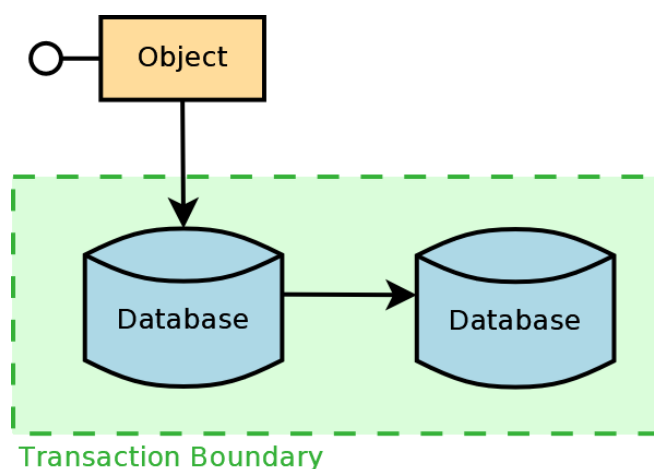


Figure 12: SQL Server transaction – distributed

Figure 12 shows a distributed SQL Server transaction. Notice how this transaction uses multiple databases. How do we do this in SQL?

```
BEGIN DISTRIBUTED TRANSACTION;  
DELETE FROM UserDb.AppUser WHERE UserID = 9;  
DELETE FROM RemoteServer.UserDb.AppUser  
    WHERE UserID = 9;  
COMMIT;
```

A very important point to note is that the Microsoft Distributed Transaction Coordinator (MS DTC) is required to be installed for this distributed transaction to work at all. (Laudenschlager & Guyer, 2016)

SQL Server supports different levels of isolation between transactions (Rabeler, et al., 2017):

- *Read uncommitted*: A transaction can read data from another transaction that has been modified but not yet committed (this is called dirty reads). This is used to minimise locking contention but should be avoided in almost all cases. There are other options to minimising locking contention while preventing dirty reads, such as using the snapshot isolation level.
- *Read committed*: The default isolation level for all SQL Server databases. (Sheldon, 2014) Other transactions can only read changes once the transaction is committed, thus avoiding dirty reads.
- *Repeatable read*: Like read committed, with the additional limitation that data that has been read by a transaction cannot be modified by another transaction until the transaction that read it completes.
- *Snapshot*: Only data that was committed before the start of the current transaction can be accessed. Note that there is the database option `ALLOW_SNAPSHOT_ISOLATION` that must be set to on for this to work.
- *Serializable*: Like repeatable read, with the additional limitation that other transactions cannot update or insert rows that would appear in queries already executed by the current transaction.

The transaction isolation level can be set in SQL and will remain set to the chosen level for the specific connection until it is set again. (Rabeler, et al., 2017) Here is a SQL example:

SET TRANSACTION ISOLATION LEVEL SNAPSHOT;

The SQL Server transaction support is used under the hood by ADO.NET and Entity Framework when implementing transactions.

3.3 ADO.NET

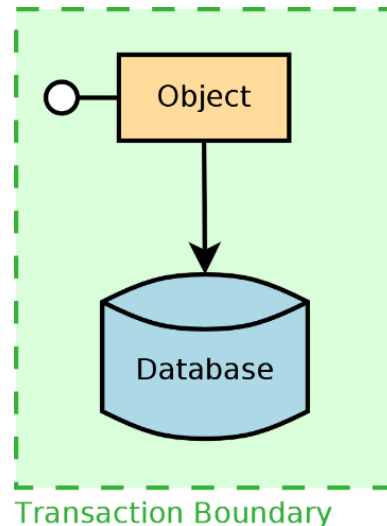


Figure 13: Simple ADO.NET transaction

When using ADO.NET, the developer must explicitly manage transactions in application code. (Rabeler, et al., 2017) This works well in the situation where a single object needs to access a single database, as shown in Figure 13. Notice that the object that is making the call now is part of the transaction.

What does this look like in C#?

```
SqlConnection connection =  
    new SqlConnection(connectionString);  
connection.Open();  
SqlCommand command =  
    new SqlCommand(queryString, connection);  
  
IDbTransaction transaction;  
transaction = connection.BeginTransaction();  
command.Transaction = transaction;  
try  
{  
    command.ExecuteNonQuery();  
    transaction.Commit();  
}  
catch  
{  
    transaction.Rollback();  
}  
finally  
{  
    connection.Close();  
}
```

To set the isolation level, pass the isolation level as a parameter to the `SqlConnection.BeginTransaction` method.

```
IDbTransaction transaction;  
transaction = connection.BeginTransaction(  
    IsolationLevel.ReadCommitted);
```

If multiple objects are involved in a transaction, and even worse multiple databases, the management of the transaction becomes very complex. (Rabeler, et al., 2017)

3.4 Entity Framework

When using Entity Framework (EF), the framework manages database transactions for you to some extent when it generates the SQL to be executed on the database. For every time that you call `DbContext.SaveChanges()`, a transaction is created and committed. (EntityFrameworkTutorial.net, 2018)

Let us look at the C# code to insert some entities into a database:

```
using (var context = new SheepContext())
{
    context.Sheep.Add(aSheep);
    context.Sheep.Add(anotherSheep);
    context.SaveChanges();

    context.Sheep.Add(thirdSheep);
    context.SaveChanges();
}
```

The class `SheepContext` in this example inherits from `DbContext`. In this code, two transactions will be created under the hood by EF – one for each call of the `SaveChanges()` method.

It is useful to know what EF does in the background. But how can we control where the transaction should start and end?

```
using (var context = new SheepContext())
{
    using (var transaction = context.Database.BeginTransaction())
    {
        try
        {
            context.Sheep.Add(aSheep);
            context.Sheep.Add(anotherSheep);
            context.SaveChanges();

            context.Sheep.Add(thirdSheep);
            context.SaveChanges();

            transaction.Commit();
        }
        catch (Exception ex)
        {
            transaction.Rollback();
        }
    }
}
```

In the above code, only one database transaction will be created. (EntityFrameworkTutorial.net, 2018)

To set the isolation level for this transaction, pass the level into the method `DbContext.Database.BeginTransaction()`.

Note that the `BeginTransaction()` and `EndTransaction()` methods were only introduced in EF 6.0. (Mohapatra, 2016)

Entity Framework Core supports the same transaction handling described above. (Rojansky, et al., 2020)

3.5 Windows Communication Foundation

One major feature of .NET Framework that was not ported to .NET 5 and its successors, is creating services using Windows Communication Foundation (WCF). It is still possible to make a call from a client .NET 5 to a .NET Framework WCF service. (Carter, et al., 2021) Hence this section only applies to .NET Framework.

When creating services in WCF, three attributes in the System.ServiceModel namespace allow you to configure transactions. (Wenzel, et al., 2017b)

The first attribute is TransactionFlowAttribute. This attribute is applied on the service contract to specify what is expected from a client calling the service with regards to transactions. (Wenzel, et al., 2017b) The TransactionFlowOption enum has three possible values (Microsoft Corporation, 2018c):

- Allowed: If the client calls the service in a transaction, that transaction can be used by the service.
- Mandatory: The client must call the service in a transaction.
- NotAllowed: The service call will fail if the client calls the service in a transaction.

Here is an example of a service contract with the TransactionFlowAttribute:

```
[ServiceContract]
public interface IUsefulService
{
    [OperationContract]
    [TransactionFlow(TransactionFlowOption.Mandatory)]
    string Method1(int value);
}
```

The second attribute is ServiceBehaviorAttribute, that is applied on the service implementation class. It has four properties related to transactions (Wenzel, et al., 2017b):

- TransactionAutoCompleteOnSessionClose: determines whether to complete the transaction when the session is gracefully terminated;
- ReleaseServiceInstanceOnTransactionComplete: indicates whether to release the service instance when the calling transaction completes;

- `TransactionIsolationLevel`: specifies the isolation level to use for all the transactions within the service; and
- `TransactionTimeout`: the time to wait for a transaction to complete before aborting.

The third and last attribute is the `OperationBehaviorAttribute`, which is applied to methods in a service implementation class. This attribute has two properties of interest:

- `TransactionScopeRequired`: This property indicates whether the method must be executed in a transaction scope. If the caller does not provide a transaction scope, one will automatically be created for the duration of this method.
- `TransactionAutoComplete`: Indicates whether the transaction in use in the method should be automatically closed if the method completes without issues.

Here is an example of a service implementation with both the `ServiceBehavior` and `OperationBehavior` attributes.

```
[ServiceBehavior(
    ConcurrencyMode = ConcurrencyMode.Single,
    ReleaseServiceInstanceOnTransactionComplete = true,
    TransactionAutoCompleteOnSessionClose = true,
    TransactionIsolationLevel = IsolationLevel.RepeatableRead,
    TransactionTimeout = "00:03:00"
)]
public class Service1 : IUsefulService
{
    [OperationBehavior(
        TransactionAutoComplete = true,
        TransactionScopeRequired = true
    )]
    public string Method1(int value)
    {
        throw new NotImplementedException();
    }
}
```

Note that not all WCF bindings support transactions. Consult (Wenzel, et al., 2018) for the full list of bindings and the features supported by each one.

Depending on how the service is configured and how the client calls it, the client can either be outside the transaction boundary, or inside it. Both cases are shown in Figure 14.

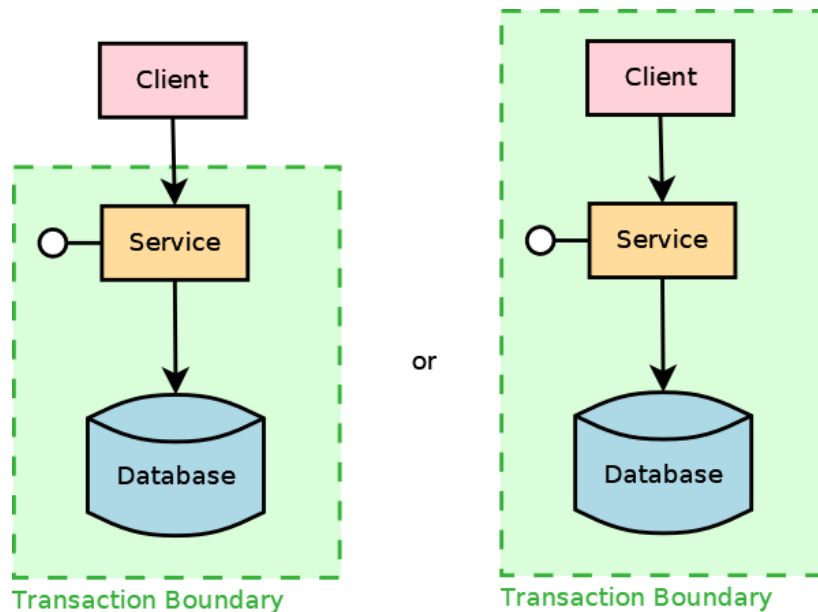


Figure 14: WCF Transactions

Note that there is a performance implication for including the client in the transaction. More calls are made over the network between the client and the service if the client is managing the transaction. So only use a transaction on the client if it is really necessary.

3.6 *TransactionScope*

Since .NET Framework version 2.0, there is a way to manage all these different transactions in a consistent way, using the classes and interfaces in the `System.Transactions` namespace. This implementation includes a transaction manager called the Lightweight Transaction Manager (LTM), which can manage non-distributed transactions that use resources that is able to be managed in this way. (Lowy, 2005)

`System.Transactions` Resource Managers are resources that can be managed using a `TransactionScope`. (Lowy, 2005) Examples of such resources include:

- Entity Framework (and even Entity Framework Core since version 2.1) communicating with a SQL Server database;
- Microsoft Message Queuing (MSMQ) (Lowy, 2005) and
- IBM WebSphere MQ.NET (IBM, 2018b).

Note that when a message is committed to a message queue, the transaction ends there. A new transaction will get created when a subscriber picks up the message to process it. More about message queues in Section 4.

Let us see how our ADO.NET example from earlier will look using `TransactionScope`:

```
using (TransactionScope scope = new TransactionScope())
{
    using (var connection = new SqlConnection(connectionString))
    {
        connection.Open();
        IDbCommand command =
            new SqlCommand(queryString, connection);
        command.ExecuteNonQuery();
    }
    scope.Complete();
}
```

See (Pal, 2018) for another example.

And how about the Entity Framework example?

```
using (TransactionScope scope = new TransactionScope())
{
    using (var connection = new SqlConnection(connectionString))
    {
        connection.Open();
        IDbCommand command =
            new SqlCommand(queryString, connection);
        command.ExecuteNonQuery();
    }
    scope.Complete();
}
```

See (Gavilán, 2018) for another example.

Both examples are non-distributed transactions – only a single database is used in each case. So, let us look at an example of a distributed transaction, mixing not only databases but also ADO.NET and Entity Framework.

```
using (TransactionScope scope = new TransactionScope())
{
    using (var connection = new SqlConnection(connectionString))
    {
        connection.Open();
        IDbCommand command =
            new SqlCommand(queryString, connection);
        command.ExecuteNonQuery();
    }
    using (var context = new SheepContext())
    {
        context.Sheep.Add(aSheep);
        context.Sheep.Add(anotherSheep);
        context.SaveChanges();
        context.Sheep.Add(thirdSheep);
        context.SaveChanges();
    }
    scope.Complete();
}
```

As with distributed transactions on SQL Server, MSDTC is required for this to work. (Vega, et al., 2016)

Note that Entity Framework Core does NOT support distributed transactions. (Gerr, 2018)

There are two more interesting facts around TransactionScopes. The first is that a TransactionScope, by default, must be created and disposed on the same thread. It might sound like a non-problem, but when using an asynchronous call to get data, this exact problem will occur. There is a way to allow the TransactionScope to flow between threads – pass a parameter into its constructor: TransactionScopeAsyncFlowOption.Enabled. However, due to timing issues, you can still get an exception if the same context is used in one transaction that can flow between threads and one that is not. (Gerr, 2018)

TransactionScopes can be nested. But a BeginTransaction() cannot be nested inside a TransactionScope. (Gerr, 2018)

4 Messaging

4.1 Definition

Microsoft Message Queuing (MSMQ) is a technology that allows applications to add messages to a queue, so that other applications can read the messages. (Microsoft Corporation, 2016) A queue is a First In, First Out (FIFO) structure. The first message that is added to the queue will also be the first message picked up by a consumer.

4.2 Configuration

MSMQ can be installed on desktop and server Windows operating systems – it is an optional Windows feature. For detailed instructions on how to install MSMQ, read (Wenzel, et al., 2017c)

To manage queues:

1. On the Windows Start Menu, search for Computer Management and run it.
2. Expand the Services and Applications node.
3. Expand the Message Queuing node.
4. Click Private Queues. Here you can view and add queues (see Figure 15).

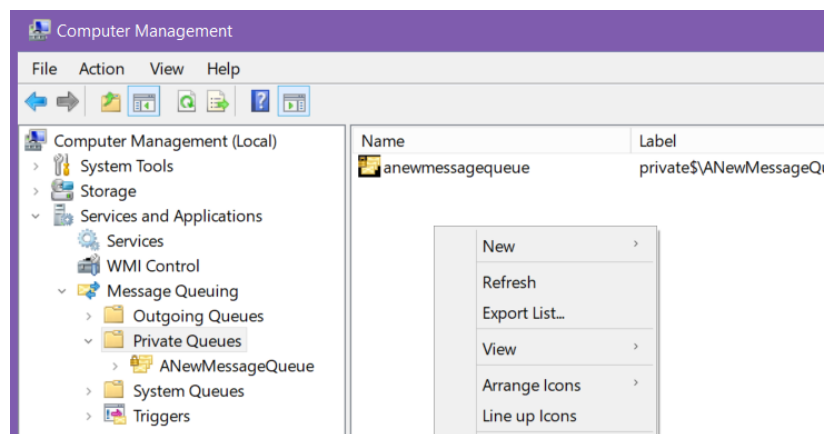


Figure 15: MSMQ Configuration

When you create a new queue, you can choose whether it must be transactional or not.

Private queues are local queues, that are maintained on the same computer. An outgoing queue is automatically created by MSMQ if you send a message to queue on another server. (Breakwell, 2008) This is illustrated in Figure 16.

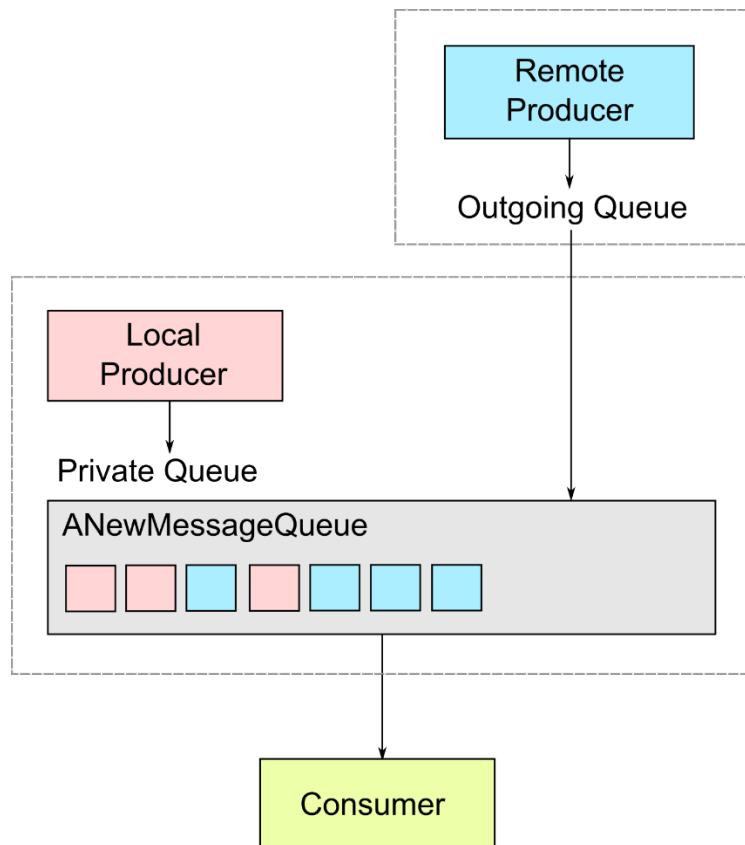


Figure 16: MSMQ In Action

You can also create public queues. These queues are published in Active Directory. (Krishnan, 2007)

4.3 Using MSMQ in C#

To use MSMQ directly (without Windows Communication Foundation (WCF)) in C#, you need to add a reference to System.Messaging to your project. (Kanjilal, 2016)

Before you use the queue, you can check whether it exists, and create it if necessary (Microsoft Corporation, 2018c):

```

string queuePath1 = @".\Private$\ANewMessageQueue";
if (!MessageQueue.Exists(queuePath1))
{
    MessageQueue.Create(queuePath1);
}
  
```

Once your queue exists, you can send a message like this:

```
string queuePath = @".\Private$\ANewMessageQueue";
string message = "Hello world!";
using (MessageQueue queue = new MessageQueue(queuePath))
{
    queue.Send(message);
}
```

Remember that MSMQ is one of the resources that can be used with a TransactionScope. So, here is the same send code, this time in a transaction:

```
using (var scope = new TransactionScope())
{
    string queuePath1 = @".\Private$\ANewMessageQueue";
    Message message = new Message();
    message.Body = "Hello world!";
    using (MessageQueue queue = new MessageQueue(queuePath1))
    {
        queue.Send(message);
    }
}
```

We can read that message from this queue like this:

```
string queuePath = @".\Private$\ANewMessageQueue";
using (MessageQueue queue = new MessageQueue(queuePath))
{
    System.Type[] arrTypes = new System.Type[1];
    arrTypes[0] = typeof(string);
    queue.Formatter = new XmlMessageFormatter(arrTypes);
    Message aMessage = queue.Receive();
    Console.WriteLine(aMessage.Body.ToString());
}
Console.ReadLine();
```

The Receive() method removes the message from the queue.

