



T100

Modelling SABSA® with ArchiMate®

Release 2.0

A SABSA Tools & Techniques Paper

Published by The SABSA Press™, an imprint of The SABSA Institute™

September 2021

Copyright © 2021, The SABSA Institute C.I.C. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the copyright owners unless it is presented in its current form as published by The SABSA Institute.

Document Title: Modelling SABSA® with ArchiMate® Release 2.0.
A SABSA Tools & Techniques Paper) 2021

Document Number: TSI T100

Published by The SABSA Press, (a trading name of The SABSA Institute C.I.C.) September 2021.

Submit comments relating to the material contained in this document to:

The SABSA Institute C.I.C, 126 Stapley Road, Hove, BN3 7FG, UK

Or by electronic mail to:

sabsapress@sabsa.org

Registered in England and Wales, No. 08439587

Trademarks

SABSA® is a registered trademark of The SABSA Institute. Other trademarks owned by The SABSA Institute are labelled with a TM mark on their first occurrence in the text.

ArchiMate®, TOGAF® ADM® & 'Boundaryless Information Flow®' are registered trademarks of The Open Group.

All other brands, companies, and product names are used for identification purposes only and may be trademarks that are the sole property of their respective owners.

This Document

This SABSA Tools & Techniques Paper describes how security architecture concepts can be expressed using ArchiMate, The Open Group's widely adopted Enterprise Architecture modelling language.

It extends and updates the 1.0 release of this paper to reflect the further work of The SABSA Institute's Modelling SABSA in ArchiMate (MSA) Working Group between June 2020 and June 2021.

It has been developed and approved by The SABSA Institute C.I.C. Board of Trustees.

Acknowledgements

Author: Steven Bradley

The MSA Working Group: Jonathan Bentley, Murray Booth, Carlos Dominguez, Patrick Dunstan, Duncan Hart, Gregory Hoel, Renil Lambert & Richard Morgan.

Contributors and Reviewers: John Czaplewski, Editor-in-Chief, The SABSA Institute; John Sherwood, Chief Architect, The SABSA Institute; Maurice Smit, Deputy Chief Architect, The SABSA Institute; John Linford, Security Forum Director, The Open Group.

Table of Contents

Section 1: Context, Motivation and Goals	5
1 Introduction	6
1.1 Background	7
1.2 What is ArchiMate?	8
1.3 Purpose	9
1.4 References	10
2 The Rationale for SABSA-ArchiMate Alignment	11
2.1 The Benefits of Modelling	11
2.2 The Case for an ArchiMate Security Perspective	13
2.3 The Benefits that Modelling can bring to SABSA	13
2.4 Vendor Neutrality	15
3 An Introduction to the ArchiMate Language	16
3.1 Core Elements	16
3.2 Core Relationships	16
3.2.1 Definition and Notation of Relationships	17
3.2.2 Derived Relationships	17
3.2.3 The Core Layers and their Elements	18
3.3 Extension Layers and Elements	21
3.3.1 Strategy Layer Extension	21
3.3.2 The Motivation Extension	21
3.3.3 Implementation and Migration Extension	22
3.3.4 Miscellaneous Elements	22
3.4 ArchiMate Extensibility	23
3.4.1 User-Defined Properties	23
3.4.2 Specialisations and Stereotypes	23
3.4.3 Overloaded Relationships	23
3.5 The ArchiMate 3.1 Framework	24

Section 2: The Security Overlay	25
4 Aligning the SABSA and ArchiMate Frameworks	26
4.1 An Overview of the Task	26
4.2 Risk & Security Modelling in ArchiMate	27
4.3 The Basic Element and Relationships	31
5 The Motivation Aspect	32
5.1 Value & Loss	32
5.2 Value Chain	34
5.3 SABSA Business Attributes	35
5.3.1 Structural Placement of Business Attributes	36
5.3.2 Traceability of Business Attributes	38
5.4 Meaning	40
5.5 Impact, Threat, Vulnerability & Risk	41
5.6 Controls: Objectives, Requirements and Measures	42
5.7 Multi-Tiered Security	43
Compliance	44
5.8 Regulations and Standards	44
5.9 Articles, Mandates and Compliance Objectives	46
5.10 Control Mechanisms	48
5.11 Trust	49
6 Modelling the Contextual Security Architecture	50
6.1 Business Assets	52
6.1.1 Capability and Value Stream	52
6.1.2 Business Object	53
6.1.3 Business Service, Interface and Service Level Agreement (SLA)	54
6.2 Business Risk	55
6.3 Business Process / Function / Interaction	55
6.4 Business Roles & Actors	57
6.4.1 Governance	58

6.4.2	Threat Actors	60
6.5	Business Geography	61
6.6	Business Time Dependencies	61
7	Modelling the Conceptual Security Architecture	62
7.1	Attribute Profiling	64
7.2	Risk Management & Strategy	66
7.2.1	Risk Management	66
7.2.2	Policy Architecture	67
7.2.3	Multi-Regulatory Compliance.....	69
7.3	Conceptual Security Services.....	71
7.4	Identity and Trust	72
7.4.1	Identity and Access Rights	72
7.4.2	Roles and Responsibilities.....	75
7.4.3	Trust.....	76
7.5	Domain Framework Model.....	79
7.6	Security Events	80
8	Modelling the Logical Security Architecture	81
8.1	Information Assets	81
8.1.1	Application Components	82
8.1.2	Security Configuration	83
8.1.3	Software Defects and Malware.....	83
8.1.4	Data Assets.....	83
8.2	Risk Modelling	85
8.2.1	Risk Modelling.....	85
8.2.2	The Scenario.....	85
8.3	Application Functionality and Services	88
8.4	Logical Access Management.....	90
8.4.1	Account	91
8.4.2	Application Role	91

8.4.3	Application Service	91
8.4.4	Application Process and Function	92
8.4.5	Application Interface	92
8.5	Logical Domains	94
8.6	Timing and Events	94
8.6.1	Application Security Events	94
9	Modelling the Physical Security Architecture	95
9.1	Data and Technology Assets.....	96
9.1.1	Artefact.....	96
9.1.2	Device and Node	97
9.2	Risk Management Practices	98
9.2.1	Defect.....	98
9.3	Process Mechanisms.....	99
9.3.1	Technology Functions and Services.....	100
9.3.2	System Software.....	100
9.4	Human-Machine Interfaces.....	100
9.4.1	Technology Interface	100
9.5	Physical Environment	100
9.6	Timing and Interrupts.....	100
9.6.1	Technology Security Events	100
10	Conclusion	104

Section 1: Context, Motivation and Goals

This section describes the context for this paper, its motivation and sets out its goals;

1 Introduction

This White Paper discusses how security architecture concepts can be expressed using ArchiMate, The Open Group's widely-adopted Enterprise Architecture (EA) modelling language. It describes a model-based approach to creating SABSA artefacts that conforms with standard EA notation and tooling.

The integration of security into EA methodologies, through the alignment of SABSA concepts with Zachman, TOGAF ADM and other popular frameworks, has been formally established for several years. In practice however, the lack of native support for security concepts in EA modelling notations (and therefore tools and processes) means that security architects have been at a disadvantage to their architectural peers.

A means of expressing security perspectives in ArchiMate would fill a deficiency impeding the realisation of a truly holistic architecture methodology that can serve the needs of all architecture practitioners with a concern for security, not just security architects.

Introduction to Version 2

The original T100 v1, published in Sept 2018, set out a case for including security concepts in the ArchiMate modelling language. It was intended as pragmatic advice for security modellers but nevertheless represented only the early experiences and ideas of a small number of practitioners in an emerging domain.

Encouraged by feedback from subject-matter experts in the SABSA and EA modelling communities, a Joint Working Group was established with The Open Group to develop these ideas toward a settled consensus of core security elements, relationships, and properties, referred to collectively as the Security Overlay. This update to T100 represents the conclusions of this Working Group's discussion and consideration.

Our ambition is for the Security Overlay to evolve into a practical resource for the ArchiMate community, offered as a considered contribution from experienced SABSA practitioners and subject-matter experts. By proposing a reference of vocabulary, patterns, and conventions in this domain, we hope to progress towards standards, further enterprise security as a practice, and encourage the wider appreciation of security concepts in general and the SABSA approach in particular.

A credible and well-considered Security Overlay should relieve each member of the emerging security modelling community from having to resolve these issues for themselves, enabling them to concentrate instead on modelling the security perspective for their systems of interest. Consolidation around such a standard would also incentivise tool-makers to facilitate support for security within their products. While this effort may or may not become that definitive standard, it is hopefully a significant contribution toward it.

This revision is a timely update that extends and enhances the original paper that reflects the work of the MSA Working Group and the general evolution of the ArchiMate language. However, the topic of security modelling in ArchiMate is by no means complete and concluded. Although the MSA Working Group, having completed its original mandate, is no longer formally constituted, the Tools & Techniques community at TSI continues as an Interest Group. Any comments, feedback or suggestions are welcomed via the Institute site.

1.1 Background

In the not-too-distant past, information security was a collection of disparate disciplines (people, processes and technologies) that were detached from the main body of enterprise architecture. Yet it is self-evident that security architecture cannot and does not exist in isolation. Its purpose is to protect the organisation's information assets and capabilities and produce artefacts that guide the Enterprise Architecture development.

Thankfully this demarcation is no longer the norm, having been challenged, first conceptually by the philosophy developed in the 'Blue Book'¹ and the subsequent integration and alignment of the SABSA approach with other EA frameworks², including TOGAF, ITIL, Zachman and DoDAF. SABSA was formally aligned and integrated with TOGAF via the 2011 Joint White Paper [Ref.3] and is now included in the Open Group Guide to Security Architecture in the TOGAF 9.2 Library [Ref. 11].

The Open Group / SABSA collaboration aligns and integrates the essential concepts of the two frameworks. It recognises the need for a truly holistic approach. Nevertheless, there are practical impediments to adopting an integrated methodology: the lack of security support in popular EA modelling languages and their implementation in modelling tools and development environments.

A model-based approach is as essential to security as it is to any other design and engineering discipline. Models amplify the efficiency and effectiveness of practitioners and provide a valuable means of establishing a common understanding with stakeholders: a pre-requisite for quality. And yet the analyst's toolkit (Threat Models, Trust Models, Risk Analysis, Control Traceability, etc.) are too often a collection of disparate artefacts, detached from central EA models. Consequently, security artefacts tend to be costly to create and maintain, synchronised and coherent with the main body of system documentation and therefore poorly integrated with related domains such as Enterprise Risk Management, Governance or Compliance.

Ultimately, real-world systems recognise a single, de-facto architecture: what ISO 42010³ calls the 'System of Interest' (SOI). It makes sense to strive towards a single holistic model⁴ of that SOI, capable of describing, validating, querying and analysing all its pertinent functional and non-functional aspects, including the security perspective.

¹ Ref.1: Enterprise Security Architecture: A Business-Driven Approach. John Sherwood, Andy Clark, and David Lynas. Boca Raton, FL: Taylor & Francis Group, LLC. 2005.

² The SABSA Institute: SABSA Executive Summary

³ ISO/IEC/IEEE 42010:2011 - Systems and Software Engineering -- Architecture Description

⁴ or a series of interconnected models forming a complete Architectural Description (AD).

1.2 What is ArchiMate?

ArchiMate is an open and independent modeling language for Enterprise Architecture published by The Open Group. It supports the description, analysis and development of Enterprise Business & IT Systems in the same way that the Unified Modelling Language (UML) describes software design or the Business Process Modelling Language (BPML) describes business processes.

The ArchiMate Framework resolves enterprise architecture into three core layers: Business, Logical (Applications & Data), and Technology Infrastructure.

In each layer, three aspects are considered: *active structure elements* (actors, roles, applications) perform *behaviour* (services realised by processes & functions) on *passive structural elements* (information, data) via a constrained set of relations. Figure 1 shows a simple example.

Subsequent releases have supplemented this core with additional layers and perspectives that now include: Strategy, Motivation, Physical and 'Implementation & Migration'. A brief primer on the notation is presented in Section 3: 'An Introduction to the ArchiMate Language'.

The current release (version 3.1, November 2019) marks the second decade of the language's maturation⁵. Its popularity is the result of its essential usefulness (developed through practitioners' feedback from earlier versions), its concise but expressive notation (easy to learn, yet capable of communicating complex patterns with precision and clarity); a wealth of literature covering everything from language semantics to good modelling style, the availability of certified, compliant modelling tools to suit all budgets (including excellent freeware) and the portability of models (via a standardised ArchiMate Exchange Format).

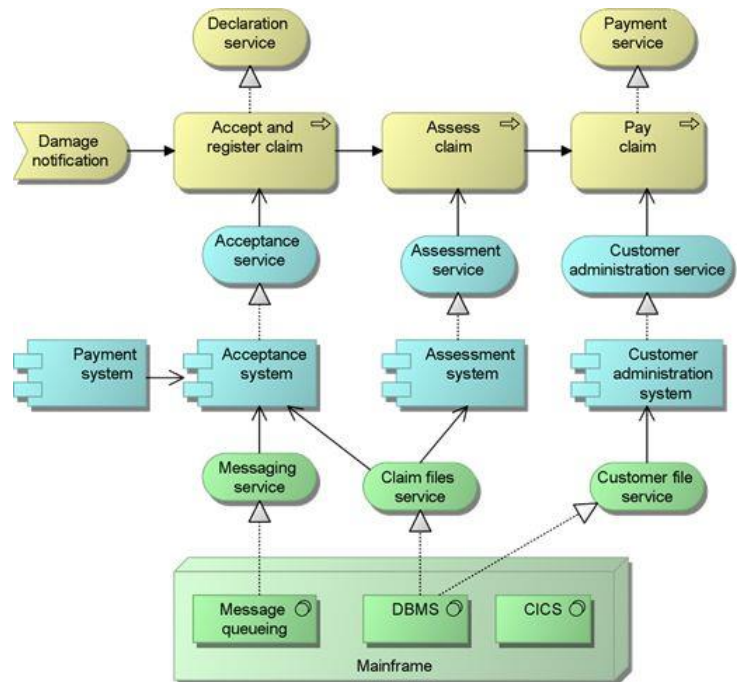


Figure 1: A small ArchiMate Diagram

⁵ ArchiMate originated from the Telematica Instituut in the Netherlands in cooperation with several Dutch partners from government, industry and academia. In 2008, stewardship was transferred to the ArchiMate Forum within The Open Group. The ArchiMate® 1.0 standard was first published as a formal technical standard in 2009.

Currently there are over 1000 certified ArchiMate practitioners worldwide and tens of thousands of ArchiMate resources have been downloaded from The Open Group's site. Archi⁶, a popular free modelling tool, is regularly being downloaded several thousand times per month.

Despite this progress, the least developed aspect of the notation remains its lack of specific support for security. This appears to have its roots in the ArchiMate manifesto, which prioritises the needs of the many (architects) over those of the few. As a result, security has occupied the no man's land between being deemed too niche to be part of the core language and yet, we would argue, too important and pervasive to be left out.

1.3 Purpose

This White Paper explores how to express security concepts in ArchiMate using only the in-built extensibility capabilities of the existing notation. It aims to be as complete as reasonably possible, using the ability to create the artefacts identified in the SABSA framework as a benchmark. This should enable a unified EA/ESA modelling environment capable of supporting in practice what the Joint White Paper [Ref. 3] and Open Group Guide to Security [Ref. 11] have established in theory.

The analysis explores the following questions:

- What is the rationale for using ArchiMate to model security concerns in general and SABSA in particular?
 - What motivates the goal of expressing security in a modelling language?
 - What does a model-driven approach offer to SABSA practitioners and other security stakeholders?
- Which kinds of security artefact are required from an ArchiMate model?
- What are the gaps and limitations of the current language specification?
- Which new elements, relationships, properties and viewpoints are needed to create these artefacts?

The analysis leads to two main focus areas:

- What can currently be achieved within the constraints of ArchiMate 3.1 language and certified tools?
- What might further be achieved via a security-specific extension to the core language, and what would such an extension look like?

⁶ Ref.9: The Archi Modelling Tool

1.4 References

Table 1: References

Ref.	Title	Source	Date
1	Enterprise Security Architecture: A Business-Driven Approach	Sherwood et al.	2005
2	What is the benefit of a model-based design of embedded software systems in the car industry?	Manfred Broy et al. Technical University Munich	2011
3	TOGAF® and SABSA® Integration	The Open Group & SABSA Institute	Oct 2011
4	How to Model Enterprise Risk Management and Security with the ArchiMate® Language	The Open Group	Mar 2017 Updated Nov 2017 ⁷
5	The ArchiMate® 3.1 Specification	The Open Group	Nov 2019
6	ArchiMate 3.1 Reference Cards	The Open Group	Nov 2019
7	Mastering ArchiMate –Edition 3.1	Gerben Wierda R&A Enterprise Architecture	Apr 2021
8	Digital Identity Guidelines	NIST Special Publication 800-63-3	Dec 2017
9	The Archi Modelling Tool	https://www.archimatetool.com	
10	R101: SABSA Matrices 2018	The SABSA Institute Academic Board	June 2018
11	Integrating Risk and Security within a TOGAF® Enterprise Architecture	The Open Group Library	March 2019
12	How to Model Enterprise Risk Management and Security with the ArchiMate® Language	The Open Group Library	Nov 2019
13	The Unified Compliance Framework	https://www.unifiedcompliance.com	N/A
14	The NIST Cybersecurity Framework	https://www.nist.gov/cyberframework	N/A

⁷ Updated for alignment with the ArchiMate 3.0.1 release.

2 The Rationale for SABSA-ArchiMate Alignment

Before immersion in a discussion of the “what” and “how” of security modelling, it’s worth stepping back to reflect on the “why”. What can we hope to achieve by augmenting ArchiMate’s security modelling capability?

2.1 The Benefits of Modelling

Models are used in virtually all science and engineering discipline to represent complex ideas or systems. They serve in many ways: to document, query, analyse, explain, explore, validate, plan, teach or predict a system’s structure or behaviour without the need for interaction with the system itself.

The ICT discipline’s drive to digitalisation for agility and competitiveness demands that organisations actively manage their enterprise architecture. The complexity of these systems is a significant challenge to enacting change and demands an effective EA modelling capability: practising EA without specialist EA tools is analogous to creating documentation without a word processor.

The ability to interact with models has many advantages over doing so with real systems. Generally, models are available earlier and by definition: faster, cheaper, more flexible, more scalable, less disruptive and safer. A study⁸ of the benefits of model-based engineering estimated an average 27% cost saving and 36% time saving (rising to 40% and 50% respectively when the model includes testing). It also found a tendency to produce “*better-architected solutions*”: evaluation of a model promotes understanding, reveals problems that can be addressed early, and allows the selection of solution building blocks (products and components) to be deferred until a more complete and well-defined set of requirements has been elicited. The characteristics of modelling that produce these benefits include:

- Resolving Complexity

Models allow modellers to choose what to focus on by abstracting out irrelevant or inconsequential details. They support navigation capabilities that provide links to related views or drill down to resolve elements in finer detail. Views can be tailored to reflect specific stakeholder concerns, eliminating unnecessary complexity and making what remains more comprehensible.

- Quality of Documentation

Documentation quality is likely to be enhanced in two aspects:

- Concise, expressive, comprehensible notation enables a more efficient, effective review process. This encourages greater involvement and feedback from stakeholders who otherwise may not devote enough time or cognitive effort to the (recurrent) review of a lengthy textual document;

⁸ Ref. 2: Benefits of Model-based Development of Embedded Software Systems in Automobiles: Broy, Kirstan TU Munich

- By generating the documentation set from the model, artefacts are referentially consistent, more easily kept up to date, better targeted to specific stakeholders, and available in multiple formats/media types (traditional document, intranet etc.) at a fraction of the effort of manual authorship.
- **Efficiency**

Human visual processing is our most powerful faculty for making sense of the surrounding world. Numerous studies⁹ have shown that visual information is absorbed and analysed from text and pictures with far less cognitive effort than from text alone. And, from an author's perspective, an ability to generate documentation from a model produces artefacts that are consistent, easily refreshed, up-to-date, and generally maintainable at higher quality for less effort.
- **Better Collaboration**

A good visual notation is concise, expressive and yet, being semi-formal, conveys precise meaning. It provides a more effective communication medium than a verbose textual description. This advantage is especially beneficial across international projects with multi-lingual teams.
- **Reuse**

Experienced modellers express recurring ideas using established notational patterns. Consistent use of such design patterns facilitates comprehension through recognition: implicitly conveying contextual information such as problem analysis, design trade-offs or emphasis on resulting system characteristics such as flexibility, extensibility, separation of concerns, policy compliance, etc.
- **Stakeholder Buy-In**

The ability to create and maintain viewpoints addressing specific stakeholder concerns and perspectives (with minimal additional effort) helps provide assurance for stakeholders about the system or project, manage their expectations, and help maintain their buy-in.
- **Standardisation**

Standards promote sharing beyond the immediate project team. Standards make it possible to benefit from a wide pool of literature, blogs, tools, trainings, certifications, etc. and facilitate the exchange of artefacts with partners, suppliers, customers and communities of practice.

⁹ Including: Levie & Lentz, 1982; Levin, Anglin, & Carney, 1987

2.2 The Case for an ArchiMate Security Perspective

ArchiMate delivers the benefits of modelling to architects and designers and is purpose-designed to support TOGAF. But just as the original TOGAF specification contained an under-developed security perspective (subsequently addressed by Refs.3 & 11), so the ArchiMate notation provides only nominal support for the representation of security concerns. Consequently, security artefacts must be produced largely outside the EA modelling environment: separate models, notations, tools and processes. This results in parallel representations of the SOI that EA and ESA teams must strive to reconcile and synchronise, to the detriment of the organisation's overall security posture.

The Open Group's mission is '*boundaryless information flow*': global interoperable access requiring relevant information to permeate through boundaries to support business processes. The SABSA framework enhances this vision with a complete and coherent security methodology that produces a comprehensive set of security artefacts. SABSA offers techniques such as domain modelling and trust modelling that support the analysis of business relationships for architectural synthesis.

With a Security Overlay extending the language with the means to express security perspectives, ArchiMate has the potential to unify these parallel but intrinsically linked domains.

