Graduate Thesis

Submitted in partial fulfillment of the requirements for graduation from the Adelphi Master's Computer Science Program

Automating Enterprise Cyber Threat Modeling

Christopher Benson

Dr. Kees Leune

Dr. David Chays

Sung Kim, J.D

Mike Fernez, M.S

5/13/2024

Abstract

Enterprises must keep their businesses safe from hostile actors, regardless of size. Protecting an enterprise benefits from creating and maintaining large-scale cyber threat models. Such models are potentially extensive and may encompass large quantities of information necessary for an enterprise's security. Based on a literature review, this thesis proposes a methodology following the design science approach to help ease the automated creation of such threat models. Paired with the methodology, I designed and implemented a proof-of-concept to support analysts to understand threats from adversaries better. The proof-of-concept is validated based on a case study.

Table of Contents

1.0 Introduction

Cyber threat analysis (CTA) matches information about vulnerabilities in an organization's current environment and determines potential cyber risks. (*Cyberthreat Analysis Tool - Identify Security Threats*, n.d.) Once an environment starts scaling up into an enterprise-level environment, maintaining and analyzing all potential threats quickly becomes complex, which can lead to mistakes. In the business world, a possible security mistake can cause an enterprise to lose irrecoverable wealth and assets (Morgan, 2022). In 2022, Suffolk County, NY, experienced a cyber attack that crippled the county and forced the local government to rely exclusively on pen and paper until the situation was resolved and systems restored (Maslin, 2022). These types of attacks take months to occur, and due to a lack of proper cybersecurity procedures, the response time for critical civilian services was significantly impacted. A tool that could stay current on the latest cyber threat vectors and help automate the creation of CTA models for professionals to look over could help prevent or mitigate the impact of such attacks.

It is important for organizations to stay up to date with their infrastructure and to know the possible threats that can threaten it. This is increasingly true with the technological world quickly moving to cloud services (IDG, 2020, 2) and the amount of data being transferred steadily increasing every year (Duarte, 2023). When moving to the cloud, it might be a common misconception that services will be more secure because they are not on-premise. However, the lack of visibility will bring a lack of foresight into how those systems operate and will make traditional cybersecurity practices less effective (Leune and Kim, 2020). The repercussions could be costly if the person in charge is careless or loses track of what is included in their infrastructure. This research seeks to help mitigate the problem and make securing an enterprise more manageable and reliable.

2.0 Research Objective

This research aims to develop and evaluate software-based solutions that generate and maintain CTA models for service-oriented enterprise models (hereafter known as SOEM). The development of this tool will be used to answer the following main overall goal of this paper: determine if CTA models can be automated and maintained legibly with SOEM. The following questions will be answered:

1) Are CTA models effective for SOEM? Many approaches to threat modeling focus on software development rather than analyzing operational service environments.

2) How beneficial would automation and maintenance of CTA models be for SOEM? If not, what else can be done to help improve CTA for SOEM?

3) Does automatically creating and maintaining CTA models for SOEM help improve overall cybersecurity situational awareness?

4) How readable are large-scale automated models?

2.1 Research Methodology

This thesis takes inspiration from a design science approach to develop a prototype that will meet the requirements of the Research Objective. The approach allows us to rapidly develop and test our prototype while adhering to the research principles due to design sciences' reliance on relevance, rigor, and the scientific method (Venable et al., 2014, p. 2). This approach to conducting research will involve five steps, each of which will lead to a working prototype that will help us answer this paper's topic: How can service-oriented enterprise models be used to enhance CTA and be created and maintained automatically?

Before listing the steps, it is important to understand what exactly design science is. The design science approach relies on problem-solving, which will help allow for innovative solutions, especially when applying this approach to developing a tool that will enable us to answer the questions presented in this paper (Brocke et al., 2020, pp. 3-4). To answer these questions, a tool is needed, but since the tool does not exist, it needs to be created. Due to the research aspect of this thesis, using a creation method that coincides with research is a good option.

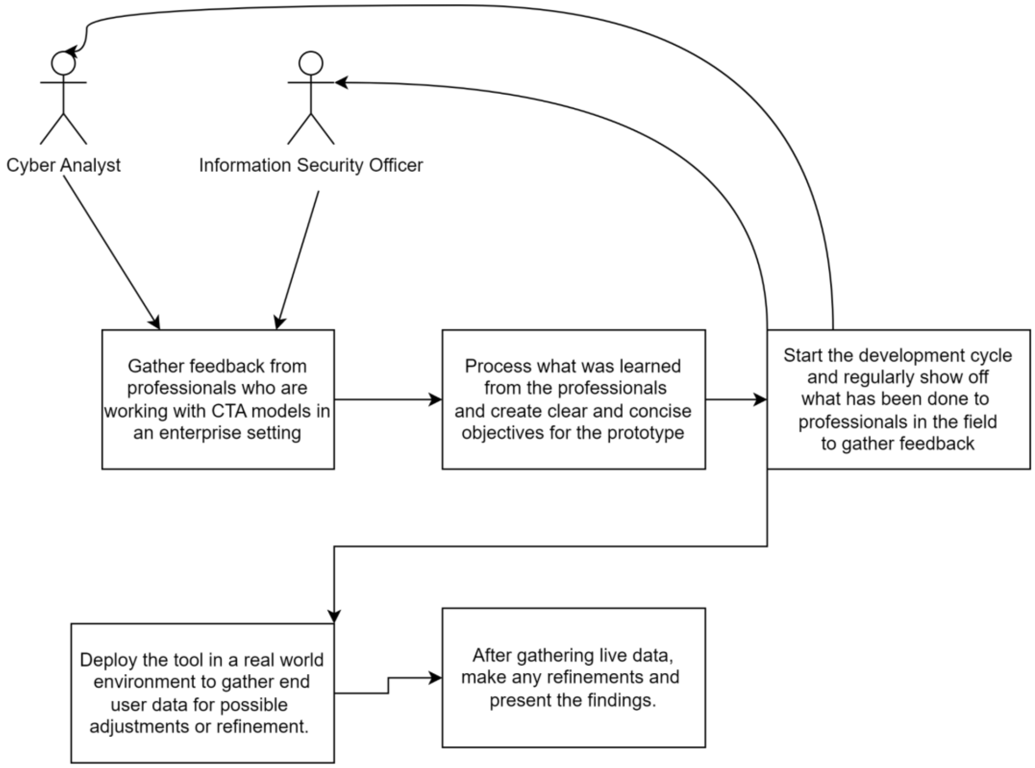

Cyber Analyst
Information Security Officer
Gather feedback from professionals who are working with CTA models in an enterprise setting
Process what was learned from the professionals and create clear and concise objectives for the prototype
Start the development cycle and regularly show off what has been done to professionals in the field to gather feedback
Deploy the tool in a real world environment to gather end user data for possible adjustments or refinement.
After gathering live data, make any refinements and present the findings.

Figure 1

A model I made based on Brocke et al. (2020) showing the five steps being taken.

Referring to Figure 1, step one uses professional feedback from those currently using CTA in an SOEM environment. It is important to gather real-world data from the source to understand better what needs to be implemented for our prototype. Step two is to derive clear objectives and restrictions when developing the prototype. With the limited timeframe of this thesis, I need to have a set of priorities and restrictions to derive results from a working prototype. In step three, with our narrowed-down goal and data from a real-world environment, software development will heavily emphasize testing and refinement. During step three, I will regularly demonstrate our work to end-users to gather their feedback on the usefulness and functionality of the prototype. This will be the longest phase, incorporating elements from steps one and two. Step four will revolve around the finished prototype being tested in a professional environment to gather professional feedback for possible refinement and adjustments. Once this is done, the design science aspect of the thesis will be completed, and the final step will be to present our findings in this paper.

3.0 Background

Before creating the proposed model, it is important to understand the fundamental pillars of Enterprise Threat Models and how they are applied at the enterprise level. These pillars are Enterprise Architecture(EA), Threat Modeling, and Enterprise Threat Modeling. Understanding EA and Threat Modeling makes understanding enterprise threat modeling possible because EA and threat modeling make up Enterprise Threat Models, as will be explored in the following subsections.

3.1 Enterprise Architecture

Enterprise Architecture translates business vision and strategy into effective enterprise change by creating, communicating, and improving the fundamental principles and models that describe the enterprise's future state and enable its evolution (Behrouz, Fathollah., 2016). With these models, the EA will represent an entire business with its constituents (Rohloff, 2005). Designing EA only on the theoretical level does not allow it to bring a more efficient use of Information Technology (IT) or help achieve business goals faster; it needs to be an integral part of the organization to be valuable and applicable (Dumitriu, Ana-Maria Popescu., 2020). When implementing EA, taking the business side and IT is important due to their importance in any enterprise and how they work together. When a sudden change occurs, businesses must react quickly, or they could be left behind. This is why EA helps an enterprise stay agile and resilient even when a sudden change occurs (White, 2022).

EA in an IT environment ensures businesses understand how IT is integrated with their enterprise. EA enables the leveraging of IT in a business, helps communications between the business and IT sectors, supports business goals, and can react quickly to market requirements with fast scalability and growth (Rohloff, 2005). EA is also used within the IT team for systems development, IT management, decision-making, and IT risk management to eliminate errors, system failures, and security breaches (White, 2022). From server upgrades, license assignments, infrastructure upgrades, and account creation, every single task can be coordinated at some level with the enterprise environment.

With EA, IT investments are also made easier due to a more precise picture being presented and the benefits stated. Using EA with IT will also provide the following benefits: (White, 2022).

1) Improve service orientation via APIs and the cloud.

2) Rationalize and less costly application portfolios.
3) Reduce the risk and cost of unsupported technology.
4) Improve information management and security.
5) Solutions to reuse existing IT assets.
6) Better performance and resilience.
7) Faster and more successful implementations and updates, and better automation.

In general, EAs are created using a framework based on the needs of a particular enterprise. One renowned framework is the TOGAF framework, which stands for The Open Group Architecture Framework. The other is the Zachman Framework for Enterprise Architecture (ZFEA). While both frameworks are essential to understand, TOGAF will be used as a basis for my proposed model in section 4.0.