If you do not specify which formatter to use when receiving the message, you will get an exception. So, why use the XmlMessageFormatter, if we sent a string? Because by default the XmlMessageFormatter is used when serialising a message. (Microsoft Corporation, 2018b)

Sending a single string in a message is not very exciting though, so let us see how we can send a custom struct containing data. Here is the definition of the Beverage struct that we want to send:

```
public struct Beverage
{
    public string Description { get; set; }
    public string Size { get; set; }
}
```

To send it using XML serialisation is just as easy as with the string message:

```
Beverage xllatte = new Beverage()
{
    Description = "Cafe Latte", Size = "Extra Large" };

string queuePath1 = @".\Private$\ANewMessageQueue";
Message message = new Message();
message.Body = xllatte;
using (MessageQueue queue = new MessageQueue(queuePath1))
{
    queue.Send(message);
}
```

And to receive the Beverage is very easy too:

```
string queuePath = @".\Private$\ANewMessageQueue";
using (MessageQueue queue = new MessageQueue(queuePath))
{
    System.Type[] arrTypes = new System.Type[1];
    arrTypes[0] = typeof(Beverage);
    queue.Formatter = new XmlMessageFormatter(arrTypes);
    Message aMessage = queue.Receive();
    Beverage b = (Beverage)aMessage.Body;
    Console.WriteLine(b.Description + " - " + b.Size);
}
Console.ReadLine();
```

The XmlMessageFormatter deserialises the message so that we have a Beverage object again on the receiving side.

When formatting messages as eXtensible Markup Language (XML), you can read the contents of the message in the Computer Management view (see Figure 17). (Stojanovic, 2015)

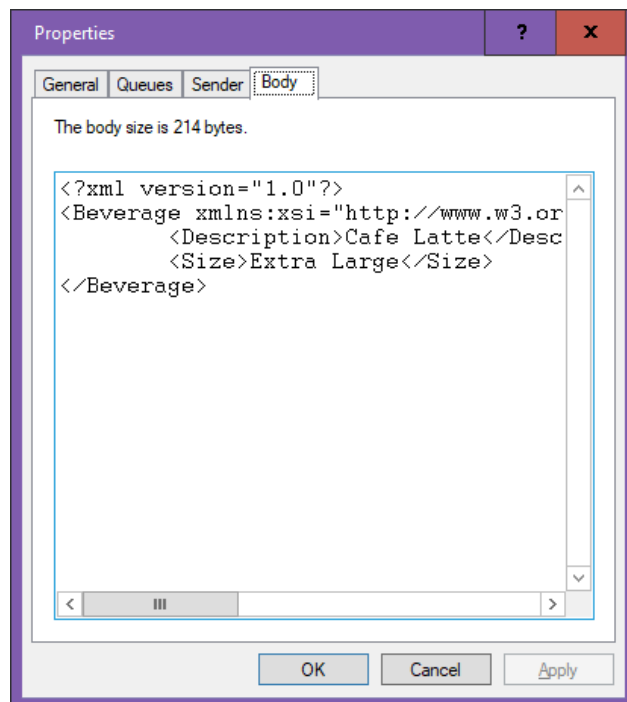


Figure 17: Beverage XML in MSMQ

But what if we want to send our message in a different format? There is also a `BinaryMessageFormatter` available which serialises the message in a binary (non-human readable) format. To use the `BinaryMessageFormatter`, the struct needs to be marked as serialisable:

```
[Serializable]
public struct Beverage
{
    public string Description { get; set; }
    public string Size { get; set; }
}
```

Then we can send the message in a binary format like this:

```
Beverage xILatte = new Beverage()
{ Description = "Cafe Latte", Size = "Extra Large" };

string queuePath1 = @".\Private$\ANewMessageQueue";
Message message = new Message();
message.Body = xILatte;
message.Formatter = new BinaryMessageFormatter();
using (MessageQueue queue = new MessageQueue(queuePath1))
{
    queue.Send(message);
}
```

And receive it:

```
string queuePath = @".\Private$\ANewMessageQueue";
using (MessageQueue queue = new MessageQueue(queuePath))
{
    queue.Formatter = new BinaryMessageFormatter();
    Message aMessage = queue.Receive();
    Beverage b = (Beverage)aMessage.Body;
    Console.WriteLine(b.Description + " - " + b.Size);
}
Console.ReadLine();
```

If you want to use a different format, you can always write your own formatter. This is exactly what Dejan Stojanovic did to use JSON as the formatter. The JSON formatter is faster and creates much smaller messages. Read more in (Stojanovic, 2015).

4.4 Handling Delivery Errors

So how do you handle a failed delivery? If the first message in a queue cannot be delivered for some reason, then all the other messages in the queue are blocked from being processed. This can be handled using dead-letter queues. (Wenzel, et al., 2017d)

When creating a message, you can set the time that that specific message will live in the queue:

```
Message message = new Message();
message.Body = x1Latte;
message.TimeToBeReceived = new TimeSpan(0, 0, 30);
message.Formatter = new BinaryMessageFormatter();
using (MessageQueue queue = new MessageQueue(queuePath1))
{
    queue.Send(message);
}
```

When this message exceeds its 30 seconds time to live, it will automatically be removed from the queue. But what if you want to keep the message somewhere for fault finding purposes? You can do this by sending it to the dead-letter queue – a system-wide queue of discarded messages.


```

Message message = new Message();
message.Body = xILatte;
message.TimeToBeReceived = new TimeSpan(0, 0, 30);
message.UseDeadLetterQueue = true;
message.Formatter = new BinaryMessageFormatter();
using (MessageQueue queue = new MessageQueue(queuePath1))
{
    queue.Send(message);
}

```

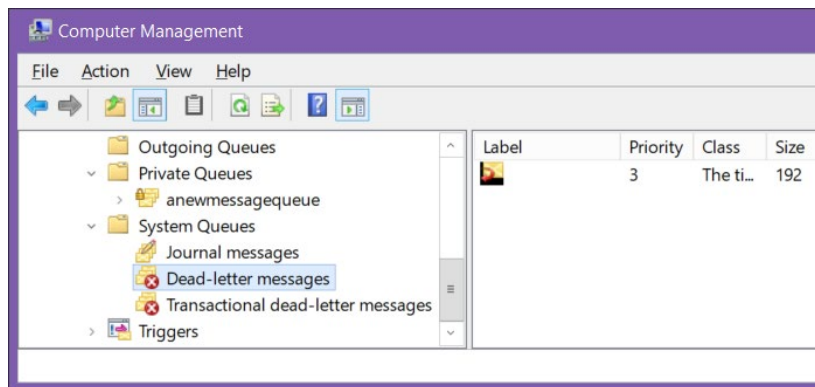


Figure 18: Message in the Dead-Letter Queue

You can also use MSMQ with WCF. Read more about it in (Wenzel, et al., 2017d) and (Wenzel, et al., 2017e).

5 Directory Services

5.1 Definition

“Directory services are software systems that store, organize and provide access to directory information in order to unify network resources. Directory services map the network names of network resources to network addresses and define a naming structure for networks.” (Technopedia, 2018b)

Remember that public queues in Microsoft Message Queuing (MSMQ) are published to Active Directory. Well, that is an example of using directory services. You don’t need to know the physical location of such a service to access it. (Technopedia, 2018b)

Lightweight Directory Access Protocol (LDAP) is a protocol for providing directory services. (TechTarget, 2008) It is used by Microsoft’s Active Directory (AD) – a set of services that run on Windows Server. (TechTarget, 2018)

5.2 *Getting Started with Apache Directory*

Since Microsoft's AD is fairly complex to configure and requires a Windows Server operating system, we will be using an open source alternative for the server – Apache Directory.

Download ApacheDS™ (the server) here:

<http://directory.apache.org/apacheds/> [Accessed 12 December 2022].

And download Apache Directory Studio™ (the configuration user interface) here:

<http://directory.apache.org/studio/> [Accessed 12 December 2022].

Note that the default port that ApacheDS runs its unencrypted connection on is 10389. Also, anonymous access is enabled by default.

To get started with the tutorial data provided by Apache:

1. Install ApacheDS (allow it to start the service when it asks during installation) and Apache Directory Studio.
2. Run Apache Directory Studio.
3. On the welcome page, click the bottom right button to go to the workspace.
4. Create a new connection with hostname localhost and port 10389.
5. Follow the instructions from the basic user guide at <http://directory.apache.org/apacheds/basic-ug/1.4.3-adding-partition.html> to add a new partition. Use the exact name!
6. Disconnect from the server.
7. Restart the ApacheDS – default service using the Windows Services Manager. (Search for Services on the Windows Start Menu.)
8. Download the tutorial data here:
<http://directory.apache.org/apacheds/basic-ug/resources/apache-ds-tutorial.ldif> [Accessed 12 December 2022].
9. Connect to the server again using Apache Directory Studio.
10. Click File and then Import.
11. Expand the LDAP Browser category and select LDIF into LDAP.
12. Click Next.
13. Browse to the tutorial data file.
14. Click Finish.

15. Download the tutorial data for Captain Hook's user here:
<https://directory.apache.org/apacheds/basic-ug/resources/captain-hook-hierarchy.ldif> [Accessed 12 December 2022].
16. Edit the file and remove the first four lines, so that the file now starts with:
dn: cn=James Hook,ou=people,o=sevenSeas
17. Import the Captain Hook file too.

Now you should have new entries in your directory, as shown in Figure 19.

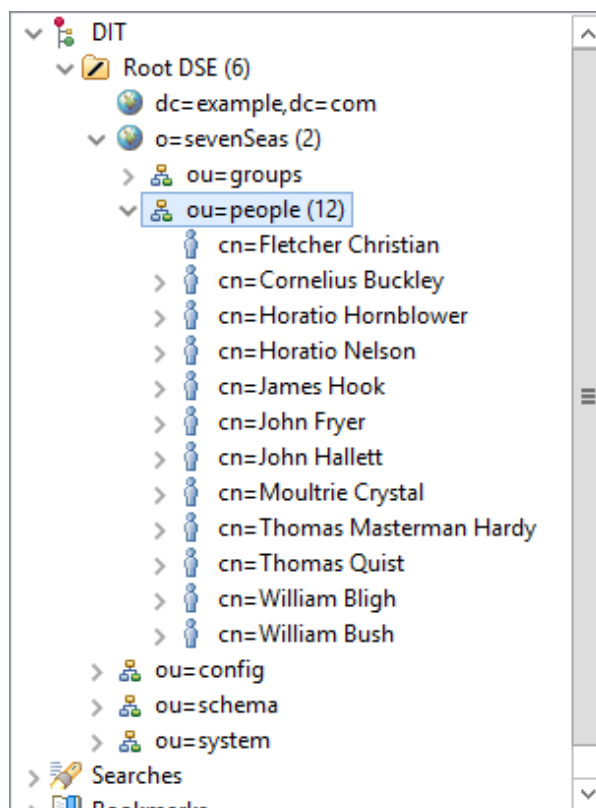


Figure 19: Imported Tutorial Data

5.3 LDAP in C#

Now that we have a directory server up and running, we can query the data like this in C#:

```
var ldapDirectoryIdentifier =
    new LdapDirectoryIdentifier("localhost", 10389, true, false);

using (var ldapConnection = new LdapConnection(ldapDirectoryIdentifier))
{
    ldapConnection.AuthType = AuthType.Anonymous;
    ldapConnection.SessionOptions.ProtocolVersion = 3;
    var request = new SearchRequest("ou=people,o=sevenSeas",
        "(objectClass=organizationalPerson)", SearchScope.Subtree, null);
    var response = (SearchResponse)ldapConnection.SendRequest(request);
    foreach (SearchResultEntry entry in response.Entries)
    {
        Console.WriteLine(entry.DistinguishedName);
    }
}
```

When using ApacheDS, it is important to include the line setting in the protocol version 3. Otherwise, you will get a Protocol Exception when trying to send the request.

To verify whether a user's credentials are correct, use the method `LdapConnection.Bind()`:

```
var ldapDirectoryIdentifier =
    new LdapDirectoryIdentifier("localhost", 10389, true, false);
var networkCredential = new NetworkCredential(
    "cn=James Hook,ou=people,o=sevenSeas", "peterPan");
using (var ldapConnection = new LdapConnection(
    ldapDirectoryIdentifier, networkCredential))
{
    ldapConnection.AuthType = AuthType.Basic;
    ldapConnection.SessionOptions.ProtocolVersion = 3;
    bool result = true;
    try
    {
        ldapConnection.Bind();
    }
    catch (Exception)
    {
        result = false;
    }
    if (result == true)
    {
        Console.WriteLine("Oh no, Hook logged in!");
    }
    else
    {
        Console.WriteLine("Hook not allowed!");
    }
}
```

Read (Peyrott, 2015) for more code examples of using LDAP in C# projects.

6 Security

6.1 Introduction

“Computer Security is the protection of computing systems and the data that they store or access.”
(University of California Santa Cruz, 2015)

Computer security is important both to enterprises and their customers. As a customer, you want to be sure that your private data, especially things like passwords, will not be revealed to the world. So not only could a leak of private information cause big losses for customers, it would also severely damage the reputation of the enterprise.

Security Breaches Close to Home

Four security breaches have been reported in South Africa since March 2017 (Niselow, 2018):

- Ster-Kinekor’s booking website was revealed to have a major security flaw in March 2017;
- Master deeds data was leaked in October 2017;
- ViewFines data was revealed in March 2018; and
- Liberty had their email repository hacked in June 2018.

6.2 Concepts

Computer security is a large field of study, so we will only be able to touch on a few of the core concepts here.

We saw in the definition of computer security that it is about protecting data. If nobody is supposed to access the data, it is easy to protect – put the data on a server that is not connected to any network. But this is rarely the case. So how can we ensure that the right people do get to access the data?

6.2.1 Authentication

The first important step in allowing access to data, is authentication. Authentication is “the process that confirms a user’s identity”. (L, 2018) When you log into a website, the software system knows who you are.

Authentication can be done based on what a user has, or what the user knows (SecurEnvoy Ltd, 2018). Let us look at some of the ways in which a user can be authenticated:

- Username and password – the most common authentication mechanism, but far from the most secure (L, 2018);
- Passwordless authentication – a link is sent to the user's email address, or a One Time Password (OTP) via Short Message Service (SMS) (Otemuyiwa, 2016);
- Biometrics – for example the fingerprint authentication used by many smartphones today (L, 2018);
- Hard token – a physical device that needs to be present to authenticate a user (Segal, n.d.);
- Multi factor authentication – using at least two different ways of authentication together, for example username and password with an OTP (SecurEnvoy Ltd, 2018).

Important: If you do decide to use authentication that uses a password, then it is essential that the password should NOT be stored in clear text in the database! So how do we avoid this? By applying a hash algorithm on the password before storing it. Hashing is a one-way transformation of data – it is impossible (or at least impractical) to get the original password from the hashed version. (Oracle Corporation, 2018)

Be aware though that not all hashing algorithms are equally strong. Some of them used to be strong enough in the past because of the relatively limited processing power that was available back then. But that is no longer the case. Read (Defuse Security, 2018) for more information about how to do effective password hashing.

6.2.2 Authorisation

Authentication is only the first step though. Just knowing who the user is does not mean that he/she should have access to everything in the system. That is where authorisation comes in.

“Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features.” (Technopedia, 2018c)

Authorisation will allow a customer to access only their own data and staff to have access to the data that is relevant to their responsibilities in the enterprise. In an e-commerce

environment, for example, the warehouse staff needs to know about the orders of all the customers so that they can pack the right goods into the right packages. But they should not have access to the payment details that a user entered at checkout.

6.2.3 Secure Transfer

When sensitive data is being transferred between a server and a client, it is important that the data should be encrypted. This means that third parties cannot intercept and read information such as credit card numbers while it is being sent to the server. This can be accomplished using a Secure Sockets Layer (SSL) certificate to secure the connection using Transport Layer Security (TLS). (DigiCert, Inc., 2018)

An SSL certificate is issued by a trusted Certification Authority (CA). The certificate is signed by the CA (Rouse & Loshin, 2018), which makes it possible to determine whether the certificate has been tampered with.

6.3 Security Design Principles

Now that we have discussed some of the concepts in security, let us look at some of the most common principles for designing secure systems.

6.3.1 Least Privilege

The principle of least privilege states that one should grant a user the least possible permissions that are required. (Shostack, n.d.) Microsoft explained why this is a good idea in the Microsoft Windows Security Resource Kit:

“Always think of security in terms of granting the least amount of privileges required to carry out the task. If an application that has too many privileges should be compromised, the attacker might be able to expand the attack beyond what it would if the application had been under the least amount of privileges possible.” (Flores, et al., 2018)

The more access a user has, the more an attacker could do if that user's credentials are compromised.

6.3.2 Complete Mediation

The complete mediation principle states that authorisation must be checked for every access for all resources. (Merritt, 2013)

6.3.3 Balance with Usability

Also called the psychological acceptability principle. (Merritt, 2013) Security measures must be easy enough to comply with to be acceptable to users.

6.3.4 Economy of Mechanism

This principle states that systems should be designed to be simple since complex designs can cause accidental unauthorised access. (Merritt, 2013), (Shostack, n.d.)

6.3.5 Defend in Depth

Have multiple layers of security, so that an attacker must get through multiple defences before gaining access to the system. (Merritt, 2013)

6.4 Security Development Lifecycle

As we already saw in Learning Unit 1, security is something that should be considered throughout the full life cycle of an enterprise software system. Microsoft recognised this and developed their Security Development Lifecycle (SDL) in response – a process for developing more secure software. This process is well documented and can be applied successfully to projects outside of Microsoft.

Download both the .doc and the .xlsx of the *Simplified Implementation of the SDL* from <https://www.microsoft.com/en-us/download/details.aspx?id=12379> This document is an excellent, concise description of the SDL. Read through the section *Simplified SDL Security Activities*, particularly focussing on the mandatory activities.

Also read more about the SDL practices in (Microsoft Corporation, 2021).

6.5 Code Access Security (CAS)

If you look at Chapter 19 of (Howard & Lipner, 2006), you will notice that there is a long list of calls that have been banned from use due to their security implications. But these are all low-level C language functions, that you will probably never encounter as a C# developer. So, what does a C# developer need to know about security at the code level?

In the .NET Common Language Runtime (CLR), there is a feature called Code Access Security (CAS). This makes it possible to control how much access a managed .NET application has to the computer that it runs on. (Agarwal, 2013)

When an application is executed, the host that runs it is granted a set of permissions. For desktop applications, the default host has full permissions. But in other cases, the permissions can be configured. So, it is important to write your application in such a way that it will not try to use features that will not be available to it. (Wenzel, et al., 2017f)

For more information about how to use CAS, read (UB, 2004) and (eTutorials.org, 2018).

6.6 Securing WCF Services

For guidance and best practices when it comes to securing WCF Services, read (Perler, et al., 2017).

For instructions on how to secure services, read (Perler, et al., 2017b).

7 Services, Orchestration and Choreography

7.1 Introduction

Enterprise software systems are used to automate business processes. What is a business process?

“A business process is a collection of linked tasks which find their end in the delivery of a service or product to a client.” (Appian, 2018)

A business process can, for example, be how to handle a new customer requesting a fibre internet connection from an Internet Service Provider (ISP). First, the customer's details would get captured in a Customer Relationship Management (CRM) system. Then the ISP would have to request that a physical fibre line be installed by a third-party fibre provider. Once that is done, the billing system needs to be updated so that the client is billed for the service. Then the service can be activated, and the customer can finally connect to the internet.

Behind the scenes, there are a lot of different services that will get called during this process. There are two conceptual ways to handle this – using service orchestration or service choreography.

7.2 Service Orchestration

In service orchestration, there is a central manager that manages the whole process. A process like this is often called a workflow (see Figure 20).

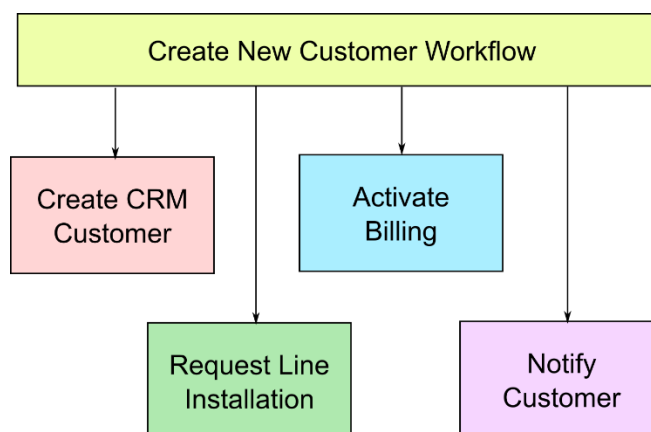


Figure 20: Service Orchestration

The advantages of using service orchestration are (Sehgal, 2018):

- A central view exists that shows the full flow of the process; and
- It is easy to track how long a process takes to complete, and whether service level agreements (SLAs) are met.