2.3 The Benefits that Modelling Can Bring to SABSA

The SABSA methodology focuses on risk-driven enterprise information security and information assurance. Its goal is to deliver security solutions and architectures designed to support critical business initiatives. Perhaps SABSA's most striking achievement is its completeness: every conceivable security service and activity practised has a logical place and sequence in the SABSA Matrices. Strategic goals in the upper layers can be traced, in both directions, to requirements and control measures in the layers below.

As elegant and comprehensive as this framework is, performing SABSA can generate volumes of complex, interdependent information that can quickly present a significant documentation management challenge. Because each cell of the SABSA Matrix is related to the cells in the same row and column, the artefacts produced from any cell must be maintained consistent with peers in the same row, align with the requirements and assumptions of the cell above, be traceable to consequences in the cell below, and managed through activities in the Management Matrix.

In practice, most artefacts undergo several review cycles. Review feedback can introduce changes that may impact several related documents. Applying these additional changes can drive additional revisions, triggering further review cycles resulting in complex, interconnected documents under perpetual change.

At the same time, IT Divisions are adopting Agile: code sprints, rapid review cycles, and direct stakeholder feedback, creating pressure to eliminate all but the most essential documentation. What might be the consequences for security?

Performing SABSA *well* requires a formidable information and document management capability to keep all artefacts up to date and consistent at the required velocity. Unfortunately, a traditional approach reliant on maintaining multiple independent artefacts (documents, spreadsheets, diagrams etc.) in a coherent state requires an effort that scales exponentially with the number of items and pace of change. The two seem irreconcilable.

A model-driven approach has the potential to transform this information management problem. By generating artefacts as views of a single underlying model, the impact of change to one item is immediately reflected everywhere. Qualities such as referential integrity can be revalidated automatically. The review cycle is streamlined (based on views reflecting stakeholder perspectives) and when all is correct, the documentation set can be auto-generated with the assurance that it is consistent with a single source of truth.

Security is such an essential aspect of modern enterprises that the benefits of modelling are worth pursuing.

The SABSA methodology has received wide global recognition as the gold standard in Enterprise Security Architecture. Yet arguably, the resources required to perform SABSA properly has been a significant barrier to adoption for some organisations. The development of sophisticated security capabilities that are compatible with existing ArchiMate tooling should significantly lower this barrier and enable SABSA's rigour to become economically viable for mainstream commercial organisations.

Complete standardisation of an ArchiMate Security Overlay is still in its infancy, and incorporation into the core language is not even a distant prospect. In the meantime, this paper represents a set of ideas that explore what can be achieved within the constraints of ArchiMate v3.1¹⁰. Hopefully, the ideas offered here will continue to be refined, developed, and extended into a settled consensus of the security practitioner community.

¹⁰ This revision of the paper reflects ArchiMate v3.1. At the time of writing, a minor revision is under development but v3.1 is latest version for which the full specification is published.

2.4 Vendor Neutrality

The SABSA Institute is committed to preserving the vendor neutrality of the SABSA methodology and its related artefacts. In keeping with this principle, every aspect introduced by the Security Overlay and described in this document is fully compatible with the ArchiMate specification and any product that supports The Open Group ArchiMate® Model Exchange File Format¹¹.

With that made clear, the ArchiMate diagrams featured in this document and related models¹² have been created using the **Archi** modelling tool [Ref. 9]. Archi is a user-friendly, full-featured ArchiMate modelling environment that is free to download and use for educational, research and other non-commercial purposes. In using Archi, we hope to minimise the entry barriers for any Reader new to ArchiMate but who wishes to try these techniques for themselves.

This paper makes full use of Archi's support for stereotypes¹³ and custom icons¹⁴, purely for expressiveness: the ability to visualise in diagrams what would otherwise require inspection of underlying properties.

Because custom iconography is a non-standard feature, the Security Overlay neither relies on it nor defines an extended icon set. Stereotyping is, however, a recognised and legitimate extension mechanism even if its expression is somewhat cosmetic (simply the use of double chevrons << >> surrounding the name).

The Security Overlay has adopted a more formal mechanism: a 'stereotype' property on the element being extended. This approach is fully interoperable because no information is lost in export in Open Exchange format: though the rendering of stereotype names will look slightly different in the target environment.

¹¹ <https://www.opengroup.org/open-group-archimate-model-exchange-file-format>.

¹² Many of which are made available from The SABSA Institute site.

¹³ Stereotyping is a mechanism in which a new element type can be introduced by extending an existing one.

¹⁴ Featured in Archi v4.9 and later.

3 An Introduction to the ArchiMate Language

This paper assumes no familiarity on the part of the Reader with the ArchiMate language. This section presents a brief introduction to the structure and grammar of the notation to help familiarise readers to the level needed to evaluate the arguments that will be put forward.

3.1 Core Elements

ArchiMate's core elements belong to one of three categories:

- **Active Structure Elements:** *entities capable of performing behaviour, such as applications or human / organisational actors and their roles relating to particular activities;*
- **Behaviour Elements:** *units of activity performed by Active Structure Elements. (Events are specific types of behaviour that denote a state change);*
- **Passive Structure Elements:** *upon which activity is performed such as business information, data (and their physical realisation in files or databases) or other consumable resources.*

This Active->Behaviour->Passive syntax provides the basic grammar of the core language rather like the *subject-verb-object* construct of a natural language. The Active & Behaviour types are refined further to support service-oriented viewpoints:

- **External Behaviour Elements (Services):** *units of functionality exposed to the external environment. Services provide a 'Service Definition' while concealing the details of their internal implementation;*
- **Internal Behaviour Elements (Processes & Functions):** *define internal processing behind services;*
- **External Active Structure Elements (Interfaces):** *access points to services;*
- **Internal Active Structure Elements (Actors, Roles, Applications & Devices):** *active entities capable of offering interfaces and performing behaviour.*

3.2 Core Relationships

ArchiMate elements relate to each other through a set of core relationships, grouped into four categories:

- **Structural relationships:** *model the static construction or composition of elements: composition, aggregation, realization, and assignment;*
- **Dependency relationships:** *how elements support each other: serving, access, influence, and association;*
- **Dynamic relationships:** *model information and control flows between elements: triggering and flow;*
- *The specialization relationship stands apart as not classified into any of the above categories.*


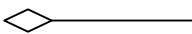
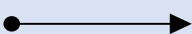
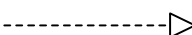
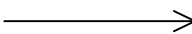


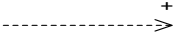




The language does not allow elements and relationships to be combined arbitrarily: each relationship is constrained to connect a predefined set of valid source and target elements, usually in a specific direction.

Relationships are directional and connect exactly two endpoints, though this latter constraint can be extended by using 'junction connectors' that allow, for example, the splitting and joining of data flows.

3.2.1 Definition and Notation of Relationships

The definition and notation for each of these relationships are shown in Table 2.

Table 2: ArchiMate Relationships

Structural Relationships		Notation
Composition	Indicates that an element consists of one or more other elements.	
Aggregation	Indicates that an element groups several other elements.	
Assignment	Expresses the allocation of responsibility, the performance of behaviour, execution.	
Realisation	Indicates that an entity plays a critical role in the creation, achievement, sustenance or operation of a more abstract entity.	
Dependency Relationships		Notation
Serving	Models that an element provides its functionality to another element.	
Access	Models the ability of behaviour elements to observe or act upon passive structure elements. Arrowhead variants express <i>access</i> , <i>read</i> , <i>write</i> or <i>read/write</i>	 
Influence	Models that an element affects the implementation or achievement of some motivation element, either positively or negatively.	
Association	An unspecified relationship, or one not represented by an existing relationship. Basic association is non-directional, ¹⁵ but specialisations may show directionality	
Dynamic Relationships		Notation
Triggering	Describes a temporal or causal relationship between elements	
Flow	Data transfer from one element to another	
Other Relationships		Notation
Specialisation	Indicates that an element is a kind of another element.	
Junction	Used to connect relationships of the same type.	● And ○ Or

To constrain the number of relationships in the vocabulary, some are overloaded, i.e. although their usage is conceptually consistent, their precise meaning adapts to the context of the source and destination elements. This philosophy will be extended when adapting relationships for security contexts.

3.2.2 Derived Relationships

Relationships have a transitive property that allows them to be traversed as chains in the forward direction. The general premise is that if an Element A relates to an Element B ($A \rightarrow B$), and B relates to an Element C ($B \rightarrow C$), then a derived relationship between A and C ($A \rightarrow C$) may be said to exist, subject to rules that take into account the relationships' type compatibility. The details of these rules have been revised with successive versions of the specification and are now too elaborate to be reproduced here. In simple terms though, each relationship is assigned a relative strength, shown in Table 3. If a derived relationship exists, its type is determined by the weakest constituent relationship in the chain.

Table 3: Relative Strength in Derived Relationships

		(← stronger)	Relationship Strength		(weaker →)		
composition	aggregation	assignment	realisation	serving	access	influence	association

¹⁵ Although association is defined as non-directional, the underlying model always identifies a source and target element. In effect, although association is visually & semantically bi-directional, in the underlying model, it is not.

3.2.3 The Core Layers and their Elements

The original ArchiMate v1.0 specification began with 3 **core** layers into which elements are placed:

- The **Business Layer** describes the products and services available to external customers of the domain being modelled. These services are realized by business processes performed by business actors/roles upon business information;
- The **Application Layer** provides the Business Layer with IT services that are realized by software components that operate on logical data objects;
- The **Technology Layer** provides the infrastructure services (data processing, storage and communications) necessary to deploy and run applications. These services are realized by hardware, networks, system software and data artefacts such as files, directories and databases.

It is important to note that ArchiMate elements represent classes rather than instances, i.e., they provide type definitions for actors, processes, data etc., not individual persons, process executions or documents.

The definition and notation for Business Layer elements are shown in Table 4.

Table 4: Business Layer Elements










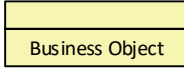
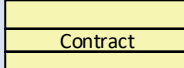
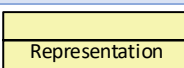
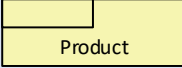
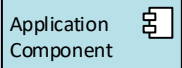
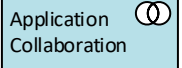
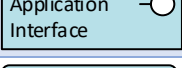
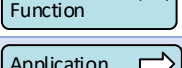
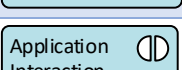
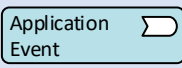
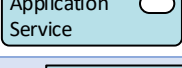
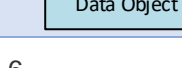
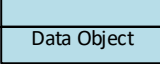
Element	Definition	Notation
Business Actor	A business entity that can perform behaviour.	Business Actor 
Business Role	The responsibility for performing specific behaviour, to which an actor can be assigned, or the part an actor plays in a particular action or event.	Business Role 
Business Collaboration	An aggregate of two or more business internal active structure elements that work together to perform collective behaviour.	Business Collaboration 
Business Interface	A point of access where a business service is made available to the environment.	Business Interface 
Business Process	A sequence of business behaviours that achieves a specific outcome, such as a defined set of products or business services.	Business Process 
Business Function	A collection of business behaviour based on a chosen set of criteria (typically required business resources and/or competencies).	Business Function 
Business Interaction	A unit of collective business behaviour performed by a collaboration of two or more business roles.	Business Interaction 
Business Event	A business behaviour element that denotes an organisational state change. It may originate from and be resolved inside or outside the organisation.	Business Event 
Business Service	An explicitly defined exposed business behaviour.	Business Service 
Business Object	A concept used within a particular business domain.	
Contract	A formal or informal specification of an agreement between a provider and a consumer that specifies the rights and obligations associated with a product.	
Representation	A perceptible form of the information carried by a business object.	

Table 4: Business Layer Elements

Element	Definition	Notation
Product	A coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers.	

The definition and notation for Application Layer elements are shown in Table 5.

Table 5: Application Layer Elements

Element	Definition	Notation
Application Component	An encapsulation of application functionality aligned to an implementation structure, which is modular and replaceable. It encapsulates its behaviour and data, exposes services, and makes them available through interfaces.	
Application Collaboration	An aggregate of two or more application components that work together to perform collective application behaviour.	
Application Interface	A point of access where application services are made available to a user, another application component, or a node.	
Application Function	Automated behaviour that an application component can perform.	
Application Process	A sequence of application behaviours that achieves a specific outcome.	
Application Interaction	A unit of collective application behaviour performed by (a collaboration of) two or more application components	
Application Event	An application behaviour element that denotes a state change.	
Application Service	An explicitly defined exposed application behaviour.	
Data Object	Data structured for automated processing.	

Similarly, the definition and notation for Technology Layer elements are shown in Table 6.

Table 6: Technology Layer Elements

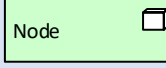
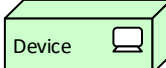
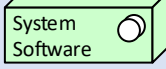
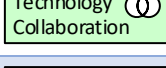
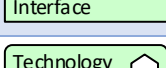
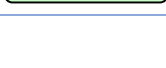
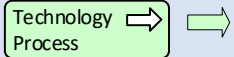
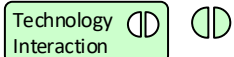
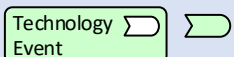
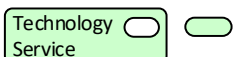
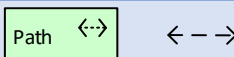
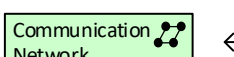
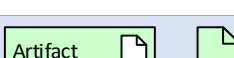
Element	Definition	Notation
Node	A computational or physical resource that hosts, manipulates, or interacts with other computational or physical resources.	
Device	A physical IT resource upon which system software and artefacts may be stored or deployed for execution.	
System Software	Software that provides or contributes to an environment for storing, executing, and using software or data deployed within it.	
Technology Collaboration	An aggregate of two or more nodes that work together to perform collective technology behaviour.	
Technology Interface	A point of access where technology services offered by a node can be accessed.	
Technology Function	A collection of technology behaviour that a node can perform.	

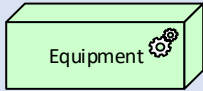
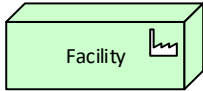
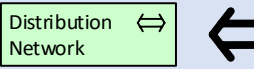
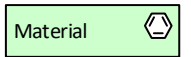
Table 6: Technology Layer Elements

Element	Definition	Notation
Technology Process	A sequence of technology behaviours that achieves a specific outcome.	Technology Process 
Technology Interaction	A unit of collective technology behaviour that is performed by (a collaboration of) two or more nodes.	Technology Interaction 
Technology Event	A technology behaviour element that denotes a state change.	Technology Event 
Technology Service	An explicitly defined exposed technology behaviour.	Technology Service 
Path	A link between two or more nodes, through which these nodes can exchange data or material.	Path 
Communication Network	A set of structures and behaviours that connects computer systems or other electronic devices for transmission, routing, and reception of data or data-based communications such as voice and video.	Communication Network 
Artefact	Data used or produced in software development or by deployment and operation of a system, e.g. a file, database, or executable.	Artefact 

ArchiMate 3.0 introduced a Physical extension to the language, shown in Table 7, to better represent technology nodes embedded in real-world objects (IoT, automation, raw materials distribution channels etc.). It contained only new structural elements, reusing the same behaviour elements as the Technology Layer.

In Archimate v3.1, the Physical extension has been fully merged as a subset of the Technology layer.

Table 7: Physical Layer Elements

Element	Definition	Notation
Equipment	One or more physical machines, tools, or instruments that can create, use, store, move or transform materials.	Equipment 
Facility	A physical structure or environment.	Facility 
Distribution Network	A physical network used to transport materials or energy.	Distribution Network 
Material	Tangible physical matter or physical elements.	Material 

3.2.3.1 Use of Colour

An observant Reader may notice that the Business, Application, and Technology and Physical elements are shown in different colours (yellow, blue, and green, respectively). However, it is important to note that while a number of colour schemes have become popularised, the ArchiMate specification itself is colour-agnostic. Colour is defined as carrying no semantics, leaving the modeller free to apply colour for visual emphasis.

The colours adopted in this section reflect those used in the ArchiMate Reference Card [Ref. 6] but generally, the colours used in model fragments elsewhere in the document follow the 9-colour scheme recommended in ‘Mastering ArchiMate’ [Reference 7].

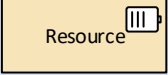
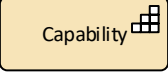
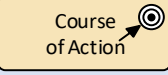
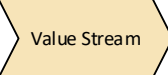
3.3 Extension Layers and Elements

Beyond the core elements, the specification has been extended with other elements relevant to EA practice.

3.3.1 Strategy Layer Extension

The Strategy elements (Capability, Resource, Course of Action and Value Stream) are used to model business strategy and capability-based planning. The notation is shown in Table 8.

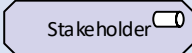
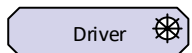
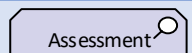
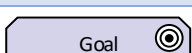
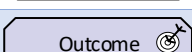
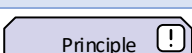
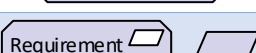
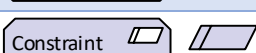

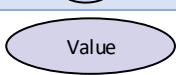
Table 8: Strategy Layer Elements

Element	Definition	Notation
Resource	An asset owned or controlled by an individual or organisation.	
Capability	An ability that an active structure element, such as an organisation, person or system, possesses.	
Course of Action	An approach or plan for configuring some capabilities and resources of the enterprise; undertaken to achieve a goal.	
Value Stream	A value stream represents a sequence of activities that create an overall result for a customer, stakeholder, or end-user.	

3.3.2 The Motivation Extension

Motivation elements form an independent aspect that models the intent and purpose of the architectural design and can be placed in any layer. The set of Motivation elements is shown in Table 9.

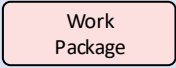
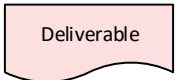
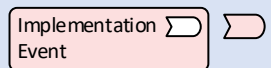
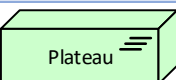

Table 9: Motivation Extension Element

Element	Definition	Notation
Stakeholder	The role of an individual, team, or organization (or classes thereof) that represents their interests in the outcome of the architecture.	
Driver	An external or internal condition that motivates an organization to define its goals and implement the changes necessary to achieve them.	
Assessment	A gap analysis comparing the current state of some stakeholder concern(s) and a future desired state.	
Goal	A high-level statement of intent, direction, or desired end state for an organization and its stakeholders.	
Outcome	An end result that has been achieved.	
Principle	A Principle represents a statement of intent defining a general property that applies to any system in a certain context in the architecture.	
Requirement	A statement of need that must be met by the architecture.	
Constraint	A factor that prevents or obstructs the realization of goals.	
Meaning	The knowledge or expertise present in, or the interpretation of, a core element in a particular context.	
Value	The relative worth, utility, or importance of a core element or an outcome.	

3.3.3 Implementation and Migration Extension

The Implementation and Migration elements model the phased evolution of Enterprise Architectures and the migration between generations of implemented architectures. They include representations of Work Package, Deliverable, Plateau and Gap as shown in Table 10.

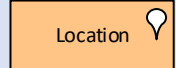
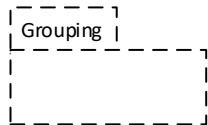
Table 10: Implementation & Migration Elements

Element	Definition	Notation
Work Package	A series of actions identified and designed to achieve specific results within specified time and resource constraints.	
Deliverable	A precisely defined outcome of a work package.	
Implementation Event	A behaviour element that denotes a state change related to implementation or migration.	
Plateau	A relatively stable state of the architecture that exists during a limited period.	
Gap	A statement of the difference between two plateaus	

3.3.4 Miscellaneous Elements

The Miscellaneous elements¹⁶ shown in Table 11 can be used to annotate any layer.

Table 11: Miscellaneous Elements

Element	Definition	Notation
Location	A place or position where structural elements can be located or behaviour can be performed.	
Grouping	An element that aggregates or composes concepts belonging together based on some common characteristic.	

¹⁶ **Note:** ArchiMate tools generally support additional visual elements to annotate diagrams (e.g. comment boxes, visual groupings) that are not part of the formal specification

3.4 ArchiMate Extensibility

The ArchiMate specification offers several mechanisms to customise the language to users' needs.

3.4.1 User-Defined Properties

All elements and relationships can be assigned user-defined properties in the form of key-value pairs.

Although the ArchiMate Exchange format supports several property data types (xs:NMTOKEN, string, boolean, currency, date, time, and number), many tools treat all property values as text. While the ArchiMate standard defines remarkably few properties, the SecurityOverlay will use them extensively to annotate security-relevant information. Occasionally, compound properties are expressed as strings using JSON format.

3.4.2 Specialisations and Stereotypes

ArchiMate offers two methods of extending the vocabulary with user-defined elements and relationships:

- **'In model'** specialisation uses the specialisation relationship introduced in Section 3.2.1 to declare one element as a *'type of'* another. Specialisation implies a refinement of definition from parent to child and an inheritance of relationships and properties, although actual tool recognition of this varies;
- **'Out of model'** specialisation (also known as a stereotype) is denoted by double chevrons surrounding the type name. It enables the introduction of a completely new concept without reference to any other. Stereotypes carry no inheritance and are constrained only by the grammar of the base type. Unlike in-model specialisation, stereotypes can be formed from both elements and relationships.

Figure 2 shows both specialisations in use. Abstract Actor, *Staff Member*, is progressively specialised into Developer and Lead Developer. Specialisation implies that both concrete Actors inherit *Staff Member* properties and that Lead Developer inherits the assignment to the Development process.

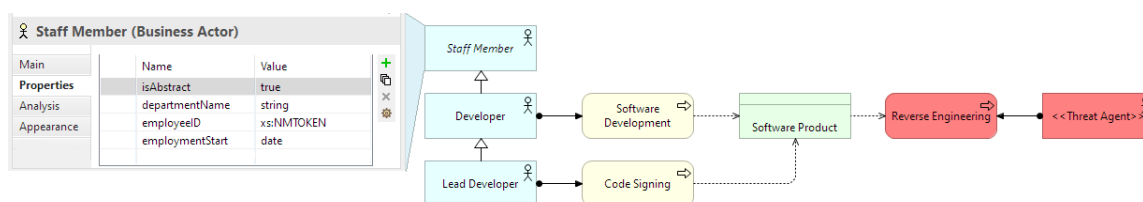


Figure 2: Specialisation vs Stereotypes

On the right-hand side, a Threat Agent has been modelled as a stereotype of Business Actor. It is bound by the same grammatical rules as the base class (e.g., it can be assigned to a business process) but is otherwise distinct from the native Actor element. The use of a stereotype enshrines malevolent intent into the element definition, using something more than colour.