3.1.1 TOGAF Framework

The TOGAF framework's main components consist of three primary domains: the Business, Application, and Infrastructure architectures (Figure 2) (Rohloff, 2005). The domain of business architecture focuses on the fundamental organization and requirements of the business based on its strategy and objectives. Application architecture gives an overview of all applications supporting the day-to-day business operations with the building blocks of enterprise applications, portal information management platforms, data repositories, and Enterprise Architecture Integration (EAI) Services. The last domain is infrastructure architecture, also called technology infrastructure, which comprises the hardware, software, and networking infrastructure required for all operations in the business. (Rohloff, 2005).

These three domains are essential to any enterprise architecture. In my work as a student system administrator at Adelphi University, I appreciated the importance of these domains firsthand. For example, I was given a set of business objectives that had to be met with a new virtual machine being created for a new Help Desk ticket application. This application would support web server infrastructure, allowing for rapid deployment of ticketing websites for testing purposes. Maintaining the university's infrastructure would only be possible with oversight and models in these areas.

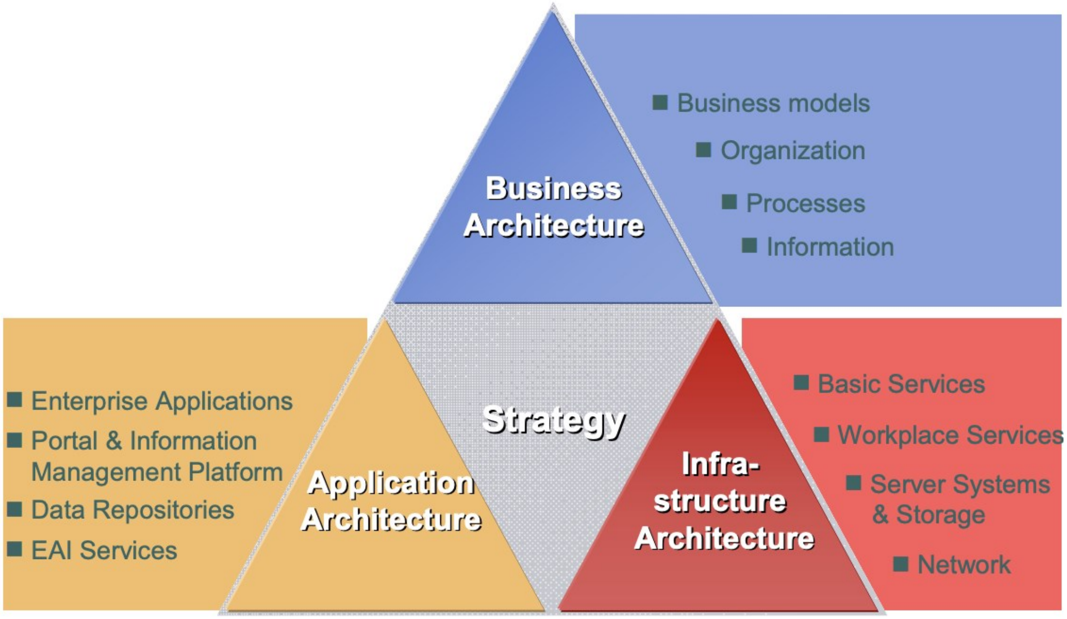
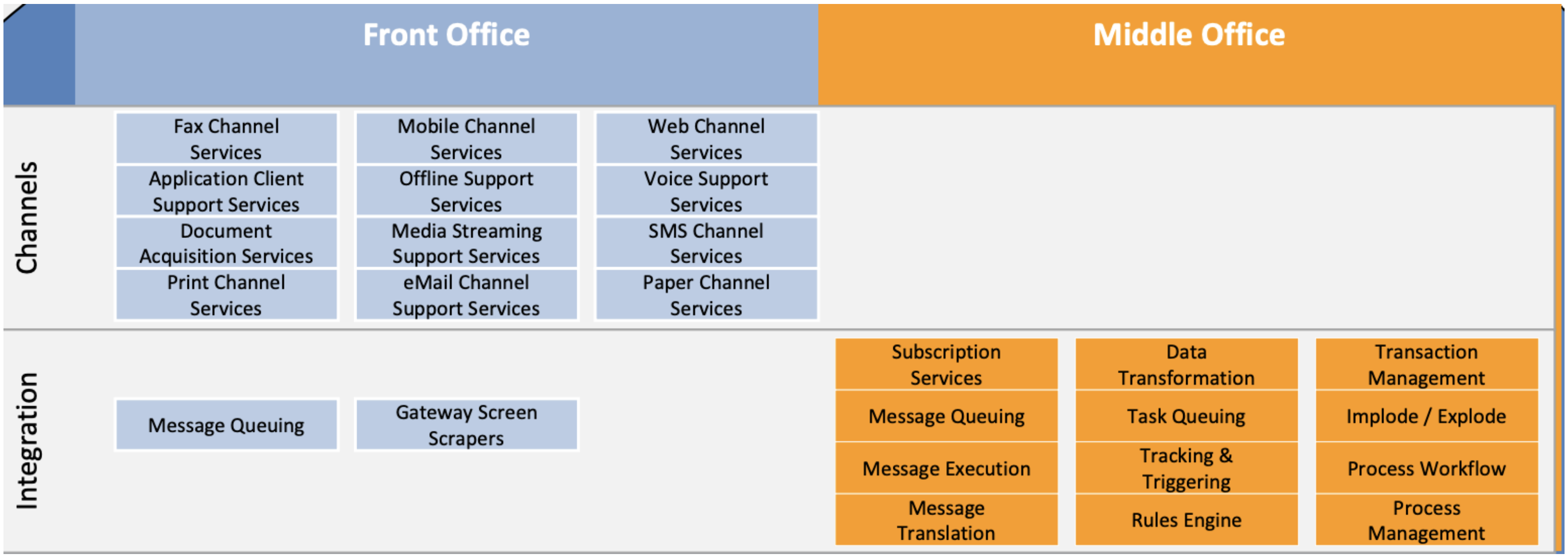


Figure 2

A graphical representation of the three primary domains of an EA framework (Rohloff, Michael., 2005).

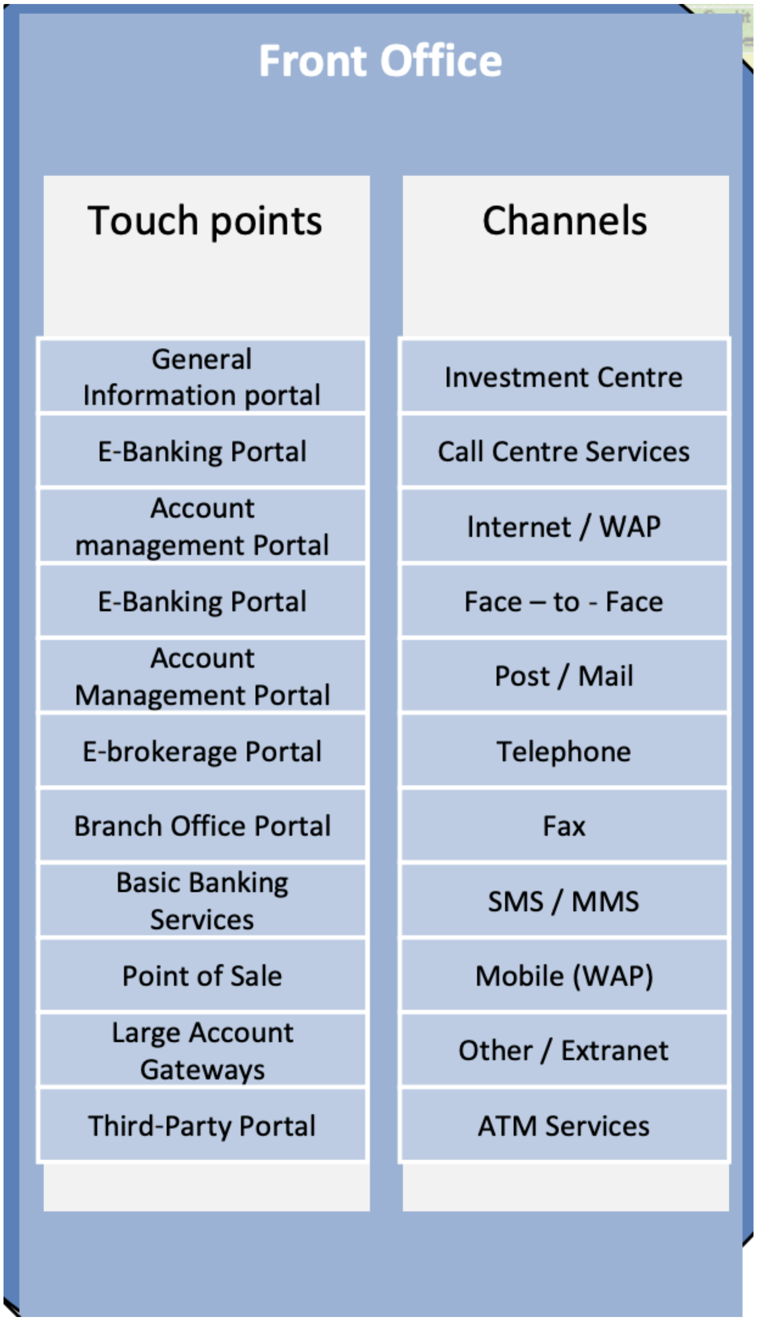
Due to TOGAF's focus on the application side of EA, the following examples will examine the application side of an enterprise and how they model it. One case study for the TOGAF methodology showcases a fictional bank trying to expand its outreach to customers through different offers and services. (The Open Group Adoption Strategies Working Group, 2010). With these new offers and services, new architecture must be carried out. Most of this case study goes over the business aspects of the case study but also focuses on the Information Systems(IS) architecture side. The IS architecture was developed based on the business's initiatives (Figure 3) and activities. Once the business model was developed, it was mapped to an IS model (Figure 4), showcasing what needed to be done to support the business model. The IS model showcases the application layer and what will be needed to reach this scenario's business goals. It also showcases the business drivers, strategies and initiatives, and activities underway within the business. (Figure 5).