The drawbacks of using service orchestration are (Sehgal, 2018):

- When new states need to be added, it requires an update to the central flow; and
- It requires Business Process Management (BPM) software to manage the flow; and
- The central orchestrator can be a bottleneck and is a central point of failure.

7.2.1 Technologies

In the Microsoft eco system, there are two technologies for creating workflows: Windows Workflow Foundation (WF) and BizTalk. Read more in (Brown, 2008).

7.3 Service Choreography

In service choreography, all the interactions between services are in the form of messages or events. There is no central controller, and each service registers for the event or message that it can handle (see Figure 21).

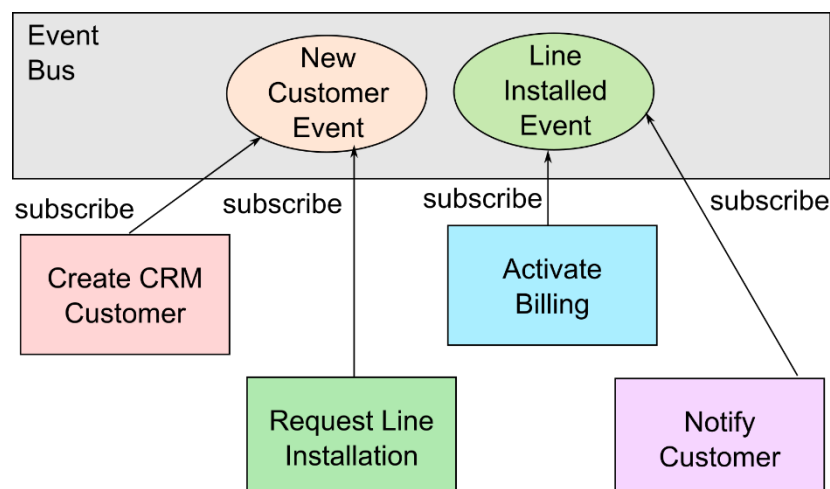


Figure 21: Service Choreography

The advantages of using service choreography are (Sehgal, 2018):

- Highly scalable;
- No central orchestrator to be a bottleneck or single point of failure; and
- Adding a new step to the process does not affect the rest of the flow.

The drawbacks of using service choreography are (Sehgal, 2018):

- There is no central view of the whole process; and
- Each microservice needs to do its own message/event processing.

In the example above, two subscribers react to the new customer event. Service choreography will handle this correctly, if the order in which the two steps happen is not important. If the CRM customer creation must happen first, and there is a chance that it might fail, then service orchestration would have been a better choice for this process.

7.3.1 Technologies

There are many ways in which the events or messages can be passed between services, for example using Microsoft Message Queuing (MSMQ).

7.4 Hybrid Solutions

In some cases, a hybrid between orchestration and choreography is necessary. Read more in (Bonham, 2017)

8 Portals

“A portal is a web-based platform that collects information from different sources into a single user interface and presents users with the most relevant information for their context.” (Liferay, 2021)

A portal is the starting point where an employee in an enterprise can gain access to all the information that is available across different systems in the organisation.

In the Microsoft eco system, Microsoft SharePoint Server is the product that offers portal functionality. See (Microsoft Corporation, 2018d) for more information.

9 Application Hosting

Once you have created an enterprise software system, the question remains as to where to host it. Let us look at the most common hosting options for enterprise software systems.

9.1 *Self-Hosted*

The first hosting possibility for a Windows Communication Foundation (WCF) service is self-hosting. This is accomplished by creating a console application or a Windows service that uses the `ServiceHost` class to host the service. (Green, 2018)

9.2 *Internet Information Services Express*

When developing web applications and services in Visual Studio, there is no need to have a full installation of Internet Information Services (IIS) on the developer computer. When running an app or service from Visual Studio, it will automatically use IIS Express instead if no IIS instance is configured. (Stackity, 2017)

IIS Express is great for development, but it is not intended to be used in production systems. There are performance and security limitations compared to full IIS, and it does not support File Transfer Protocol (FTP). (Stackity, 2017)

Read (Jana, 2014) for additional useful information about IIS Express.

9.3 *Internet Information Services*

Internet Information Services (IIS) can be used to host not only WCF services, but also web applications. IIS provide automatic activation as well as health monitoring of services. (Green, 2018b)

Read (Green, 2018b) for an example of how to do this.

9.4 *Windows Process Activation Service*

Another way to host a WCF service is by using Windows Process Activation Service (WAS). It is installed together with IIS but needs to be activated separately.

Read (Green, 2018b) for an example of how to do this.

9.5 Docker Container

“A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.” (Docker Inc., 2018)

This sounds a lot like microservices, which are standalone services that can be deployed independently. This is indeed one of the important use cases for hosting services in Docker containers. (Docker Inc., 2018b)

So, why choose to use a container at all? Some of the major reasons are (Lander, 2017):

- Consistency – Because the container includes everything that the application needs to run, it will work the same no matter where the container runs;
- Lightweight – Running a container does not use a lot of system resources; and
- Sharing – It is easy to distribute an image of your application to others.

Containers can be used for development, testing, and production. (Lander, 2017)

Docker started off as a Linux technology, and that meant that you could only use .NET Core with Docker. But now Windows Server 2016 and Windows 10 both support a container experience using Windows. This means that both .NET Core and .NET Framework can now be hosted in Docker containers. (Lander, 2017)

9.6 Azure Cloud

When deploying applications to the Microsoft Azure cloud, there are four main ways to deploy applications (Luijbregts, 2017):

1. Azure Service Fabric – Deploying an application in this way means that it will scale automatically. Setting up the application is however not trivial.
2. Virtual Machines – This gives you the most control over what is installed, but it also means that it takes the most effort to get up and running from scratch.
3. Containers – You can deploy applications that are already running in a container (including Docker).

4. Azure App Services – You can deploy various kinds of applications, taking advantage of capabilities such as automatic scaling and authentication. You do not need to worry about what is installed in terms of the environment, but that also means that you do not have any control over it.

10 Recommended Additional Reading

If you are interested in learning more about Microsoft Azure, have a look at the very comprehensive content available on Microsoft Learn: <https://docs.microsoft.com/en-us/learn/azure/> [Accessed 12 December 2022].

Read this very detailed article by Juval Lowy if you want to learn more about TransactionScope:

- Lowy, J. *Introducing System.Transactions in the .NET Framework 2.0*. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973865\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973865(v=msdn.10)?redirectedfrom=MSDN) [Accessed 12 December 2022].

11 Revision Exercises

11.1 Revision Exercise 1

Where can transactions be managed in an enterprise software system?

11.2 Revision Exercise 2

What is the difference between service orchestration and service choreography?

12 Solutions to Revision Exercises

12.1 Revision Exercise 1

Read Section 0 of this learning unit.

12.2 Revision Exercise 2

Read Section 7 of this learning unit.

Learning Unit 4: Optimising Application Performance

Learning Objectives:

- Discuss the performance implications of using Strings in loops.
- Use asynchronous calls to optimise application performance.
- Explain the performance implications of using reflection.
- Explain what n+1 queries are.
- Explain how to avoid n+1 queries.
- Explain the performance implications of using cursors.
- Explain how cross applies can be used to optimise performance.
- Differentiate between chunky services and chatty services.
- Use caching to improve application performance.
- Use lists optimally to avoid performance issues.

Material used for this learning unit:

- Microsoft Visual Studio 2019.
- *Fundamentals of Computer Programming with C#*, Chapter 13 (Nakov & Kolev, 2013).

How to prepare for this learning unit:

- Before you attend the lectures, read all sections of the learning unit in this module manual.

My notes

1 Introduction

When thinking about application performance optimisation, there are two kinds of metrics to consider: how long the code takes to execute and how much resources (such as memory) it uses while doing so. In many cases, there is a trade-off to be made between the two. For example, caching more information in random access memory (RAM) improves speed at the cost of using more memory.

It is important to identify which aspects of an application will have the greatest impact on performance and start with improving those aspects. A slow call to read a file from disk does not matter as much if it is done only once when the client application starts up. But saving even 10ms on a method that is called thousands of times will make a big difference.

In this learning unit, we will explore the most common issues that can cause bad performance. By just avoiding these few issues, the software that we write will already perform a lot better.

1.1 Measuring Execution Time

But before we jump into the details, let us consider for a moment how to accurately measure execution time in C#. This is very useful to study the impact of code in isolation.

```
Stopwatch stopwatch = Stopwatch.StartNew();  
String s = "";  
for (int i = 0; i < 5000; i++)  
{  
    s += "a";  
}  
stopwatch.Stop();  
Console.WriteLine("Operation took " +  
    stopwatch.ElapsedMilliseconds + "ms");
```

The output of this application indicates that it took 5ms. So, one iteration of the for loop happened in a very short period indeed. To get a more accurate measurement, it is useful to repeat the operation thousands of times. (Robert & Bell, 2011)

There is a lot of other things going on at the same time as the code that is being tested. In a non-real-time operating system, there is no guarantee exactly of how much of the resources are available at any point in time to the application. So, to be sure that the duration that is measured is statistically significant, make the measurement multiple times and calculate the average. Have a look at (Vladov, 2018) for a good code example of how to do this.

When comparing the performance of different ways to solve the same problem, make sure that apples are compared to apples. Accidentally including (or excluding) code can skew the results.

2 C# Topics

2.1 String Concatenation and StringBuilder

Simple string concatenation (using the + operator) in C# (and for that matter in Visual Basic (Soni, 2006) and Java too (Coosner, 2020)) is a slow process. (Kayal, 2016) It is not noticeable when two or three strings are concatenated, but when many strings are concatenated in a loop, the effect becomes very pronounced indeed.

A generally much faster approach is to use the StringBuilder class. So, let us try it out and see what the difference is. Here is the test code:

```
string toAppend = "abcdefghijklmnopqrstuvwxyz";
int numberOfLoops = 35000;
Stopwatch stopwatch = Stopwatch.StartNew();
string s1 = "";
for (int i = 0; i < numberOfLoops; i++)
{
    s1 += toAppend;
}
stopwatch.Stop();
Console.WriteLine(" Concatenation took " +
    stopwatch.ElapsedMilliseconds + "ms");

stopwatch.Restart();
StringBuilder builder = new StringBuilder();
for (int i = 0; i < numberOfLoops; i++)
{
    builder.Append(toAppend);
}
string s2 = builder.ToString();
stopwatch.Stop();

Console.WriteLine(" StringBuilder took " +
    stopwatch.ElapsedMilliseconds + "ms");
```

Here is the result of one such a run:

```
Concatenation took 12330ms  
StringBuilder took 1ms
```

That is a dramatic difference indeed – more than 12 seconds vs. around one thousandth of a second to perform the same operation!

What happens behind the scenes in string concatenation that is so much slower? The important point to note is that strings in C# are immutable. So, when doing the operation `str1 += str2`, a new string is created out of the two strings, and `str1` then points to the new string. (Herbie, 2005) `StringBuilder` on the other hand, was specifically designed to do this as efficiently as possible.

Because of all the strings that get created during string concatenation (and remember that with each iteration the strings are getting longer and longer), it also uses more memory than `StringBuilder`. (Kayal, 2018) They will get garbage collected eventually, but in the meantime, they take up a lot of space. And thus, it makes sense that the garbage collector also works harder when using concatenation. (Herbie, 2005)

The interesting point that Dr Herbie noticed is that for a small number of strings, string concatenation is faster. (Herbie, 2005) But 2005 is a long time ago, so let us test it out:

```
string toAppend = "abcdefghijklmnopqrstuvwxy";  
int numberOfLoops = 10;  
Stopwatch stopwatch = Stopwatch.StartNew();  
string s1 = "";  
for (int i = 0; i < numberOfLoops; i++)  
{  
    s1 += toAppend;  
}  
stopwatch.Stop();  
Console.WriteLine(" Concatenation took " +  
    stopwatch.ElapsedTicks + " ticks");  
  
stopwatch.Restart();  
StringBuilder builder = new StringBuilder();  
for (int i = 0; i < numberOfLoops; i++)  
{  
    builder.Append(toAppend);  
}  
string s2 = builder.ToString();  
stopwatch.Stop();  
  
Console.WriteLine(" StringBuilder took " +  
    stopwatch.ElapsedTicks + " ticks");
```

Now we must measure the elapsed time in ticks, to measure accurately enough. And here is the result:

```
Concatenation took 10 ticks  
StringBuilder took 15 ticks
```

Concatenation is indeed faster. Why? Because `StringBuilder` is a complex object to instantiate. So below a certain threshold, it is faster to use concatenation. (Herbie, 2005)

Even with a small set though, the garbage collector must work harder with concatenation than with the `StringBuilder`. (Herbie, 2005)

If the process needs to be repeated multiple times in the same run of the application, reuse the `StringBuilder` instance instead of creating a new one. That will be more memory efficient. (Microsoft Corporation, 2018e)

For more interesting points about string handling in C#, read (Hare, 2010).

2.2 *Async*

Asynchronous programming using the `async` and `await` keyword can be very useful. If you need to make multiple service calls that each take a second or two to complete, for example, these can be done in parallel using `async/await`, instead of having to do them one by one. This means that the result can be displayed to the user a few seconds earlier than it would have been.

But there are two points to consider when using `async` in this way. The first is that having to do multiple service calls might be a sign of sub-optimal service design (see Section 4 of this learning unit). So, first consider whether the services can be changed to require fewer calls, before using `async` to mask a different problem elsewhere in the system.

The second point is that `async` and `await` comes with a performance penalty, since it adds overhead to the process. (Teplyakov, 2018) (Kalapos, 2014) Each `await` operation takes around 4µs and it also allocates around 300 bytes of memory. This is negligible in comparison if the method that is called takes a second or two to complete. But if the method is very fast, the overhead will be noticeable. (Teplyakov, 2018)

Read (Teplyakov, 2018) for more information about optimising performance with `async` and `await`.

2.3 Reflection

Reflection provides a lot of possibility in terms of flexibility, since it allows for reading type metadata programmatically. (Kanjilal, 2016) Read (Kanjilal, 2016) if you want to brush up on the details of how reflection works.

But reflection is notoriously slow. So, let us experiment with calling a method using reflection. Here is the very simple class with a single method that we will be using:

```
class MyClass
{
    public int CallMe()
    {
        return 0;
    }
}
```

Since the CallMe method does not take any parameters, calling it using reflection is as simple as possible.

Side note: There is some overhead associated with calling a method. In (Kayal, 2018b) we see that doing a single Console.WriteLine() is almost 10 times slower than calling it without the method. So even though the method just returns a hardcoded value, we will still see that it takes a bit of time to run.

```
MyClass myObject = new MyClass();
int numberOfRepeats = 10000;

Stopwatch stopwatch = Stopwatch.StartNew();
for (int i = 0; i < numberOfRepeats; i++)
{
    myObject.CallMe();
}
stopwatch.Stop();
Console.WriteLine(" Direct call completed in " +
    stopwatch.ElapsedTicks + " ticks");

object myOtherObject = new MyClass();

stopwatch.Restart();
for (int i = 0; i < numberOfRepeats; i++)
{
    myObject.GetType().GetMethod("CallMe").
        Invoke(myObject, null);
}
stopwatch.Stop();

Console.WriteLine(" Reflective call completed in " +
    stopwatch.ElapsedTicks + " ticks");
```



```
Direct call completed in 1741 ticks
Reflective call completed in 22321 ticks
```

We can see that reflection is a lot slower than direct access for methods. But there is a way to improve the performance. Let us add another test where we get the method once and reuse it:

```
stopwatch.Restart();
MethodInfo method = myObject.GetType().GetMethod("CallMe");
for (int i = 0; i < numberOfRepeats; i++)
{
    method.Invoke(myObject, null);
}
stopwatch.Stop();

Console.WriteLine(" Reflective call with cache completed in " +
    stopwatch.ElapsedTicks + " ticks");
```

```
Direct call completed in 1152 ticks
Reflective call completed in 26486 ticks
Reflective call with cache completed in 15459 ticks
```

That is much better but still a lot slower than the direct access way.

Reflection is powerful and can certainly be worthwhile in many cases. But when considering whether to use reflection, remember about the performance implications.

For more information read (Warren, 2016).

3 Databases and Entity Framework

3.1 *n+1 Queries*

In Entity Framework (EF), as in object relational mappers (ORMs) in general, the *n+1* query a significant performance issue. In fact, it can be considered an anti-pattern. (Rump, 2018)

The *n+1* refers to reading one entry from the database, and then iterating through *n* children related to that parent entity. And doing a database query for each of the *n* children. (Fink, 2010)

For this experiment, we will use two entities – a Customer that can have many Orders.

```
public class CustomerData : DbContext
{
    public CustomerData()
        : base("name=CustomerData")
    {
    }

    protected override void OnModelCreating(
        DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Order>()
            .HasRequired<Customer>(s => s.Customer)
            .WithMany(g => g.Orders)
            .HasForeignKey<int>(s => s.CustomerId);
    }

    public virtual DbSet<Customer> Customers { get; set; }
    public virtual DbSet<Order> Orders { get; set; }
}

public class Customer
{
    public Customer()
    {
        Orders = new List<Order>();
    }

    public int Id { get; set; }
    public string Name { get; set; }
    public virtual List<Order> Orders { get; set; }
}

public class Order
{
    public int Id { get; set; }
    public string Description { get; set; }
    public int CustomerId { get; set; }
    public virtual Customer Customer { get; set; }
}
```

How do we create the n+1 problem?

```

using (CustomerData context = new CustomerData())
{
    Stopwatch stopwatch = Stopwatch.StartNew();
    context.Database.Log = s =>
        System.Diagnostics.Debug.WriteLine(s);

    var customers = context.Customers.ToList();
    foreach (Customer c in customers)
    {
        foreach (Order o in c.Orders)
        {
            Console.WriteLine("orderid: " + o.Description);
        }
    }
    stopwatch.Stop();
    Console.WriteLine("n+1 queries took " +
        stopwatch.ElapsedMilliseconds + " ms");
}

```

To see what queries are performed on the database, the database log output is directed to the diagnostics console in this line:

```

context.Database.Log = s =>
    System.Diagnostics.Debug.WriteLine(s);

```

By default, lazy loading is on in EF. So, when we call

```
var customers = context.Customers.ToList();
```

only the list of customers is queried from the database. Looking at the output window in Visual Studio, we see the following query that is executed:

```

SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Name] AS [Name]
FROM [dbo].[Customers] AS [Extent1]

```

So far so good. This is the one in the n+1. But then in the orders loop, one query is called to get the orders for each of the customers. Here is the first one:

```

SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Description] AS [Description],
    [Extent1].[CustomerId] AS [CustomerId]
FROM [dbo].[Orders] AS [Extent1]
WHERE [Extent1].[CustomerId] = @EntityKeyValue1

-- EntityKeyValue1: '1' (Type = Int32, IsNullable = false)

```

... and the 571st one...

```

SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Description] AS [Description],
    [Extent1].[CustomerId] AS [CustomerId]
FROM [dbo].[Orders] AS [Extent1]
WHERE [Extent1].[CustomerId] = @EntityKeyValue1

-- EntityKeyValue1: '571' (Type = Int32, IsNullable = false)

```

How can this situation be avoided? It is easy to do with EF:

```

var customers = context.Customers.Include("Orders").ToList();
foreach (Customer c in customers)
{
    foreach (Order o in c.Orders)
    {
        Console.WriteLine("orderid: " + o.Description);
    }
}

```

What gets executed now? Just a single query, although a slightly more complex one:

```

SELECT
    [Project1].[Id] AS [Id],
    [Project1].[Name] AS [Name],
    [Project1].[C1] AS [C1],
    [Project1].[Id1] AS [Id1],
    [Project1].[Description] AS [Description],
    [Project1].[CustomerId] AS [CustomerId]
FROM ( SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Name] AS [Name],
    [Extent2].[Id] AS [Id1],
    [Extent2].[Description] AS [Description],
    [Extent2].[CustomerId] AS [CustomerId],
    CASE WHEN ([Extent2].[Id] IS NULL) THEN CAST(NULL AS int) ELSE 1 END AS [C1]
    FROM [dbo].[Customers] AS [Extent1]
    LEFT OUTER JOIN [dbo].[Orders] AS [Extent2] ON [Extent1].[Id] = [Extent2].[CustomerId]
) AS [Project1]
ORDER BY [Project1].[Id] ASC, [Project1].[C1] ASC

```

So, what is the performance difference between the two? The table below shows the average time over 10 runs of the code in each case, running against a local database.