3.4.3 Overloaded Relationships

ArchiMate uses relationship overloading: the same symbol represents different but semantically compatible concepts in different contexts. For example, assignment is used both to bind Actors to Roles and to deploy Artefacts to Nodes. Realisation depicts both the implementation of services and the persistence of logical data on physical storage. Overloading helps constrain the vocabulary with minor loss of precision. The Security Overlay will further overload relationships in keeping with this general principle.

3.5 The ArchiMate 3.1 Framework

Figure 3 arranges the layers (rows) and aspects (columns) described previously to show the overall structure of the ArchiMate framework. Generally speaking, elements in any layer only connect to elements in the same layer or those of same aspect in the layers directly above or below^{17 18}.

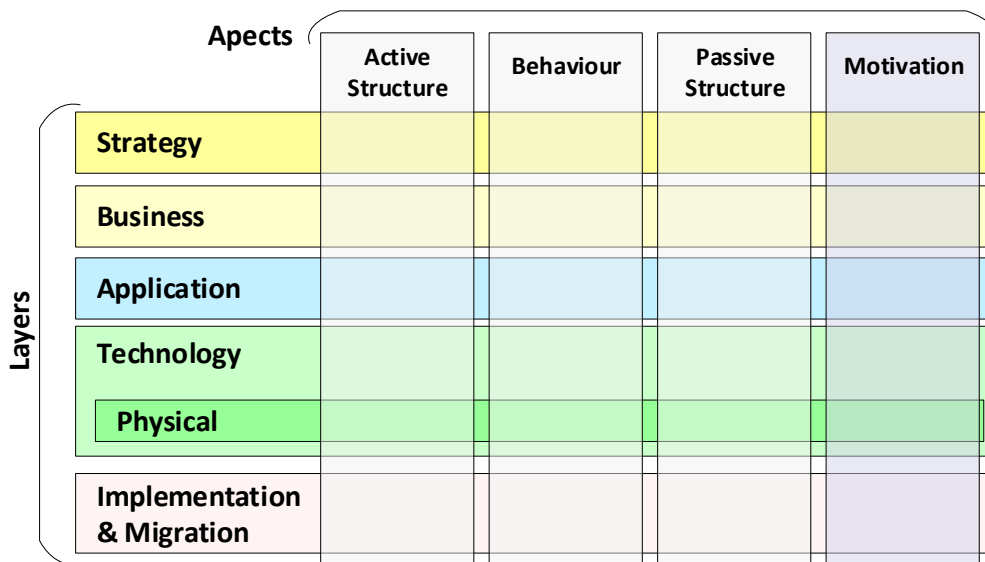


Figure 3: The ArchiMate 3.1 Framework

To complete this section, Figure 4 reproduces the mapping of ArchiMate to the TOGAF ADM.

Since the mapping between TOGAF and SABSA has been defined [Refs.3 & 11], any mapping between ArchiMate and SABSA must be consistent with this alignment.

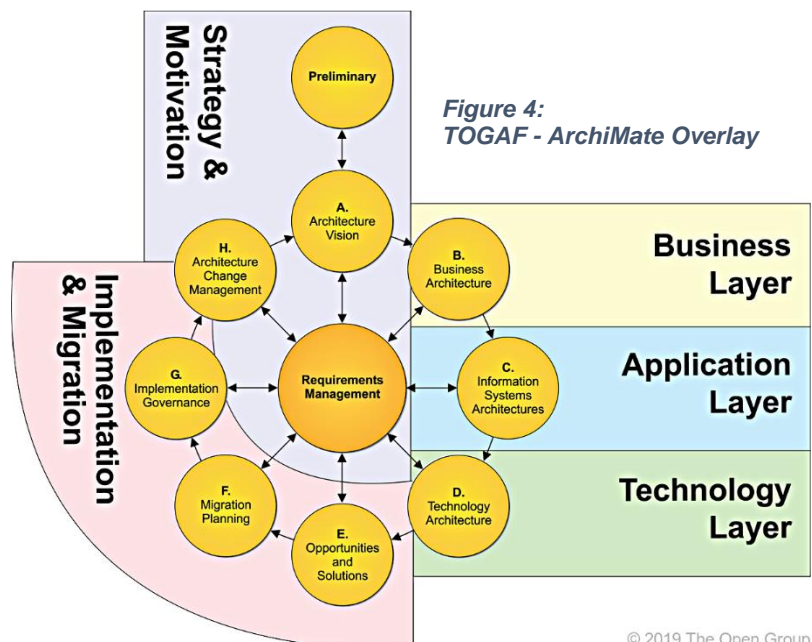


Figure 4: TOGAF - ArchiMate Overlay

© 2019 The Open Group

¹⁷ This statement refers to best practice when creating the underlying ArchiMate model. In diagrams, visual shortcuts are possible and valid when implemented through the use of derived relations.

¹⁸ Motivational elements being the notable exception: they may connect with elements of any layer or aspect.

Section 2: The Security Overlay

This section discusses a Security Overlay for ArchiMate 3 that focuses on extending the architectural layers and Motivation aspect to create a model-based framework for SABSA. It uses ArchiMate's inbuilt extensibility capabilities to propose a Security Overlay: a set of security properties for existing ArchiMate elements, supplemented by a set of new stereotyped elements, to express security concepts.

The section's structure first addresses ArchiMate's Motivation aspect and then works through the layers of the SABSA Matrix, top-down, left to right, referencing elements from the corresponding layers of the ArchiMate framework.

Each element is presented in terms of its purpose, design rationale, description, and examples of intended usage. The documentation here is intended to be informative rather than normative.

For a more formal definition of the Security Overlay, an accompanying XML Schema is provided via The SABSA Institute website. The schema is constructed of three namespaces:

- `tog`: The Open Group: used for property schema definitions for standard ArchiMate elements;
- `tsi`: The SABSA Institute: used to define new stereotyped elements for security concepts;
- `custom`: defines a set of enumerated types that can be customised to suit the organisation's needs.

4 Aligning the SABSA and ArchiMate Frameworks

4.1 An Overview of the Task

Figure 5 illustrates, from a high-level perspective, the task of aligning the SABSA and ArchiMate frameworks.

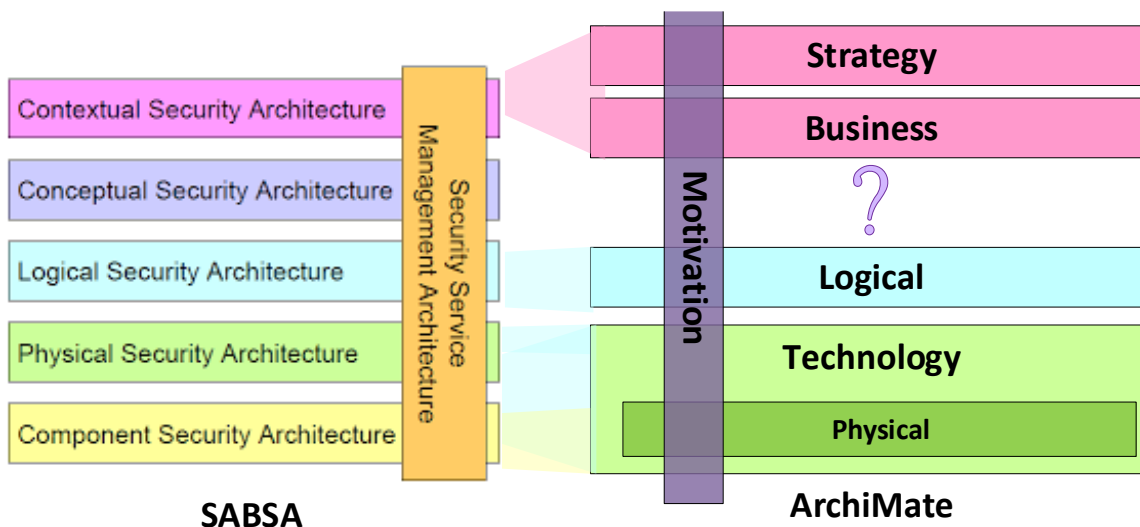


Figure 5: Aligning the SABSA and ArchiMate Frameworks

As a first approximation and allowing for different interpretations of the label '*Physical*', the layers of the SABSA and ArchiMate frameworks broadly correspond. However, the glaring exception is that ArchiMate has no equivalent of SABSA's Conceptual Security Architecture in which to model security concepts (except for its intersection with the Motivation (Why) column). A main aim of this paper is to find practical solutions that address this gap within the constraints of the ArchiMate specification and with an eye on tool capabilities.

A more subtle deficiency is that, even though ArchiMate's core layers have SABSA equivalents, the elements defined in these layers lack '*essential*' security properties: a 'Business Object' ought reasonably to carry a data classification; a 'Business Role' an indication of privileged or non-privileged access etc. Although ArchiMate supports the embellishment of elements and relationships using properties, the standard defines very few. Much of this paper is concerned with identifying these relevant security properties.

The ArchiMate Physical layer was added in v3.0 to model material world entities (equipment, materials, facilities, distribution networks). Its support for physical devices aligns with SABSA's Component layer.

Although SABSA's Security Service Management architecture has no equivalent layer in the specification, it can be expressed in practical models. The recommendation of Ref. 7 is to partition the Architectural Description into three planes: a primary architecture showing how the EA supports business processes at run-time; a secondary management plane that shows how the primary EA is created, deployed, and operated; and a tertiary plane that deals with ownership, administration and governance. SABSA Management processes can be modelled using orthodox notation in the secondary architecture with governance structures (Collaboration) and responsibilities (RACI roles) modelled in the tertiary architecture.

4.2 Risk & Security Modelling in ArchiMate

The ArchiMate standard defines a visual modelling language with a notation for documenting snapshots that describe, analyse and communicate Enterprise Architecture concerns at a point in time. The standard provides no special provision for security, nor does it exclude it from the “many concerns” of Enterprise Architecture. Although the standard makes no mention of security in its formal definition, the specification [Ref. 5] contains several ‘informative’ examples on a security theme.

- Information domains are cited as an appropriate use of the Grouping element, described as “a set of users, their information objects and a security policy” (§4.5.1 p.20).
- Driver is defined as “an external or internal condition that motivates an organization to define its goals and implement the changes necessary to achieve them”. Security concerns are cited as an example (§6.2.2 p.41). Note that Driver expresses both positive or negative motivation, mirroring SABSA’s view of risk as an enabler (exploiting opportunity), a threat to be countered (mitigation), or both.
- In illustrating ArchiMate’s capacity for language extension through specialisation (§15.2 p.117), several examples are given of specialised elements that could form part of a notional Security Overlay:
 - **Assessment** is specialised into Risk and Vulnerability
 - **Business Event** is specialised into Threat Event and Loss Event
 - **Business Actor** is specialised into Threat Agent
 - **Goal** is specialised into Control Objective
 - **Principle** is specialised into Business Policy
 - **Requirement** is specialised into Control Measure
 - **Grouping** is specialised into Risk Domain
- The standard also illustrates the stereotyped Event and Motivation elements to model the derivation of Control Measures from threat and vulnerability analysis via risk analysis and control objectives.

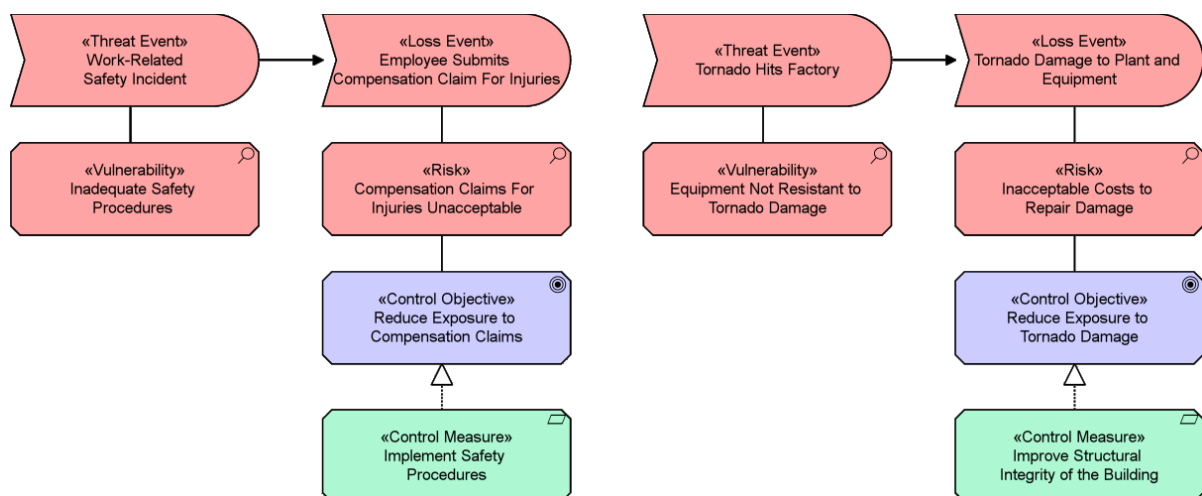


Figure 6: The Risk-Modelling example in the ArchiMate 3.1 Specification

Informative examples cannot be taken as either normative or complete: they illustrate only how the language *could* be used rather than how it *should* be used. Nevertheless, the examples provide some insight into how the ArchiMate authors envisaged risk and security concerns being expressed.

They confirm that security modelling should stem from the specialisation of Motivation elements by providing clear guidance on four elements (Assessment, Principle, Goal and Requirement) and a recommended use for one other (Driver). The semantics of these elements are a natural fit for their application to security. It also proposes Grouping for Trust Domains and specialised Events for the materialisation of threats and losses.

Security events may occur in all core layers and so should not be limited to specialisations of Business Event. ArchiMate 3.0 introduced Application layer events (suitable for modelling input validation errors, time-outs) and Technical layer events (malware detection, data leak alert, back-up failure, etc.).

The example uses non-directional association between Assessment and Event to convey the idea that a Threat Event can exploit a Vulnerability and a Risk can materialise into a Loss Event. In the direction Assessment to Event, the specification permits only the association relationship. In the opposite direction (Event to Assessment), it allows both association and influence. Let's examine these options.

The history of events is a relevant factor in making an Assessment, so depicting an Event influencing an Assessment can make sense if that is what is intended, e.g. in the context of modelling a threat or risk analysis process even if it is not the choice in this example.

What is shown is an unnamed bi-directional association. How should we interpret this? In the direction Event to Assessment, we could say that a Threat Agent / Event “*exploits*” a Vulnerability; a Loss Event *materialises* a Risk. In the opposite direction (Assessment to Event), it seems intuitive to read the association as simply binding the analysis with the object of that analysis.

Apart from the ambiguity, all these interpretations carry an inherent directionality and, in the absence of a better bi-directional interpretation, could be improved by using the directed association introduced by v3.1. Thus the interpretation (Threat Agent / Event <<*exploits*>> Vulnerability; Loss Event <<*materialises*>> Risk), is expressed explicitly in Figure 7.

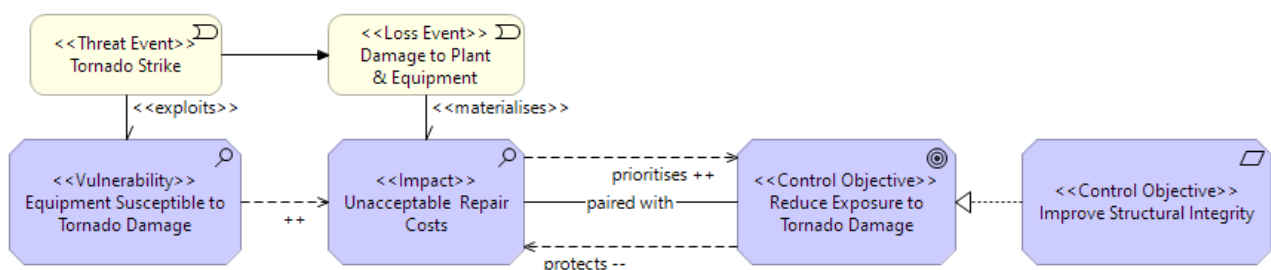


Figure 7: Risk Modelling Relationships

Figure 6 also uses non-directional association between Goal (Control Objective) and Assessment (Risk). Although the specification allows an influence relationship in either direction, the association relationship feels like a natural fit to express a correlation between an Assessment and a Control Objective that model analysis

may need to navigate in either direction. The influence relationship could be used to express more nuance; for example, discovering a new security flaw would raise both the criticality and prioritisation of the associated Control Objective, and deployment of mitigating controls would correspondingly reduce exposure to the vulnerability. These augmentations have also been redrawn in Figure 7.

We now turn to the representation of Control Objective and Control Measure; both modelled as stereotyped Motivational elements: Goal and Requirement, respectively. While the first of these (Goal – Control Objective) is intuitive and uncontroversial, the latter (Requirement – Control Measure) is worthy of comment. By using Requirement as the base element, a Control Measure is being defined here as a *requirement* for a control rather than the *implementation* of a control that the term <<Control Measure>> might suggest. We might ponder what type of element might represent the control implementation: something concrete and deployable that provides effective protection?

This question has been addressed in The Open Group paper: *How to Model ERM and Security with the ArchiMate Language* [Ref. 12], leading to the expanded example, reproduced in Figure 8.

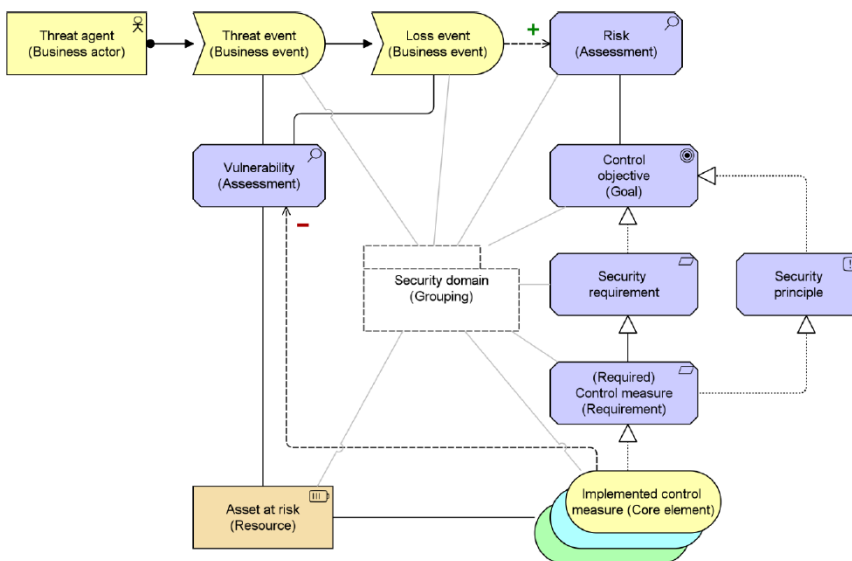


Figure 8: Mapping of Risk and Security Elements

Here a Security Requirement is specialised by a Control Measure (no longer stereotyped), which is then realised by a control implemented by core Services. Though the diagram is non-normative, the definition of Control Requirement on page 25 describes the general case: “A control requirement is realized by core entities used for mitigation, treatment, and control.”

Although ArchiMate lacks a specific way to marking an entity as a ‘Control Implementation’, they become *de-facto* Controls through the realisation of Control Measures. This is a topic that we shall return to later.

Both references confirm the use of a realisation chain from Control Objective to implementation. This relationship is structural (strong) and a good semantic fit where a single Control Measure suffices.

Complications arise where multiple measures are necessary to achieve the required objective: here, the

default semantics of realisation¹⁹ implies that each measure is an OR'ed alternative (e.g. multiple software implementations of the same service definition). Security realisations often need AND semantics: a set of requirements, all of which are necessary to achieve an objective of end-to-end protection or defence in depth. The specification allows two other relationships: influence and association. Let us consider whether either of these would be a better fit.

Association is ArchiMate's '*relationship of last resort*': used for linking elements without any precise meaning beyond "*an unspecified relationship or one that is not represented by another ArchiMate relationship.*" If we accept that realisation is a good semantic fit, there is no need to use the '*relationship of last resort*'. If not, it would be better to specialise realisation (strong & inherently directional) than use a form of association.

Influence, though stronger than association, is much weaker than realisation. It implies that implementing a set of Measures will impact the posture vis-à-vis the Control Objective rather than delivering it assuredly. Control Measures and Control Objectives can conflict in that some Measures undermine some Objectives while reinforcing others positively. The *influence* relationship is the only one capable of expressing positive and negative effects and therefore merits a place.

Realisation, then, is the best choice for the structural relationship between Control Objectives, Control Requirements, and actual Control Measures, with influence reserved for 'cross-coupling'. A means of expressing AND/OR semantics is discussed on Page 42.

Finally, the specification suggests *Driver* as the appropriate element to model security concerns but offers no example of how this should be related to other elements. For guidance, the ArchiMate Motivation meta-model is reproduced in Figure 9, where Driver (as a form of the abstract Motivation element) may *associate* with, *influence* or be *influenced* by any other Motivation element.

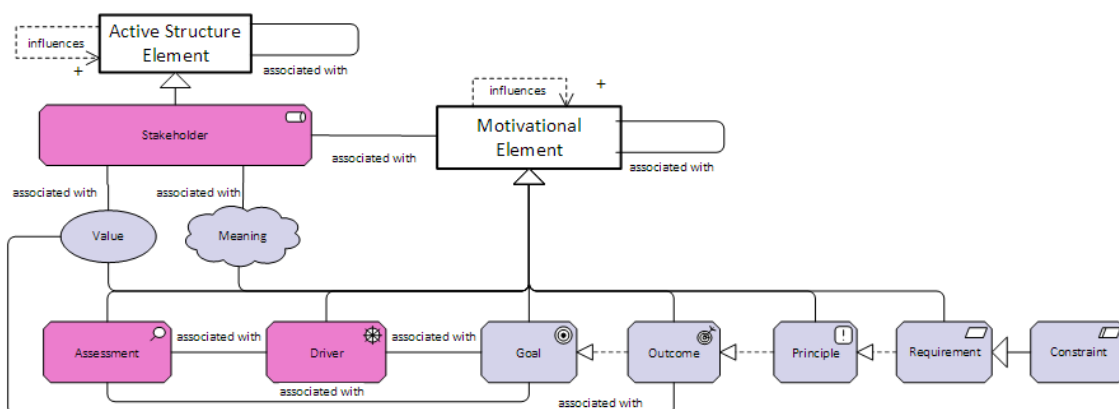


Figure 9: ArchiMate Motivation Meta-Model

¹⁹ To indicate that more abstract entities are realized, completely but not exclusively, by means of more tangible entities.