	Front Office			Middle Office		
Channels	Fax Channel Services	Mobile Channel Services	Web Channel Services			
	Application Client Support Services	Offline Support Services	Voice Support Services			
	Document Acquisition Services	Media Streaming Support Services	SMS Channel Services			
	Print Channel Services	eMail Channel Support Services	Paper Channel Services			
Integration				Subscription Services	Data Transformation	Transaction Management
	Message Queuing	Gateway Screen Scrapers		Message Queuing	Task Queuing	Implode / Explode
				Message Execution	Tracking & Triggering	Process Workflow
				Message Translation	Rules Engine	Process Management

Figure 3

The business model and activities (The Open Group Adoption Strategies Working Group, 2010).

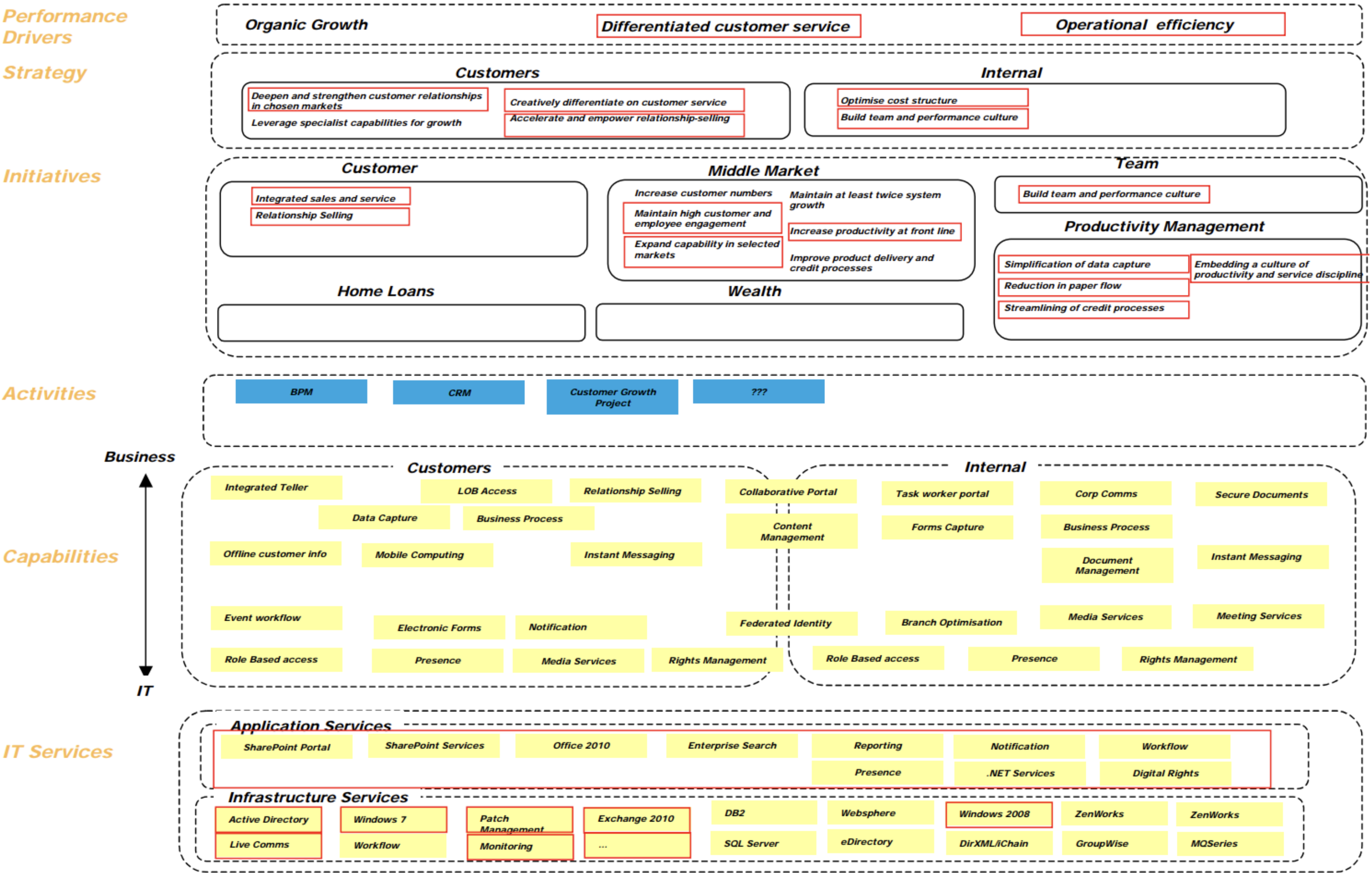


Front Office

Touch points	Channels
General Information portal	Investment Centre
E-Banking Portal	Call Centre Services
Account management Portal	Internet / WAP
E-Banking Portal	Face – to - Face
Account Management Portal	Post / Mail
E-brokerage Portal	Telephone
Branch Office Portal	Fax
Basic Banking Services	SMS / MMS
Point of Sale	Mobile (WAP)
Large Account Gateways	Other / Extranet
Third-Party Portal	ATM Services

Figure 4

The information systems model to support the business (The Open Group Adoption Strategies Working Group, 2010).



Figures 5,

Business and IT model together (The Open Group Adoption Strategies Working Group, 2010).

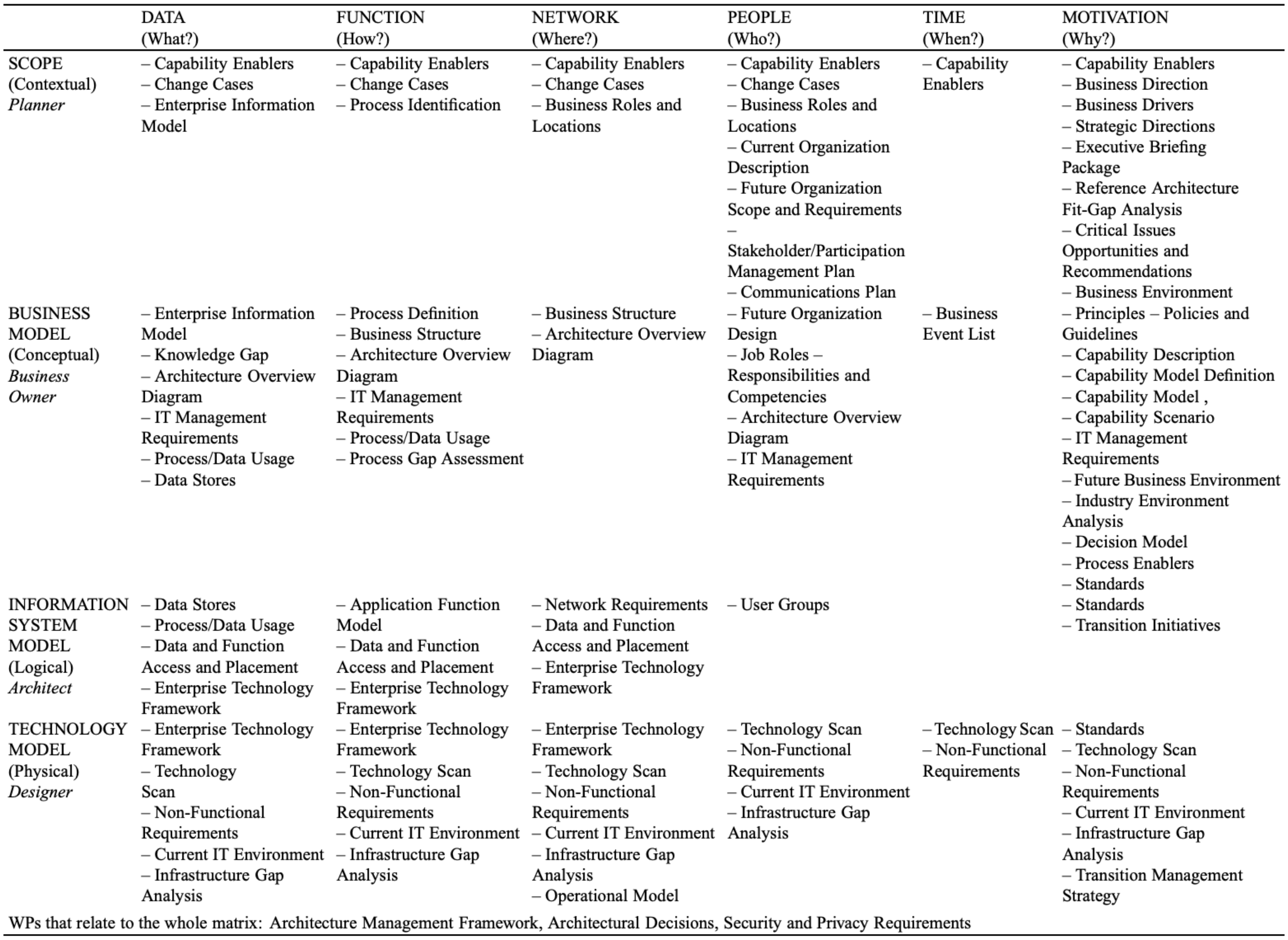
Another example of the TOGAF methodology involves a fictional Small Medium Enterprise called XYZ that focuses on dormitory house accommodation services. (Hen et al., 2021). For this case study, XYZ needs six applications to run its business successfully: a resident portal, rental application (for residents and outsiders), payment application, inventory application, asset management and digital recording application, and tenant front office application. After the case study, the authors were able to suggest an implementation of their model based on the company's needs and budget and how to implement the information systems portion of the model.

The TOGAF methodology is more graphic-oriented, with charts and diagrams. In contrast, the ZFEA methodology is more text-based and involves answering questions. For creating a tool to automate this process, the TOGAF framework will be more applicable and a good starting point for developing an automated tool to create EA models.

3.1.2 ZFEA Framework

Another EA methodology is the Zachman Framework for Enterprise Architecture (ZFEA). This framework was designed after observing that various engineered objects such as computers, buildings, and airplanes can be classified according to the fundamental abstractions or interrogatives, namely what, how, where, who, when, and why. There are also six perspectives: the executive, business manager, architect, engineer, technician, and user. These two groups create a six-by-six grid, which forms the basis of an EA (Gerber et al., 2020).