	N+1	Single Query
11 Customers	489 ms	442 ms
1011 Customers	4 872 ms	994 ms

If these queries were running against a remote database server, the time difference would have been even more pronounced. This is because the queries and returned data would have been transmitted over a network instead of across a local connection.


Side note: Calling a service n+1 times performs even worse.

3.2 Cursors vs Cross Apply


In Structured Query Language (SQL), a cursor can be used to iterate over table entries, row by row. It can be tempting to use for developers that are not used to the set-based paradigm of SQL. But the fact of the matter is that SQL is optimised for set-based operations. So, cursors are far slower. (Hibbert, 2016)

For this investigation, we will use two tables:

Customers

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>	

Orders

	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	Description	nvarchar(MAX)	<input checked="" type="checkbox"/>	
	CustomerId	int	<input type="checkbox"/>	
	Date	date	<input checked="" type="checkbox"/>	

These are the tables created by Entity Framework (EF) in Section 3.1, with the Date field added to Orders.

Let us look at a silly example that uses a cursor to iterate through the customers, to find the Id of the order with the first date.

```

declare @id int, @name nvarchar(max), @firstorderid int

declare customer_cursor cursor for
select Customers.id, Customers.Name
from Customers;

open customer_cursor

fetch next from customer_cursor
into @id, @name

while @@FETCH_STATUS = 0
begin
    select top 1 Customers.Id, Customers.Name, Orders.Id as FirstOrderId
    from Customers
    left join Orders on Customers.Id = Orders.CustomerId
    where Customers.Id = @id
    order by Orders.Date;

    fetch next from customer_cursor
    into @id, @name
end
close customer_cursor;
deallocate customer_cursor;

```

This took around nine seconds to execute with 150 records in the customer table. With 1011 customers, it took almost a minute to complete.

How can the same be done in the set-based paradigm in SQL Server? By using an apply operator.

```

select Customers.id, Customers.Name,
       FirstOrder.Id as FirstOrderId
from Customers
outer apply
(
    select top 1 Orders.Id, Orders.Description
    from Orders
    where Orders.CustomerId = Customers.Id
    order by Orders.Date
) FirstOrder;

```

With the original 1011 customers from the previous test data set, this outer apply query completed in less than a second. So, it is very clear that cursors are very slow indeed and should be used with great care.

Note: The apply operators were first introduced by SQL Server and is not supported by all other database engines. However, there is an equivalent in standard SQL – lateral join. (Buen, 2013)

Read more about how cross apply and outer apply works in (Richardson, 2018).

Read (Hibbert, 2016) for a great example of how to replace a cursor that updates records with a cross apply using a table value function.

4 Service Design

4.1 *Chunky vs Chatty Services*

Chatty services (also called fine grained services) require the service consumer to make several calls to get the desired data. Chunky services (or coarse grained services) often require only one call to get the desired data. All services are on a continuum between chatty and chunky. (Gotsch, 2018)

Sequentially calling chatty services to get the same data that could have been supplied by a single chunky service is slower. (University of Washington, 2010)

So, what is better – chunky or chatty services? The answer is that it depends on the needs of the service consumers. A service consumer should be able to get the desired data with as few calls as possible. (Gotsch, 2018) But only the desired data, not more than that.

If the service consumer only ever wants to get the personal information for a single customer, then including all the details of all the orders for the customer in the response too makes the service too chunky. The consumer must wait for the large response to get compiled and transmitted, all to just use a few fields in the customer record and discard the rest.

On the other hand, if the service consumer needs to get the orders for a consumer, the service would be considered too chatty if the orders need to be loaded one by one, making a service call for each one. This is like the $n+1$ query problem.

The optimal service granularity is a compromise between chatty and chunky. (Gotsch, 2018)

4.2 Caching

“Caching is a common technique that aims to improve the performance and scalability of a system. It does this by temporarily copying frequently accessed data to fast storage that’s located close to the application. If this fast data storage is located closer to the application than the original source, then caching can significantly improve response times for client applications by serving data more quickly.” (Narumoto, et al., 2017b)

Data that is read often but changes infrequently is the best candidate for caching. (Narumoto, et al., 2017b)

In our example of customers that have orders, those orders would typically have an order status. It could be something like awaiting payment or shipped. Typically, information like this would be stored in a table with Id and Description columns. Something like this:

OrderStatus

	Name	Data Type	Allow Nulls	Default
PK	Id	int	<input type="checkbox"/>	
	Description	nvarchar(MAX)	<input checked="" type="checkbox"/>	

```
public class OrdersModel : DbContext
{
    public OrdersModel()
        : base("name=OrdersModel")
    {
        Database.SetInitializer(new StatusInitializer());
    }

    public virtual DbSet<OrderStatus> OrderStatuses { get; set; }
}

public class StatusInitializer : DropCreateDatabaseAlways<OrdersModel>
{
    protected override void Seed(OrdersModel context)
    {
        context.OrderStatuses.Add(
            new OrderStatus() { Description = "Awaiting Payment" });
        context.OrderStatuses.Add(
            new OrderStatus() { Description = "In Progress" });
        context.OrderStatuses.Add(
            new OrderStatus() { Description = "Packing" });
        context.OrderStatuses.Add(
            new OrderStatus() { Description = "Shipped" });
        base.Seed(context);
    }
}

public class OrderStatus
{
    public int Id { get; set; }
    public string Description { get; set; }
}
```


Let us create a very simple Windows Communication Foundation (WCF) service that reads all the possible statuses from the database.

```
public class OrdersService : IOrdersService
{
    public IEnumerable<OrderStatus> GetOrderStatuses()
    {
        using (OrdersModel context = new OrdersModel())
        {
            context.Database.Log = s =>
                System.Diagnostics.Debug.WriteLine(s);
            return context.OrderStatuses.ToList();
        }
    }
}
```

Now if we run the service and invoke it multiple times using the WCF Test Client, we will see that the below query is executed every time the service is called:

```
SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Description] AS [Description]
FROM [dbo].[OrderStatus] AS [Extent1]
```

But the data is the same every time since this is data that will rarely change! So how can this information be cached, instead of reading it from the database all the time? By using an ObjectCache. (De Rycke, 2012)

```
public class OrdersService : IOrdersService
{
    private const string StatusesKey = "orderStatuses";

    public IEnumerable<OrderStatus> GetOrderStatuses()
    {
        using (OrdersModel context = new OrdersModel())
        {
            context.Database.Log = s =>
                System.Diagnostics.Debug.WriteLine(s);

            ObjectCache cache = MemoryCache.Default;

            if (cache.Contains(StatusesKey))
                return (IEnumerable<OrderStatus>)cache.Get(StatusesKey);
            else
            {
                IEnumerable<OrderStatus> statuses =
                    context.OrderStatuses.ToList();
                CacheItemPolicy cacheItemPolicy =
                    new CacheItemPolicy();
                cacheItemPolicy.AbsoluteExpiration =
                    DateTime.Now.AddHours(1.0);
                cache.Add(StatusesKey, statuses, cacheItemPolicy);
                return statuses;
            }
        }
    }
}
```

A reference to `System.Runtime.Caching` is required for this.

Now if we invoke the service multiple times, the query will get executed only the first time. And the data will only get read from the database again after an hour.

But what if we have a lot of different services that we want to use caching with? We can define a custom WCF behaviour, that can be applied to all our services.

What is an operation behaviour in WCF? It is code that gets executed before and after our service code. Remember when we looked at transactions in WCF in learning unit 3. The operation behaviour attribute where we specify whether a `TransactionScope` is required is an example of this.

```
public class Service1 : IUsefulService
{
    [OperationBehavior(
        TransactionAutoComplete = true,
        TransactionScopeRequired = true
    )]
    public string Method1(int value)
    {
        throw new NotImplementedException();
    }
}
```

If we define our own service behaviour as an attribute, we could reduce the code for our service to this:

```
public class OrdersService : IOrdersService
{
    [CachingOperationBehaviour(60)]
    public IEnumerable<OrderStatus> GetOrderStatuses()
    {
        using (OrdersModel context = new OrdersModel())
        {
            context.Database.Log = s =>
                System.Diagnostics.Debug.WriteLine(s);

            return context.OrderStatuses.ToList();
        }
    }
}
```

The service code is so much cleaner and the caching code is created and maintained in a single place.

To define a new operation behaviour, we must implement the `IOperationBehaviour` interface. And to make it possible to use it as an attribute, we must also inherit from `Attribute`. Here is the behaviour class:

```
public class CachingOperationBehaviour : Attribute, IOperationBehavior
{
    private readonly int timeout;

    public CachingOperationBehaviour(int timeoutInMinutes)
    {
        timeout = timeoutInMinutes;
    }

    public void ApplyDispatchBehavior(OperationDescription operationDescription,
        DispatchOperation dispatchOperation)
    {
        dispatchOperation.Invoker = new CachingOperationInvoker(
            dispatchOperation.Invoker, dispatchOperation, timeout);
    }

    public void AddBindingParameters(OperationDescription operationDescription,
        BindingParameterCollection bindingParameters)
    {
    }

    public void ApplyClientBehavior(OperationDescription operationDescription,
        ClientOperation clientOperation)
    {
    }

    public void Validate(OperationDescription operationDescription)
    {
    }
}
```

Notice that our new behaviour only does something for `ApplyDispatchBehaviour`. This will get called whenever the service is invoked. Read more about the behaviours in (Boursi, 2010).

Now all that remains is the implementation of the invoker. Add a reference to `Newtonsoft.Json`.

```

internal class CachingOperationInvoker : IOperationInvoker
{
    private IOperationInvoker invoker;
    private readonly string operationName;
    private readonly int timeout;

    public CachingOperationInvoker(IOperationInvoker invoker,
        DispatchOperation dispatchOperation, int TimeoutInMinutes)
    {
        this.invoker = invoker;
        this.operationName = dispatchOperation.Name;
        this.timeout = TimeoutInMinutes;
    }

    public ObjectCache CacheProvider
    {
        get { return MemoryCache.Default; }
    }

    public bool IsSynchronous => invoker.IsSynchronous;

    public object Invoke(object instance, object[] inputs, out object[] outputs)
    {
        string payload = JsonConvert.SerializeObject(inputs);
        string key = instance.GetType().ToString() + ":" + operationName +
            "-" + payload;

        Tuple<object, object[]> cacheItem = CacheProvider.Get(key) as
            Tuple<object, object[]>;
        if (cacheItem != null)
        {
            outputs = cacheItem.Item2;
            return cacheItem.Item1;
        }
        else
        {
            CacheItemPolicy cacheItemPolicy = new CacheItemPolicy()
            { AbsoluteExpiration = DateTimeOffset.Now.AddMinutes(timeout) };
            object result = invoker.Invoke(instance, inputs, out outputs);
            CacheProvider.Add(new CacheItem(key,
                new System.Tuple<object, object[]>(result, outputs)),
                cacheItemPolicy);
            return result;
        }
    }

    public object[] AllocateInputs()
    {
        return invoker.AllocateInputs();
    }

    public IAsyncResult InvokeBegin(object instance, object[] inputs,
        AsyncCallback callback, object state)
    {
        return invoker.InvokeBegin(instance, inputs, callback, state);
    }

    public object InvokeEnd(object instance, out object[] outputs,
        IAsyncResult result)
    {
        return invoker.InvokeEnd(instance, out outputs, result);
    }
}

```

The invoke method checks whether it has the response cached first before invoking the actual service.

Since .NET Framework 4.6.1, the caching mechanism that was already available in ASP.NET can also be used for HTTP GET requests on WCF HTTP services. Read (Wenzel, et al., 2017g) for more information.

4.3 Lists

It can happen that a service consumer needs to query multiple entities by their identifiers. For example, if a client needs to get order numbers 4, 8 and 52; one way of solving this would be for the service consumer to call a `GetOrderByID` service multiple times – one for each order. But that is far from optimal, since both the service and the database is called multiple times.

So, in this case, provide a service that allows the caller to pass multiple IDs to search for:

```
public class OrdersService : IOrdersService
{
    public List<Order> GetOrdersByIds(List<int> ListOfIDs)
    {
        using (CustomerData context = new CustomerData())
        {
            return context.Orders.Where(
                o => ListOfIDs.Contains(o.Id)).ToList();
        }
    }
}
```

This will mean that only one call needs to be made to the service and only one database call is made by the service too.

5 Recommended Additional Reading

For more information about the performance of C# code, read (Kayal, 2016), (Kayal, 2018), (Kayal, 2018b) and (Kayal, 2018c).

For more details about SQL performance, read (Fritchey, 2012).

For more information about REST API design, have a look at (Subramaniam, 2014).

For examples of how to use output caching in ASP.NET MVC, read (Anderson, et al., 2009).

6 Revision Exercises

6.1 *Revision Exercise 1*

What is the most performant way to concatenate strings in loops?

6.2 *Revision Exercise 2*

Are cursors the recommended way to update records in a database? Why/Why not?

7 Solutions to Revision Exercises

7.1 Revision Exercise 1

Read Section 2.1 of this learning unit.

7.2 Revision Exercise 2

Read Section 3.2 of this learning unit.

Learning Unit 5: Methodologies and Architecture Frameworks

Learning Objectives:

- Contrast various common software development methodologies.
- Explain what DevOps is.
- Explain what continuous integration and continuous delivery is.
- Discuss the challenges involved in modelling large enterprises.
- Compare ITIL, TOGAF and the Zachman framework.

Material used for this learning unit:

- Internet access.

How to prepare for this learning unit:

- Before you attend the lectures, read all sections of the learning unit in this module manual.

My notes

1 Introduction

When only one developer works on a small software development project, there is no need to follow any special processes. But this is very rarely the case in real world projects. When the project is large and many different people fulfilling different roles participate, the need arises to have a common methodology. That way everybody can know what is expected of them and what they can expect from the process in return.

In this learning unit, we will explore several topics related to how software is developed and managed in large enterprises.

2 Software Development Methodologies

A software development methodology is “a framework that is used to structure, plan, and control the process of developing an information system.” (Vskills, 2018)

Many different methodologies exist, each with its own advantages and disadvantages. To a large extent, newer methodologies were created in response to the drawbacks of older ones.

We will cover several important methodologies here, but this is by no means an exhaustive list of all the methodologies.

2.1 Waterfall

The Waterfall methodology was invented in 1970. It was based on the work of Henry Ford on his car assembly lines in 1913. (Sacolick, 2018)

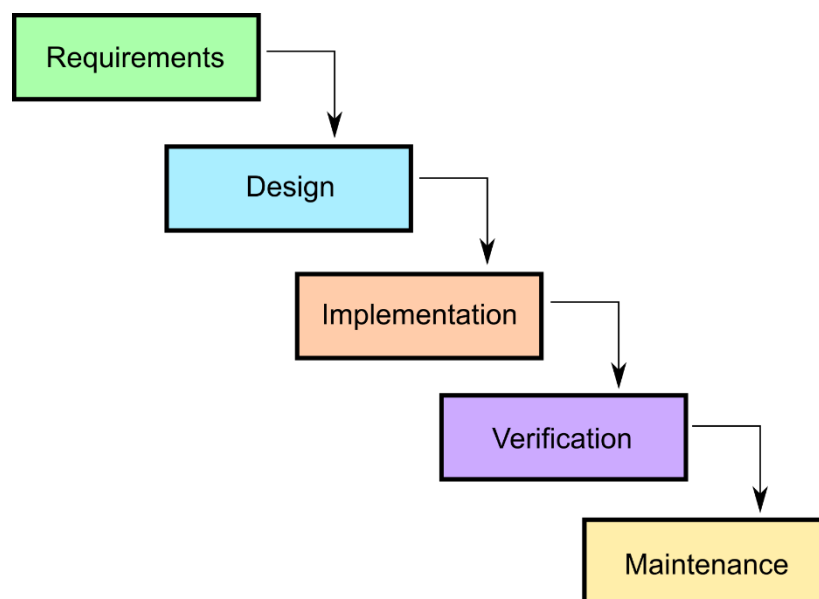


Figure 22: Waterfall Methodology

In the Waterfall methodology, the different phases (see Figure 22) in the software development lifecycle (SDLC) are followed linearly. Each phase must be completed before moving on to the next one. (Adobe, 2021) All the requirements are gathered up front and documented in detail. Once the specification is set in stone, the software can be designed. And so on.

Waterfall works well with traditional project management, where everything is planned up front and tightly managed.

This is the same process that is used in engineering projects such as bridge construction. (Roseke, 2012) It is a very sensible methodology for something like a bridge, where the requirements are fixed, and you get only one shot at building the right thing. Getting halfway with building the bridge only to realise that it will not be able to carry the required weight is an expensive mistake to fix.

When Waterfall was first introduced, software was very different from what it is today. Things changed slowly and the computers and their software were large and complex. (Sacolick, 2018) Today, everything changes rapidly, including requirements.

Even though Waterfall is generally considered to be outdated, there are still cases where it is used successfully. For small projects (that take less than 100 hours of development), Waterfall is a simple and clear process to follow. (Kuprenko, 2017)

The aerospace and defence industries also still heavily rely on the Waterfall process. In this instance, the motivation behind using the Waterfall process is the safety requirements of the software that is developed. Imagine, for example, how dire the consequences could be if the navigation system on board a passenger liner malfunctioned.