4.3 The Basic Element and Relationships

The Security Overlay defines basic properties that can be applied to every element and relationship type. They are described here, but to avoid repetition, they are not necessarily referenced in every place where they would typically appear.

They include:

- An **identifier**: used to provide a globally unique reference for the element within the model;
- An ***isAbstract*** flag, set to false by default, indicating that an element cannot be instantiated; it must have a non-abstract specialisation. Visually, an abstract element name is written in italics;
- **Stereotype/stereotypeOf** used to bind stereotyped elements to base elements using something other than the double chevrons name format;
- **Cardinality** indicators at the source and target ends of relationships are expressed in UML notation, i.e. n (exactly n), n..m (range n to m), * (any). The meaning of the cardinality varies with context.

5 The Motivation Aspect

Building on the previous section, Figure 10 shows the full Security Overlay for the Motivation metamodel that formalises the non-normative examples into a specification that better supports the security perspective.

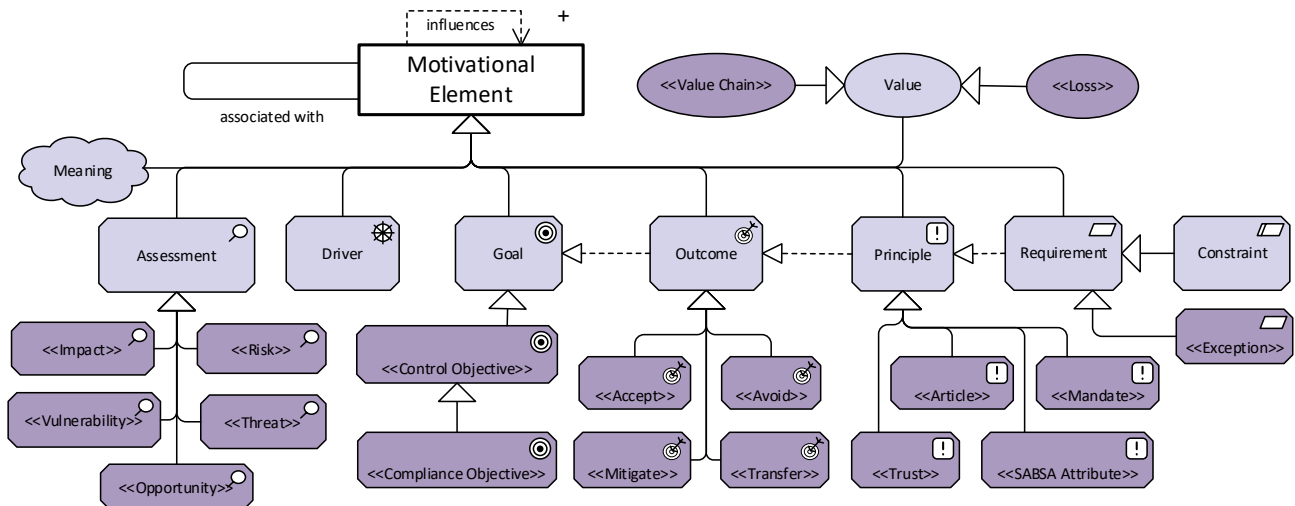


Figure 10: Security Enhanced Motivation Metamodel

This section describes how these elements can be used to express security motivations: the identification and valuation of assets, analysis of threats & vulnerabilities, evaluation & treatment of risks and the identification of Control Objectives and Control Requirements.

5.1 Value & Loss

The “What” column of the SABSA Architecture Matrix [Ref. 10] acknowledges the business value of assets. Business Value may be expressed in several forms (financial, legal, brand, social, economic, health & safety) in combination or with others. This evolution brings it closer to ArchiMate, which has no element to represent an abstract Asset but instead offers the ability to associate a **Value** element with any model element.

Models containing many Values will encounter the issue of providing sufficient unique names. This can be overcome through adoption of a naming convention that reflects the associated Element name. Figure 11(a) shows the simple association of a Value with a generic element, thereby marking it as an asset.

A tertiary relationship is used to represent the Stakeholder to whom the Value is important (Figure 11(b) lower) is recommended over the in-line association (Figure 11(b) upper). The former pattern scales better when a Stakeholder associates the same Value with multiple assets or a Value is appreciated by multiple Stakeholders. It avoids the risk of ambiguity / unintended coupling that could otherwise occur in the underlying model even if not explicitly drawn, as in Figure 11(c).

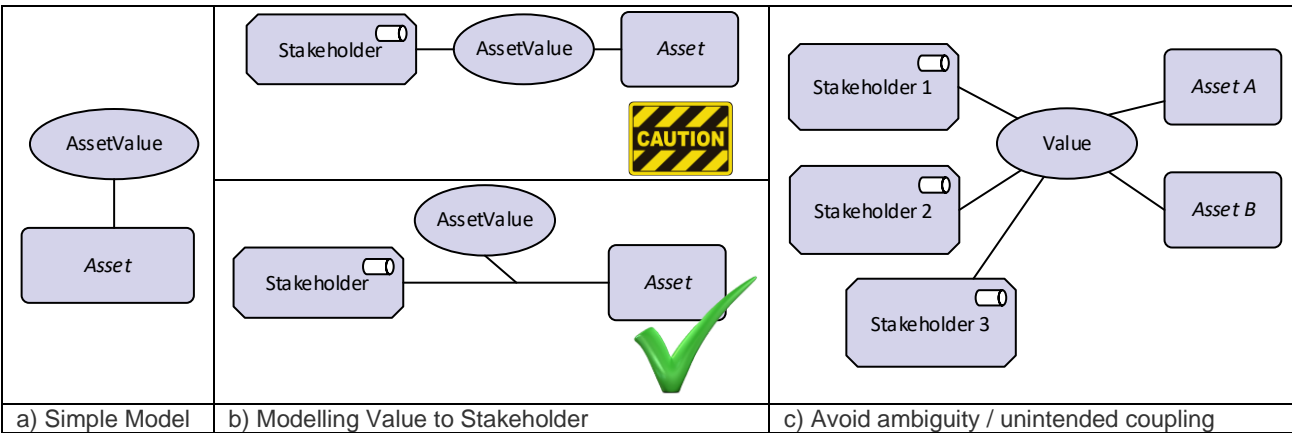
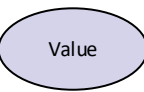
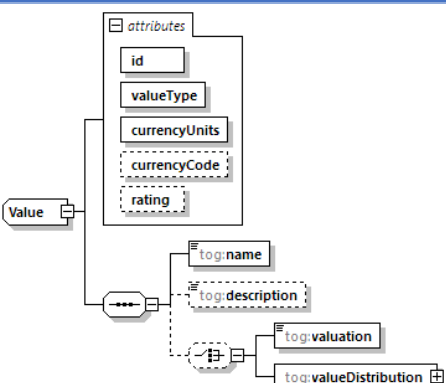



Figure 11: Modelling Assets using Value

Because security is concerned with enhancing value and protecting assets, the Value element (and a “Loss” specialisation introduced by this Security Overlay) provide a basis for modelling risk²⁰. A good textual description of the nature and scale of Value and Loss is essential for a complete model. Enabling automated analysis of Value and Loss requires capturing key characteristics of these elements in a structured way. This can be achieved using custom properties. Table 12 shows proposed properties for the Value and Loss elements. Financial values, to support quantitative analyses, are expressed as a 3-point range consisting of a lower and upper bound (typically the 90% confidence interval) and a mode (most likely) value.

Table 12: Proposed Value Property Overlay

Element	Properties	Schema
	<p>The properties of the Value and Loss elements include the following key-value pairs.</p> <ul style="list-style-type: none">valueType classifies the asset in terms of a set of enumerated values: FINANCIAL, REGULATORY, REPUTATION; <p>The set of valueType can be drawn from a standardised set or customised as required.</p>	
	<p>The set of valueTypes can be drawn from a standardised set or customised as required.</p> <ul style="list-style-type: none">currencyUnits and currencyCode describe the scale and denomination, respectively;rating supports a qualitative scale (e.g. from VLOW to VHIG) for non-financial valuations;valuation denotes the value, either as a single point value or a confidence interval.	

²⁰ More on this topic in Section 5.1.5.

5.2 Value Chain

The Security Overlay defines <<Value Chain>> as a stereotype of Value, specifically for use with process architecture rather than assets and will be discussed further in Section 6.1: Capability and Value Stream elements.

The key measures of meta-processes are rooted in cost-accounting: production cost, sales price, and margin. The organisation seeks to identify which activities generate value and which are inefficient or uneconomic. The security perspective is concerned with enhancing and protecting the value of the value generators while containing the costs of the inefficient process. The ValueChain stereotype models the breakdown of costs and is therefore, principally a financial measure. The Security Overlay can be used to map the composition of a Value Chain to any behavioural element in the EA model, as shown in Figure 12.

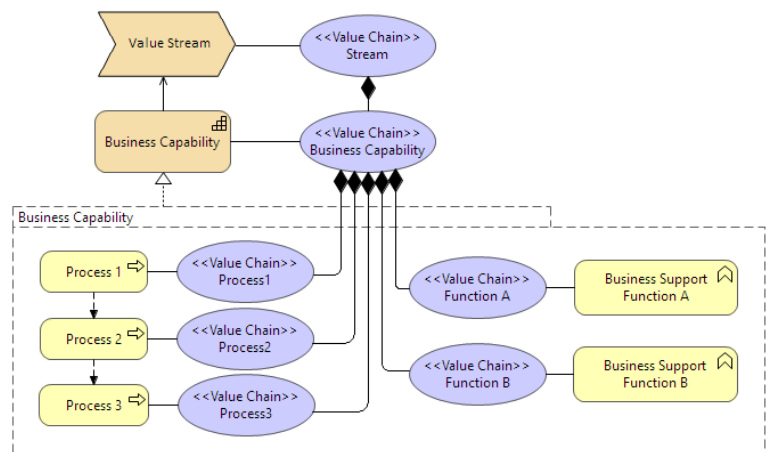


Figure 12: Composition of Value Chains

The properties proposed for the Value Chain stereotype enable the margin to be calculated as the sum of its constituent sub-chains. The ArchiMate 3.1 specification allows the Value element to be decomposed into compositions and aggregations of the same type. Composition seems the safer option here because, unlike aggregation, it should prevent cost accounting figures from being counted multiple times.

Table 13: Value Chain Properties

Element	Properties	Schema
	<p>The Value Chain element extends the Value element with properties to express value in currency units.</p> <p>The definition supports the expression of production costs (input materials, process handling and consumption of support services) and the manifest worth in the final value.</p> <p>The margin can also be expressed as a derived value, calculated as the difference between total costs and the final delivered value.</p>	

5.3 SABSA Business Attributes

Business Attributes represent the essential qualities of the Stakeholders' ideal system, to be promoted, protected and enhanced in the target architecture if the enterprise is to fulfil its mission. They are also atomic Singletons: describing a single indivisible characteristic rather than an elaborate composition of requirements. Attributes are essential to the SABSA approach but have no equivalent representation in ArchiMate.

At closer examination however, the definition of ArchiMate's Principle element as *“an intended property of a system ... a general property that applies to any system in a certain context ... motivated by some goal or driver”* is semantically highly compatible for representing SABSA Business Attributes in the Security Overlay. Figure 13 recreates the SABSA Business Attributes catalogue of Principle stereotypes.

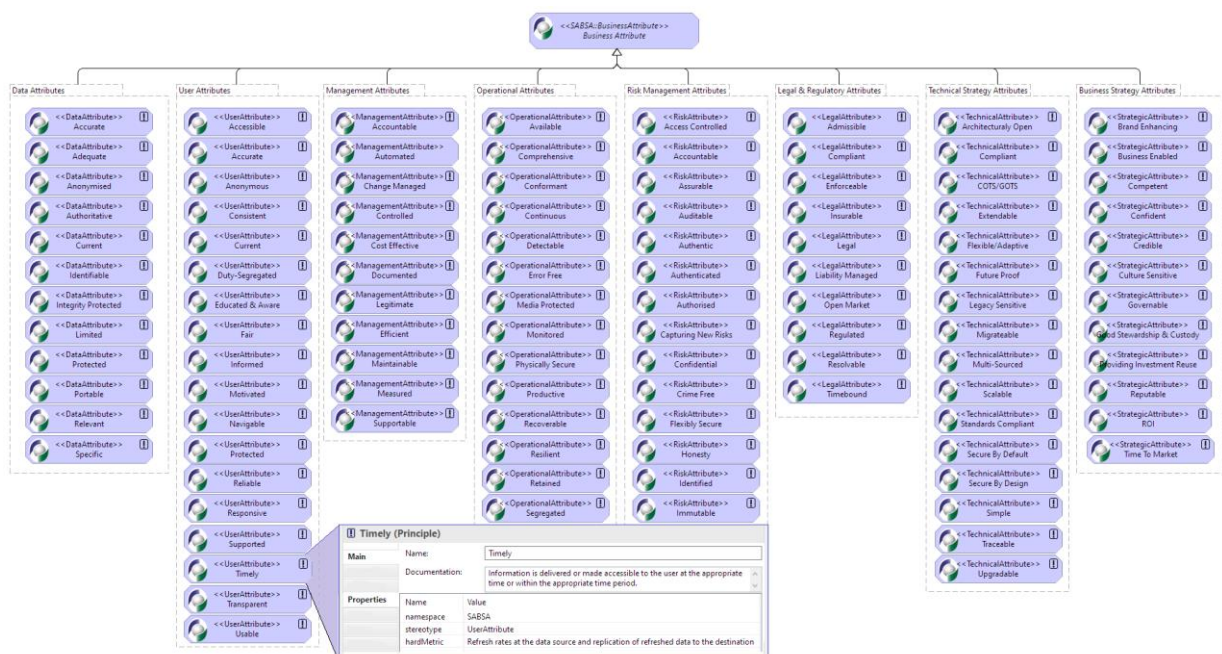


Figure 13: SABSA Business Attributes represented in ArchiMate

A few points to note about this Business Attributes representation:


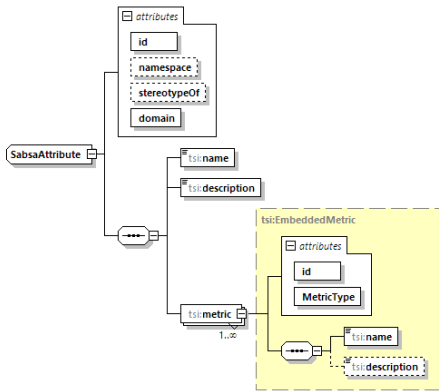
- Attributes are modelled as a stereotype of Principle, distinguishing them from conventional uses of the Principle element such as “Cloud First” and enables it to define a distinct set of properties.
- The taxonomy is structured as an abstract base Attribute, specialised into domains using Groupings. This approach offers benefits in practical modelling and various analysis scenarios where:
 - Domains act as a namespace that supports the construction of qualified names enabling Attributes to be overloaded with an appropriate definition and metrics for different contexts²¹.
 - Grouped attributes promote consistency and correctness in modelling. E.g., Stakeholders with legal concerns should only be offered associations with Legal and Regulatory attributes.

²¹ For example, Accountable is defined in both Management (assignment of accountability) and Operational (being held to account for actions taken) contexts.

- It enables Attributes with the same non-qualified name to appear on the same diagram.

The SABSA Attribute stereotype has been modelled with properties that reflect those published in Ref. 1.

Table 14: SABSA Attribute Properties

Element	Properties	Schema
	<p>The SABSA Attribute stereotype defines properties that express:</p> <ul style="list-style-type: none"> • a name: an adjective by which it is commonly recognised; • a domain: a grouping that qualifies the name by acting a specific namespace; • a description defining of the attribute; • a sequence of metrics, each with a name and description and categorised as either hard or soft. 	

5.3.1 Structural Placement of Business Attributes

In the SABSA method, business-layer Attributes are refined and resolved at lower layers into contributing Attributes with sub-system scope and ultimately addressed by Control Objectives (**Goals**) achieved through the realisation of Control Requirements and Constraints.

The dictionary definition of 'principle' is: *'something more abstract than policy and objectives and meant to govern both'*. In the abstraction hierarchy of Figure 14, principles are universal, singleton concepts at the apex, with policies, control objectives and requirements beneath.

This concept of a principle is not universal: the influential COSO Enterprise Risk Management framework positions 'Principles' between Objectives and Controls, as in Figure 15.

ArchiMate follows the COSO paradigm²²: i.e. Principles are used as maxims to guide Designers *towards* Outcomes and Goals (Figure 16). SABSA Attributes align with the dictionary: as aspirational ideals towards which the architecture should align, through the achievement of costed, concrete, contextualised goals and requirements, but which may never be fully attained.

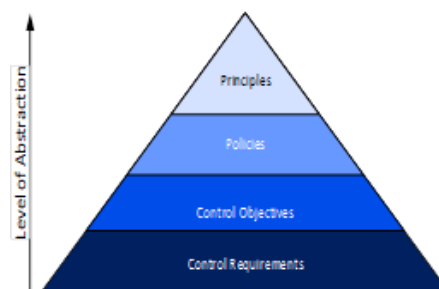


Figure 14: Hierarchy of Abstraction

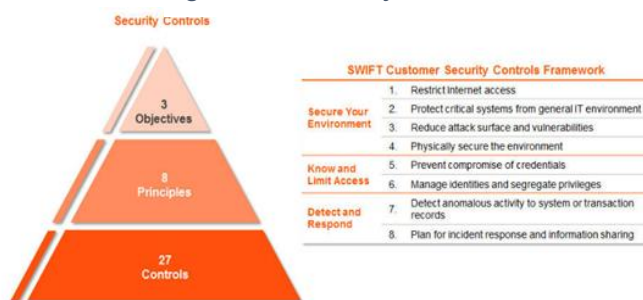


Figure 15: The COSO Model

²² Note the example from the ArchiMate specification which proposes the Principle element to represent policy.

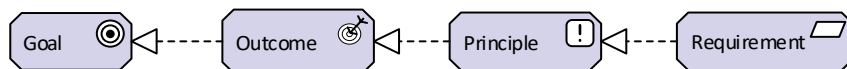


Figure 16: Principle in the ArchiMate Motivation Hierarchy

This presents a few problems for SABSA's intended usage. Take the example shown in Figure 17.

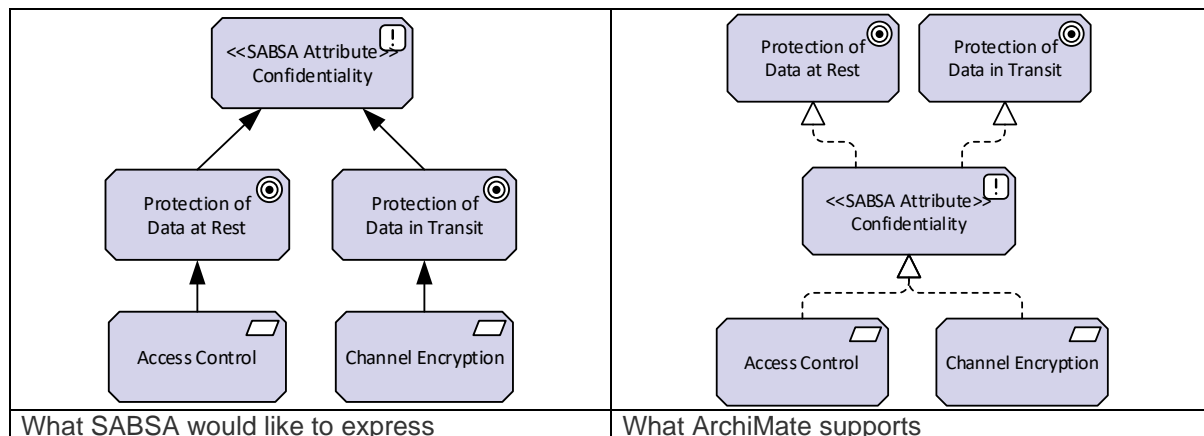


Figure 17: Highlighting the Control Hierarchy Mismatch (i)

The SABSA view (left) expresses the Attribute (*Confidentiality*) as being satisfied by specific Goals (*'Protection of Data at Rest'* and *'Protection of Data at in Transit'*) at various points in the model. These are to be achieved by distinct, verifiable Requirements (*'Access Control'* and *'Channel Encryption'*), respectively.

The ArchiMate metamodel (right) places *Confidentiality* in the centre of the structure and, by doing so, forces the realisation paths to merge. The resulting model suggests *'Protection of Data at Rest'* might be realised through *'Channel Encryption'* and *'Protection of Data in Transit'* through *'Access Control'*.

The result is both unintentional and incorrect. It is also difficult to disentangle because, even if the structures are *drawn* as distinct views, they remain merged in the underlying model, causing a problem for automated analysis which sees both paths as legitimate and is consequently unable to resolve the modeller's intent.

A workaround is to create parallel stacks (Figure 18), each with its own copy of *'Confidentiality'*. Note that the copies should have distinct names because ArchiMate represents *'classes'* rather than *'objects'*.