An example of the ZFEA methodology showcases a scenario of an IBM department in Finland to help apply the IBM Global Services Method to small EA-oriented projects conducted in Finland, including both IT strategy aspects and EA aspects. (Ylimäki, Halttunen, 2005) The ZFEA approach uses the what, how, where, who, when, and why framework and with which group is responsible. When applied to IT, the case study developed the following diagram in Figure 6, which also encompasses other areas relating to the IBM department.



	DATA (What?)	FUNCTION (How?)	NETWORK (Where?)	PEOPLE (Who?)	TIME (When?)	MOTIVATION (Why?)
SCOPE (Contextual) *Planner*	– Capability Enablers – Change Cases – Enterprise Information Model	– Capability Enablers – Change Cases – Process Identification	– Capability Enablers – Change Cases – Business Roles and Locations	– Capability Enablers – Change Cases – Business Roles and Locations – Current Organization Description – Future Organization Scope and Requirements – Stakeholder/Participation Management Plan – Communications Plan	– Capability Enablers	– Capability Enablers – Business Direction – Business Drivers – Strategic Directions – Executive Briefing Package – Reference Architecture Fit-Gap Analysis – Critical Issues Opportunities and Recommendations – Business Environment
BUSINESS MODEL (Conceptual) *Business Owner*	– Enterprise Information Model – Knowledge Gap – Architecture Overview Diagram – IT Management Requirements – Process/Data Usage – Data Stores	– Process Definition – Business Structure – Architecture Overview Diagram – IT Management Requirements – Process/Data Usage – Process Gap Assessment	– Business Structure – Architecture Overview Diagram	– Future Organization Design – Job Roles – Responsibilities and Competencies – Architecture Overview Diagram – IT Management Requirements	– Business Event List	– Principles – Policies and Guidelines – Capability Description – Capability Model Definition – Capability Model , – Capability Scenario – IT Management Requirements – Future Business Environment – Industry Environment Analysis – Decision Model – Process Enablers – Standards
INFORMATION SYSTEM MODEL (Logical) *Architect*	– Data Stores – Process/Data Usage – Data and Function Access and Placement – Enterprise Technology Framework	– Application Function Model – Data and Function Access and Placement – Enterprise Technology Framework	– Network Requirements – Data and Function Access and Placement – Enterprise Technology Framework	– User Groups		– Standards – Transition Initiatives
TECHNOLOGY MODEL (Physical) *Designer*	– Enterprise Technology Framework – Technology Scan – Non-Functional Requirements – Current IT Environment – Infrastructure Gap Analysis	– Enterprise Technology Framework – Technology Scan – Non-Functional Requirements – Current IT Environment – Infrastructure Gap Analysis	– Enterprise Technology Framework – Technology Scan – Non-Functional Requirements – Current IT Environment – Infrastructure Gap Analysis – Operational Model	– Technology Scan – Non-Functional Requirements – Current IT Environment – Infrastructure Gap Analysis	– Technology Scan – Non-Functional Requirements	– Standards – Technology Scan – Non-Functional Requirements – Current IT Environment – Infrastructure Gap Analysis – Transition Management Strategy

WPs that relate to the whole matrix: Architecture Management Framework, Architectural Decisions, Security and Privacy Requirements

Figure 6

This shows the ZFEA framework applied to a small set of EA-oriented projects. (Ylimäki Tanja., Halttunen Veikko., 2005)

3.2 Threat Modeling

Threat modeling is a structured process for identifying and understanding potential threats and developing and prioritizing mitigations to protect valuable assets in the system (Shi et al., 2022). Threats are possible harms that occur from a vulnerability or flaw in the design of software applications (Shafiq et al., 2014). When a threat is discovered, it can be exploited and used maliciously, leading to unwanted consequences. With threat modeling, threats can be identified early on and help gauge how much investment is needed to secure a system. When properly applied, threat modeling can become effective on the enterprise level to alleviate threats and maintain positive feedback among those who apply it (Shi et al., 2022).

Threat modeling also analyzes software, organizational network systems, and industrial areas such as the energy sector. Security experts and engineers working side by side could create these threat models. This may not always be seen as the best approach due to the lack of available experts at times. Some believe it may be developing the system to create their own threat model due to having a better understanding of how the security model works and to take ownership of their own work (Shostack, 2008).

Threat modeling requires risk analysis, usually conducted during the system's design phase. When applying threat modeling, a definition of risk is required because each organization has different needs and risk assessments. One organization might have a different approach compared to another organization, but it will be based on their needs. It is important first to apply the following four questions when applying a threat model (Shostack, 2014).

1) What are you building?
2) What can go wrong?
3) What should be done about the things that can go wrong?
4) Was the threat model applied correctly by the modeler?

Understanding these questions will help threat modelers better understand what they are creating and why they are doing it. It will also give them a better path and an end goal for their threat model.

There are multiple different methods for tackling threat modeling. One threat modeling method could follow the following process: What are we building, what can go wrong, what are we going to do about it, and finally, did we do a good enough job? (Yskout, et al. 2020). Yskout et al. then analyzed twenty threat modeling projects. They noted that each project followed the same structure(Figure 7) (Yskout et al., 2020):

1) Stakeholders will agree on the scope and goal of the project.
2) A model of the system, usually a flow chart or whiteboard diagram, is created in a modeling session and finalized by a threat modeling expert.

3) An elicitation session is held to uncover threats, which are then ranked by experts afterward.

4) Experts prepare a review meeting while colleagues perform quality assurance checks. Finally, the results are presented to the stakeholders to determine whether they are satisfactory.

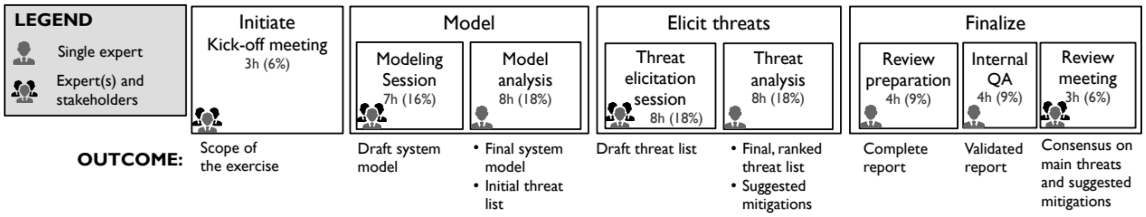


Figure 7,

A model going over the four steps of threat modeling based on 20 threat modeling projects. (Yskout et al., 2020)

STRIDE is another threat modeling method, the most widely used model. The model can cover the six threats of threat modeling: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (Hussain et al., 2014). Once the STRIDE model has been constructed, security threats are mapped and analyzed, creating a DREAD model used to rate each identified threat from the STRIDE model. DREAD stands for Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability (Hussain et al., 2014). When STRIDE and DREAD are completed, any possible mitigation measures are recommended. The only downside of STRIDE is its reliance on fully manual modeling without any automation.

3.2.1 Threat Modeling Examples

Real-life threat models include a case study with the New York City Cyber Command (NYC3), which is responsible for defending NYC from cyber attacks. NYC is a city that supports 60 million visitors each year and has over 300,000 government employees (Stevens et al., 2018). To minimize the time an employee spends on their duties, the NYC3 used the Center of Gravity threat modeling method due to its top-down approach and insight into what an adversary might be thinking. The case study results showed that after 120 days of completing the study, the NYC3 implemented eight new categories of controls directly based on the actionable defense plans developed by the participants in the study. Also, improvement was seen in the following areas: Testing readiness, securing accounts, protecting physical network assets, crowdsourcing assessments, sensor coverage, protecting legacy systems, protecting against data corruption, and reducing human error (Stevens et al., 2018). These improvements are exactly what this paper is seeking to improve but in an automated environment that will allow quick and easy adoption through the creation of threat models for enterprise environments.

Microsoft is a high-profile example of an enterprise using threat modeling in the day-to-day development of its software and products. Adam Shostack created the Security Development Lifecycle (SDL), which Microsoft employed using a set of processes applied to all Microsoft services that pose a severe security or privacy risk (Shostack, 2008). The SDL process involves four steps: diagraming, threat enumeration, mitigation, and verification. Diagraming uses data flow diagrams, which help show processes, where data is stored, how it is connected, and what entities it encompasses (Table 1). Threat enumeration under Microsoft uses the STRIDE methodology for each element presented in the Diagraming step. Mitigation uses the model to propose a practical resolution using a redesign, standard mitigations, unique mitigations, or just accepting the risk. Finally, verification ensures that the entire threat model process has been thoroughly conducted according to Microsoft policy and that all risks have been mitigated (Shostack, 2008).

Name	**External Entity**	**Process**	**Data Flow**	**Data Store**
Representation	Rectangular Box	Circle	Directed Arrow	Parallel Lines

Definition	Things outside your control	Code	How information flows between other elements	Data at Rest
Examples	People, other systems, websites	exes, assemblies, COM components	Function Calls	Files, databases, registry keys

Table 1
Shows an example of Diagramming (Shostack, 2008).

A detailed understanding of threat modeling and how to apply it to the real world is essential if working on a tool to help automate threat modeling, especially on the enterprise level, which usually involves lots of valuable moving parts. Before moving into the research methodology and the scope of this paper, it is important to fully understand what enterprise threat modeling is and how it is currently applied to the real world.

3.3 Enterprise Threat Modeling

Enterprise threat modeling aims to create a consistent model or blueprint of an enterprise's structure and organization. The model or blueprint would include the overall goals, processes, and information systems present in the enterprise (Grov et al., 2019). This type of threat modeling is referred to as Enterprise Architecture analysis. When these models are created, they help increase the general understanding of enterprise systems and facilitate various forms of security analysis using attack simulations and threat analysis (Xiong et al., 2022).

When understanding Enterprise Threat Modeling, it is essential to understand Enterprise Architecture (EA) and how it relates to Enterprise Threat Modeling. EA helps to describe the fundamental artifacts of business and IT and how they relate to one another (Xiong et al., 2019). This relation is defined by the National Institute of Standards and Technology and contains the following:

1. Business architecture drives information architecture.
2. Information architecture prescribes information systems architecture.
3. Information systems architecture identifies the data architecture

4. Data architecture suggests specific data delivery systems.
5. Data delivery systems (software, hardware, and communications) support the data architecture.