2.1.1 Advantages

- The methodology is straightforward to learn and apply. (TatvaSoft, 2015)
- The projects are easy to manage because the methodology is so inflexible. (TatvaSoft, 2015)
- Detailed documentation is produced, which will enable new developers to get all the information that is required to get started. (Lvivity, 2018)
- The completion date and total cost can be determined up front. (Lvivity, 2018)

2.1.2 Disadvantages

- Waterfall does not work if the requirements are not well understood up front or are likely to change. (TatvaSoft, 2015)
- The process has very little flexibility, making it hard and expensive to cope with changes. (Lvivity, 2018)
- Functional software is only available at the end of the project. (TatvaSoft, 2015)
- The client is not involved in the development, and only gets to use the product when it is complete. (Lvivity, 2018)

2.2 Rational Unified Process

The Rational Unified Process (RUP) was first described under this name in 1998, by the company Rational. Rational was later acquired by IBM. (Kruchten, 2004)

In RUP, there are also phases like in Waterfall. And the four phases (inception, elaboration, construction, and transition) take place in order. But unlike Waterfall, this process is completed in multiple iterations. Each iteration only deals with a part of the larger system, and functional software is delivered at the end of each iteration. (Ambler, 2005)

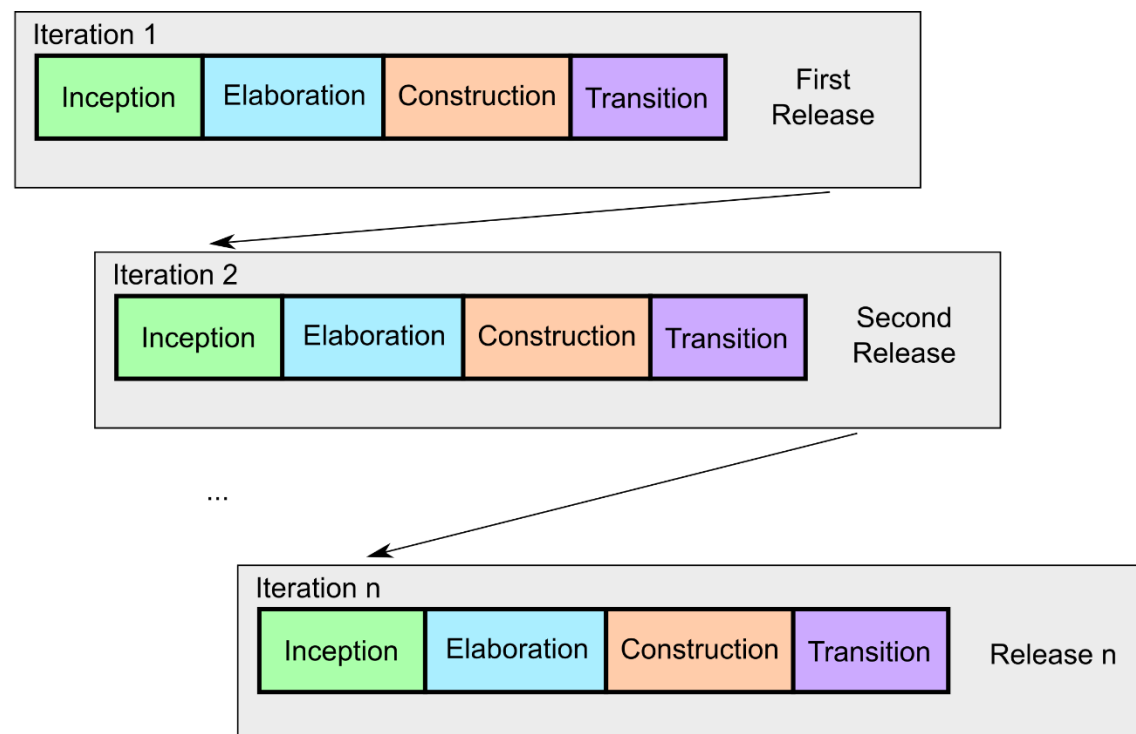


Figure 23: RUP Methodology

2.2.1 Advantages

- Exact documentation is created during the process. (Powell-Morse, 2017b)
- Stakeholders can be frequently informed of progress, and they can see the software early on. (Ambler, 2005)
- The process is flexible enough to handle changes in the requirements. (Powell-Morse, 2017b)
- Integration occurs continuously instead of at the end of the whole project. (Powell-Morse, 2017b)

2.2.2 Disadvantages

- RUP is a complex methodology that can be daunting to learn. (Powell-Morse, 2017b)
- Tasks are assigned to specific team members, and each team member therefore needs to be an expert to ensure that the right artefacts are produced. (Powell-Morse, 2017b)
- RUP is in some cases still not flexible enough. (Green, 2016)
- The focus on continuous integration can cause confusion during the testing stages. (TatvaSoft, 2015)

2.3 *Agile Manifesto*

With the introduction of the internet, the software development landscape changed. Innovation and the ability to keep up with rapidly changing demands became more and more important. Out of this, in 2001, the agile manifesto was born. (Sacolick, 2018)

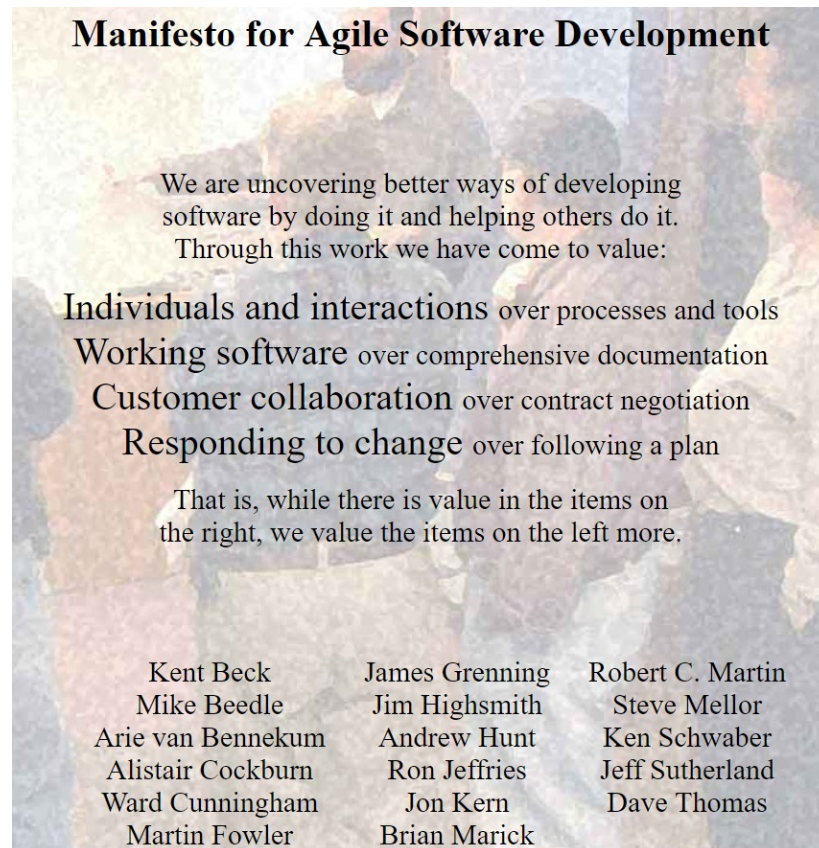


Figure 24: Agile Manifesto from <http://agilemanifesto.org/>

This section is the odd one out amongst the methodologies, since agile is not actually a software development methodology. Agile consists of the set of four values shown in Figure 24, as well as 12 principles (see Figure 25). But agile development does not actually prescribe any practices, and so is not a methodology. (Gratis, 2018)

There are, however, several methodologies that are agile, that all strive to adhere to the values and principles in their own unique ways. Examples of agile methodologies are:

- Extreme Programming (TatvaSoft, 2015);
- Scrum (Tyagi, 2020);
- Kanban (Radigan, 2018); and
- Feature-Driven Development (FDD) (Synopsis, 2017).

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 25: Agile Principles from
<http://agilemanifesto.org/principles.html> [Accessed 12
December 2022].

2.4 Extreme Programming

Extreme Programming (XP) is the first agile methodology that we will touch on. It was first used at Chrysler in the mid-1990s. (Agile Alliance, 2018b) At its core, XP has 12 practices that are all taken to the extreme (see Figure 26). (Kuprenko, 2017)

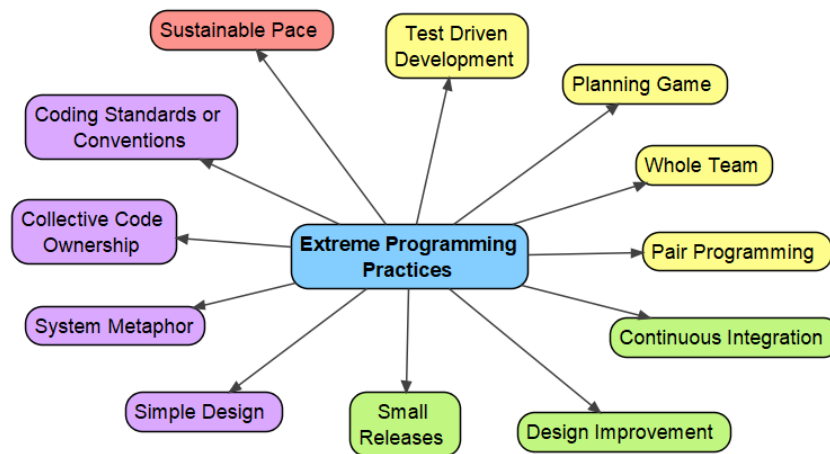


Figure 26: Extreme Programming Practices

XP also has its own set of values: “communication, simplicity, feedback, courage, and respect”. (Agile Alliance, 2018b)

XP is meant to be used in a small team that works in the same physical office, using a technology that can have automated unit and functional tests. It is ideal in the situation where requirements change often. (Agile Alliance, 2018b)

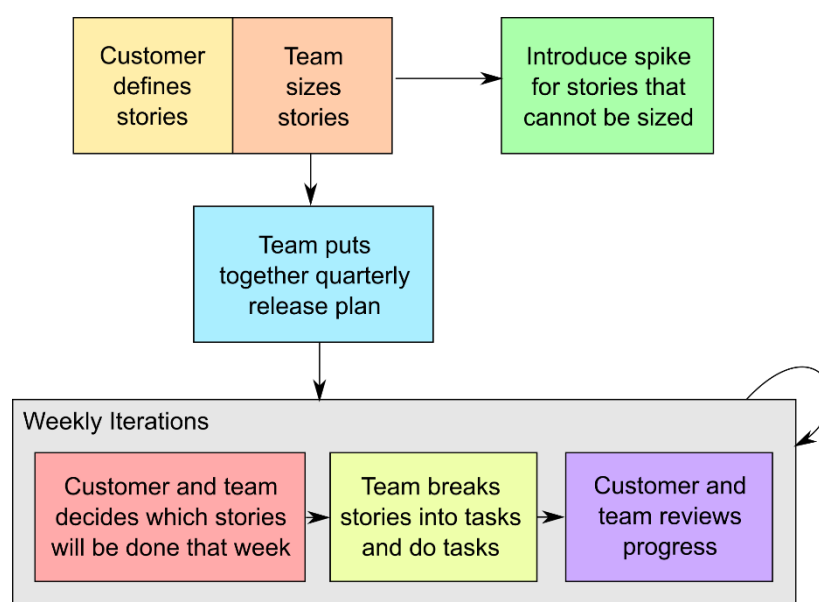


Figure 27: Extreme Programming Practices

2.4.1 Advantages

- XP works well when productivity needs to be high. (Green, 2016)
- Customers are very involved in XP projects, allowing them to contribute their opinions early. (TatvaSoft, 2015)
- The developers on an XP project personally commit to the schedules. (TatvaSoft, 2015)

2.4.2 Disadvantages

- Not all people can work in an environment that is so unstructured. (Green, 2016)
- XP requires lots of face-to-face meetings with the customer, which can be expensive. (TatvaSoft, 2015)
- The full project scope is not known, making it impossible to provide a price for the whole project. (TatvaSoft, 2015)

2.5 Scrum

The initial concepts of Scrum were described in 1986, and it has been refined several times. (Krishnamurthy, 2012)

Scrum consists of principles, aspects, and processes. The principles are mandatory for all Scrum projects (SCRUMstudy, 2017):

1. “Empirical Process Control”;
2. “Self-organisation”;
3. “Collaboration”;
4. “Value-based Prioritisation”;
5. “Time-boxing” and
6. “Iterative Development”.

The five Scrum aspects are elements that are managed through Scrum: organisation, business justification, quality, change and risk. (SCRUMstudy, 2017)

The basic flow of a Scrum project is shown in Figure 28. In Scrum, the product owner plays an important role. A product owner is the business user of the system that is developed, who can define and prioritise the business requirements. A prioritised product backlog is then created, containing all the requirements of the system. Out of that, several stories are picked for completion during a sprint. A sprint is a time-boxed period of

development, usually between two and six weeks in duration. (SCRUMstudy, 2017)

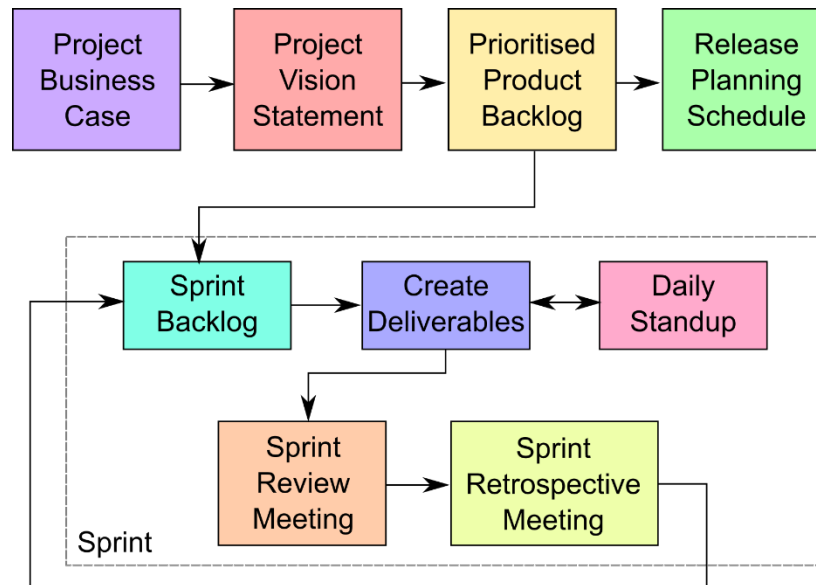


Figure 28: Scrum Flow

Every day during a sprint, there is a daily stand-up meeting of maximum 15 minutes, where the team reports on progress. At the end of the sprint, there is a review meeting where the system is demonstrated to the product owner and accepted if criteria are met. At this point, the system will usually be deployed. (SCRUMstudy, 2017)

The last meeting in a sprint is the retrospective. Here the team asks themselves what worked well, and what did not work well during the sprint. And typically, a single thing is chosen to change to the process for the next sprint. This encourages continuous process improvement. (SCRUMstudy, 2017)

2.5.1 Advantages

- Features with the highest business value is delivered first. (SCRUMstudy, 2017)
- Large projects are broken up into more manageable chunks. (Das, 2018)
- The product owner and other stakeholders get to provide feedback early and often. (Das, 2018)

2.5.2 Disadvantages

- Since there is no specific end date, scope creep can be introduced. (Das, 2018)
- Losing a team member will have a big impact on the project. (Das, 2018)

2.6 Kanban

Kanban has an incredibly long history, considering that it is an agile methodology. The basic concepts behind Kanban were first used by Toyota in the 1940s on the factory floor. While there have certainly been refinements to the process since then, the just in time (JIT) concept is still at the heart of Kanban. (Radigan, 2018)

Kanban is based on three principles (CollabNet, 2018):

- Graphically show what you are currently working on;
- Only work on a small amount of work at a time; and
- When one task is finished, take the next one from the backlog.

In Kanban, tasks are displayed on a Kanban board (see Figure 29 for an example). Tasks move from left to right through the board. In this example, the “In Progress” column is limited to only two tasks, one for each of the two developers working on the project. This helps to avoid taking on too much work at the same time. (Radigan, 2018)

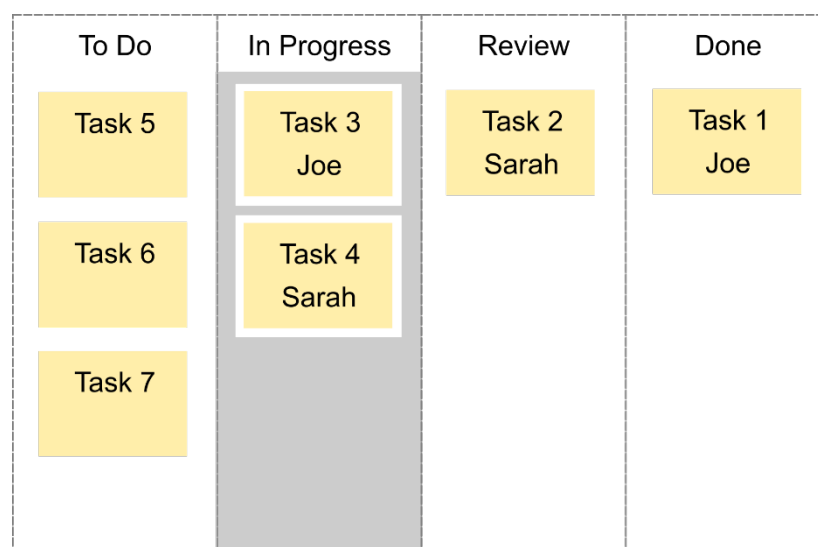


Figure 29: Kanban Board

Kanban has a lot in common with Scrum, but there are some important differences. In Kanban, there are no sprints. So, the workflow and delivery schedule are continuous. No specific roles are defined and change to requirements can happen at any time. (Radigan, 2018)

When transitioning from a Waterfall approach to agile, it is easier to change to Kanban than to Scrum. Since Waterfall teams are used for continuous development, although with no deliverables until the end of the project, it is easier to transition to the continuous development and delivery model of Kanban.

2.6.1 Advantages

- Kanban is an extreme flexible methodology. (Yodis, 2016)
- Kanban is a simple methodology that can be learned very easily. (Yodis, 2016)
- Software can be delivered very regularly. (Yodis, 2016)

2.6.2 Disadvantages

- Since there is no specific timeframe for a delivery, timing of releases can be difficult. (Yodis, 2016)
- If the Kanban board is not updated regularly, the process will not work well. (Yodis, 2016)

3 DevOps

3.1 What is DevOps?

“DevOps is the blending of tasks performed by a company’s application development and systems operations teams.” (TechTarget, 2017)

Before DevOps, the development team typically developed and tested their software on independent environments. And once the software is done, the operations team would take over and deploy the software to the live servers. (TechTarget, 2017)

In DevOps, the idea is to bring development and operations closer together, so that the same team does the development and deployment. Typically, these deployments are automated as much as possible. (TechTarget, 2017)

3.2 Goals of DevOps

The goals of DevOps are:

- Deploy software more frequently. (CollabNet, 2018b)
- Shorten the time to market of new products. (CollabNet, 2018b)
- Improve the success rate of new releases. (CollabNet, 2018b)
- Improve the turnaround time on bug fixes. (CollabNet, 2018b)
- Recover from failures more quickly. (CollabNet, 2018b)

3.3 Continuous Integration, Delivery and Deployment

On the DevOps continuum, there are three different levels: continuous integration, continuous delivery and continuous deployment. (CollabNet, 2018b) Let us look at each of these.

“Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.” (ThoughtWorks, 2018)

If you have not worked on a large project with many developers before, you may not appreciate the value of continuous integration yet. But imagine for a moment a team of five developers all working on the same code base. If one developer makes a change to an Application Programming Interface (API) but does not check in that change to the repository for a couple of days, the next time that he or she does commit code a lot of other people’s work is likely to no longer compile, let alone work because they were still blissfully unaware of the change in API and were coding against the old version.

The longer a developer neglects to push code, the more likely conflicts become when the code is merged. And while modern source control tools do help with merging, it is not a position that developers should get themselves into in the first place.

Building the source code using an automated build tool also eliminates the type of problems that can happen where “it works on my PC!” For example, if a developer forgets to check in a new file, the build will work perfectly well on the developer’s own PC. But when the build server tries to build it, the missing file will cause the build to fail.

Once continuous integration is in place, the next level is continuous delivery.

“Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.”
(Humble, 2017)

In continuous delivery, more automation and testing are added to ensure that the code that is build is always ready to deploy. (CollabNet, 2018b) But it does not mean that every change is immediately deployed. (Caum, 2013)

The last level is continuous deployment. This means that software that is built is deployed to live environments without any manual steps. (CollabNet, 2018b)

This might seem like an impossible goal. But there are several well-known companies that does do continuous deployment: Netflix, Etsy, Amazon, Pinterest, Flickr and Google. (CollabNet, 2018b)

3.4 Tools

While a change in culture is an important part of DevOps, the tools that automate the process are an essential part of the process. Here are some of the most common tools that are used in DevOps.

- Source Code Repository
 - Git (CollabNet, 2018b)
 - GitLab (Kantor, 2018)
 - Team Foundation Server (TFS) (CollabNet, 2018b)
- Build Server
 - Jenkins (CollabNet, 2018b)
 - Bamboo (Levy, 2015)

- Test Automation
 - Selenium (CollabNet, 2018b)
 - Watir (CollabNet, 2018b)
- Configuration Management
 - Puppet (CollabNet, 2018b)
 - Chef (CollabNet, 2018b)
 - Terraform (Kantor, 2018)
 - Ansible (Kantor, 2018)
 - SaltStack (Brikman, 2016)
 - CloudFormation (Brikman, 2016)
- Containers
 - Docker (Kantor, 2018)
- Container orchestration
 - Kubernetes (Kantor, 2018)
 - Amazon EC2 Container Service (ECS) (Kantor, 2018)
- Monitoring
 - Elasticsearch, Logstash, and Kibana (ELK) (Kantor, 2018)
 - Prometheus (Kantor, 2018)
 - Nagios (Levy, 2015)

4 Enterprise Architecture

4.1 What is Enterprise Architecture?

Enterprise architecture is a discipline that functions on a much higher level than the architecture of a single application. In large enterprises, there are often groups of people tasked with maintaining the enterprise architecture.