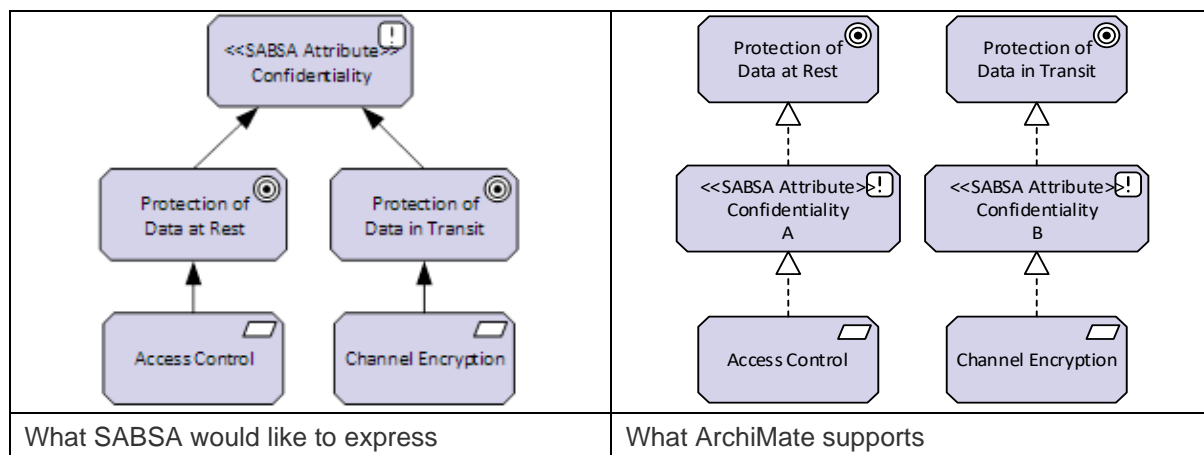


Figure 18: Highlighting the Control Hierarchy Mismatch (ii)

Although this approach achieves the required segregation and is entirely in line with the specification, creating distinct *Confidentiality* elements for each stack is inelegant. Attributes are singletons, so any duplication involves redundancy and maintenance issues. If a good model is meant to reflect reality, this is poor style: a concept of different ‘flavours’ of confidentiality is not natural in the real world.

A better solution (Figure 19) eschews the standard structure but remains legal and reflects the original intent.

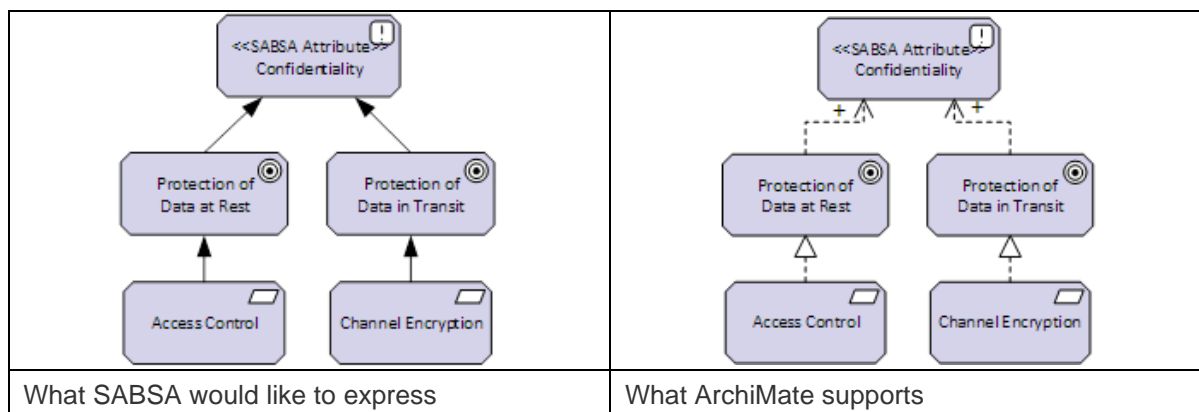


Figure 19: Achieving the desired Hierarchy

Via a convention in which SABSA Attributes are only ever **influenced** by Control Objectives, the desired structure can be achieved within legal grammar. The choice aligns with ArchiMate’s definition of **influence** as “a traceable motivational path” where the “motivation element is achieved to a certain degree. That is to say, SABSA Attributes are *augmented* by Control Objectives rather than attained in an absolute sense.

5.3.2 Traceability of Business Attributes

Another essential requirement of SABSA Attribute modelling is the ability to trace their refinement through the architectural layers, also implemented using influence relationships, as shown in Figure 20.

Element reuse is an issue encountered when creating moderately-sized Attribute hierarchies concerns.

Consider a situation where ‘Compliant’ is a desired SABSA Attribute that occurs twice in the model: once to indicate GDPR compliance on personal data in the Business layer and again in the context of FIPS compliance for a Hardware Security Module in the Technology layer.

The Modeller encounters the following problems:

- Because ArchiMate is *class-* rather than an *instance-*oriented, modelling tools generally forbid (or complain bitterly) if the same element appears more than once in the same view.

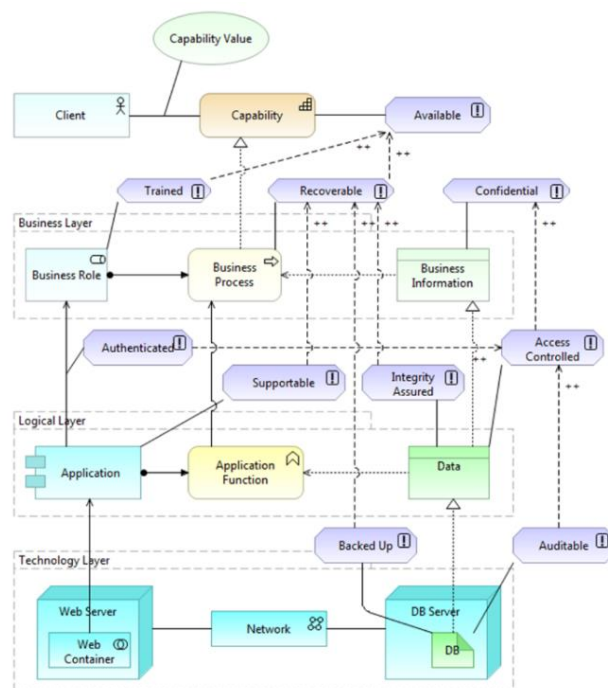


Figure 20: Attribute Traceability across Layers

- Using the Attribute once, associating it to both targets, is both possible and 'legal' but:
 - It is more difficult to provide definitions and metrics that are compatible with multiple contexts;
 - the reused Attribute becomes a hub of merged relationships, warping the hierarchical structure.
- Artificial 'flavours' of global, abstract concepts such as Attributes are an inelegant design practice.

Because SABSA Attributes are global singletons, reuse within a model runs the risk of unintended coupling. In cases where an Attribute's definition is compatible, and reuse doesn't warp the layer hierarchy, it is possible to accommodate different metrics using an externalised 'Meaning' element (See 0).

In all other cases, the best solution is to use distinct instances from the Attribute taxonomy (Figure 13). In this example, this would mean using the 'Compliant' Attribute from the Technical Strategy category for use with the HSM and then one from the 'Legal & Regulatory' category for GDPR. As distinct elements (same short name, different qualified names), both Attributes can legitimately appear on the same diagram without creating the hub/warping effects associated with what we might refer to as a 'Singularity': a black hole at the heart of our model.

The influence relationship conveys the same sense of intangibility that justified its use with Control Objectives. There is a subtle difference, however. Attributes are pure abstract singletons in a taxonomy that is reused across models, projects, and enterprises. When *directly* connected to each other by 'concrete' relationships in a model, they also become connected in the abstract: universally and forever.

It's perhaps an esoteric point, but it is important to note that in reality, SABSA Attributes are just abstract concepts with no intrinsic correlation. The relationship only arises as an "echo" of a real correlation in the context of the specific real-world system being modelled.

5.4 Meaning

ArchiMate's Meaning element is not extended by the Security Overlay other than by the addition of optional properties. The element plays a useful role in representing concepts such as Identity and in the application/adaptation of SABSA Attributes to multiple contexts.

As described in the previous section, the SABSA Attribute taxonomy incorporates sample attributes with notional definitions and hard/soft metrics via user-defined properties. While these properties can and should be customised for each new context, because Business Attributes are modelled as singletons, any definition and metrics chosen will apply to everything that the Attribute touches, i.e., in every context where the Attribute is referenced.

As we have seen, this causes problems where an Attribute is applied to multiple points in the same model. Consider, for example, whether there is a single *Confidentiality* Attribute enhanced by the two Control Objectives and four Controls of Figure 21. How can we express context-sensitive definitions and metrics?

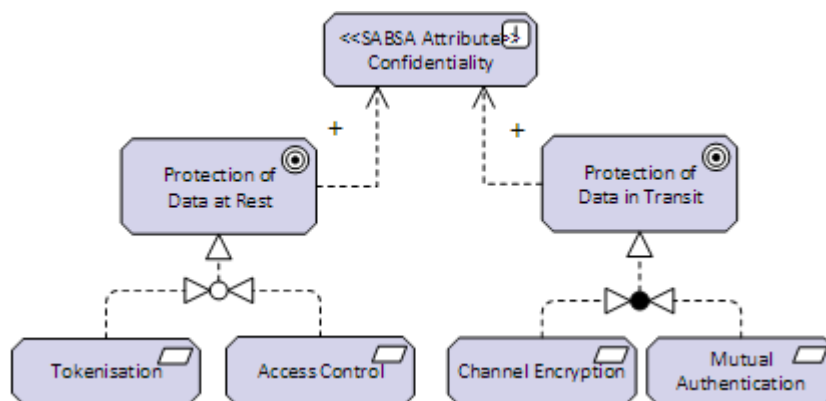


Figure 21: Applying Attribute Metrics to Multiple Controls

The Security Overlay recommends the use of **Meaning** elements to re-interpret the Attribute for each context. Figure 22 shows how the meaning of *Confidentiality* (i.e. its definition and metrics) can be externalised and then adapted to *Tokenisation*, *Access Control* and the other requirements using Meaning elements.

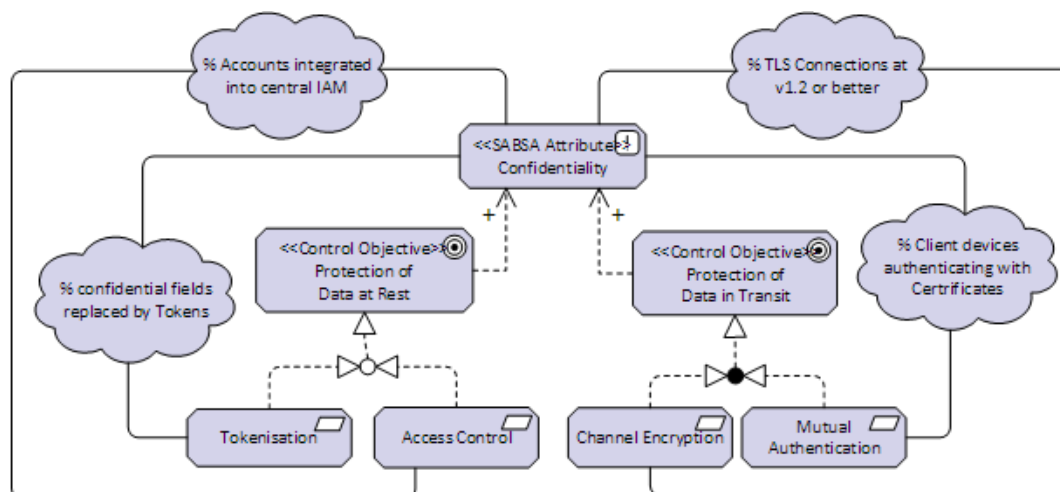



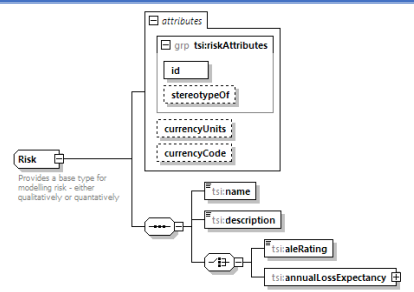
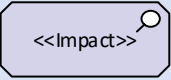
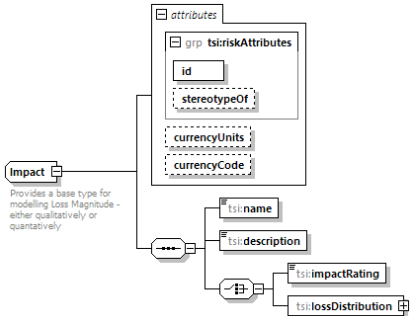

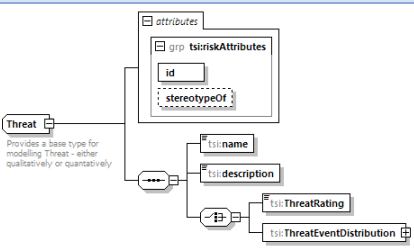
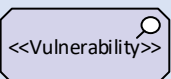
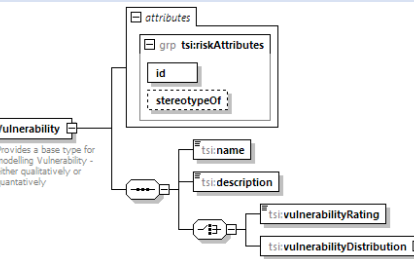
Figure 22: Use of Meaning to define context-sensitive Metrics

5.5 Impact, Threat, Vulnerability & Risk

Guided by the ArchiMate Specification (See 4.2), threats, vulnerabilities, impacts and risks are modelled as stereotyped **Assessment** elements, providing a good semantic fit that reflects the subjective uncertainty inherent in assessing these unknowns.

Neither SABSA nor the majority of ISMS standards are prescriptive about risk assessment methodologies. The Security Overlay attempts to accommodate any approach commonly practised: qualitative or quantitative, by providing support for methods based on enumerated labels (e.g. HIGH, MEDIUM, LOW) and numerical calculation (e.g. Bayesian methods or Monte Carlo simulation). Usage will be discussed in Section 7.2 Risk Management & . The properties of the basic stereotypes are shown in Table 15.

Table 15: Risk Element Properties

Element	Properties	Schema
	<p>The Risk stereotype expresses annual loss expectancy as a qualitative rating or a quantitative probability distribution.</p> <ul style="list-style-type: none"> currencyCode & currencyUnits qualify the scale and value of the expected loss. 	
	<p>The Impact stereotype expresses impact (loss magnitude) as a qualitative rating or a quantitative probability distribution.</p> <ul style="list-style-type: none"> currencyCode & currencyUnits qualify the scale and value of the expected loss. 	
	<p>The Threat stereotype expresses vulnerability as a qualitative rating or a quantitative probability distribution.</p>	
	<p>The Vulnerability stereotype expresses vulnerability as a qualitative rating or a quantitative probability distribution.</p>	

5.6 Controls: Objectives, Requirements and Measures

This section models the SABSA's Conceptual/Motivation cell that deals with Control Objectives. As usual, our starting point is the ArchiMate specification, discussed in Section 4.2, in which Goal is specialised as Control Objective and Requirement is specialised as Control Measure. We have already encountered some semantic questions about this model. The Security Overlay retains the <<Control Objective>> stereotype of Goal, retains Requirement for security requirements without further stereotyping and introduces a new <<Control>> stereotype to represent the control implementation, discussion of which is deferred to Section 5.10.

We first concentrate on the Motivational elements: Control Objective and Requirement.

To model a Control Objective realised through a (*Control*) Requirement seems a good semantic fit. In practice, though, a single Requirement is insufficient to express *End-to-End Security* or *Defence-in-Depth*.

For the Security Overlay, a means of conveying more complex arrangements is needed. Consider the slightly extended example of Figure 23 (left). We wish to express that the 'Protection of Data at Rest' objective can be achieved through 'Access Control' or 'Tokenisation' and that the 'Protection of Data in Transit' objective must be implemented with both 'Channel Encryption' and 'Mutual Authentication'.

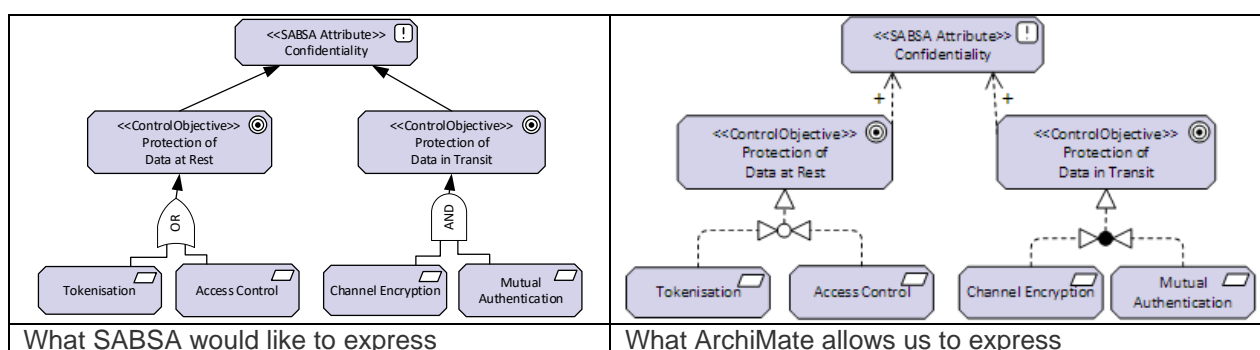


Figure 23: Achieving Complex Requirements

A convenient solution exists via ArchiMate's AND and OR Junctions Figure 23 (right), which, since version 3.0, can be used to join a broader set of relationships²³ of the same type.

²³ Extended from dynamic relationships to include all dependency relationships, assignment and realization.

5.7 Multi-Tiered Security

SABSA's Conceptual/Process cell discusses the combination and layering of Controls for 'Defence-in-Depth' or 'End-to-End Protection'. The Blue Book [Ref. 1] describes a 4-tier model: Prevent, Contain, Detect & Notification, and Recovery & Restoration. The NIST Cybersecurity Framework [Ref. 14] identifies 5 control functions: Identify, Protect, Detect, Respond, & Recover. The Security Overlay adds the category 'Deter' to cover sanctions expressed in policies & contractual clauses. The *Identify* set (including Asset Management, Business Environment, Governance, Risk Assessment and Risk Management Strategy) are modelled in the secondary & tertiary architectures rather than the primary, run-time architecture. The properties for the Control elements have been derived from analysis and use of several commonly used control frameworks. The resulting schema is shown in Table 16.

Table 16: Control Element Properties

Element	Properties	Schema
<div data-bbox="150 1106 331 1173"> <<Control Objective>> </div> <div data-bbox="150 1189 331 1256"> Requirement </div> <div data-bbox="150 1272 331 1339"> Constraint </div>	<p>The Control set (Objectives, Requirements and Constraints) define a similar set of properties.</p> <p>The namespace and refCode properties provide a means to link the element to an external reference document such as a remediation plan.</p> <p>The status property is used to track the element's lifecycle using enumerated types: MANDATORY, ADVISORY, WITHDRAWN & PENDING.</p> <p>The applicability property can be used to indicate the element's scope: COMPONENT, SYSTEM, ORGANISATION or GLOBAL.</p> <p>The baseline property is used to indicate the element's level in the baselining tiers defined by some standards and maturity models, e.g. STANDARD, ENHANCED or ASSURED.</p> <p>The protection profile expresses a control strength for each layer of the multi-tiered defence.</p>	

Figure 24 shows how this works in practice. Profile properties are declared in the Control Objective element to define its protection requirement and in each Requirement element to describe its protection capability.

We see, in this example, that the Control Objective '*Prevent Unauthorised Modification*' defines an assurance requirement of '*Moderate*' across all tiers. While there is no single Requirement that can satisfy this profile in full, the aggregate set of Requirements meet and exceed the profile: an Acceptable Use Policy that sanctions unauthorised access satisfies the '*deter*' aspect, Tamper Detection mechanisms delivers the '*detect*' capability, Regular Back-Up, fulfilled by a BCP infrastructure service, provides the '*recover*' capability, etc.

By standardising these properties within the Security Overlay, an analysis tool could verify that the protection profile required by the Control Objective is met or exceeded by the set of Requirements proposed to realise it.

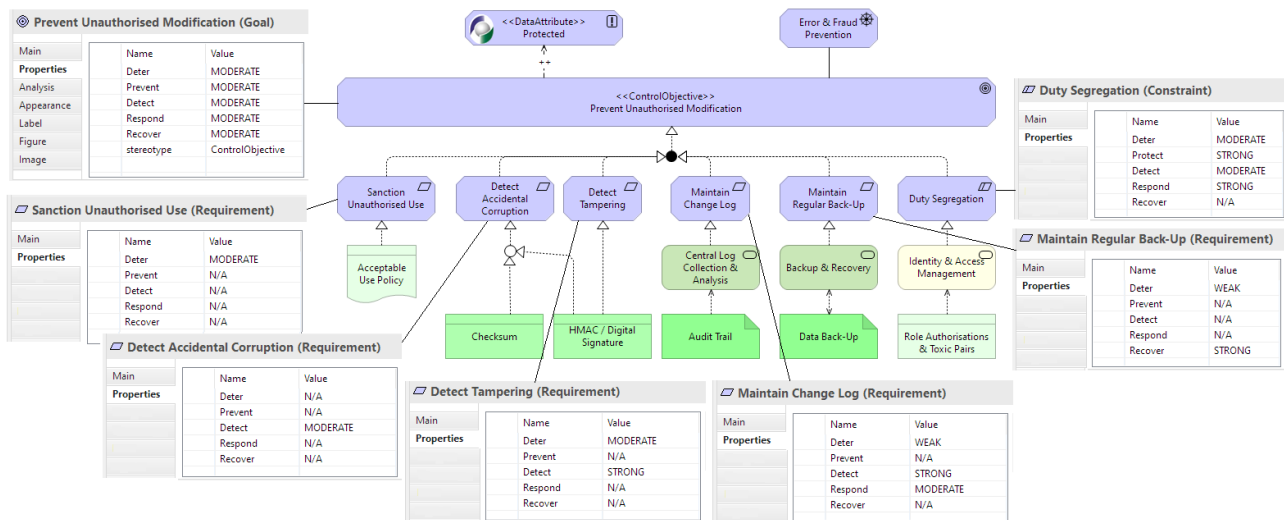


Figure 24: Example of Multi-Tiered Security

Compliance

Risk analysis is far from being the only source of controls. As the digital economy becomes increasingly critical to society's functioning and prosperity, organisations are being subjected to regulatory demands that are becoming ever more numerous, extensive, rigorous, frequent and held to ever-higher levels of scrutiny.

The "Why" cells of Contextual and Conceptual cells address this important and demanding driver. The Security Overlay has been extended with additional stereotypes to support the representation of Compliance Objectives and guidance on managing the complexity created by the challenges of simultaneous multi-regulatory compliance.