Due to the nature of enterprises' businesses, understanding this relationship is important so that a proper threat model can be created while having a complete understanding of how the EA operates.

Enterprise models should be applied to enterprises of any size. Even if an enterprise has only a handful of employees, it should still be necessary to model infrastructure for current and future use. Not doing so will cause more issues as the enterprise grows and more is added to keep up with demand. Even if an enterprise does not plan to scale up, having a model to threat analyze will help the enterprise stay more secure and allow future additions to integrate with legacy systems.

Enterprise threat models are unique compared to other threat models since they account for size and scope. For example, Microsoft is responsible for having a 70% market share of the OS ecosystem (*Desktop Operating System Market Share 2013-2023*, 2023). Having a working threat model is essential to organizations like Microsoft, with extensive market share to prevent vulnerabilities and the damage that may result from them. There are downsides to enterprise architecture development, such as no appreciation from project sponsors, lack of time, and insufficient tool support. Intuition is often the only decision driver, with no reuse of already gained knowledge. This usually leads to acceptance issues and quality problems with the software architectures under construction (Zimmermann et al., 2007).

Meta models are at the core of Enterprise Architecture design; they provide a clear view of the structure and dependencies between relevant parts of the organization. As talked about previously, Microsoft employed its security development lifecycle for all of its software products, especially those that posed a significant security or privacy risk. Microsoft's approach exemplifies how threat modeling is used in the enterprise environment. However, it is also an example of how they rely on STRIDE as a primary aspect of their threat modeling style, which is known to involve the manual creation of threat models without the help of automation. The threat modeling methods discussed in the previous section also apply to enterprise threat modeling. However, the models will need to be scalable and readable to be of any use for analysis.

4.0 Proposed Model

After a thorough literature review, a model was developed based on the fundamental principles of enterprise architecture (see Section 3.1) and the overarching purpose of threat modeling (see Section 3.2). Our design is an evolution of the model proposed by Leune and Kim (2021). The proposed model represents the enterprise architecture principles in a threat model format, allowing for analysis of an entire enterprise and its components, depending on which layer is being examined. The business, application, and implementation/technology layers (Figure 8) are represented in the model, each with its components and channels.

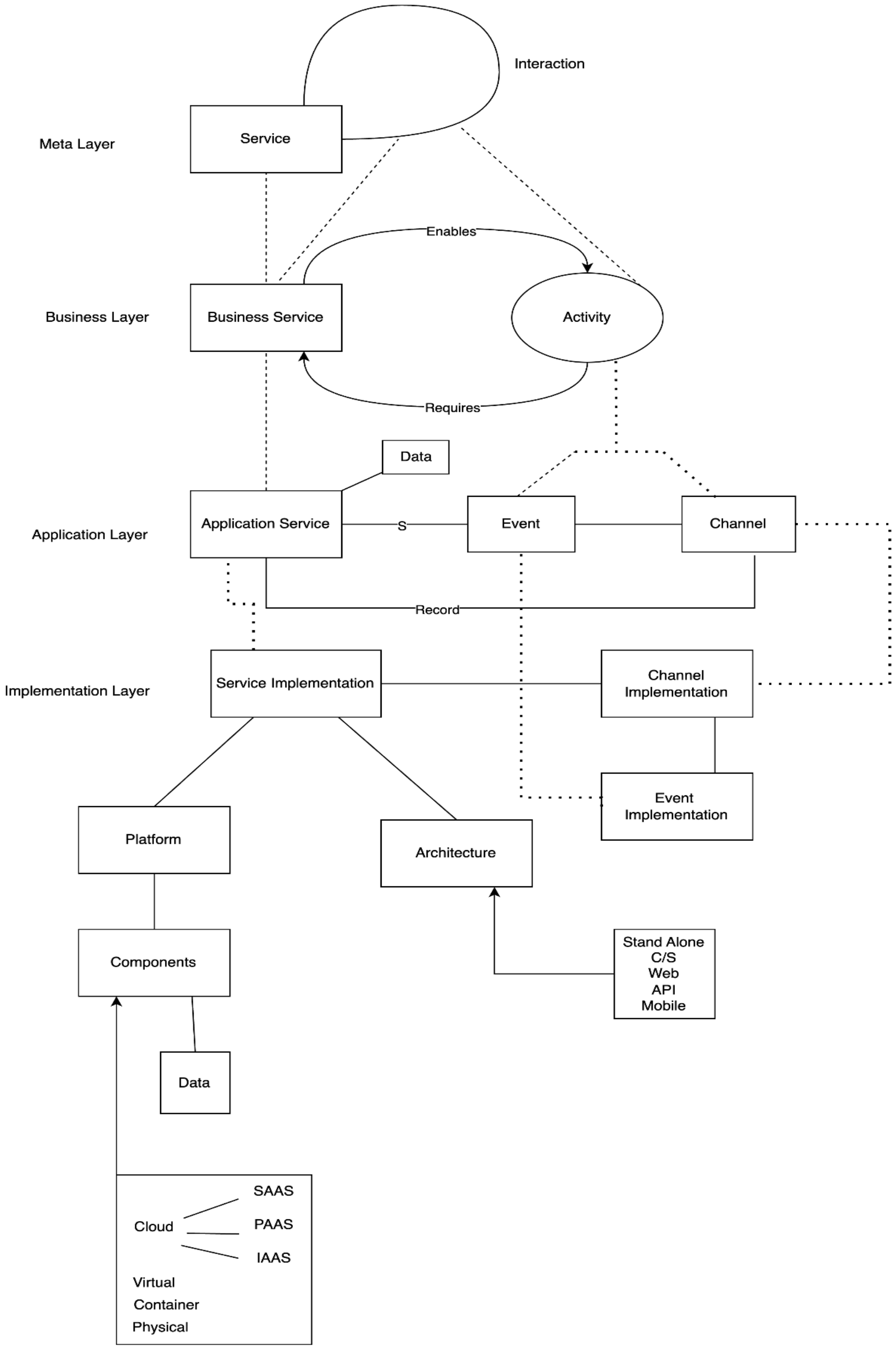

Interaction
Meta Layer
Service
Enables
Business Layer
Business Service
Activity
Requires
Data
Application Layer
Application Service
S
Event
Channel
Record
Implementation Layer
Service Implementation
Channel Implementation
Event Implementation
Platform
Architecture
Components
Stand Alone
C/S
Web
API
Mobile
Data
Cloud
SAAS
PAAS
IAAS
Virtual
Container
Physical

Figure 8,

The literature review derived model showcasing the main principles of enterprise architecture.

4.1 The Meta Layer

4.1.1 Purpose of Layer

The meta layer provides foundational definitions based on the *Service* and how that *Service* interacts with the rest of the model. This provides the model with a connection to the business layer.

4.1.2 Layer Elements

Definition 1 (*Service)*. A *Service* provides a measure of work (Leune and Kim, 2020).

Definition 2 (*Interaction*). An *Interaction* is a way to engage and make use of a *Service*.

The meta layer's modeling primitives are the *Service* and its *Interaction* with the other layers; a service is only connected through interactions.

4.2 The Business Layer

4.2.1 Purpose of Layer

Like the concepts introduced in Section 3.1 (Enterprise Architecture), The *Business Layer* focuses on the business's fundamental organization and requirements. It is based on the organization’s strategy and objectives. These fundamentals are applied to the *Business Layer* and give an overview of what the service will offer the business and its activities.

4.2.3 Layer Elements

Definition 3 (*Business Service*). A *Business Service* is a type of *Service* that focuses on an organization's needs and enables *Activities*.

Definition 4 (*Activity*). An *Activity* is only possible with a *Business Service and* describes what the *Business Service* enables for the organization.

The business layer comprises two modeling primitives: the B*usiness Service* and an *Activity*. The Business Service enables a specific activity and derives directly from the service primitive in the meta layer. For example, an email service will enable a business to use email.

4.3 The Application Layer

4.3.1 Purpose of Layer

The Application Layer identifies the functional components that provide a Business Service. The *Activity*-concept introduced in the Business Layer is refined into *Events* and *Channels*. It focuses on the data passing through, how events and channels feed into the service, and how the application layer bridges the business and implementation layers.

4.3.2 Layer Elements

Definition 5 (*Application Service*). An *Application Service* is a more narrowed-down version of a *Business Service* and details that specify the function being performed by the *Application Service*.

Definition 6 (*Event*). An *Event* is a result of an action taken by the *Application Service.*

Definition 7 (*Channel*). A *Channel* represents a flow of data from one location to the next.

The *Application Layer* consists of the following modeling primitives: The *Application Service, Event, Channel, and Data* primitives. For example, if the *Business Service* is an authentication service, then the *Application*

Service would be a type of authentication service, such as multi-factor authentication (MFA). *Event* and *Channel* are derived from the primitive activity in the business layer.

An *Event* results from a service's actions, such as push notifications and authentication requests. Each action has a flow of data fed through a *Channel*. These *Channels* simply act as a medium and pass data from one location to another, usually for processing and authentication. The data from the *Channel* is recorded and sent to the application.

4.4 The Implementation Layer

4.4.1 Purpose of Layer

The *Implementation Layer* shows how the previous layers are implemented on a hardware and software level. Each layer now has a purpose; the *Business Layer* represents the why, the *Application Layer* is the what, and the *implementation layer* is the how. For each *Application Service*, there must be a *Service Implementation*. Therefore, for each *Business Service*, there must be a *Service Implementation*. Without the *Implementation Layer*, the other layers could not function.

4.4.2 Layer Elements

Definition 8 (*Service Implementation*). A *Service Implementation* is how the *Application Service* is implemented.

Definition 9 (*Channel Implementation*). The *Channel Implementation* is how *Channels* are implemented into the *Service*.

Definition 10 (*Event Implementation*). The *Event Implementation* is how *Events* are implemented into the *Service*.

Definition 11 (*Platform*). A *Platform* is the name of the service, along with its data and hardware.

Definition 12 (*Architecture*). *Architecture* is the connection type being implemented into the service.