“Enterprise architecture (EA) is a discipline for proactively and holistically leading enterprise responses to disruptive forces by identifying and analyzing the execution of change toward desired business vision and outcomes. EA delivers value by presenting business and IT leaders with signature-ready recommendations for adjusting policies and projects to achieve target business outcomes that capitalize on relevant business disruptions.” (Gartner, 2018)

From this definition, we see that enterprise architecture supports the larger business strategy by ensuring that the IT systems enable the enterprise to reach its objectives. While a software developer rarely has direct input into the enterprise architecture, it is useful to understand that all software projects fit into the larger enterprise architecture.

The Open Group additionally indicates in the introduction to *The Open Group Architecture Framework* (TOGAF) that enterprise architecture should improve the integration of disjointed processes. (The Open Group, 2018)

In the rest of this section, we will look at three different frameworks and methodologies that operate at an enterprise architecture level in large organisations:

- Information Technology Infrastructure Library (ITIL) is a framework for managing IT services all the way through their life cycle. (Priyadharshini, 2018)
- TOGAF framework describes a methodology for describing the architecture of an enterprise. (White, 2018)
- The Zachman framework is an ontology – a structured way to describe an enterprise. (Zachman, 2008)

TOGAF and Zachman can be used together to arrive at a description of the enterprise architecture – TOGAF describes the process while Zachman provides the template or framework to fill in.

4.2 Information Technology Infrastructure Library (ITIL) Framework

Information Technology Infrastructure Library (ITIL) is a framework for managing information technology (IT) services in an organisation. A service in this context is something that delivers value to a customer. (Priyadharshini, 2018)

ITIL was first introduced by the British government in the 1980s. In the early 1990s, large companies and governments in Europe adopted the framework. (ILX Group, 2018)

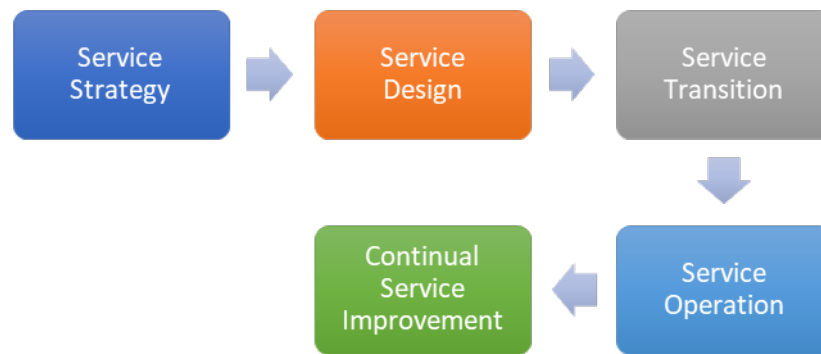


Figure 30: ITIL Process Categories

There are five categories of processes that are described in ITIL, as shown in Figure 30. Each category contains a list of specific processes. For example, under Service Design there is Service Level Management, Availability Management, and IT Security Management to name a few. (Priyadharshini, 2018)

ITIL is not a standard to comply to, but instead contains guidelines for how to manage IT services. (Burton, 2016) ISO 20000 is an example of a measurable standard for IT service management (ITSM). If ITIL is implemented well, it will aid in being ISO 20000 certifiable. (Zitek, 2018)

Whether ITIL is still relevant in a world where DevOps is becoming more important is a much-debated question. It can be argued that ITIL, if properly implemented, can be complementary to DevOps instead of opposed to it. Read more in (Daine, 2018).

4.3 The Open Group Architecture Framework (TOGAF)

TOGAF was first created in 1995 by the Open Group. (White, 2018)

At the heart of TOGAF is the Architecture Development Method (ADM), as shown in Figure 31. The cycle contains all the steps to be taken when a new business requirement is identified. (The Open Group, 2018).

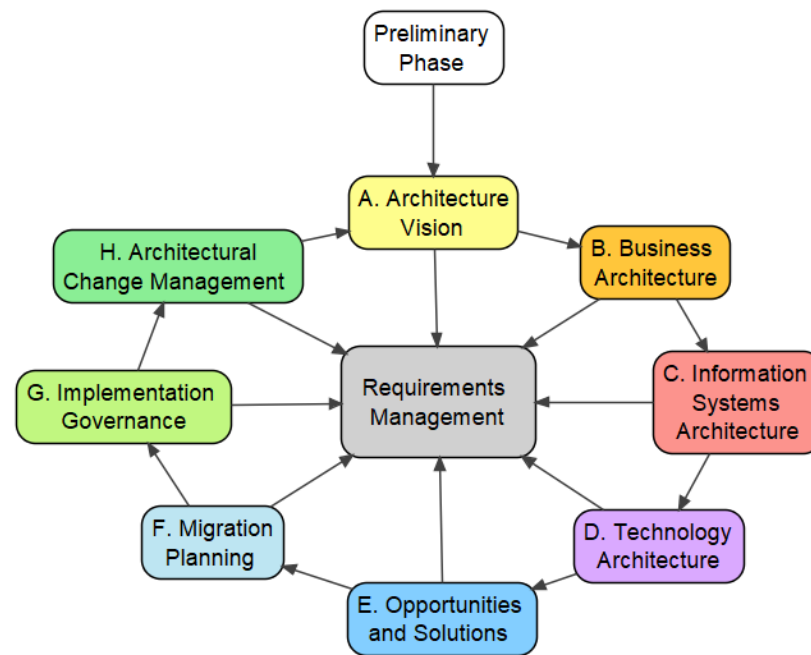


Figure 31: TOGAF ADM Cycle

In the preliminary phase, the new business requirement is identified. For example, when an Internet Service Provider (ISP) decides to start providing fibre connections for the first time, this new product needs to be implemented in the enterprise's systems. In the architecture vision step, the high-level details of the proposed system are described. This can for example be that there must be a new map where consumers can see fibre coverage and a new application form for the fibre product needs to be created.

As each of the steps from A to H are completed, the information that is gathered in the step is stored in some central information store, denoted by requirements management in the diagram. So, once the high-level details are known, this is recorded.

Steps B, C and D elaborate on the various aspects of the system to be built. In the opportunities and solutions step, a plan is created for what will be available by when. In the migration planning step, the external time constraints are considered to determine when the newly available features will be deployed.

The implementation step includes all the software development efforts and from an enterprise architecture perspective, this is about monitoring to make sure that the architecture that was defined is correctly implemented.

The last step is used to review the process and decide on any future changes to be made to the whole process.

TOGAF is not very prescriptive when it comes to the exact nature of the artefacts that are created during the process. But Zachman is one of the recommended frameworks for structuring the information in the central information store.

4.4 Zachman Framework

The Zachman framework was first created in 1984, and it has been updated several times since. (Zachman, 2011)

Figure 32 shows a simplified view of the 6 x 6 table that is the basis of the Zachman framework. Each row describes the enterprise at a level that has a specific audience. Each column answers a specific question.

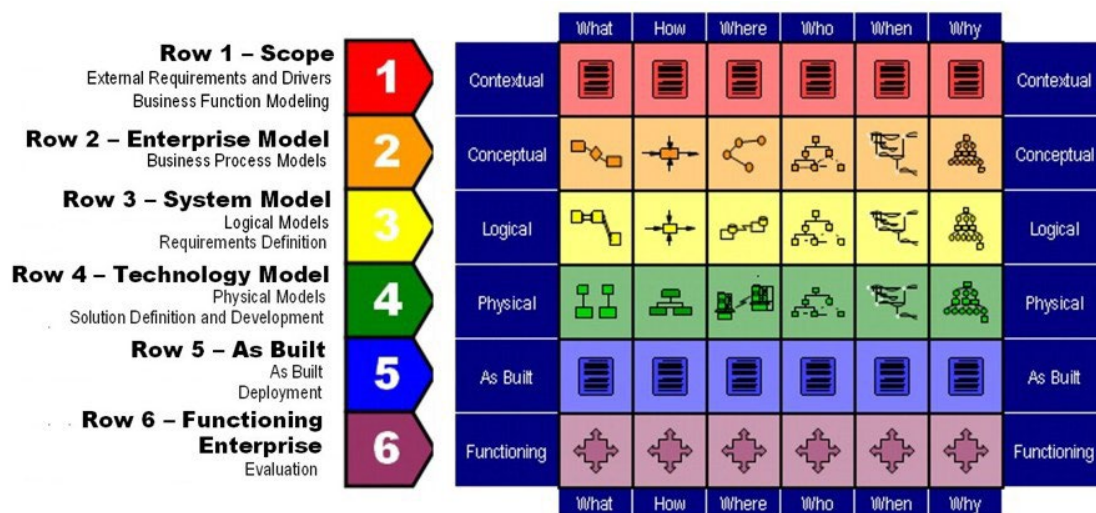


Figure 32: Zachman Framework (from (Tej, 2019))

For the full diagram showing the audience for each row and the typical elements that are described in each of the blocks, see (Zachman, 2008).

Each cell in a completed Zachman table can be anywhere from a simple statement to a link to another very detailed diagram.

5 Recommended Additional Reading

Learn more about the standards used in avionics systems in (Horváth, 2018).

Read more about the Rational Unified Process in (Ambler, 2005).

Read more about Extreme Programming in (Agile Alliance, 2018b).

Read more about Scrum in the SBOK™ Guide (SCRUMstudy, 2017).

To learn more about Scrum, and get Scrum Fundamentals Certified for free, visit the SCRUMstudy SFC course page: <https://www.scrumstudy.com/certification/scrum-fundamentals-certified> [Accessed 12 December 2022].

To learn more about The Open Group Architecture Framework (TOGAF), read the official standard at <http://pubs.opengroup.org/architecture/togaf92-doc/arch/> [Accessed 12 December 2022].

6 Revision Exercises

6.1 Revision Exercise 1

What are the differences between Scrum and Kanban?

6.2 Revision Exercise 2

What are the goals of DevOps?

7 Solutions to Revision Exercises

7.1 Revision Exercise 1

Read Sections 2.5 and 2.6 of this learning unit.

7.2 Revision Exercise 2

Read Section 3.2 of this learning unit.

Concluding Remarks

Think back all the way to Learning Unit 1 for a moment. Remember the very first question that I posed to you back then?

“What is an enterprise software system?”

Take a moment to reflect on your answer that you wrote down then. How would you answer that question today? How has your perception of enterprise software systems changed?

I hope that this has been a rewarding journey and that you feel confident in your newly acquired knowledge and skills!

Bibliography

Adobe, 2021. What is the Waterfall methodology?. [Online] Available at: <https://www.workfront.com/project-management/methodologies/waterfall> [Accessed 11 December 2023].

Agarwal, V. V., 2013. .NET Code Access Security (CAS). [Online] Available at: <https://www.c-sharpcorner.com/UploadFile/84c85b/net-code-access-security-cas/> [Accessed 11 December 2023].

Agile Alliance, 2018b. Extreme Programming. [Online] Available at: <https://www.agilealliance.org/glossary/xp/> [Accessed 11 December 2023].

Agile Alliance, 2018. Glossary: AntiPattern. [Online] Available at: <https://www.agilealliance.org/glossary/antipattern> [Accessed 11 December 2023].

Amarasinghe, et al., 2015. Reading 23: Locks and Synchronization. [Online] Available at: <http://web.mit.edu/6.005/www/fa15/classes/23-locks/> [Accessed 11 December 2023].

Ambler, S. W., 2005. A Manager's Introduction to The Rational Unified Process (RUP). [Online] Available at: <http://www.ambysoft.com/downloads/managersIntroToRUP.pdf> [Accessed 11 December 2023].

Anderson, R., Addie, S., Pasic, A. & Dykstra, T., 2009. Improving Performance with Output Caching (C#). [Online] Available at: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/improving-performance-with-output-caching-cs> [Accessed 11 December 2023].

Appian, 2018. Business Process Definition. [Online] Available at: <https://www.appian.com/bpm/definition-of-a-business-process/> [Accessed 11 December 2023].

Atwood, J., 2006. Code Smells. [Online] Available at: <https://blog.codinghorror.com/code-smells/> [Accessed 11 December 2023].

Beal, V., 2010. Enterprise Applications Explained. [Online] Available at: https://www.webopedia.com/quick_ref/enterprise_application.asp [Accessed 12 December 2022].

Bisson, S., 2017. .Net Framework or .Net Core? When to use which. [Online] Available at: <https://www.infoworld.com/article/3180478/development-tools/net-framework-or-net-core-when-to-use-which.html> [Accessed 11 December 2023].

Bonham, A., 2017. Microservices—When to React Vs. Orchestrate. [Online] Available at: <https://medium.com/capital-one-developers/microservices-when-to-react-vs-orchestrate-c6b18308a14c> [Accessed 11 December 2023].

Boursi, A., 2010. IOperationBehavior—How to customize behaviors for the WCF operation?. [Online] Available at: <https://wcpro.wordpress.com/2010/12/22/ioperationbehavior/> [Accessed 11 December 2023].

Breakwell, J., 2008. How do I create an MSMQ outgoing queue?. [Online] Available at: <https://docs.microsoft.com/en-gb/archive/blogs/johnbreakwell/so-what-is-an-outgoing-queue-and-what-does-msmq-do-with-it-it> [Accessed 11 December 2023].

Brikman, Y., 2016. Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation. [Online] Available at: <https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c> [Accessed 11 December 2023].

Brown, K., 2008. BizTalk Server 2006 or WF? Choosing the Right Workflow Tool for Your Project. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/cc303238\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/cc303238(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Buen, M., 2013. SQL Server Said, PostgreSQL Said. APPLY and LATERAL. [Online] Available at: <http://www.anicehumble.com/2013/09/sql-server-said-postgresql-said-apply-lateral.html> [Accessed 11 December 2023].

Burns, A. et al., 2018. Cross-Platform mobile development in Visual Studio. [Online] Available at: <https://docs.microsoft.com/en-gb/visualstudio/cross-platform/cross-platform-mobile-development-in-visual-studio> [Accessed 11 December 2023].

Burton, R., 2016. ITIL® Misconceptions: “ITIL is a standard to adhere to”. [Online] Available at: <https://www.axelos.com/news/blogs/september-2016/itil-misconceptions-itil-is-standard-to-adhere-to> [Accessed 11 December 2023].

Carr, R., 2009. Gang of Four Design Patterns. [Online] Available at: <http://www.blackwasp.co.uk/gofpatterns.aspx> [Accessed 11 December 2023].

Carter, P., A, A., Addie, S. & Wenzel, M., 2018. Choosing between .NET Core and .NET Framework for server apps. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server> [Accessed 11 December 2023].

Carter, P. et al., 2021. .NET vs. .NET Framework for server apps. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server> [Accessed 11 December 2023].

Caum, C., 2013. Continuous Delivery Vs. Continuous Deployment: What's the Diff?. [Online] Available at: <https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff> [Accessed 11 December 2023].

CollabNet, 2018b. What is DevOps? The Ultimate Guide to DevOps. [Online] Available at: <https://resources.collab.net/devops-101/what-is-devops> [Accessed 12 December 2022].

CollabNet, 2018. What Is Kanban? An Introduction to Kanban Methodology. [Online] Available at: <https://resources.collab.net/agile-101/what-is-kanban> [Accessed 11 December 2023].

Coosner, L., 2020. "String" + "Concatenation" = "Slow". [Online] Available at: <https://www.incusdata.co.za/blog/programming/string-concatenation-slow/> [Accessed 11 December 2023].

Crouch, S., 2018. Developing maintainable software. [Online] Available at: <https://www.software.ac.uk/resources/guides/developing-maintainable-software> [Accessed 11 December 2023].

Daine, G., 2018. ITIL® Vs. DevOps! 25 Influential Experts Share Their Insights (Is ITIL® Agile Enough?). [Online] Available at: <https://purplegriffon.com/blog/is-til-agile-enough> [Accessed 11 December 2023].

Dalbey, J., 1998. Non-functional Requirements. [Online] Available at: <http://users.csc.calpoly.edu/~jdalbey/SWE/QA/nonfunctional.html> [Accessed 11 December 2023].

Das, C., 2018. Scrum Project Management – Pros and Cons. [Online] Available at: <https://www.simplilearn.com/scrum-project-management-article> [Accessed 11 December 2023].

De Rycke, P., 2012. Caching in WCF Services: Part 1. [Online] Available at: <https://pieterderycke.wordpress.com/2012/04/09/caching-in-wcf-services-part-1/> [Accessed 11 December 2023].

Defuse Security, 2018. Salted Password Hashing - Doing it Right. [Online] Available at: <https://crackstation.net/hashing-security.htm> [Accessed 11 December 2023].

Dejaeger, G., 2010. Programming antipatterns. [Online] Available at: <https://glenndejaeger.wordpress.com/2010/04/05/programming-antipatterns/> [Accessed 11 December 2023].

Despodovski, R., 2017. Microservices vs. SOA – Is There Any Difference at All?. [Online] Available at: <https://dzone.com/articles/microservices-vs-soa-is-there-any-difference-at-all> [Accessed 11 December 2023].

DevIQ, 2018. Code Smells: Symptoms of Possible Deeper Problems. [Online] Available at: <https://deviq.com/code-smells/> [Accessed 11 December 2023].

DigiCert, Inc., 2018. What is SSL, TLS and HTTPS?. [Online] Available at: <https://www.websecurity.symantec.com/security-topics/what-is-ssl-tls-https> [Accessed 24 September 2018].

Docker Inc., 2018b. Microservices. [Online] Available at: <https://www.docker.com/solutions/microservices> [Accessed 12 December 2022].

Docker Inc., 2018. What is a Container. [Online] Available at: <https://www.docker.com/resources/what-container> [Accessed 11 December 2023].

Eiffel.org, 2018. .NET. [Online] Available at: <https://www.eiffel.org/doc/solutions/.NET> [Accessed 11 December 2023].

EntityFrameworkTutorial.net, 2018. Transaction in Entity Framework. [Online] Available at: <http://www.entityframeworktutorial.net/EntityFramework6/transaction-in-entity-framework.aspx> [Accessed 11 December 2023].

Eriksson, U., 2015. The difference between functional and non-functional requirements. [Online] Available at: <https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/> [Accessed 11 December 2023].

eTutorials.org, 2018. 7.2 Programming Code-Access Security. [Online] Available at: <http://etutorials.org/Programming/Programming+.net+security/Part+II+.NET+Security/Chapter+7.+Permissions/7.2+Programming+Code+Access+Security/> [Accessed 11 December 2023].

Fink, G., 2010. Select N+1 Problem – How to Decrease Your ORM Performance. [Online] Available at: <https://dzone.com/articles/select-n1-problem-%e2%80%93-how> [Accessed 12 December 2022].

Flores, J. et al., 2018. Implementing Least-Privilege Administrative Models. [Online] Available at: <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models> [Accessed 11 December 2023].

Fowler, M., 2006. CodeSmell. [Online] Available at: <https://martinfowler.com/bliki/CodeSmell.html> [Accessed 11 December 2023].

Fowler, M., 2011. CQRS. [Online] Available at: <https://martinfowler.com/bliki/CQRS.html> [Accessed 11 December 2023].

Fowler, M., 2014. EnterpriseApplication. [Online] Available at: <https://martinfowler.com/bliki/EnterpriseApplication.html> [Accessed 11 December 2023].

Fritchey, G., 2012. The Seven Sins against TSQL Performance. [Online] Available at: <https://www.red-gate.com/simple-talk/sql/performance/the-seven-sins-against-tsql-performance/> [Accessed 11 December 2023].

Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995. Design Patters: Elements of Reusable Object-Oriented Software. 1st ed. Indianapolis: Addison-Wesley.
Gartner, 2018. Enterprise Architecture (EA). [Online] Available at: <https://www.gartner.com/it-glossary/enterprise-architecture-ea/> [Accessed 11 December 2023].

Gavilán, F., 2018. Entity Framework Core 2.1: Ambient Transactions (new functionality!). [Online] Available at: <https://gavilan.blog/2018/08/02/entity-framework-core-2-1-ambient-transactions-new-functionality/> [Accessed 11 December 2023].

GeeksforGeeks, 2018. Operating System | Process Management | Deadlock Introduction. [Online] Available at: <https://www.geeksforgeeks.org/operating-system-process-management-deadlock-introduction/> [Accessed 11 December 2023].