Having analysed several regulations, legislation, and industry or sector standards that organisations must certify against, the Security Overlay provides the following elements that enable these external references to be incorporated into the model as importable architectural reference modules.

5.8 Regulations and Standards

Compliance controls are defined in public standards. The level of adherence demanded spans a broad spectrum: some are mandatory carrying the force of law, some are pre-requisite for a licence to operate, while others may be important in demonstrating good governance, fulfilling market expectations or measuring capability maturity. As a container element to model these publications, the Security Overlay offers two stereotypes of Representation:

- Regulation: for compliance obligations that are enforced by law;
- Standard: for compliance obligations that are driven by commercial or technical considerations.

Both Regulations and Standards tend to follow a regular structure. Although substantial standards, such as NIST SP1500 (Big Data), consist of several volumes (modelled as a composite <<Standard>>), it is more

typical for the document to be partitioned into Chapters or Sections devoted to a particular theme. These top-level sub-divisions are modelled as regular Groupings, as shown in the examples in Figure 25Figure 25: The Structure of Standards and Regulations.

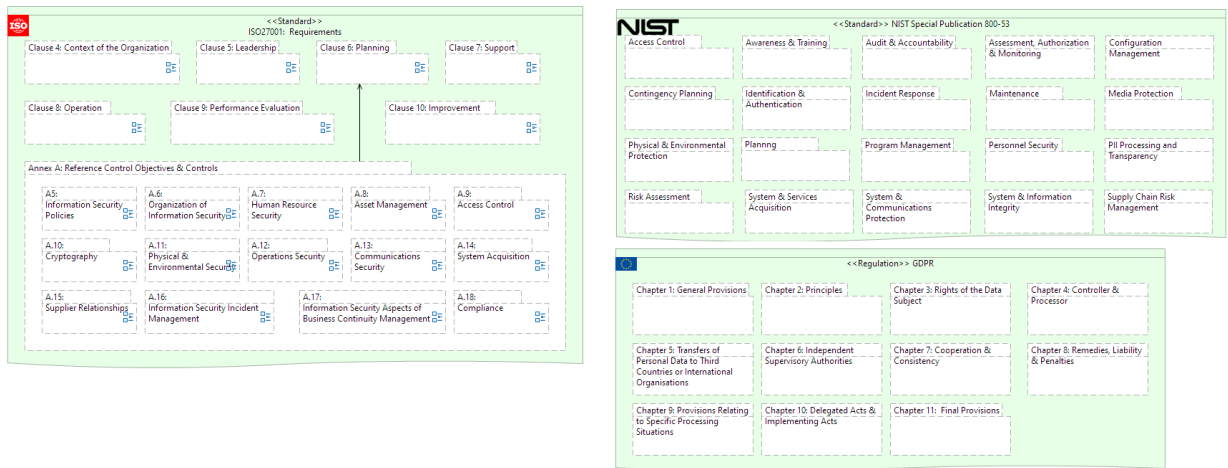


Figure 25: The Structure of Standards and Regulations

The schema definitions of Standard and Regulation are given in Table 17Table 14.

Table 17: Standard & Regulation Properties

Element	Properties	Schema
	<p>Standard and Regulation are both modelled as stereotypes of Representation and define a similar set of properties.</p> <p>The namespace, publisher, and sourceURL properties provide a means to link the element to an external reference document.</p> <p>version and publishDate provide support for version control, while expiryDate can be used to show that the standard has been superseded or withdrawn.</p> <p>Regulation extends Standard with one additional property, effectiveDate, enabling legislation to be modelled and used ahead of the date it comes into legal force.</p>	

5.9 Articles, Mandates, and Compliance Objectives

Within Chapters/Sections, these documents follow a similar structure, albeit using different vocabularies.

Legal texts tend to consist of articles (schedules), paragraphs and clauses, numbered for cross-reference. Articles serve multiple purposes: for example, they can be used to define terminology, to assert scope and applicability, to cite earlier mandates that empower the issuing body with authority to act in the given domain. Only after considerable preamble do the Articles generally turn to address what needs to be accomplished. Articles consist of paragraphs and sub-paragraphs that are eventually built from short, well-focused clauses.

To model this in ArchiMate, the Security Overlay draws an equivalence between legal paragraphs and Compliance Objectives (an analogue of the risk-based Control Objective) and between legal clauses and Requirements/Constraints. This structure fits reasonably well as clauses tend to be succinct and focused. By definition, adherence to the clauses is necessary and sufficient to satisfy the objective of the paragraph.

The question remains of how to model the articles themselves. As with SABSA Attributes, we need to represent an abstract concept, the law, an ideal to be pursued in letter and in spirit by meeting specific, achievable goals. Again, the Security Overlay turns to a stereotype of Principle, <<Article>>: and the convention of only referencing it through influence relationships.

In practical terms, legal articles affecting IT architecture are imported using the <<Article>> stereotype; the documentation field is used to fully reproduce the article text. <<Compliance Objective>> elements are then used to translate the formal style of legal paragraphs into a plain language goal description that is more suitable to a technical audience. Requirements, Constraints and <<Exception>> (a stereotype of Requirement that de-scopes the stated objective) are then used to spell out fine-grained, actionable statements, again in plain language, that mirror the legal clauses. The example of Figure 26 shows how this has been implemented for Article 7 of the GDPR.

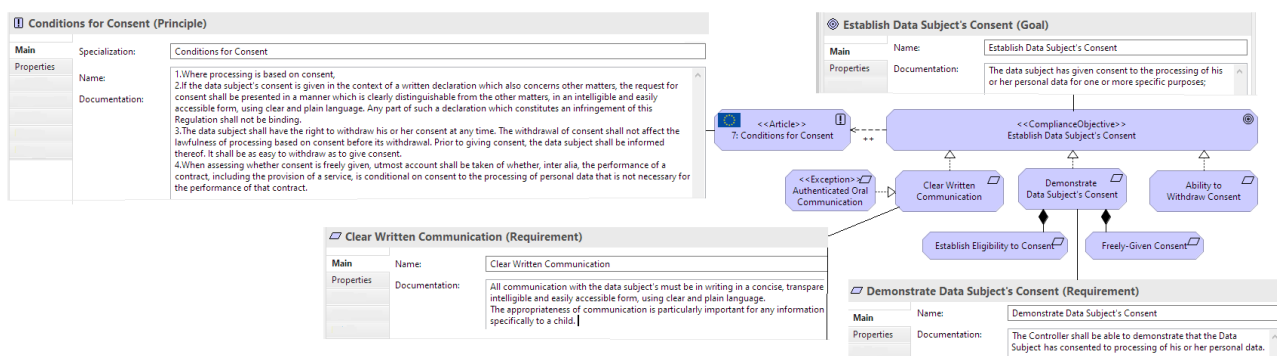


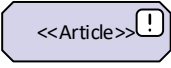
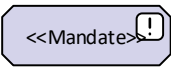
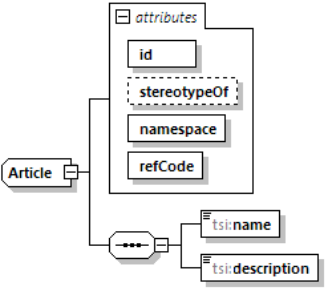

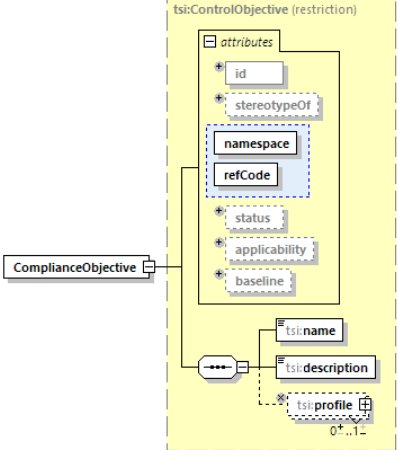
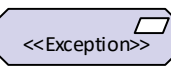
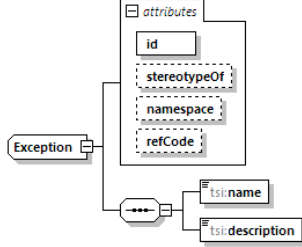
Figure 26: Articles and Compliance Objectives

This same 3-tier pattern (Concept—Goal—Requirement) is also evident in sectoral and technical standards. While the stereotypes already identified are re-usable at the Goal and Requirement levels, neither the <<SABSA Attribute>> nor <<Article>> are intuitive names for the top-level concepts found in these standards.

Consensus on naming in the standards is absent: they are variously referred to as security principles, overarching objectives, strategic goals or capabilities – all of which clash with ArchiMate vocabulary. The term currently adopted in the Security Overlay is <<Mandate>> (also used by UCF [Ref. 13] to abstract over similar control objectives defined in disparate standards), but the final choice is still work-in-progress.

The stereotypes Article and Mandate are, broadly speaking, synonyms with minor tuning of the schema to reflect the more stringent regulatory context as shown in Table 18.

Table 18: Compliance Conceptual Element Properties

Element	Properties	Schema
 	<p>Article & Mandate are declared as stereotypes of Principle and define properties that express:</p> <p>The namespace and refCode properties provide a means to reference the Article/Mandate in an external reference document declared in the enclosing Regulation/Standard.</p> <p>The description reproduces the original text in full.</p> <p>The elements are synonyms of each other with no difference in the schema definition.</p>	
	<p>Compliance Objective is a restricted extension of Control Objective in which the namespace and refCode properties are required.</p> <p>Compliance Objectives are generally stronger than Control Objectives. Being derived from regulatory or standards compliance, they tend to be mandatory and non-negotiable.</p> <p>By contrast, Control Objectives can be derived from risk analysis or maturity models and therefore governed by internal strategy, policy, and project timescales with more flexibility in prioritisation, scheduling and reporting.</p> <p>The description translates sections of the original text, often expressed in formal, legalistic style, into plain language for the intended audience.</p>	
	<p>Exception provides a visual means to de-scope a Compliance Objective, Requirement or Constraint;</p> <p>Without such an element, caveats and exceptions in the detailed text might be overlooked.</p> <p>The namespace and refCode properties are optional.</p>	

Models of commonly used regulations and standards are highly re-usable. We hope and intend that ArchiMate models of these references will be developed and then shared among the community. Please keep watch on The SABSA Institute website for such contributions, or better still, if you are the first to model a particular reference, please consider sharing it via TSI's MSA Forum.

5.10 Control Mechanisms

The previous sections have identified a need for 3 fundamental elements to model a control architecture:

- An abstract concept of an ideal that is being pursued, such as a SABSA Attribute, a state of regulatory (Article) or standards (Mandate) compliance, modelled as a stereotype of Principle;
- A specific, realisable, defined state of a system component that represents a sufficient realisation of the ideal if achieved, maintained and evidenced. These are modelled as Control Objectives and Compliance Objectives, stereotypes of Goal. The architecture will aim for this state by a given date. The emphasis is on bringing a risk within appetite or demonstrating compliance with the law at an acceptable cost and schedule. Objectives are not aiming for the achievement of perfection;
- Control Requirements, Constraints and Exceptions specifying granular actions and restrictions that, when performed in the right way, in the right place, at the right time, and for the right cost, achieve the overall objective.

Section 5.6 observed that Motivational elements are only capable of expressing intent. A complete model needs a means of representing the concrete mechanisms that implement the requirements in the target architecture. Although these by definition cannot be Motivational elements, the document deals with them here for completeness of the topic.

Services are often cited as recode the means of realising security requirements. This approach works well in many cases, e.g. to model Identity & Access Management, Vulnerability Management, Security Incident Event Management (SIEM). In practice however, there are also many controls for which service implementation feels contrived. Figure 27 shows Controls manifest in many forms: nominated accountability, a policy clause, a process design, a logical control or a physical device.

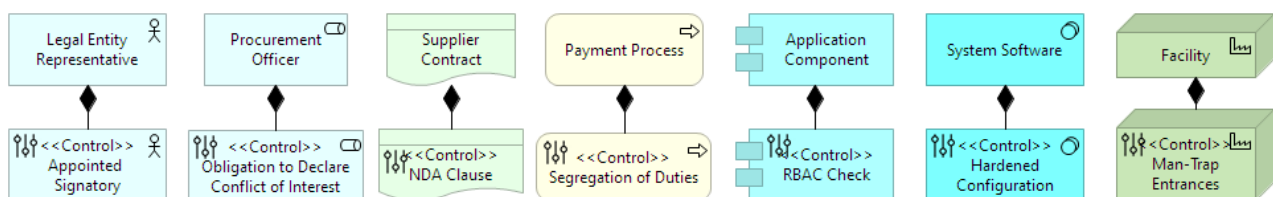
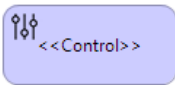
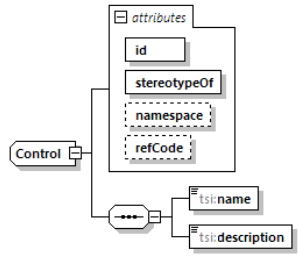


Figure 27: Use cases and iconography for Control

What can be concluded from this set of examples is:

- Controls are often a component of an element of the same type;
- Control stereotypes cannot be formed from a single base type but require multiple base types.

Table 19: Control Properties

Element	Properties	Schema
	<p>Control is unique in the Security Overlay in that it can extend any base type.</p> <p>It is used to emphasise a standalone element or a component of a composite element that addresses a specific requirement.</p> <p>Its properties express:</p> <ul style="list-style-type: none"> An optional source and refCode link it to an external document. <p>Though Controls are often a component of an element of the same type, no rule requires this to be necessarily so.</p>	

5.11 Trust

The final Motivational element introduced by the Security Overlay is Trust: another stereotype of Principle.

The element enables the explicit modelling of any trust implicit in an interaction between Parties.

The definition is given here, but discussion of this element's usage is deferred until the section: Governance

Governance runs like a seam through the People column of the SABSA Matrix, exhibiting two main aspects:

- Organisation structures representing decision-making bodies, review boards, steering committees etc. They can be adequately modelled as Collaborations to which all the participating Roles can be assigned;
- Individual involvement as might be expressed by a RACI (Responsible / Accountable / Consulted / Informed) matrix;

RACI presents an interesting design consideration in ArchiMate. The language is predicated on a grammar in which Active Structure's (Actor or Role) access to Passive Structure (the Asset) is mediated by a Behaviour element in a "Subject–Verb–Object" syntax. Applying this to RACI requires us to consider what the 'Process of Being Accountable' means and what would it look like. Instead of contriving some pattern that fits this grammar, the Security Overlay proposes to stereotype a directed association: <<RACI>> specifically for this purpose. The example of Figure 31 shows how it can be used.

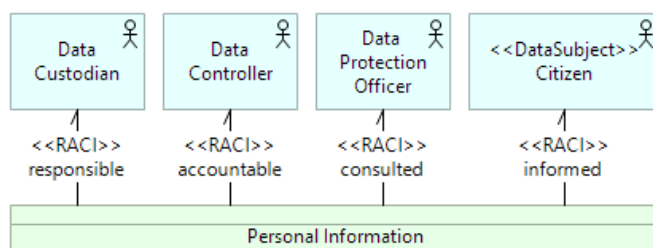


Figure 31: Representing RACI Relationships

A note of caution about using generic names for RACI targets: care should be taken not to create Singularities.

Consider the example of Figure 32a: 'Data Owner' is a poor choice of name for a RACI role. It is likely to be re-used in multiple contexts in multiple views. While it still may be possible to understand the modeller's intent from each view in isolation, the entanglement in the underlying model, shown in Figure 32b, renders automated analysis impossible.

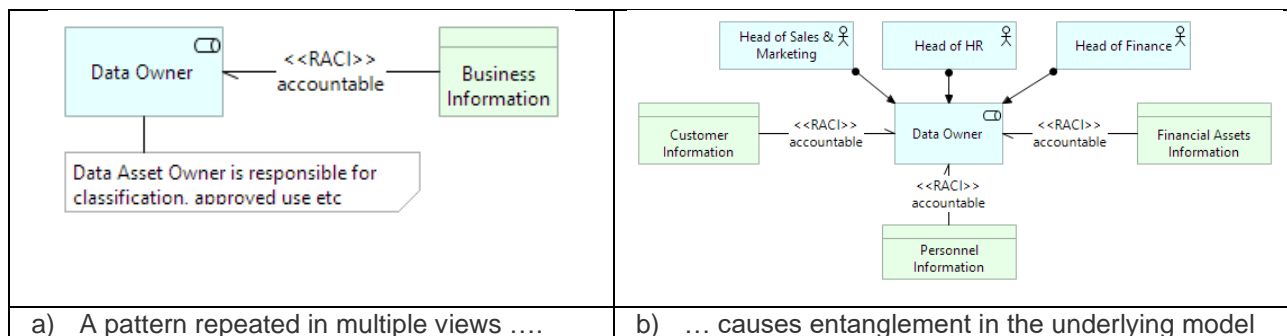


Figure 32: Avoid RACI Entanglement

Alternative patterns that avoid this issue are shown in Figure 33.

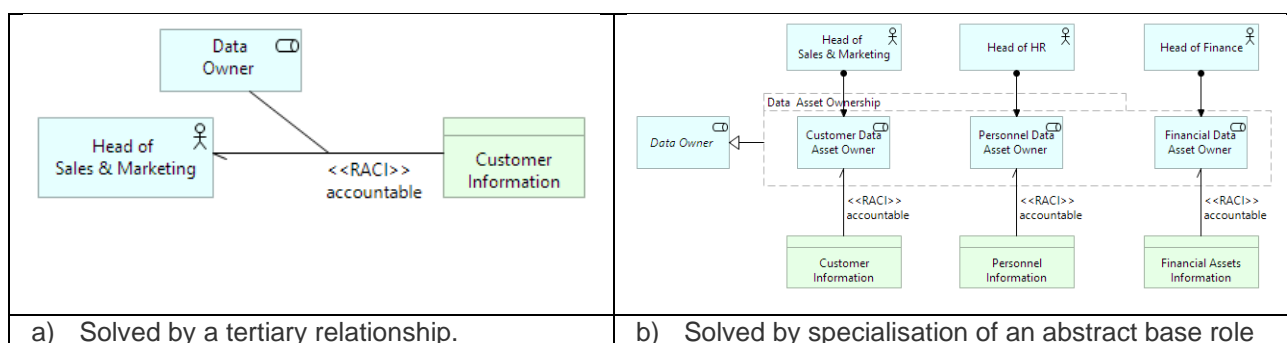


Figure 33: Recommended RACI Patterns

Table 28: RACI Properties

Relationship	Properties	Schema
<p> <code><<RACI>></code> </p>	<p>RACI is a stereotype of Association in which the relationship name is restricted to the enumerated values: RESPONSIBLE, ACCOUNTABLE, CONSULTED or INFORMED.</p> <p>Directionality is fixed to TRUE and directed towards the Actor or Role.</p> <p>sourceCardinality expresses the number of items that are within the scope of the RACI association.</p> <p>targetCardinality is normally 1.</p>	<p>The schema shows a relationship named 'RACI' with a group of attributes 'tog:relationshipAttributes'. This group contains 'id', 'sourceCardinality', and 'targetCardinality'. The relationship is also associated with a 'tsi:name' attribute and a 'tsi:description' attribute.</p>

5.11.1 Threat Actors

There are three possible ways to model threat actors, depicted in Figure 34:

- As a stereotype of Actor: a malicious organisation that is known to pose a threat;

- As a stereotype of Role representing a malicious intent, engaged against the system;
- As an adverse action (behaviour or event) that occurs by accident or design;

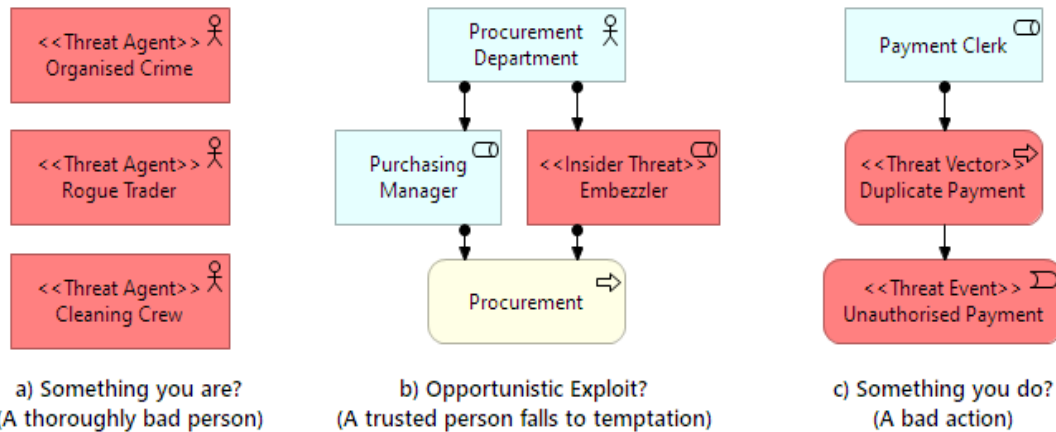


Figure 34: Possibilities for Modelling Threat Actors

Option (a) is seductively intuitive. It provides a useful node for collating information about various Threat Agent constituencies (population size, motivation, capability, determination, etc.), all useful to quantitative risk modelling. It's fine for cases where the perpetrator isn't explicitly identified (e.g. Organised Crime & Rogue Trader). It may even be reasonable to identify a particular Hacker Collective that poses a concern. Caution should be exercised if using <<Threat Agent>> to label specific Actors (competitors, states and certainly not internals, who may even have access to the model) as malicious. Defaming the Cleaning Crew in this way is unacceptable.

Option (b) suffers from the same issue: if we accept that internal staff are overwhelmingly honest and well-intentioned, it is likely to cause great offence if they are modelled with an explicit assumption of criminality.

Option (c) is similarly problematic when alluding to malicious intent. It also misrepresents exceptional behaviour as a formal process. Modelling accidental errors and omissions in this way is less contentious.

Although analysis of Insider threats is unavoidable, we should be conscious of these sensibilities in our models. It is better to contain analysis of misplaced trust and abuse to specific risk views that can be restricted from the main published model. A better style in which sensitive threat-related elements have a degree of separation from (i.e. are not directly attached to) Actors and Roles is shown in Figure 35.