Definition 13 (*Components*). Components are the combination of data and *hardware* and how they relate to the *Service Implementation.*

The *Implementation Layer* consists of the following modeling primitives: *Service Implementation, Channel Implementation, Event Implementation, Platform, Architecture, and Components*. The *Service Implementation* derives from the *Application Implementation* and consists of two separate primitives: the *Platform* and *Architecture*. The *Platform* has a set of *Components* that determine the type of data being used, the type of hardware, and the name of the *Service*. These *Services* could be physical, virtual, containerized, or in the cloud as a software, platform, or infrastructure as a *Service* model. The *Architecture* primitive determines the service's connection: mobile, API, web, client/server, and/or stand-alone. The *Event* and *Channel implementations* are derived from the *Event* and *Channel* primitives in the *Application Layer*. The *Event and Channel Implementations* are implemented depending on the service implementation requirements.

5.0 Validation

5.1 Case Study

A case study of Adelphi University's Information Technology infrastructure was conducted to determine the proposed model's validity. The university is a valid choice for a case study due to its enterprise structure and reliance on technology, utilizing both on-premise and cloud-based applications to support the University's academic and administrative processes. The services focused on during the case study are Active Directory, eCampus, Slate, Navigate, and Duo for MFA. Due to their importance for the university, these services are used daily and would be good to implement in the proposed model.

eCampus[1] is an on-premise online web-based portal enabling students, faculty, and administrators to access other applications. Slate[2] is a cloud-based website tracking prospective students. It is used from the moment of inquiry until they are accepted to Adelphi University and have committed to attending by making a tuition deposit. This system enables Admissions to maintain one-to-one contact with students through text or

[1] https://portal.adelphi.edu

[2] https://slate.org/

email. It also facilitates tracking whether students attended events and if housing deposits were made. Navigate[3] is an online cloud-based hub for faculty to view student records, record notes about advising meetings, and issue reports or alerts if needed. It is accessed using a web browser, requiring HTTP over TLS. Navigate also provides a mobile app, which is not used at Adelphi.

Furthermore, while the Navigate platform supports direct use by students, Adelphi has only made the service available to faculty and administrators. Active Directory is an on-premise directory service used to create, manage, and delete users, organizational units, and control groups. Active Directory can only be accessed through Adelphi’s internal network and with Active Directory directly installed on an individual's computer. Finally, Duo[4] is used by the University as a form of MFA for faculty and staff.

5.1.1 Authentication

Authentication is required for each application service used by students, administrators, faculty, and staff. Students and faculty/staff primarily use

[3] https://eab.com/solutions/navigate360/
[4] https://duo.com/

eCampus. Students must only log onto these services with a username and password. They will only use Duo if they become student employees. If students also accept on-campus appointments and need access to protected services, they will be provided with a separate work account. This secondary account may have MFA services associated with it.

In a typical authentication exchange, Faculty and Staff will use both a username/password and Duo to access services provided by Adelphi. Duo generally does not prompt for a second credential when authenticating on the campus network, as access from known IP space is considered a form of 2-factor authentication. However, this varies depending on the security policies applied to each service. After logging in with the correct credentials, Duo will send a push notification to the user's registered cellphone or smartwatch. Alternatively, users can receive an SMS text message or a phone call to their previously known telephone number. If the user passes an authentication challenge, they will authenticate to the service they are accessing, and determining who gets access to what services depends on the employee's role. For example, only the System Administrator teams can access Active Directory. Admissions and at least one individual from each Adelphi academic department have access to Slate.

5.1.2 Logging and Audit

As an educational institution, Adelphi values privacy and confidentiality as a significant priority to ensure secure and protected data. Even during a compromise, limiting what data is available per service is essential to reducing the amount of sensitive information in case of a breach. Slate only keeps student social security numbers accessible to a select few individuals, and any student data is only accessible to those with access to Slate. Auto logs are also available to check when a change to a record is made so that any abnormalities are recorded. Applications will also limit what information is stored. For example, eCampus will usually only store information regarding a user's username and email.

Logs are another way to validate that data is authentic. Whenever a user releases information to an application on the portal or successfully/fails to log in, a log will be created for audit purposes. These logs are stored on a local disk; then, the single sign-on logs are streamed to Splunk, where they are processed. Only the application administrators and information security teams have access to these logs. Active Directory contains the most access on this list; if a malicious actor gains complete control over

Active Directory, they can reset passwords and gain access to any account they wish. Also, they would have access to faculty/staff locations, university ID numbers, emails, and permissions. Fortunately, the list of users with this privilege is only a select few, and logs are created for each transaction. These logs are stored on the domain controllers, readable in event viewers, and exported to Splunk for analysis. All events, authentication, synchronization, and system events are logged and remain on the domain controller until space is needed.

5.1.3 Backups

For redundancy, backups are performed weekly to a tape drive with a year or more shelf life. Duo automatically records a log whenever an authentication, telephony, or administrator action occurs. For privacy, Duo does not have access to any device and will only contain information regarding the device's current version number, security settings, and the names of the services used for Duo authentication.

5.1.4 Integration

Duo is integrated into each of the services in the case study because it is the primary MFA provider. Slate feeds information into SAAS and will export data as needed per department. eCampus integrates with about 150 separate applications using SimpleSAMLphp, directly integrated into Active Directory, Duo, and Azure. Active Directory directly syncs up with Google Workspace for automatic Google Account creation, deletion, and updates.

These services represent a good test for the model; the infrastructure is extensive and relied upon for the college's successful operation. Using these services as a case study will allow the model to be incorporated into the university to determine what systems could be vulnerable and what can be mitigated.

5.2 Case Study Results

I validated the proposed modeling approach by using it to formally represent my case study findings. The results from the case study showed examples of the three key enterprise architecture domains: the business, application, and infrastructure domains. The information gathered during the study is also transferable to a model format. Given these two connections, I validated our proposed modeling approach by using it to represent our case study findings formally. To illustrate this approach, I examined Adelphi University's authentication systems and used our proposed method to represent these findings.

5.3 Business Layer in Case Study

The business architecture domain focuses on the business's fundamental organization and requirements based on its strategy and objectives. This is associated with the *Business Layer,* as Adelphi's authentication services contribute to the university’s efforts to maintain security amongst faculty and staff accounts. Users require protection so that their individual data is private and only accessible to them. However, no system is foolproof, with weak passwords and limited MFA options allowing for unwanted intrusions.

Keeping accounts safe and secure will enable Adelphi to stay safe and prevent intrusions that may compromise Adelphi’s business as an educational institution. Using an authentication service also enables faculty and staff to access secure information they would be unable to access otherwise.

When the proposed business layer modeling primitives are applied to the case study, the following model is created in Figure 9:

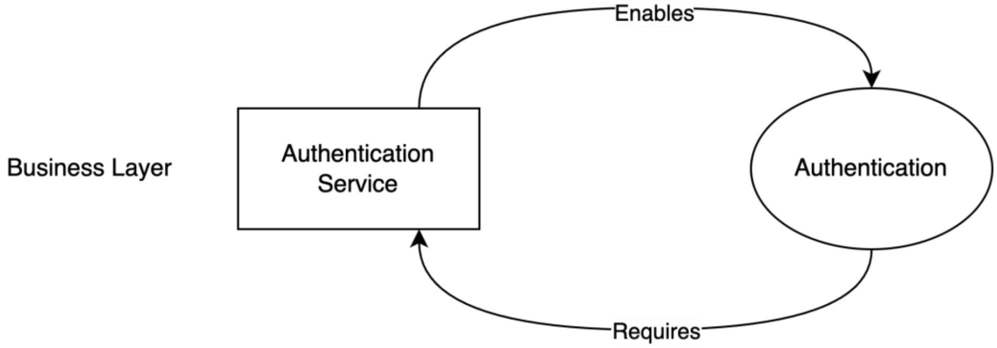


Figure 9

Shows the business layer case study applied to the proposed model.

Figure 9 shows that the Authentication Service(Business Service) described above enables Authentication (the Activity). Meanwhile, Authentication requires the Authentication Service to operate.

5.4 Application Layer in Case Study

The *Application Layer* focuses on the authentication service and consists of three components: the application services, the events, and the channels. *Application Services* are the *Services* being provided. For the Case Study, we focus on the MFA service. A user attempting to log in would trigger an authentication *Event*, and the *Channel* is the data exchanged between the service and the service provider. Each component seen in Figure 10 will be examined further in the following three subsections.

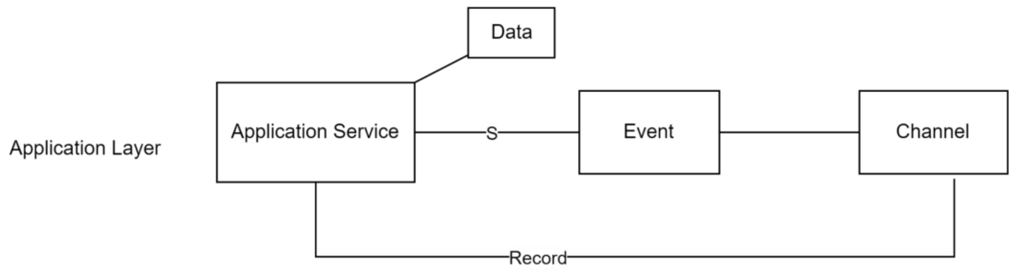


Figure 10

This figure shows the application layer.

5.4.1 Application Services

Adelphi employs an MFA service as an extra layer of security when faculty and staff access sensitive applications. This additional authentication also protects against malicious users with the username and password but not access to the legitimate user's MFA device. The first authentication layer is username/password authentication. When a user is authenticated, the MFA prompts the user for a token, usually paired with the user's device.

5.4.2 Events

An *Event* is triggered when a user attempts to log into an Adelphi that requires MFA. Given that MFA is used by faculty and staff, they would have to authenticate through the application, and it depends on where the user is located and which application they are trying to access. The *Event* could result in an MFA push, where the user presses a button on a secondary device to confirm they sent a request, response to an SMS/voice call request, where a code is sent to the user's phone number for confirmation or an authenticator code that the user must enter which refreshes every number of seconds on the authenticator app. If an authentication fails, the application issues another *Event* back to the MFA service, denying the user access. For example, if a user is trying to access an application with MFA, they will receive a notification asking to confirm their identity through a push notification; after confirming or denying, the *Event* is sent back to the MFA *Service* with the user’s response which will determine whether or not the user is granted access.