GeeksforGeeks, 2021. Introduction of Deadlock in Operating System. [Online] Available at: <https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/> [Accessed 11 December 2023].

Gerard, N., 2017. Software Design - Scalability (Scale Up|Out). [Online] Available at: <https://gerardnico.com/code/design/scalability> [Accessed 12 December 2022].

Gerr, P., 2018. Entity Framework Core: Use TransactionScope with Caution!. [Online] Available at: <http://weblogs.thinktecture.com/pawel/2018/06/entity-framework-core-use-transactionscope-with-caution.html> [Accessed 21 September 2018].

Goodreads Inc, 2018. Quote by Eleanor Roosevelt: “Learn from the mistakes of others. You can't li...”. [Online] Available at: <https://www.goodreads.com/quotes/6521824-learn-from-the-mistakes-of-others-you-can-t-live-long> [Accessed 11 December 2023].

Gotsch, S., 2018. REST API Design. [Online] Available at: <https://www.jamasoftware.com/blog/rest-api-design/> [Accessed 11 December 2023].

Graca, H., 2017. Architectural Styles vs. Architectural Patterns vs. Design Patterns. [Online] Available at: <https://herbertograca.com/2017/07/28/architectural-styles-vs-architectural-patterns-vs-design-patterns/> [Accessed 11 December 2023].

Gratis, B., 2018. 5 Reasons Agile isn't working for your team. [Online] Available at: <https://backlog.com/blog/5-reasons-agile-isnt-working-your-team/> [Accessed 11 December 2023].

Green, R., 2018b. Hosting Windows Communication Foundation Services. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ee939285\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ee939285(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Green, R., 2018. Self Hosting Windows Communication Foundation Services. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ee939340\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ee939340(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Green, S., 2016. Choose Your Project Management Methodology: Pros and Cons of Agile, Waterfall, PRiSM and more. [Online] Available at: <https://www.workflowmax.com/blog/choose-your-project-management-methodology-pros-and-cons-of-agile-waterfall-prism-and-more> [Accessed 11 December 2023].

Hare, J. M., 2010. C# Fundamentals: String Concat() vs. Format() vs. StringBuilder. [Online] Available at: <http://geekswithblogs.net/BlackRabbitCoder/archive/2010/05/10/c-string-compares-and-concatenations.aspx> [Accessed 11 December 2023].

Harris, T., 2002. The case of the 500-mile email. [Online] Available at: <https://www.ibiblio.org/harris/500milemail.html> [Accessed 11 December 2023].

Herbie, D., 2005. Performance considerations for strings in C#. [Online] Available at: <https://www.codeproject.com/Articles/10318/Performance-considerations-for-strings-in-C> [Accessed 11 December 2023].

Hibbert, B., 2016. Performance Improvement for Cursors in Stored Procedures. [Online] Available at: <https://www.sqlservercentral.com/articles/performance-improvement-for-cursors-in-stored-procedures> [Accessed 11 December 2023].

Hooks, I., 1993. 'Writing Good Requirements'. Proceedings of the Third International Symposium of the INCOSE, Volume 2. Available at: https://www.regexperts.com/wp-content/uploads/2015/07/writing_good_requirements.htm [Accessed 11 December 2023].

Horváth, Á., 2018. Standards in Avionics System Development. [Online] Available at: https://inf.mit.bme.hu/sites/default/files/materials/taxonomy/term/445/13/13_CES_DO-178B.pdf [Accessed 12 December 2022].

Howard, M. and Lipner, S., 2006. The Security Development Lifecycle. 1st ed. Redmond: Microsoft Press.

Humble, J., 2017. What is Continuous Delivery?. [Online] Available at: <https://continuousdelivery.com/> [Accessed 11 December 2023].

IBM, 2016. Service-oriented architecture (SOA). [Online] Available at: https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.eql.pg.doc/topics/pegl_serv_overview.html [Accessed 11 December 2023].

IBM, 2018. ACID properties of transactions. [Online] Available at: https://www.ibm.com/support/knowledgecenter/SSGMCP_5.1.0/com.ibm.cics.ts.prod/uctooverview.doc/concepts/acid.html [Accessed 11 December 2023].

IBM, 2018b. Distributed transactions in .NET. [Online] Available at: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.5.0/com.ibm.mq.dev.doc/q029290_.htm [Accessed 11 December 2023].

IEEE, 2017. Interoperability. [Online] Available at: <http://eitbokwiki.org/Interoperability> [Accessed 11 December 2023].

ILX Group, 2018. History of ITIL. [Online] Available at: <https://www.itiltraining.com/zar/blog/itil-history> [Accessed 11 December 2023].

Information Builders, 2018. Enterprise Information Portals (EIP) and Portal Integration. [Online] Available at: <http://www.informationbuilders.fr/EIP-enterprise-information-portal> [Accessed 11 December 2023].

International TechneGroup Incorporated, 2018. Interoperability. [Online] Available at: <https://www.iti-global.com/interoperability> [Accessed 11 December 2023].

Jana, A., 2014. 5 Internal things that you should know about IIS Express. [Online] Available at: <https://dailydotnettips.com/5-internal-things-that-you-should-know-about-iis-express/> [Accessed 11 December 2023].

Kalapos, G., 2014. Behind the .NET 4.5 Async Scene: The performance impact of Asynchronous programming in C#. [Online] Available at: <https://www.dynatrace.com/news/blog/behind-net-4-5-async-scene-performance-impact-asynchronous-programming-c/> [Accessed 11 December 2023].

Kanjilal, J., 2016. How to work with MSMQ in C#. [Online] Available at: <https://www.infoworld.com/article/3060115/application-development/how-to-work-with-msmq-in-c.html> [Accessed 11 December 2023].

Kanjilal, J., 2016. How to work with reflection in C#. [Online] Available at: <https://www.infoworld.com/article/3027240/application-development/how-to-work-with-reflection-in-c.html> [Accessed 11 December 2023].

Kantor, I., 2018. How To Become a DevOps Engineer In Six Months or Less. [Online] Available at: <https://medium.com/@devfire/how-to-become-a-devops-engineer-in-six-months-or-less-366097df7737> [Accessed 11 December 2023].

Kayal, S., 2016. 5 Tips to Improve Performance of C# Code. [Online] Available at: <https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-performance-of-C-Sharp-code/> [Accessed 11 December 2023].

Kayal, S., 2018. 5 Tips to Improve Your C# Code: Part 2. [Online] Available at: <https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-your-C-Sharp-code-part-2/> [Accessed 11 December 2023].

Kayal, S., 2018b. 5 Tips to Improve Your C# Code: Part 1. [Online] Available at: <https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-your-C-Sharp-code-part-1/> [Accessed 11 December 2023].

Kayal, S., 2018c. 5 Tips to Improve Performance of C# Code: Part 3. [Online] Available at: <https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-performance-of-C-Sharp-code-part-3/> [Accessed 11 December 2023].

Keyhole Software, 2018. Blockchain For the Enterprise. [Online] Available at: <https://keyholesoftware.com/wp-content/uploads/Blockchain-For-The-Enterprise-Keyhole-White-Paper.pdf> [Accessed 11 December 2023].

Krishnamurthy, V., 2012. A Brief History of Scrum. [Online] Available at: <https://www.techwell.com/techwell-insights/2012/10/brief-history-scrum> [Accessed 11 December 2023].

Krishnan, R., 2007. Message Queuing using C#. [Online] Available at: <https://www.c-sharpcorner.com/article/message-queuing-using-C-Sharp/> [Accessed 11 December 2023].

Kruchten, P., 2004. An Introduction to the Rational Unified Process. [Online] Available at: <http://www.informit.com/articles/article.aspx?p=169549&seqNum=5> [Accessed 11 December 2023].

Kuprenko, V., 2017. Top 5 Software Development Methodologies. [Online] Available at: <https://project-management.com/top-5-software-development-methodologies/> [Accessed 11 December 2023].

Lai, E., 2008. Microsoft may have 2,000 developers working on Windows 7. [Online] Available at: <https://www.computerworld.com/article/2532600/operating-systems/microsoft-may-have-2-000-developers-working-on-windows-7.html> [Accessed 11 December 2023].

Laker, P. and Cenerelli, K., 2012. Concurrency Design Pattern. [Online] Available at: <https://social.technet.microsoft.com/wiki/contents/articles/13210.concurrency-design-pattern.aspx> [Accessed 11 December 2023].

Laker, P. and Jester, B., 2012b. Thread Pool Design Pattern. [Online] Available at: <https://social.technet.microsoft.com/wiki/contents/articles/13245.thread-pool-design-pattern.aspx> [Accessed 11 December 2023].

Lander, R., 2017. Using .NET and Docker Together. [Online] Available at: <https://blogs.msdn.microsoft.com/dotnet/2017/05/25/using-net-and-docker-together/> [Accessed 11 December 2023].

Lander, R., 2019. Introducing .NET 5. [Online] Available at: <https://devblogs.microsoft.com/dotnet/introducing-net-5/> [Accessed 11 December 2023].

Larsen, P. and Ohlinger, M., 2016. What is HIS. [Online] Available at: <https://docs.microsoft.com/en-us/host-integration-server/what-is-his> [Accessed 11 December 2023].

Laudenschlager, D. and Guyer, C., 2016. BEGIN DISTRIBUTED TRANSACTION (Transact-SQL). [Online] Available at: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/begin-distributed-transaction-transact-sql?view=sql-server-2017> [Accessed 11 December 2023].

Laudenschlager, D., Milener, G., Roth, J. and Guyer, C., 2016. BEGIN TRANSACTION (Transact-SQL). [Online] Available at: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/begin-transaction-transact-sql?view=sql-server-2017> [Accessed 11 December 2023].

Levy, T., 2015. 9 Open Source DevOps Tools We Love. [Online] Available at: <https://devops.com/9-open-source-devops-tools-love/> [Accessed 11 December 2023].

Lewis, J. and Fowler, M., 2014. Microservices: a definition of this new architectural term. [Online] Available at: <https://martinfowler.com/articles/microservices.html> [Accessed 11 December 2023].

Liferay, 2021. What is a Web Portal?. [Online] Available at: <https://www.liferay.com/resources/l/web-portal> [Accessed 11 December 2023].

Lin, C., Likness, J., Hashimi, S. I. and Outlaw, R., 2018. Create an ASP.NET Core web app in Azure. [Online] Available at: <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-dotnet> [Accessed 11 December 2023].

L, J., 2018. Security Authentication vs. Authorization | What's the Difference?. [Online] Available at: <https://swoopnow.com/security-authentication-vs-authorization/> [Accessed 11 December 2023].

Lowe, S. A., 2018. Get your feet wet with domain-driven design: 3 guiding principles. [Online] Available at: <https://techbeacon.com/get-your-feet-wet-domain-driven-design-3-guiding-principles> [Accessed 11 December 2023].

Lowy, J., 2005. Introducing System.Transactions in the .NET Framework 2.0. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973865\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973865(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Luijbregts, B., 2017. Which Azure Deployment Model Should You Use? 4 Ways To Deploy. [Online] Available at: <https://stackify.com/azure-deployment-models/> [Accessed 11 December 2023].

Luijbregts, B., 2018. Demystify the .NET Ecosystem. [Online] Available at: <https://dzone.com/articles/demystify-the-net-ecosystem-understand-its-runtime> [Accessed 11 December 2023].

Lvivity, 2018. Waterfall Methodology: Advantages, Disadvantages And When to Use It?. [Online] Available at: <https://lvivity.com/waterfall-model> [Accessed 11 December 2023].

McGraw, G., 2004. Software security. [Online] Available at: <https://www.synopsys.com/blogs/software-security/software-security/> [Accessed 11 December 2023].

Merritt, T., 2013. 9 Software Security Design Principles. [Online] Available at: <https://dzone.com/articles/9-software-security-design> [Accessed 11 December 2023].

Microsoft Corporation, 2000. Usability in Software Design. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/ms997577\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/ms997577(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Microsoft Corporation, 2005. Building Connected Systems: The .NET Framework and the Microsoft Enterprise Application Development Platform. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973223\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/dotnet/articles/ms973223(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Microsoft Corporation, 2010b. Simplified Implementation of the Microsoft SDL. [Online] Available at: <https://www.microsoft.com/en-us/download/details.aspx?id=12379> [Accessed 11 December 2023].

Microsoft Corporation, 2010. Chapter 3: Architectural Patterns and Styles. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658117\(v=pandp.10\)](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658117(v=pandp.10)) [Accessed 11 December 2023].

Microsoft Corporation, 2016. Message Queuing (MSMQ). [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/ms711472\(v=vs.85\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/ms711472(v=vs.85)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Microsoft Corporation, 2018. .NET Downloads for Linux, macOS, and Windows. [Online] Available at: <https://www.microsoft.com/net/download> [Accessed 11 December 2023].

Microsoft Corporation, 2018b. How to write to and read from Microsoft Message Queuing in Visual C#. [Online] Available at: <https://support.microsoft.com/en-us/help/815811/how-to-write-to-and-read-from-microsoft-message-queuing-in-visual-c> [Accessed 11 December 2023].

Microsoft Corporation, 2018c. Message Class. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/api/system.messaging.message?view=netframework-4.7.2> [Accessed 11 December 2023].

Microsoft Corporation, 2018c. TransactionFlowOption Enum. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/api/system.servicemodel.transactionflowoption?view=netframework-4.7.2> [Accessed 11 December 2023].

Microsoft Corporation, 2018d. SharePoint Server. [Online] Available at: <https://docs.microsoft.com/en-us/sharepoint/sharepoint-server> [Accessed 11 December 2023].

Microsoft Corporation, 2018e. How to improve string concatenation performance in Visual C#. [Online] Available at: <https://support.microsoft.com/en-gb/help/306822/how-to-improve-string-concatenation-performance-in-visual-c> [Accessed 11 December 2023].

Microsoft Corporation, 2021. What are the Microsoft SDL practices?. [Online] Available at: <https://www.microsoft.com/en-us/securityengineering/sdl/practices> [Accessed 11 December 2023].

Miri, I., 2017. Microservices vs. SOA. [Online] Available at: <https://dzone.com/articles/microservices-vs-soa-2> [Accessed 11 December 2023].

Mogul, J. C., 2005. Emergent (Mis)behavior vs. Complex Software Systems. [Online] Available at: <http://www.hpl.hp.com/techreports/2006/HPL-2006-2.pdf> [Accessed 11 December 2023].

Mohapatra, M., 2016. Transaction In .NET. [Online] Available at: <https://www.c-sharpcorner.com/article/transaction-in-net/> [Accessed 11 December 2023].

Nakov, S. and Kolev, V. e. a., 2013. Fundamentals of Computer Programming with C#. Sofia: [ebook] Available at: <http://www.introprogramming.info> [Accessed 11 December 2023].

Narumoto, M., Mancebo, J., Dasaev, V. and Palamakumbura, S., 2017. Command and Query Responsibility Segregation (CQRS) pattern. [Online] Available at:

<https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs> [Accessed 11 December 2023].

Narumoto, M. et al., 2017b. Caching. [Online] Available at: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/caching> [Accessed 11 December 2023].

Niselow, T., 2018. Five massive data breaches affecting South Africans. [Online] Available at: <https://www.fin24.com/Companies/ICT/five-massive-data-breaches-affecting-south-africans-20180619-2> [Accessed 11 December 2023].

Ohlinger, M., Sharkey, K. and Cai, S., 2017. Getting started with BizTalk Server. [Online] Available at: <https://docs.microsoft.com/en-us/biztalk/core/getting-started-with-biztalk-server> [Accessed 11 December 2023].

Oracle Corporation, 2018. Password Hashing. [Online] Available at: https://docs.oracle.com/cd/E26180_01/Platform.94/ATGPersProgGuide/html/s0506passwordhashing01.html [Accessed 11 December 2023].

Otemuyiwa, P., 2016. How Passwordless Authentication Works. [Online] Available at: <https://auth0.com/blog/how-passwordless-authentication-works/> [Accessed 11 December 2023].

Oxford Dictionary, 2018. system. [Online] Available at: <https://en.oxforddictionaries.com/definition/system> [Accessed 11 December 2023].

Pal, T., 2018. Handling Transactions in .NET Using TransactionScope. [Online] Available at: https://www.codeguru.com/csharp/.net/net_data/handling-transactions-in-.net-using-transactionscope.html [Accessed 11 December 2023].

Pan, J., 1999. Software Reliability. [Online] Available at: https://users.ece.cmu.edu/~koopman/des_s99/sw_reliability/ [Accessed 11 December 2023].

Patel, N., 2018. How Loading Time Affects Your Bottom Line. [Online] Available at: <https://neilpatel.com/blog/loading-time/> [Accessed 11 December 2023].

Perler, B. et al., 2017b. Securing Services and Clients. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/securing-services-and-clients> [Accessed 11 December 2023].

Perler, B. et al., 2017. Security Guidance and Best Practices. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/security-guidance-and-best-practices> [Accessed 11 December 2023].

Petrusha, R., Ciubotariu, F.-C., Wenzel, M. and Hubbard, J., 2018. Visual Basic Guide. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/visual-basic/> [Accessed 11 December 2023].

Peyrott, S., 2015. Using LDAP and Active Directory with C# 101. [Online] Available at: <https://auth0.com/blog/using-ldap-with-c-sharp/> [Accessed 11 December 2023].

Pine, D. et al., 2020. What's new in .NET 5. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/core/dotnet-five> [Accessed 11 December 2023].

Powell-Morse, A., 2017b. Rational Unified Process: What Is It And How Do You Use It?. [Online] Available at: <https://airbrake.io/blog/sdlc/rational-unified-process> [Accessed 11 December 2023].

Powell-Morse, A., 2017. Domain-Driven Design – What is it and how do you use it?. [Online] Available at: <https://airbrake.io/blog/software-design/domain-driven-design> [Accessed 11 December 2023].

Priyadharshini, 2018. ITIL: Key Concepts and Summary. [Online] Available at: <https://www.simplilearn.com/itil-key-concepts-and-summary-article> [Accessed 11 December 2023].

Rabaler, C. et al., 2017. SET TRANSACTION ISOLATION LEVEL (Transact-SQL). [Online] Available at: <https://docs.microsoft.com/en-us/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-2017> [Accessed 11 December 2023].

Radigan, D., 2018. What is kanban?. [Online] Available at: <https://www.atlassian.com/agile/kanban> [Accessed 11 December 2023].

Richards, M., 2015. Software Architecture Patterns. Sebastopol: O'Reilly Media Inc.

Richardson, B., 2018. The Difference between CROSS APPLY and OUTER APPLY in SQL Server. [Online] Available at: <https://www.sqlshack.com/the-difference-between-cross-apply-and-outer-apply-in-sql-server/> [Accessed 11 December 2023].

Ritchie, P., 2010. The Difference between an Anti-Pattern and a Code Smell. [Online] Available at: <https://blogs.msmvps.com/peterritchie/2010/02/03/the-difference-between-an-anti-pattern-and-a-code-smell/> [Accessed 11 December 2023].

Robert, K. and Bell, A., 2011. How To: Measure execution time in C#. [Online] Available at: <https://www.codeproject.com/Tips/162553/How-To-Measure-execution-time-in-C> [Accessed 11 December 2023].

Rojansky, S. et al., 2020. Using Transactions. [Online] Available at: <https://docs.microsoft.com/en-us/ef/core/saving/transactions> [Accessed 11 December 2023].

Roseke, B., 2012. The Life Cycle of an Engineering Project. [Online] Available at: <http://www.projectengineer.net/the-life-cycle-of-an-engineering-project/> [Accessed 11 December 2023].

Rouse, M., 2005. portal software. [Online] Available at: <https://searchcio.techtarget.com/definition/portal-software> [Accessed 11 December 2023].

Rouse, M., Jones, S. and Hughes, A., 2017. T-SQL (Transact-SQL). [Online] Available at: <https://searchsqlserver.techtarget.com/definition/T-SQL> [Accessed 11 December 2023].

Rouse, M. and Loshin, P., 2018. certificate authority (CA). [Online] Available at: <https://searchsecurity.techtarget.com/definition/certificate-authority> [Accessed 11 December 2023].