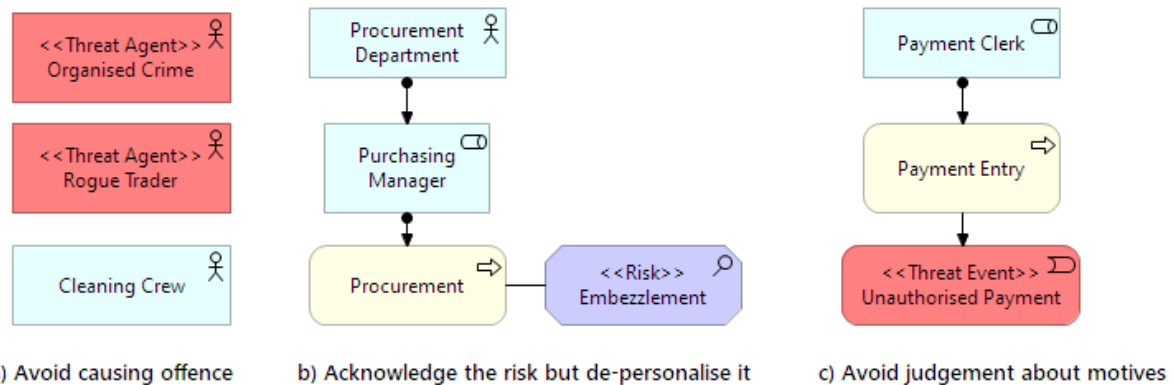

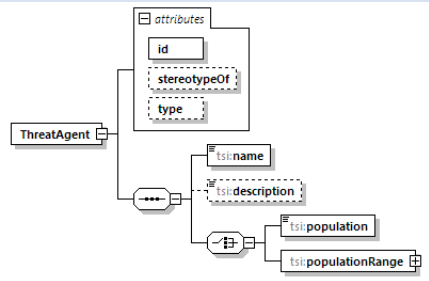


Figure 35: Sensitivity in the Representation of Threats

Table 29: Threat Agent Properties

Relationship	Properties	Schema
	<p>Threat Agent is a stereotype of Business Actor and contains the same set of properties, except that:</p> <ul style="list-style-type: none"> it cannot be designated as a Data Subject; its type property can take the value 'TECHNICAL' to represent computer viruses and other types of malware. 	

5.12 Business Geography

Business geography is easily modelled using the ArchiMate Location element unadorned. Location is intended to represent both physical or conceptual space. A means of marking a Location as a security domain (conceptual space) is discussed in Section 7.5: Domain Framework Model.

5.13 Business Time Dependencies

The SABSA Time cell is concerned with the delivery schedule of goals and responding to events.

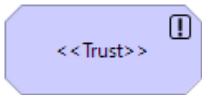
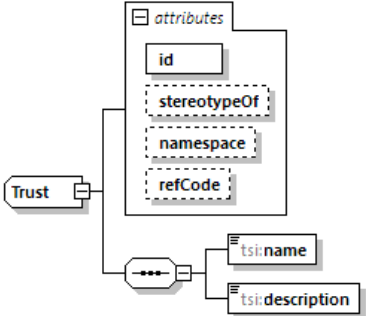
Target dates in ArchiMate are intrinsic to the definition of a Goal (a desired state to be reached by a defined point in time) and can be made explicit through Implementation and Migration views.

The Business Event element can be used unadorned. A means of using a Business Event as a Security event (conceptual), as used in the example of Figure 35, is discussed in Section 0:

Security Events.

Modelling the Conceptual Security Architecture.

Table 20: Trust Properties

Element	Properties	Schema
	<p>Trust enables the explicit modelling of any trust implicit in an interaction between two Parties.</p> <p>Its properties express:</p> <ul style="list-style-type: none">• An optional source and refCode link it to an external document.	

6 Modelling the Contextual Security Architecture

The latest edition of the SABSA framework is defined in the 2018 revision of the SABSA Matrices [Ref. 10]. Rows corresponding to Contextual Architecture artefacts and activities are reproduced in Table 21.

Table 21: SABSA Contextual Architecture

Contextual		Assets (What)	Motivation (Why)	Process (How)	People (Who)	Location (Where)	Time (When)
		Business Goals & Decisions	Business Risk	Business Meta-Processes	Business Governance	Business Geography	Business Time Dependence
		Business Value; Business Asset Taxonomy	Opportunities & Threats Inventory	Business Value Chain; Business Capabilities, Services & Functions	Operational Structure & the Extended Enterprise	Inventory of Buildings, Sites, Territories, Jurisdictions etc.	Time Dependencies of Business Goals & Value Creation
		Business Driver Development	Business Risk Analysis	Capability Management	Relationship Management	Supply Chain Management	Performance Management
Contextual	Activities	Business Benchmarking; Identification of new Drivers	Analysis of Internal & External Risk Factors	Managing Processes & Capabilities for Stakeholder Value	Managing Suppliers, Service Providers, Customers & Employees	Supply & Demand Management; Deployment & Consumption	Define & Monitor Business-Driven Performance Targets

This outline enables us to visualise the set of processes and activities necessary to create and maintain an architectural description (AD) of the Contextual Architecture, expressed in ArchiMate in Figure 28.

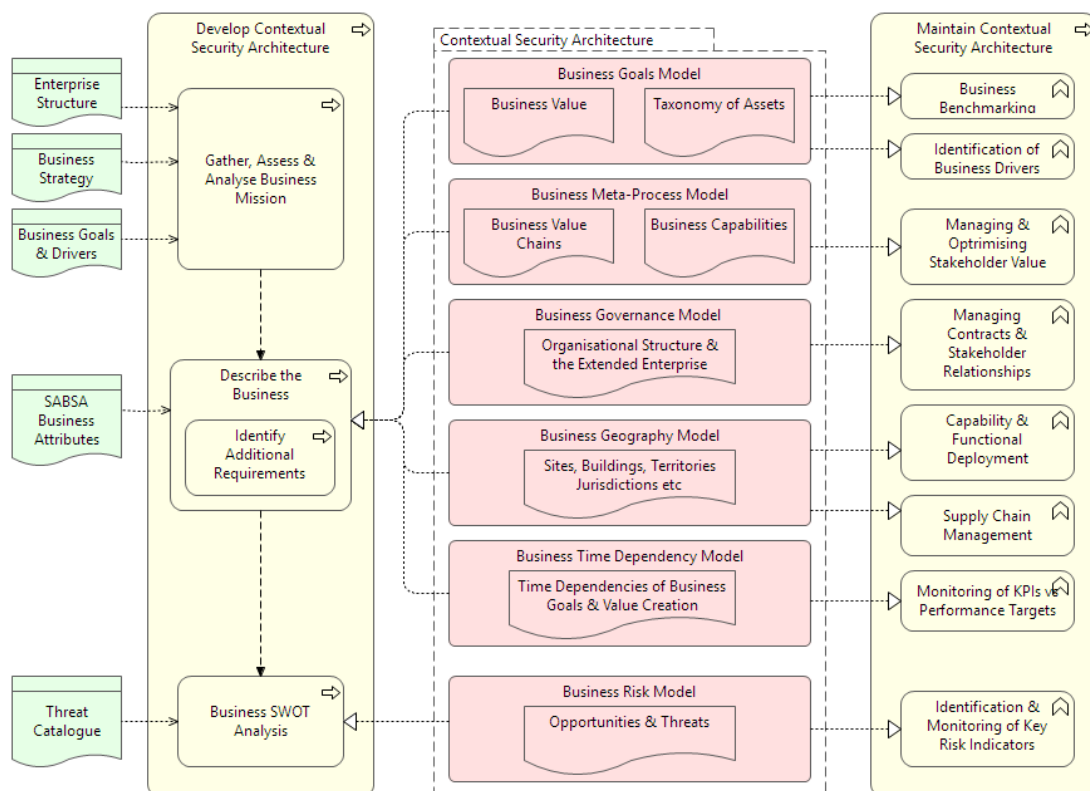
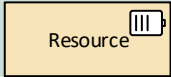
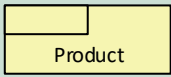
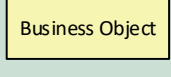
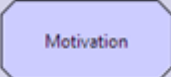











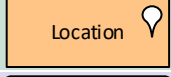
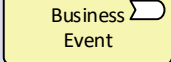


Figure 28: Developing and Maintaining the Contextual Security Architecture

This section discusses how elements from ArchiMate's Strategy & Business layers can be customised to express these Contextual artefacts as views of an ArchiMate model.

Table 22 **Error! Reference source not found.** summarises the native ArchiMate elements that are available to model the Contextual Layer.

Table 22: Contextual Elements

	SABSA Matrix	ArchiMate Element	Description
Contextual Layer	Asset (What)		<p>ArchiMate does not offer a generic, Asset element, As discussed in the previous section, assets are denoted by associating a Value element to represent its value, worth, utility or general importance to the organisation.</p> <p>Business Value is typically associated with the external appreciation of goods, services, information, knowledge, or money, normally as part of a customer-provider relationship, but supported by internal capabilities and processes at all layers that should be traced in the model.</p>
		 	<p>Value is most often expressed in financial terms. Still, non-monetary value is also essential to business, e.g. practical/functional value (including the right to use a resource or service, a certification or licence to operate) and the value of information, knowledge or intellectual property.</p> <p>Many Business and Strategic elements have intrinsic value, such as Resource (physical assets), Capability (the ability to harness resources to produce wealth) via Value Streams, the expected Outcome of some project or initiative, Business Information (Information Assets), Products, and Services that are bound to Contracts (commercial assets).</p>
	Motivation (Why)		<p>Motivation is expressed using elements from ArchiMate's Motivation aspect across all layers.</p>
	Process (How)	      	<p>Processes and meta processes can be expressed using behavioural elements from the Business and Strategy Value layers, respectively.</p> <p>The value generated by activity is most often expressed in financial, cost accounting terms. The previous section introduced the <<ValueChain>> stereotype as a means of expressing this.</p>
	People (Who)	   	<p>The active structure elements of the Business layer are used to represent organisational structure and relationships that can be formalised into contracts or as RACI roles.</p>
	Location (Where)		<p>The Location element represents a conceptual or physical place or position where concepts are located.</p>
	Time (When)		<p>The Business Event element denotes the moment or moments at which the event happens.</p>

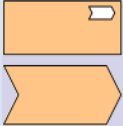
Many of these elements need to be elaborated with properties that bear on the system's security posture to perform effective security analysis. Others need to be stereotyped so that they can be assigned security properties when occurring in a security context. Achieving a consensus about the set of Element properties required to support security is a major goal of the Security Overlay initiative.

6.1 Business Assets

6.1.1 Capability and Value Stream

ValueStream is a behavioural element introduced to the Strategy layer in ArchiMate 3.1. The definition and notation are shown in Table 23.

Table 23: The Value Stream Element

Element	Definition	Notation
Value Stream	<p>A value stream represents a sequence of activities that create an overall result for a customer, stakeholder, or end-user.</p> <p>It can also be used to model a Value Chain, which according to the TOGAF definition, is a high-level view of an organisation: how it works, how it delivers value, how its functions are organised within an operating model.</p>	

A key principle of a Value Stream is that its value is always defined from the perspective of the Stakeholder: the consumer of the product, service, or deliverable and not on its intrinsic value: i.e. the cost of production. The value of Value Stream can be modelled in the ArchiMate language using the Value element. An example of a Value Stream is provided by the specification [Ref. 5 p. p.56] and reproduced in Figure 29.

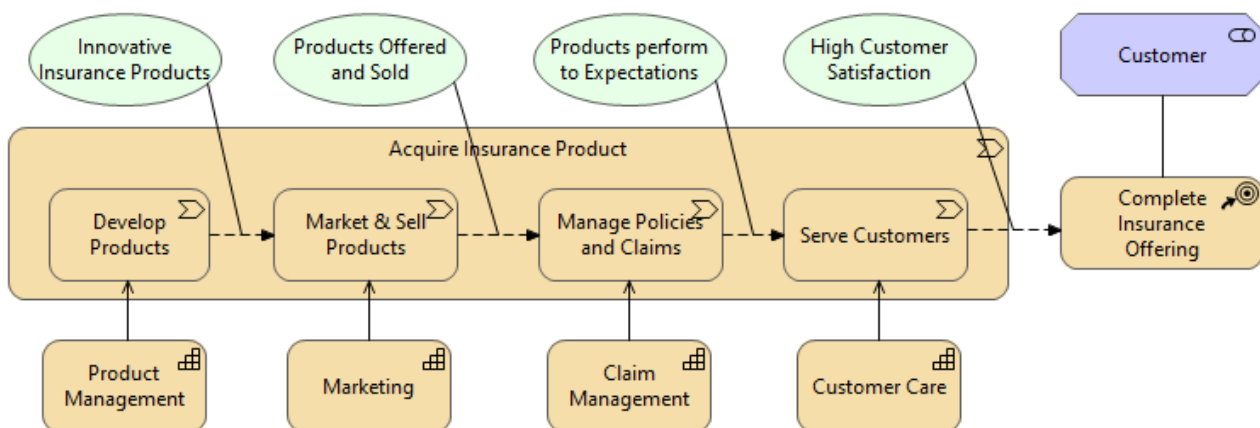


Figure 29: Value Stream Modelling

The Security Overlay proposes that the cost-accounting properties of processes and meta-processes should be externalised using a <<Value Chain>> stereotype to facilitate factoring and composition across all process elements in the Strategy & Business layers.

6.1.2 Business Object

Business Objects represent Information Assets. Its Security Overlay properties are shown in Table 24.

Table 24: Business Object Security Properties

Element	Properties	Schema
<div>Business Object</div>	<p>The Security Overlay defines the following properties:</p> <ul style="list-style-type: none"> a confidentiality classification to declare sensitivity to unauthorised disclosure; an integrity classification to declare sensitivity to unauthorised modification; an authenticity classification as assurance of the information's authenticity/origin; a classification of privacy sensitivity in terms of Personal Identifiable Information (pii), particularly if that information pertains to a child (couldBeMinor) and the need to be maintained up to date (reviewPeriod); any minimum retention period required by policy or legislation; <p>Many of these properties use enumerated types that can be tailored to the organisation.</p>	

The example of Figure 30 shows the relation with SABSA Attributes: a Business Object (Medical Record) has confidentiality property of 'CONFIDENTIAL' and is also tagged with the SABSA Attribute of the same name.

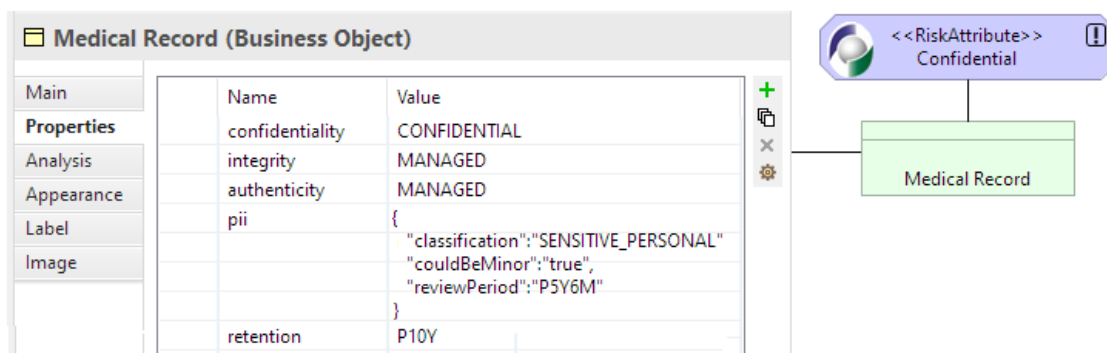


Figure 30: Element Properties and SABSA Attributes

This pattern begs the question: 'why is the confidentiality aspect of this information being modelled twice?

Consider this scenario: classified documents have been published on the Internet. Are they still secret?

In one sense, they are: the classification endures. These are still highly sensitive documents, and their publication may cause tangible harm. In another sense, they are clearly not because they are irrevocably in the public domain. Herein lies the answer: the element property is a protective marking, assigned by the Owner and immutable over the classification lifecycle. The Attribute represents a precious and fragile status, highlighted by the Architect as a characteristic that the enterprise architecture must protect.

The answer provides a useful insight into model validation. It would be curious if the information classified as CONFIDENTIAL were not linked to the Confidential Attribute as an outcome of the Attribute Profiling process. It would be similarly remarkable if the Attribute were associated with information classified as PUBLIC.

Modelling offers the possibility of SABSA Attributes being validated against such properties and patterns.

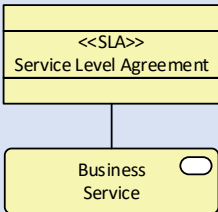
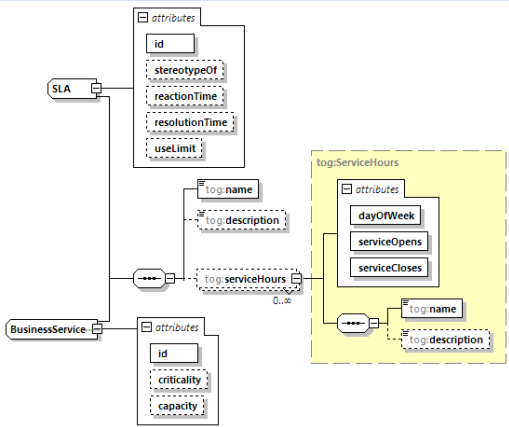
6.1.3 Business Service, Interface and Service Level Agreement (SLA)

In terms of asset analysis, the most important aspects of Service elements are those committed in a Service Level Agreement²⁴ that incurs penalties if not delivered. SLAs are enforceable contracts that set out expectations of service type, quality, and the resolution processes and remedies when these are not met. They may include a combination of technical services (applications or infrastructure), business (management) services (e.g. Support, Subscription, Accounting), data or products.

The SLA element's documentation should summarise the key contracted services, responsibilities and service level expectations. These can then be be modelled as SLA metrics using Security Overlay properties that reflect the most commonly occurring service parameters, i.e. availability, recovery time and recovery point objectives. A bespoke Requirement / Constraint may also be used where a simple property is insufficient.

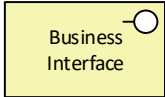
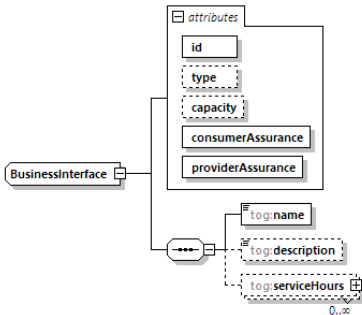
Business layer services need a distinct property set when they are offered through human interfaces. They may include reaction & response times, service windows or capacity limitations. Business services may be offered via several Interfaces (appointment, telephone, form submission etc.), each of which may exhibit different delivery characteristics: cost, capacity, service hours, etc. From a security perspective, these interfaces may also offer different possibilities for the authentication of consumers: by telephone, for example, being generally less reliable than a physical meeting. Table 25 shows a proposed schema.

Table 25: Business Service Properties

Element	Properties	Schema
	<p>Business Services are delivered via processes supported by human intervention.</p> <p>The SLA consolidates information on all contracted services and their delivery expectations as a combination of free text and user-defined properties.</p> <p>Services and SLAs exhibit similar properties from the perspective of Provider & Consumer respectively:</p> <ul style="list-style-type: none"> • The service availability window • committed reaction and response times on the handling of service requests; • limits on the number of requests submitted; • the business criticality of the service; • the maximum capacity of the service 	

²⁴ Or to a lesser extent, an Operational Level Agreement (OLA) between business functions in the same organisation.

Table 25: Business Service Properties

Element	Properties	Schema
	<p>Business Services may be offered via multiple interfaces, each offering a different capacity, service window, or other quality of service parameters.</p> <p>The interface type is an enumeration of possible channels: TELEPHONE, MAIL, APPOINTMENT etc.</p> <p>Interfaces also differ in their ability to identify or authenticate the Service Consumer and Provider: consumerAssurance & providerAssurance.</p>	

In terms of validation, SLA metrics are strong candidates for Attributes Profiling during security analysis. Once tagged with an Attribute, model validation rules can ensure that it is supported by Control Objectives and Requirements that satisfy the protection profile. It would also be possible to detect a service that depends on other services that do not meet its minimum service levels requirements.

Interfaces and services exposed over a domain boundary merit analysis using a Trust.

6.2 Business Risk

Risk, threats, vulnerabilities and opportunities can be modelled at the Business and Strategy level using the orthodox ArchiMate approach discussed in Section 4.2: Risk & Security Modelling in ArchiMate. The Security Overlay adds stereotypes of Motivation layer elements for this purpose, introduced in Section 5.5: Impact, Threat, Vulnerability & Risk.

6.3 Business Process / Function / Interaction

Just as Data Owners classify business information, business processes are often categorised in terms of their criticality to core business mission, capabilities and value chains. Security analysis needs to be able to recognise such processes and protect them accordingly.

Behavioural elements may also be deemed sensitive due to the way they operate on information. Element properties should reflect the privilege level expected to perform the activity and those of the access relation to reflect the sensitivity of the information and the nature of the access: Behaviour element properties include:

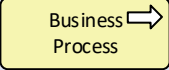
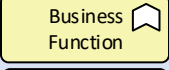
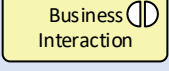
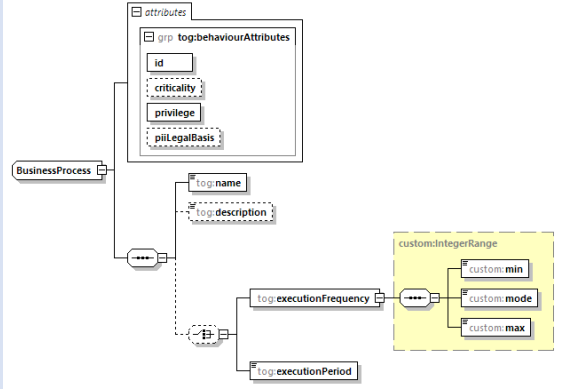
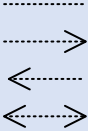
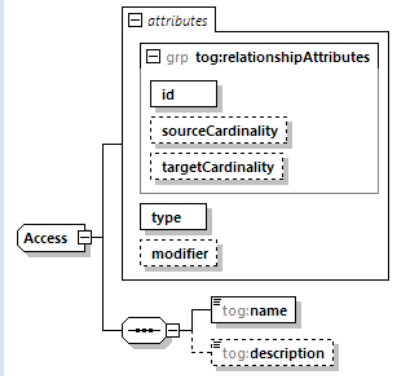
- a criticality indicator that denotes the importance of the activity to the business;
- a 'privilege level' indicator that denotes behaviour requiring some degree of restriction, perhaps because it handles high-value information or has the potential to impact other users;
- If the processing handles personal data, it should state the legal basis for such processing;

and for the access relation through which behaviour elements operate on Business Objects:

- The standard access modifiers (read, write, read-write & access) can be refined using a controlled vocabulary that includes: publish, subscribe, delete, copy, rename, encrypt, set access profile etc.;

- cardinality, i.e. whether the process operates on a single object or has access to multiple objects.