5.4.3 Channels

Channels allow the exchange of events between application services. For example, when a Duo authentication event is requested. The request is packaged and sent to the Duo cloud server, where the information is analyzed and then returned to the originating device with a pass-or-fail confirmation.

5.4.4 Application Layer Model

When the proposed application layer modeling primitives are applied to the case study, the following model is created.

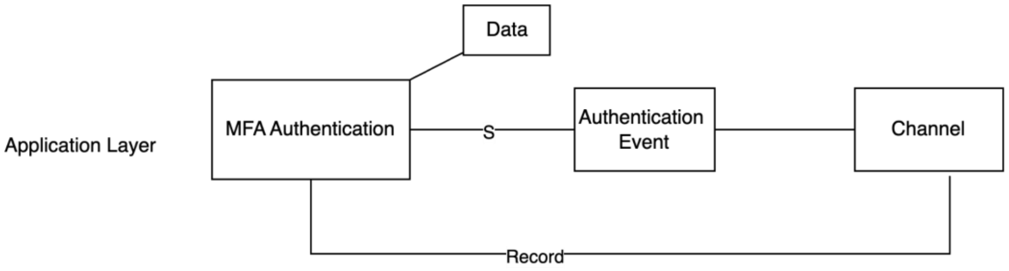


Figure 11

This shows the application layer case study applied to the proposed model.

Figure 11 shows that the MFA Authentication(*Application Service*) sends an Authentication Event(*Event*), which is then received by a channel and recorded to the MFA authentication.

5.5 Implementation Layer in Case Study

Adelphi employs an MFA called Duo to enable push and code authentication. As discussed earlier in Section 5.1, Duo is used by faculty and staff to access specific data-protected applications that require additional security. First, the faculty/staff member enters a username and password, and if successful, an additional authentication step is performed using Duo. If the user is on campus, additional IP address checks may be used if the application supports it, as this would count as another form of MFA.

Duo is provided as Software-as-a-Service and owned by the digital communications company Cisco. It acts as a *Platform* for MFA. Clients access the service *Platform* through the Duo Mobile app on a cellular device via SMS, text, or voice telephone. The *Architecture* for Duo is web-based. *Event Implementations* are implemented into the authentication app, with the *Channel Implementation* implementing a connection between the user's authentication method and the Duo cloud service for process and review. When the proposed *Implementation Layer* modeling primitives are applied to the case study, the following model in Figure 12 is illustrated.

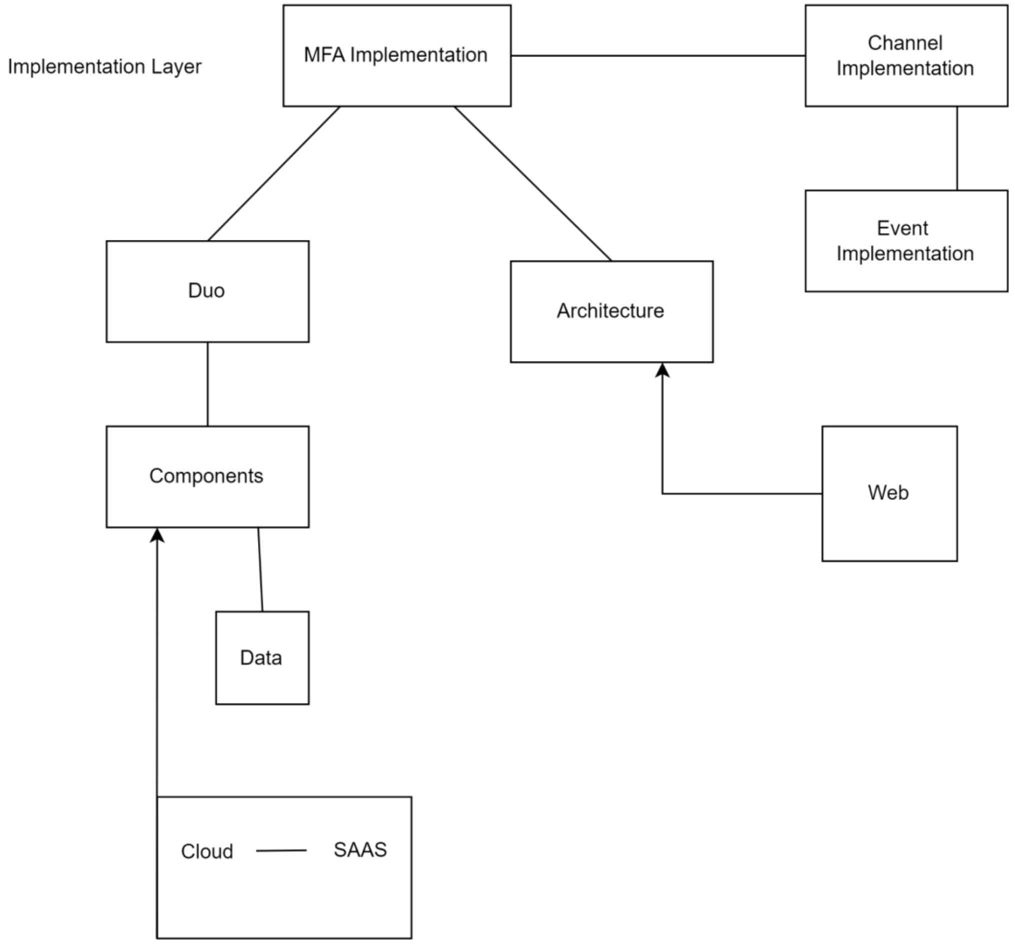


Figure 12

This shows the *Implementation Layer* case study applied to the proposed model.

Figure 12 shows that the *Implementation Layer* comprises the MFA Implementation(*Service Implementation*). The MFA Implementation contains the *Channel Implementation*, which subsequently contains the *Event Implementation*. The MFA Implementation also includes

Duo(*Platform*) and the *Architecture*, which is Web-based. Duo comprises *components* that consist of *Data* and the cloud as a platform and a service.

5.6 Multi-Factor Authentication Model

Based on the three domains, the MFA service fits into the proposed model to illustrate the interconnectivity between each domain and how the model can demonstrate an Enterprise Threat Modeling Approach to a real-world Enterprise. Now that all three layers have been incorporated into the proposed model, each can be linked to form the following model in Figure 13, with the *Meta Layer* being added to indicate the service and its interaction with the *Business Layer*.

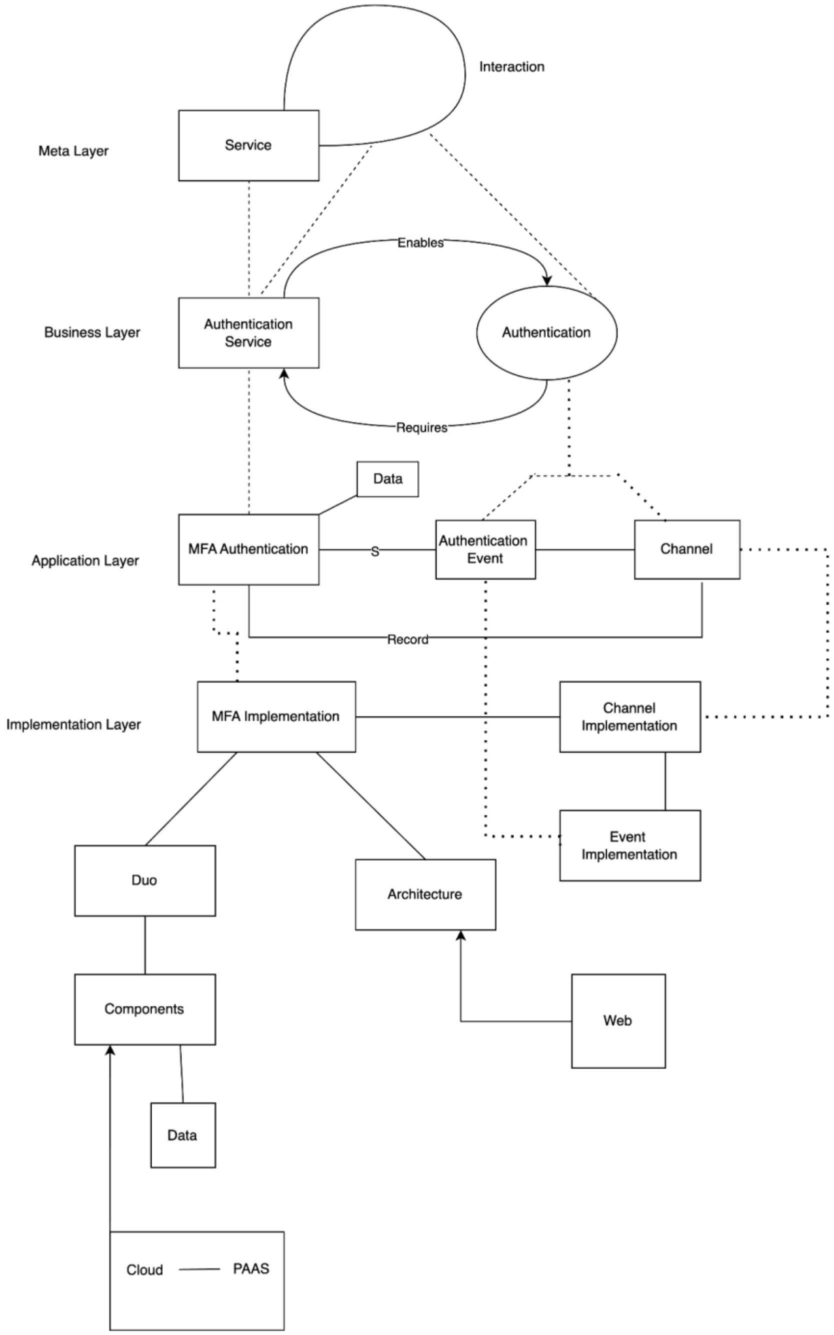


Figure 13

The case study applied to the proposed model.