Rump, R., 2018. Common Entity Framework Problems: N + 1. [Online] Available at: <https://www.brentozar.com/archive/2018/07/common-entity-framework-problems-n-1/> [Accessed 11 December 2023].

Sacolick, I., 2018. What is agile methodology? Modern software development explained. [Online] Available at: <https://www.infoworld.com/article/3237508/agile-development/what-is-agile-methodology-modern-software-development-explained.html> [Accessed 11 December 2023].

Schedlbauer, M., 2011. The Quest For Good Requirements. [Online] Available at: <https://www.batimes.com/articles/the-quest-for-good-requirements.html> [Accessed 11 December 2023].

SCRUMstudy, 2017. A Guide to the Scrum Body of Knowledge (SBOK™ Guide) – Third edition. [ebook] Available at: <https://www.scrumstudy.com/sbokguide/> [Accessed 11 December 2023] ed. Avondale, Arizona: SCRUMstudy.

SecurEnvoy Ltd, 2018. What is 2FA?. [Online] Available at: <https://www.securenvoy.com/two-factor-authentication/what-is-2fa.shtm> [Accessed 11 December 2023].

Segal, B., n.d.. Key Differences Between Hard and Soft Tokens. [Online] Available at: <https://telnyx.com/resources/hard-token-vs-soft-token> [Accessed 11 December 2023].

Sehgal, A., 2018. Big Data Pipeline: Orchestration vs. Choreography. [Online] Available at: <https://www.sapientglobalmarkets.com/blog/big-data-pipeline-orchestration-choreography> [Accessed 11 December 2023].

Seth, A., 2017. DevOps & Change Management In The Enterprise World. [Online] Available at: <https://clearbridgemoible.com/devops-change-management-in-the-enterprise-world/> [Accessed 11 December 2023].

Sharma, R. B., 2011. Difference in layer and tier architecture. [Online] Available at: <https://www.codeproject.com/Tips/277818/Difference-in-layer-and-tier-architecture> [Accessed 11 December 2023].

Sheldon, R., 2014. Questions About T-SQL Transaction Isolation Levels You Were Too Shy to Ask. [Online] Available at: <https://www.red-gate.com/simple-talk/sql/t-sql-programming/questions-about-t-sql-transaction-isolation-levels-you-were-too-shy-to-ask/#fourth> [Accessed 11 December 2023].

Shostack, A., n.d.. The Security Principles of Saltzer and Schroeder. [Online] Available at: <https://adam.shostack.org/blog/the-security-principles-of-saltzer-and-schroeder/> [Accessed 11 December 2023].

Shute, G., 2016. Design Patterns. [Online] Available at: https://www.d.umn.edu/~gshute/softeng/new/design_patterns/design_patterns.xhtml [Accessed 11 December 2023].

Slot, M., 2017. Databases and Distributed Deadlocks: A FAQ. [Online] Available at: <https://www.citusdata.com/blog/2017/08/31/databases-and-distributed-deadlocks-a-faq/> [Accessed 11 December 2023].

Soni, R., 2006. String Concatenation vs String Builder – The performance hit! See it to believe it :o). [Online] Available at: <https://blogs.msdn.microsoft.com/rahulso/2006/08/29/string-concatenation-vs-string-builder-the-performance-hit-see-it-to-believe-it-o/> [Accessed 11 December 2023].

Sonnino, B., 2017. Internet of Things - Working with Raspberry Pi and Windows 10. [Online] Available at: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2017/may/internet-of-things-working-with-raspberry-pi-and-windows-10> [Accessed 11 December 2023].

SourceMaking.com, 2018a. Design Patterns. [Online] Available at: https://sourcemaking.com/design_patterns [Accessed 11 December 2023].

Sourcemaking.com, 2018b. AntiPatterns. [Online] Available at: <https://sourcemaking.com/antipatterns> [Accessed 11 December 2023].

Sourcemaking.com, 2018c. Code Smells. [Online] Available at: <https://sourcemaking.com/refactoring/smells> [Accessed 11 December 2023].

Sourcemaking.com, 2018d. Spaghetti Code. [Online] Available at: <https://sourcemaking.com/antipatterns/spaghetti-code> [Accessed 11 December 2023].

Sourcemaking.com, 2018e. The Blob. [Online] Available at: <https://sourcemaking.com/antipatterns/the-blob> [Accessed 11 December 2023].

Sourcemaking.com, 2018f. Continuous Obsolescence. [Online] Available at: <https://sourcemaking.com/antipatterns/continuous-obsolescence> [Accessed 11 December 2023].

Sourcemaking.com, 2018g. Golden Hammer. [Online] Available at: <https://sourcemaking.com/antipatterns/golden-hammer> [Accessed 11 December 2023].

Sprott, D. and Wilkes, L., 2004. Understanding Service-Oriented Architecture. [Online] Available at: [https://docs.microsoft.com/en-us/previous-versions/aa480021\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/aa480021(v=msdn.10)?redirectedfrom=MSDN) [Accessed 11 December 2023].

Stackify, 2017. What is IIS Express? How It Works, Tutorials, and More. [Online] Available at: <https://stackify.com/what-is-iis-express/> [Accessed 11 December 2023].

Stojanovic, D., 2015. MSMQ JSON message formatter. [Online] Available at: <https://dejanstojanovic.net/aspnet/2015/october/msmq-json-message-formatter/> [Accessed 11 December 2023].

Stringfellow, A., 2017. What Is N-Tier Architecture?. [Online] Available at: <https://dzone.com/articles/what-is-n-tier-architecture> [Accessed 11 December 2023].

Subramaniam, P., 2014. REST API Design - Resource Modeling. [Online] Available at: <https://www.thoughtworks.com/insights/blog/rest-api-design-resource-modeling> [Accessed 11 December 2023].

Synopsis, 2017. Top 4 software development methodologies. [Online] Available at: <https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/> [Accessed 11 December 2023].

TatvaSoft, 2015. Top 12 Software Development Methodologies & its Advantages / Disadvantages. [Online] Available at: <https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/> [Accessed 11 December 2023].

Technopedia, 2018b. Directory Services. [Online] Available at: <https://www.techopedia.com/definition/18887/directory-services> [Accessed 11 December 2023].

Technopedia, 2018c. Authorization. [Online] Available at: <https://www.techopedia.com/definition/10237/authorization> [Accessed 11 December 2023].

Technopedia, 2021. Thrashing. [Online] Available at: <https://www.techopedia.com/definition/4766/thrashing> [Accessed 11 December 2023].

Techopedia, 2018. Enterprise Application (EA). [Online] Available at: <https://www.techopedia.com/definition/24804/enterprise-application-ea> [Accessed 11 December 2023].

TechTarget, 2008. LDAP (Lightweight Directory Access Protocol). [Online] Available at: <https://searchmobilecomputing.techtarget.com/definition/LDAP> [Accessed 11 December 2023].

TechTarget, 2017. DevOps. [Online] Available at: <https://searchitoperations.techtarget.com/definition/DevOps> [Accessed 11 December 2023].

TechTarget, 2018. Active Directory. [Online] Available at: <https://searchwindowsserver.techtarget.com/definition/Active-Directory> [Accessed 11 December 2023].

Tej, P., 2019. The Enterprise Blockchain. [Online] Available at: <https://medium.com/seattle-technology-solutions/the-enterprise-blockchain-504344221cf6> [Accessed 11 December 2023].

Teplyakov, S., 2018. The performance characteristics of async methods in C#. [Online] Available at: <https://blogs.msdn.microsoft.com/seteplia/2018/01/25/the-performance-characteristics-of-async-methods/> [Accessed 11 December 2023].

The Open Group, 2018. 1. Introduction. [Online] Available at: <http://pubs.opengroup.org/architecture/togaf92-doc/arch/> [Accessed 11 December 2023].

The Open Group, 2018. 4. Introduction to Part II. [Online] Available at: <http://pubs.opengroup.org/architecture/togaf92-doc/arch/> [Accessed 11 December 2023].

ThoughtWorks, 2018. Continuous Integration. [Online] Available at: <https://www.thoughtworks.com/continuous-integration> [Accessed 11 December 2023].

thwink.org, 2014. Emergent Behavior. [Online] Available at: <http://www.thwink.org/sustain/glossary/EmergentBehavior.htm> [Accessed 11 December 2023].

Tolk, A., Diallo, S. Y. and Turnitsa, C. D., 2013. Applying the Levels of Conceptual Interoperability Model in Support of Integrability, Interoperability, and Composability for System-of-Systems Engineering. *Journal of systemics, cybernetics, and informatics*, 5(5), pp. 65-74.

Tuliper, A., 2014. Unity : Developing Your First Game with Unity and C#. [Online] Available at: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp> [Accessed 11 December 2023].

Tyagi, N., 2020. 7 Types Of Agile Methodologies. [Online] Available at: <https://www.analyticssteps.com/blogs/7-types-agile-methodologies> [Accessed 11 December 2023].

UB, 2004. Understanding .NET Code Access Security. [Online] Available at: <https://www.codeproject.com/Articles/5724/Understanding-NET-Code-Access-Security> [Accessed 11 December 2023].

University of California Santa Cruz, 2015. Introduction to Computer Security. [Online] Available at: <https://its.ucsc.edu/security/training/intro.html> [Accessed 11 December 2023].

University of Washington, 2010. A look at chatty vs chunky RESTful web services. [Online] Available at: <https://blogs.uw.edu/ontheroa/2010/02/12/a-look-at-chatty-vs-chunky-restful-web-services/> [Accessed 11 December 2023].

Varun, 2015. Designing a Thread Pool Framework Part 1: What's the need of a Thread Pool. [Online] Available at: <https://thispointer.com/designing-a-thread-pool-framework-part-1-whats-the-need-of-a-thread-pool/> [Accessed 11 December 2023].

Vega, D., Wenzel, M. and Dykstra, T., 2016. Working with Transactions. [Online] Available at: <https://docs.microsoft.com/en-us/ef/ef6/saving/transactions> [Accessed 11 December 2023].

Viswav, P., 2019. Microsoft announces .NET 5, the next big release. [Online] Available at: <https://mspoweruser.com/microsoft-announces-net-5-the-next-big-release/> [Accessed 11 December 2023].

Vladov, P., 2018. Measure C# Code Performance. [Online] Available at: <https://www.pvladov.com/2012/06/measure-code-performance.html> [Accessed 5 October 2018].

Vskills, 2018. Software Development Methodology. [Online] Available at: <https://www.vskills.in/certification/tutorial/software-development-methodology/> [Accessed 11 December 2023].

Warren, M., 2016. Why is reflection slow?. [Online] Available at: <http://mattwarren.org/2016/12/14/Why-is-Reflection-slow/> [Accessed 11 December 2023].

Wasson, M. and Bennage, C., 2018. Event-driven architecture style. [Online] Available at: <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven> [Accessed 11 December 2023].

Watts, S., 2017a. Enterprise Application Software Defined: How Is It Different from Other Software?. [Online] Available at: <https://www.bmc.com/blogs/enterprise-application-software-defined-how-is-it-different-from-other-software/> [Accessed 11 December 2023].

Watts, S., 2017b. Microservices vs SOA: What's the Difference?. [Online] Available at: <https://www.bmc.com/blogs/microservices-vs-soa-whats-difference/> [Accessed 11 December 2023].

Wenzel, M., Erlend, Crizanto, T. and Jones, M., 2017g. Caching Support for WCF Web HTTP Services. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/caching-support-for-wcf-web-http-services> [Accessed 11 December 2023].

Wenzel, M., Erlend, Jones, M. and Hoffman, M., 2017d. Using Dead-Letter Queues to Handle Message Transfer Failures. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/using-dead-letter-queues-to-handle-message-transfer-failures> [Accessed 11 December 2023].

Wenzel, M., Erlend, Jones, M. and Hoffman, M., 2017e. How to: Exchange Queued Messages with WCF Endpoints. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/how-to-exchange-queued-messages-with-wcf-endpoints> [Accessed 11 December 2023].

Wenzel, M., Erlend, Jones, M. and Hoffman, M., 2018. System-provided bindings. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/system-provided-bindings> [Accessed 11 December 2023].

Wenzel, M., Erlend, Jones, M. and Latham, L., 2017b. ServiceModel Transaction Attributes. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/servicemodel-transaction-attributes> [Accessed 11 December 2023].

Wenzel, M. et al., 2017f. Code Access Security Basics. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/security/security-changes> [Accessed 11 December 2023].

Wenzel, M., Jones, M., Latham, L. and yishengjin1413, 2017c. Installing Message Queuing (MSMQ). [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wcf/samples/installing-message-queuing-msmq> [Accessed 11 December 2023].

Wenzel, M., Myers, A., Jones, M. and Latham, L., 2017. Transaction Fundamentals. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/data/transactions/transaction-fundamentals> [Accessed 11 December 2023].

White, S. K., 2018. What is TOGAF? An enterprise architecture methodology for business. [Online] Available at: <https://www.cio.com/article/3251707/methodology-frameworks/what-is-togaf-an-enterprise-architecture-methodology-for-business.html> [Accessed 11 December 2023].

Wigley, A. and Nixon, J., 2015. Windows 10 - An Introduction to Building Windows Apps for Windows 10 Devices. [Online] Available at: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2015/may/windows-10-an-introduction-to-building-windows-apps-for-windows-10-devices> [Accessed 11 December 2023].

Williams, L., 2021. Livelock: What is, Example, Difference with Deadlock. [Online] Available at: <https://www.guru99.com/what-is-livelock-example.html> [Accessed 11 December 2023].

Yodis, 2016. Kanban Vs Scrum Benefits, Similarities, Pros and cons. [Online] Available at: <https://www.yodiz.com/blog/kanban-vs-scrum-benefits-similarities-pros-and-cons/> [Accessed 11 December 2023].

Zachman, J. A., 2008. The Concise Definition of The Zachman Framework. [Online] Available at: <https://www.zachman.com/about-the-zachman-framework> [Accessed 11 December 2023].

Zachman, J. P., 2011. The Zachman Framework Evolution. [Online] Available at: <https://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution> [Accessed 11 December 2023].

Zitek, N., 2018. ISO 20000 and ITIL – How are they related? [Online] Available at: <https://advisera.com/20000academy/knowledgebase/iso-20000-and-til-how-are-they-related/?icn=free-knowledgebase-20000&ici=bottom-iso-20000-and-til-how-are-they-related-txt> [Accessed 11 December 2023].

Intellectual Property

Plagiarism occurs in a variety of forms. Ultimately though, it refers to the use of the words, ideas or images of another person without acknowledging the source using the required conventions. The IIE publishes a Quick Reference Guide that provides more detailed guidance, but a brief description of plagiarism and referencing is included below for your reference. It is vital that you are familiar with this information and the Intellectual Integrity Policy before attempting any assignments.

Introduction to Referencing and Plagiarism

What is 'Plagiarism'?

'Plagiarism' is the act of taking someone's words or ideas and presenting them as your own.

What is 'Referencing'?

'Referencing' is the act of citing or giving credit to the authors of any work that you have referred to or consulted. A 'reference' then refers to a citation (a credit) or the actual information from a publication that is referred to.

Referencing is the acknowledgment of any work that is not your own, but is used by you in an academic document. It is simply a way of giving credit to and acknowledging the ideas and words of others.

When writing assignments, students are required to acknowledge the work, words or ideas of others through the technique of referencing. Referencing occurs in the text at the place where the work of others is being cited, and at the end of the document, in the bibliography.

The bibliography is a list of all the work (published and unpublished) that a writer has read in the course of preparing a piece of writing. This includes items that are not directly cited in the work.

A reference is required when you:

- Quote directly: when you use the exact words as they appear in the source;
- Copy directly: when you copy data, figures, tables, images, music, videos or frameworks;
- Summarise: when you write a short account of what is in the source;
- Paraphrase: when you state the work, words and ideas of someone else in your own words.

It is standard practice in the academic world to recognise and respect the ownership of ideas, known as intellectual property, through good referencing techniques. However, there are other reasons why referencing is useful.

Good Reasons for Referencing

It is good academic practice to reference because:

- It enhances the quality of your writing;
- It demonstrates the scope, depth and breadth of your research;
- It gives structure and strength to the aims of your article or paper;
- It endorses your arguments;
- It allows readers to access source documents relating to your work, quickly and easily.

Sources

The following would count as 'sources':

- Books,
- Chapters from books,
- Encyclopaedias,
- Articles,
- Journals,
- Magazines,
- Periodicals,
- Newspaper articles,
- Items from the Internet (images, videos, etc.),
- Pictures,
- Unpublished notes, articles, papers, books, manuscripts, dissertations, theses, etc.,
- Diagrams,
- Videos,
- Films,
- Music,
- Works of fiction (novels, short stories or poetry).

What You Need to Document from the Hard Copy Source You are Using

(Not every detail will be applicable in every case. However, the following lists provide a guide to what information is needed.)

You need to acknowledge:

- The words or work of the author(s),
- The author(s)'s or editor(s)'s full names,
- If your source is a group/ organisation/ body, you need all the details,
- Name of the journal, periodical, magazine, book, etc.,
- Edition,
- Publisher's name,
- Place of publication (i.e. the city of publication),
- Year of publication,
- Volume number,
- Issue number,
- Page numbers.

What You Need to Document if you are Citing Electronic Sources

- Author(s)'s/ editor(s)'s name,
- Title of the page,
- Title of the site,
- Copyright date, or the date that the page was last updated,
- Full Internet address of page(s),
- Date you accessed/ viewed the source,
- Any other relevant information pertaining to the web page or website.

Referencing Systems

There are a number of referencing systems in use and each has its own consistent rules. While these may differ from system-to-system, the referencing system followed needs to be used consistently, throughout the text. Different referencing systems cannot be mixed in the same piece of work!

A detailed guide to referencing, entitled Referencing and Plagiarism Guide is available from your library. Please refer to it if you require further assistance.

When is Referencing Not Necessary?

This is a difficult question to answer – usually when something is 'common knowledge'. However, it is not always clear what 'common knowledge' is.

Examples of 'common knowledge' are:

- Nelson Mandela was released from prison in 1990;
- The world's largest diamond was found in South Africa;
- South Africa is divided into nine (9) provinces;
- The lion is also known as 'The King of the Jungle'.
- $E = mc^2$
- The sky is blue.

Usually, all of the above examples would not be referenced. The equation $E = mc^2$ is Einstein's famous equation for calculations of total energy and has become so familiar that it is not referenced to Einstein.

Sometimes what we think is 'common knowledge', is not. For example, the above statement about the sky being blue is only partly true. The light from the sun looks white, but it is actually made up of all the colours of the rainbow. Sunlight reaches the Earth's atmosphere and is scattered in all directions by all the gases and particles in the air. The smallest particles are by coincidence the same length as the wavelength of blue light. Blue is scattered more than the other colours because it travels as shorter, smaller waves. It is not entirely accurate then to claim that the sky is blue. It is thus generally safer to always check your facts and try to find a reputable source for your claim.

Important Plagiarism Reminders

The IIE respects the intellectual property of other people and requires its students to be familiar with the necessary referencing conventions. Please ensure that you seek assistance in this regard before submitting work if you are uncertain.

If you fail to acknowledge the work or ideas of others or do so inadequately this will be handled in terms of the Intellectual Integrity Policy (available in the library) and/ or the Student Code of Conduct – depending on whether or not plagiarism and/ or cheating (passing off the work of other people as your own by copying the work of other students or copying off the Internet or from another source) is suspected.

Your campus offers individual and group training on referencing conventions – please speak to your librarian or ADC/ Campus Co-Navigator in this regard.

Reiteration of the Declaration you have signed:

1. I have been informed about the seriousness of acts of plagiarism.
2. I understand what plagiarism is.
3. I am aware that The Independent Institute of Education (IIE) has a policy regarding plagiarism and that it does not accept acts of plagiarism.
4. I am aware that the Intellectual Integrity Policy and the Student Code of Conduct prescribe the consequences of plagiarism.

5. I am aware that referencing guides are available in my student handbook or equivalent and in the library and that following them is a requirement for successful completion of my programme.
6. I am aware that should I require support or assistance in using referencing guides to avoid plagiarism I may speak to the lecturers, the librarian or the campus ADC/ Campus Co-Navigator.
7. I am aware of the consequences of plagiarism.

Please ask for assistance prior to submitting work if you are at all unsure.