6.0 Conclusion And Future Work

In this thesis, I set out to answer the following questions: Are CTA models effective for SOEM? How beneficial would automation and maintenance of CTA models be for SOEM? Does automatically creating and maintaining CTA models for SOEM help improve cybersecurity situational awareness? How readable are large-scale automated models? Ultimately, I was able to answer only one of our research objectives

CTA models are effective for SOEM, as indicated by the proposed model and the case study. The proposed model is based on the Enterprise Architecture TOGAF framework and incorporates threat modeling, when combined, creates an enterprise threat model. A case study of Adelphi University's Information Technology infrastructure was conducted to determine the proposed model's validity. The model held true when Adelphi’s MFA service was applied to the proposed model. The enterprise business, application, and infrastructure domains were reflected in the model as the business, application, and implementation layers.

The other three questions dealt with the automation of the proposed model, but due to time constraints, the automation tool still needs to be fully implemented. Therefore, the questions cannot be answered as of right now. However, preliminary code results (see Appendix A) show that automation could be possible by using inherited classes to illustrate the proposed model and the case study. Future work will explore these results as the code is refined into possible automated code.

Bibliography

Brocke J, Hevner A, Maedche A,. (2020). Introduction to Design Science Research. 10.1007/978-3-030-46781-4_1.

Cyberthreat Analysis Tool - Identify Security Threats. (n.d.). SolarWinds. Retrieved January 20, 2024, from https://www.solarwinds.com/security-event-manager/use-cases/cyber-threat-analysis

Dumitriu D and Popescu M, (2020). Enterprise Architecture Framework Design in IT Management, Procedia Manufacturing, Volume 46, 2020, Pages 932-940, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2020.05.011.

Desktop operating system market share 2013-2023. (2023, September 4). Statista. Retrieved October 29, 2023, from https://www.statista.com/statistics/218089/global-market-share-of-windows-7/

Duarte, F. (2023, April 3). *Amount of Data Created Daily (2023)*. Exploding Topics. Retrieved October 10, 2023, from https://explodingtopics.com/blog/data-generated-per-day

Behrouz F and Fathollah M, (2016). A Systematic Approach to Enterprise Architecture Using Axiomatic Design, Procedia CIRP, Volume 53, 2016, Pages 158–165, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2016.07.012. (https://www.sciencedirect.com/science/article/pii/S2212827116307478)

Federal Enterprise Architecture Framework. (2023, September 6). CMS. Retrieved November 16, 2023, from https://www.cms.gov/data-research/cms-information-technology/enterprise-architecture/federal-enterprise-architecture-framework

Gerber, A., le Roux, P., Kearney, C., & van der Merwe, A. (2020). The Zachman Framework for Enterprise Architecture: An Explanatory IS Theory. Responsible Design, Implementation, and Use of Information and Communication Technology: 19th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2020, Skukuza, South Africa, April 6–8, 2020, Proceedings, Part I, 12066, 383–396. https://doi.org/10.1007/978-3-030-44999-5_32

Grov, G., Mancini, F., Mestl, E.M.S. (2019). Challenges for Risk and Security Modelling in Enterprise Architecture. In: Gordijn, J., Guédria, W., Proper, H. (eds) The Practice of Enterprise Modeling. PoEM 2019. *Lecture Notes in Business Information Processing*, vol 369. Springer, Cham. https://doi.org/10.1007/978-3-030-35151-9_14

Hen, H., Andry., (2021). Enterprise Architecture Design Using TOGAF ADM Framework (SME Case Study: Dormitory House). 9. 95–99.

Hussain S, Kamal A, Ahmad S, Rasool G, Iqbal S,. (2014). THREAT MODELLING METHODOLOGIES: A SURVEY. 26. 1607-1609.

IDG (2020). 2020 IDG Cloud Computing Survey. Technical report, idg.

Leune, K and Kim, S. (2020). Supporting Cyber Threat Analysis with Service-Oriented Enterprise Modeling. *In Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 385-394.

KTH. (2018, January 17). KTH. Retrieved October 10, 2023, from https://www.kth.se/cs/nse/research/software-systems-architecture-and-security/projects/old-projects/cysemol/description-1.432380

Yskout K., Heyman T., Van Landuyt D., Sion L., Wuyts K., and Joosen W., "Threat modeling: from infancy to maturity," 2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), Seoul, Korea (South), 2020, pp. 9–12.

Maslin, S. (2022, November 28). *How a Cyberattack Plunged a Long Island County Into the 1990s*. The New York Times. Retrieved September 24, 2023, from https://www.nytimes.com/2022/11/28/nyregion/suffolk-county-cyber-attack.html

Morgan, S. (2022, December 8). *Cybercrime To Cost The World $10.5 Trillion Annually By 2025*. Cybercrime Magazine. Retrieved October

14, 2023, from https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

The Open Group Adoption Strategies Working Group. (2010, April). *Case Study/White Paper Template*. The Open Group Publications Catalog. Retrieved November 21, 2023, from https://pubs.opengroup.org/onlinepubs/7698999699/toc.pdf

Rohloff, M. (2005). Enterprise Architecture - Framework and Methodology for the Design of Architectures in the Large.. 1659–1672.

Shostack, A. (2008). Experiences Threat Modeling at Microsoft.

Shostack, A. (2014). *Threat Modeling: Designing for Security*. Wiley.

Stevens, R., Votipka, D., Redmiles, E.M., Ahern, C., Sweeney, P., & Mazurek, M.L. (2018). The Battle for New York: A Case Study of Applied Digital Threat Modeling at the Enterprise Level. *USENIX Security Symposium*.

Venable, J., Pries-Heje, J., Baskerville, R. (2014, November 11). European Journal of Information Systems. *FEDS: a Framework for Evaluation in Design Science Research*, *25*, 77–89. https://link.springer.com/article/10.1057/ejis.2014.36#citeas

White, S. K. (2022, November 23). *What is enterprise architecture? A framework for transformation*. CIO. Retrieved November 14, 2023, from https://www.cio.com/article/222421/what-is-enterprise-architecture-a-framework-for-transformation.html

W. Xiong, P. Carlsson, R. Lagerström, "Re-using Enterprise Architecture Repositories for Agile Threat Modeling," 2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW), Paris, France, 2019, pp. 118–127, doi: 10.1109/EDOCW.2019.00031.

Xiong, W., Legrand, E., Åberg, O. et al. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. Softw Syst Model 21, 157–177 (2022). https://doi.org/10.1007/s10270-021-00898-7

Ylimäki, T and Halttunen, V. (2005). Method engineering in practice: A case of applying the Zachman framework in the context of small enterprise architecture-oriented projects. Information Knowledge Systems Management. 5. 189-209.

Zimmermann, O, Gschwind, T, Küster J, Leymann, F, Schuster N,. (2007). N.: Reusable Architectural Decision Models for Enterprise Application Development. In: Overhage, Sven (ed.); Szyperski et al. (ed.); Reussner, Ralf (ed.); Stafford et al. (ed.): Third International Conference on the Quality of Software-Architectures (QoSA, 2007), pp. 15-32. 10.1007/978-3-540-77619-2_2.

Z. Shi, K. Graffi, D. Starobinski, N. Matyunin, "Threat Modeling Tools: A Taxonomy," in IEEE Security & Privacy, vol. 20, no. 4, pp. 29–39, July-Aug. 2022, doi 10.1109/MSEC.2021.3125229.

Appendix

Appendix A

```
#Chris Benson
#case-study.py

from application import *

def setup_structures():
   #Splunk Log Service
   SEMS = ApplicationService() #Splunk Event Management Service

   #Login service
   loginSvc = ApplicationService()

   #Duo Example...
   duoMFARespCh = Channel(service=loginSvc)
   duoMFARespEvt = Event(channel=duoMFARespCh)
   duoSvc = ApplicationService(events=[duoMFARespEvt])

   #ECampus Example
   eCampusRespCh = Channel(service=loginSvc)
   eCampusRespEvt = Event(channel=eCampusRespCh)
   eCampusSVC = ApplicationService(events=[eCampusRespEvt])

   #Log Example
   directoryService = ApplicationService()
   domainEvent = Event() #Indexing channel?\
   #


setup_structures()
```

```
#meta.py
class Service:

   __interactions = [] # A list of other service with which we
Interact
```

```
   pass
```

```
#business.py
from meta import Service

from typing import List, ForwardRef #ForwardRef allows reference to
class not yet defined


class BusinessService(Service):  #BusinessService inherits Service
   def __init__(self, requires: List[ForwardRef('Activity')]): # a
list of Activities required to provide this Service
       self.requires = requires
       pass

class Activity:
   def __init__(self, enables: List[BusinessService]): # a list of
BusinessServices enabled by this Activity
       self.enables = enables
       pass
```

```
#application.py
from business import BusinessService

class MissingServiceException(Exception):
   pass


class Event:
   def __init__(self, channel=[]):
       self.channel = channel #The Channel via which this event can be
sent


class ApplicationService(BusinessService):
   def __init__(self, events: list[Event] = []):
       self.events = events # list of Events that can be sent by this
service
   pass
```

```
class Channel:
   def __init__(self, service: ApplicationService = None):
       if service is None:
           raise MissingServiceException()
       self.__service = service # The service to which this channel
provides its events
```

```
#implementation.py
from application import ApplicationService
from application import Channel
from application import Event

from typing import List, ForwardRef #ForwardRef allows reference to
class not yet defined

class Platform:
   def __init__(self, components: List[ForwardRef('Component')] =
None):
       self.components = components # Components

class Architecture:
   def __init__(self, serviceType):
       self.serviceType = serviceType #Just the type of service this
provides

class ServiceImplementation(ApplicationService):
   def __init__(self, platform: Platform, architecture: Architecture,
events: List[ForwardRef('EventImplementation')]):
       self.platform = platform
       self.architecture = architecture
       self.events = events # a list of EventImplementations


class ChannelImplementation(Channel):
   def __init__(self, service: ServiceImplementation):
       self.service: ServiceImplementation

class EventImplementation(Event):
   def __init__(self, channel: ChannelImplementation):
       self.channel = channel


class Component:
   def __init__(self, data, contains=None, ):
       if contains is None:
           self.contains = []
       self.contains = contains# Components, string perhaps?
       self.data = data #TBD, what is data?


class CloudComponent(Component):
   pass

class SAASImplementation(CloudComponent):
   pass

class PAASImplementation(CloudComponent):
   pass

class IAASImplementation(CloudComponent):
   pass
```