Table 26: Business Behaviour Properties

Element	Properties	Schema
  	<p>The business behaviour elements that denote:</p> <ul style="list-style-type: none"> • the criticality to the business; • a 'privilege' level that indicates whether the process is performed on behalf of the assigned Actor, another user, as a supervisor of other users or in general administration; • the legal basis for any processing of personal data; • the executionFrequency & period are used to express as a confidence range, i.e. how often the process is performed e.g. 10 - 20 times per week <p>In addition, Business Interaction enumerates any need for segregation of duties between the participants.</p>	
	<p>Modifiers for access relations include:</p> <ul style="list-style-type: none"> • read: SCAN, SEARCH, SUBSCRIBE, VERIFY • write: CREATE, APPEND, PUBLISH; • read/write: UPDATE, SIGN, ENCRYPT, DECRYPT; • access: COPY, MOVE, DELETE, ERASE, RENAME, ARCHIVE, SET_READONLY, SET_ACCESS <p>sourceCardinality expresses how many process executions may be occurring in parallel; targetCardinality indicates how many data records can be accessed per execution</p>	

Several validations become feasible with these properties:

- critical behaviour elements must be associated with at least one SABSA Attribute that characterises the nature and degree of the criticality
- the privilege level of the behaviour must be sufficient for:
 - creation rights on non-repudiable information;
 - read access to confidential/personal information or modifying access to high integrity information;
- behaviour elements declaring no processing of personal data should not be accessing personal data;
- behaviour elements accessing personal data must declare the legal basis for processing;
- identification of sensitive processes associated with access to high volumes of sensitive information requiring high assurance security services, e.g. strong authentication.

6.4 Business Roles & Actors

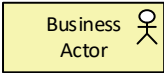
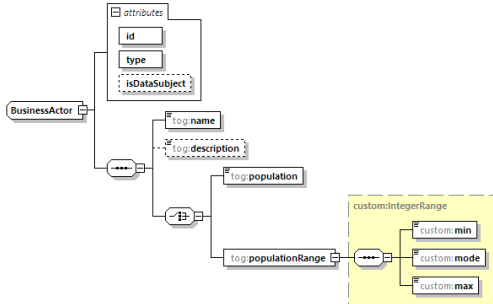
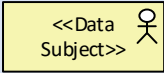
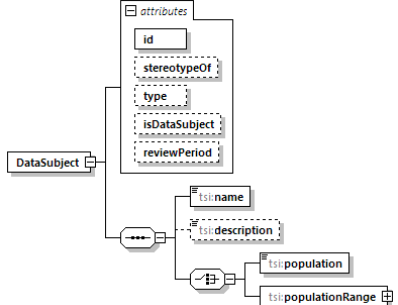

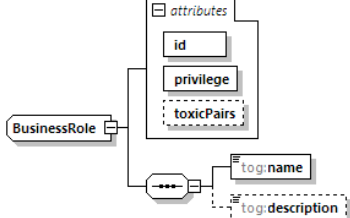
In ArchiMate, Actors represent human or organisational entities that can be assigned to Roles that describe:

- the extent of their responsibilities with respect to a given business process;
- their use of business and application services.

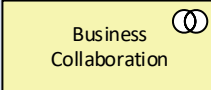
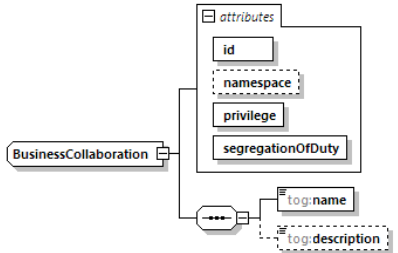
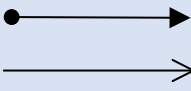
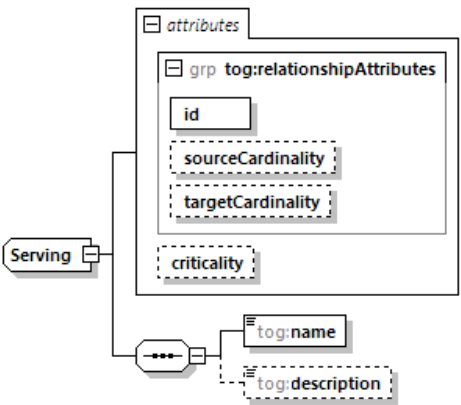
Security analysts should be able to describe the following relevant information from the model:

- whether an Actor is human or organisational (e.g. a corporate legal entity, department or team)²⁵;
- if human, whether they qualify as a Data Subject and if so, the possibility of being a minor;
- whether any Role to which the Actor has been assigned is considered privileged;
- whether the Role is considered part of a 'Toxic Pair' in a process requiring segregation of duties;
- the expected number of Actors (individuals/entities) assigned to a Role (many to one)
- the expected number of service requests (normal and peak demand) generated by a constituency.

Table 27: Actor & Role Properties

Element	Properties	Schema
	<p>A Business Actor should declare:</p> <ul style="list-style-type: none"> • its population size expressed as a point value or an expected range; • its type (HUMAN or ORGANISATION); • a flag, if also a Data Subject, e.g. a member of staff in the context of an HR system; 	
	<p>If the Actor is primarily a Data Subject, e.g. an external Actor such as a citizen, this can be emphasised by using the DataSubject stereotype. DataSubject offers additional properties:</p> <ul style="list-style-type: none"> • a flag to signal the possible participation of minors; • A review period defining how often any personal data records should be reviewed and updated. 	
	<p>A Business Role should indicate</p> <ul style="list-style-type: none"> • the privilege level assigned to the role, using the same set of enumerations defined earlier; • toxicPairs to mark the role's sensitivity to any segregation of duty conflicts. 	

²⁵ The ArchiMate specification does not envisage a technical entity being modelled as an Actor. On one hand, it would not be appropriate for technical entities to appear in the Business layer but on the other, they are Principals that have identities, need to authenticate, are granted access rights to consume services etc More on this topic in **Error! Reference source not found.**

	<p>Business Collaboration is the ArchiMate element to which multiple roles can be assigned.</p> <p>It is the natural place model any process with a Separation of Duty requirement.</p> <p>The segregationOfDuty property indicates whether segregation should be enforced between participating Actors (i.e. individuals possibly in the same role) or by distinct Roles.</p>	
<h2>Relationships</h2>		
	<p>Assignment relationships just support the basic cardinality properties:</p> <p>sourceCardinality expresses the number of Actors assigned to a given Role or the number of active Roles performing a given behaviour simultaneously.</p> <p>targetCardinality is normally 1.</p> <p>Serving relationships extend the basic relationship schema with a criticality property that denotes the importance of the service to the Consumer.</p> <p>targetCardinality indicates the number of concurrent service consumers/sessions.</p> <p>sourceCardinality indicates the number of request/response exchanges generated for each consumer process/session execution.</p>	

These properties support the following validations:

- Data Subjects, which can only be specialised from human Actors, must be associated with at least one element of Business Information with the PII property set;
- with respect to Toxic Groups that are segregated by Role²⁶)
 - an Actor being assigned to multiple Roles belonging to the same Toxic Group
 - a technical account participating in Role requiring segregation of duties;
- that the rate of service requests remains within the service's capacity;
- identification of high-security risk roles: associated with large constituencies of Actors having execute critical processes or sensitive access services;

6.4.1 Governance

Governance runs like a seam through the People column of the SABSA Matrix, exhibiting two main aspects:

- Organisation structures representing decision-making bodies, review boards, steering committees etc. They can be adequately modelled as Collaborations to which all the participating Roles can be assigned;
- Individual involvement as might be expressed by a RACI (Responsible / Accountable / Consulted / Informed) matrix;

²⁶ Rather than segregated on the basis of distinct individuals who may share a common organisational or business role.

RACI presents an interesting design consideration in ArchiMate. The language is predicated on a grammar in which Active Structure's (Actor or Role) access to Passive Structure (the Asset) is mediated by a Behaviour element in a "Subject–Verb–Object" syntax. Applying this to RACI requires us to consider what the 'Process of Being Accountable' means and what would it look like. Instead of contriving some pattern that fits this grammar, the Security Overlay proposes to stereotype a directed association: <<RACI>> specifically for this purpose. The example of Figure 31 shows how it can be used.

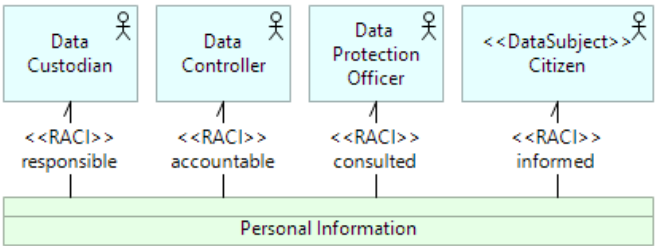


Figure 31: Representing RACI Relationships

A note of caution about using generic names for RACI targets: care should be taken not to create Singularities.

Consider the example of Figure 32a: 'Data Owner' is a poor choice of name for a RACI role. It is likely to be re-used in multiple contexts in multiple views. While it still may be possible to understand the modeller's intent from each view in isolation, the entanglement in the underlying model, shown in Figure 32b, renders automated analysis impossible.

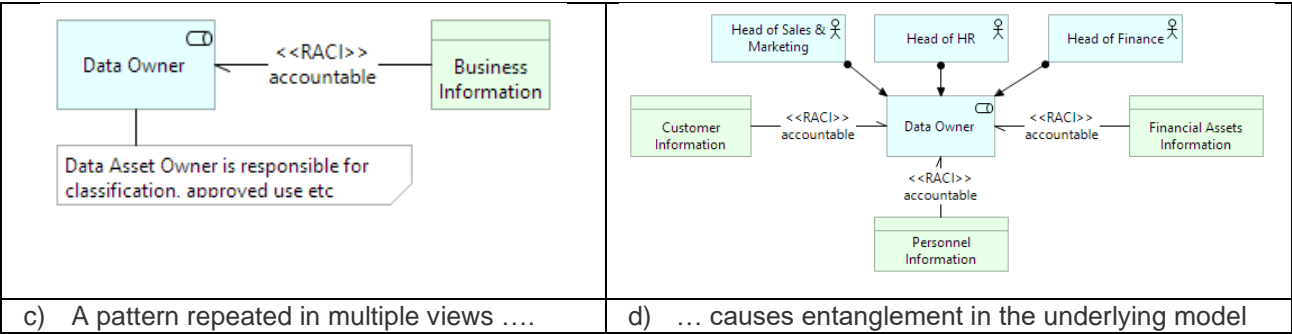


Figure 32: Avoid RACI Entanglement

Alternative patterns that avoid this issue are shown in Figure 33.

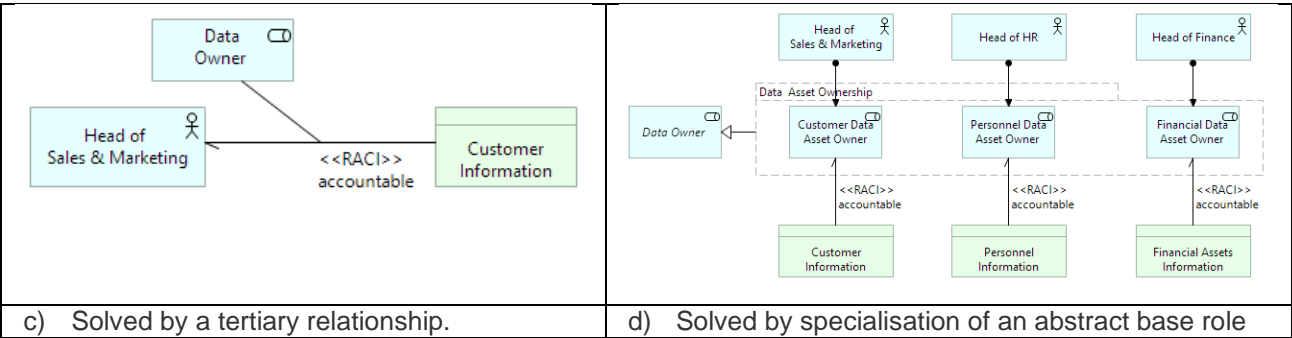
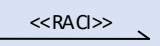
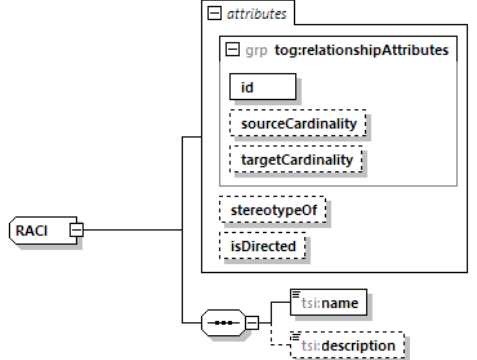


Figure 33: Recommended RACI Patterns

Table 28: RACI Properties

Relationship	Properties	Schema
	<p>RACI is a stereotype of Association in which the relationship name is restricted to the enumerated values: RESPONSIBLE, ACCOUNTABLE, CONSULTED or INFORMED.</p> <p>Directionality is fixed to TRUE and directed towards the Actor or Role.</p> <p>sourceCardinality expresses the number of items that are within the scope of the RACI association.</p> <p>targetCardinality is normally 1.</p>	

6.4.2 Threat Actors

There are three possible ways to model threat actors, depicted in Figure 34:

- As a stereotype of Actor: a malicious organisation that is known to pose a threat;
- As a stereotype of Role representing a malicious intent, engaged against the system;
- As an adverse action (behaviour or event) that occurs by accident or design;

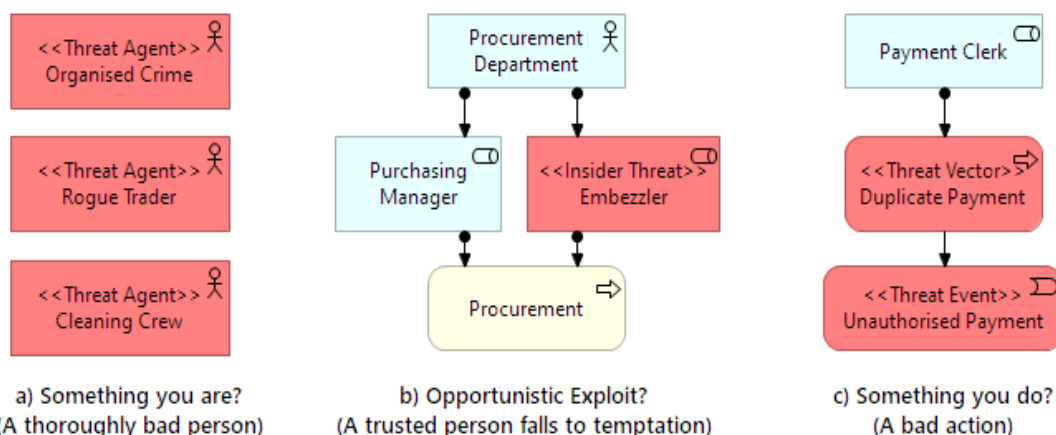


Figure 34: Possibilities for Modelling Threat Actors

Option (a) is seductively intuitive. It provides a useful node for collating information about various Threat Agent constituencies (population size, motivation, capability, determination, etc.), all useful to quantitative risk modelling. It's fine for cases where the perpetrator isn't explicitly identified (e.g. Organised Crime & Rogue Trader). It may even be reasonable to identify a particular Hacker Collective that poses a concern. Caution should be exercised if using <<Threat Agent>> to label specific Actors (competitors, states and certainly not internals, who may even have access to the model) as malicious. Defaming the Cleaning Crew in this way is unacceptable.

Option (b) suffers from the same issue: if we accept that internal staff are overwhelmingly honest and well-intentioned, it is likely to cause great offence if they are modelled with an explicit assumption of criminality.

Option (c) is similarly problematic when alluding to malicious intent. It also misrepresents exceptional behaviour as a formal process. Modelling accidental errors and omissions in this way is less contentious.

Although analysis of Insider threats is unavoidable, we should be conscious of these sensibilities in our models. It is better to contain analysis of misplaced trust and abuse to specific risk views that can be restricted from the main published model. A better style in which sensitive threat-related elements have a degree of separation from (i.e. are not directly attached to) Actors and Roles is shown in Figure 35.

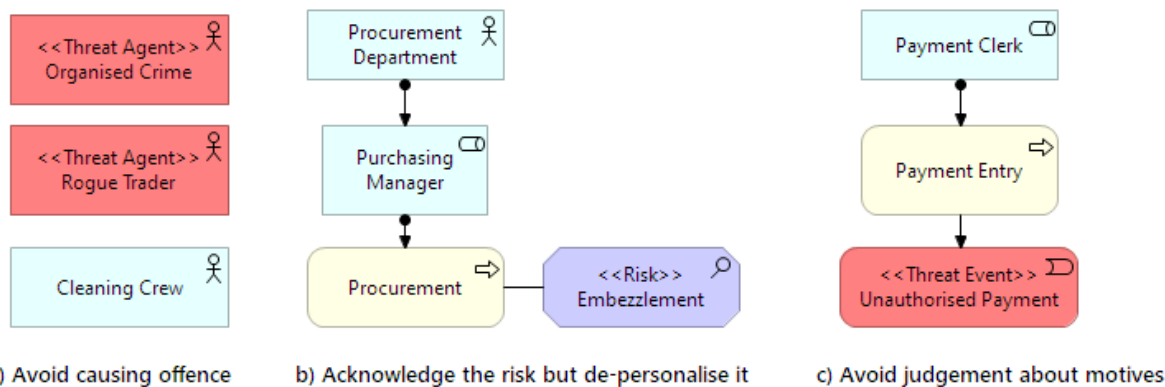

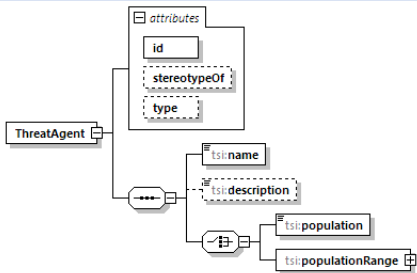


Figure 35: Sensitivity in the Representation of Threats

Table 29: Threat Agent Properties

Relationship	Properties	Schema
	<p>Threat Agent is a stereotype of Business Actor and contains the same set of properties, except that:</p> <ul style="list-style-type: none">it cannot be designated as a Data Subject;its type property can take the value 'TECHNICAL' to represent computer viruses and other types of malware.	

6.5 Business Geography

Business geography is easily modelled using the ArchiMate Location element unadorned. Location is intended to represent both physical or conceptual space. A means of marking a Location as a security domain (conceptual space) is discussed in Section 7.5: Domain Framework Model.

6.6 Business Time Dependencies

The SABSA Time cell is concerned with the delivery schedule of goals and responding to events.

Target dates in ArchiMate are intrinsic to the definition of a Goal (a desired state to be reached by a defined point in time) and can be made explicit through Implementation and Migration views.

The Business Event element can be used unadorned. A means of using a Business Event as a Security event (conceptual), as used in the example of Figure 35, is discussed in Section 0:

Security Events.

7 Modelling the Conceptual Security Architecture

The Conceptual Architecture provides the architect's view of the System of Interest. It is concerned with bridging real-world entities from the Contextual layer into abstractions for the Logical layer. The Security Architect must also devise a strategy to protect these assets in the virtual world: to assure that ICT systems reliably deliver the key capabilities supporting the mission despite threats and uncertainty.

The rows corresponding to the Conceptual Architecture's artefacts and activities are defined in the 2018 revision of the SABSA Matrices [Ref. 10], reproduced in Table 30.

Table 30: SABSA Conceptual Architecture

		Assets (What)	Motivation (Why)	Process (How)	People (Who)	Location (Where)	Time (When)
Conceptual	Artefacts	Business Value & Knowledge Strategy	Risk Management, Strategy & Objectives	Strategies for Process Assurance	Security & Risk Governance; Trust Framework	Domain Framework	Time Management Framework
		Business Attributes Taxonomy & Profile	Enablement & Control Objectives; Policy Architecture; Risk Categories; Risk Management	Operational Process Inventory; Process Mapping Framework	Owners Custodians & Users; Trust Modelling Framework	Security Domain Concepts & Framework	Through-Life Risk Mgmt Framework; Attribute Perf. Targets
	Activities	Proxy Asset Development	Developing Risk Objectives	Delivery Planning	Role Management	Business Portfolio Management	Service Level Definition
		Defining Business Attributes Profile: Performance Criteria, KPIs & KRIs	Maintaining the Risk Modelling Framework; Risk Analysis on Business Attributes Profile	SLA Planning; BCP; Financial Planning; Transition Planning; Services Catalogue	Maintain Trust Modelling Framework; Liabilities; Define Roles & Responsibilities	Business Footprint; Points of Supply & Access	Manage Performance Criteria & Targets; Abstracting Attribute Performance Targets

This outline enables us to visualise the set of processes and activities necessary to create and maintain an Architectural Description (AD) of the Conceptual Architecture, expressed in ArchiMate in Figure 36.

Although the Conceptual layer has no ArchiMate equivalent, many of the asset (What) and motivation (Why) concepts (attribute profiles, metrics, risk, compliance, control objectives & layering etc.) have already been discussed as stereotypes of Motivation elements (Section 5: The Motivation Aspect).

This section discusses the practical use of these elements in risk management process assurance, identifies conceptual security services (How) and makes proposals for the remaining artefacts from the People (Who), Location (Where) and Time domain (When) models.

Finally, it considers how the Conceptual layer can be inserted between the Contextual and Logical layers in an ArchiMate model without breaking existing models or abusing the language (too much!).

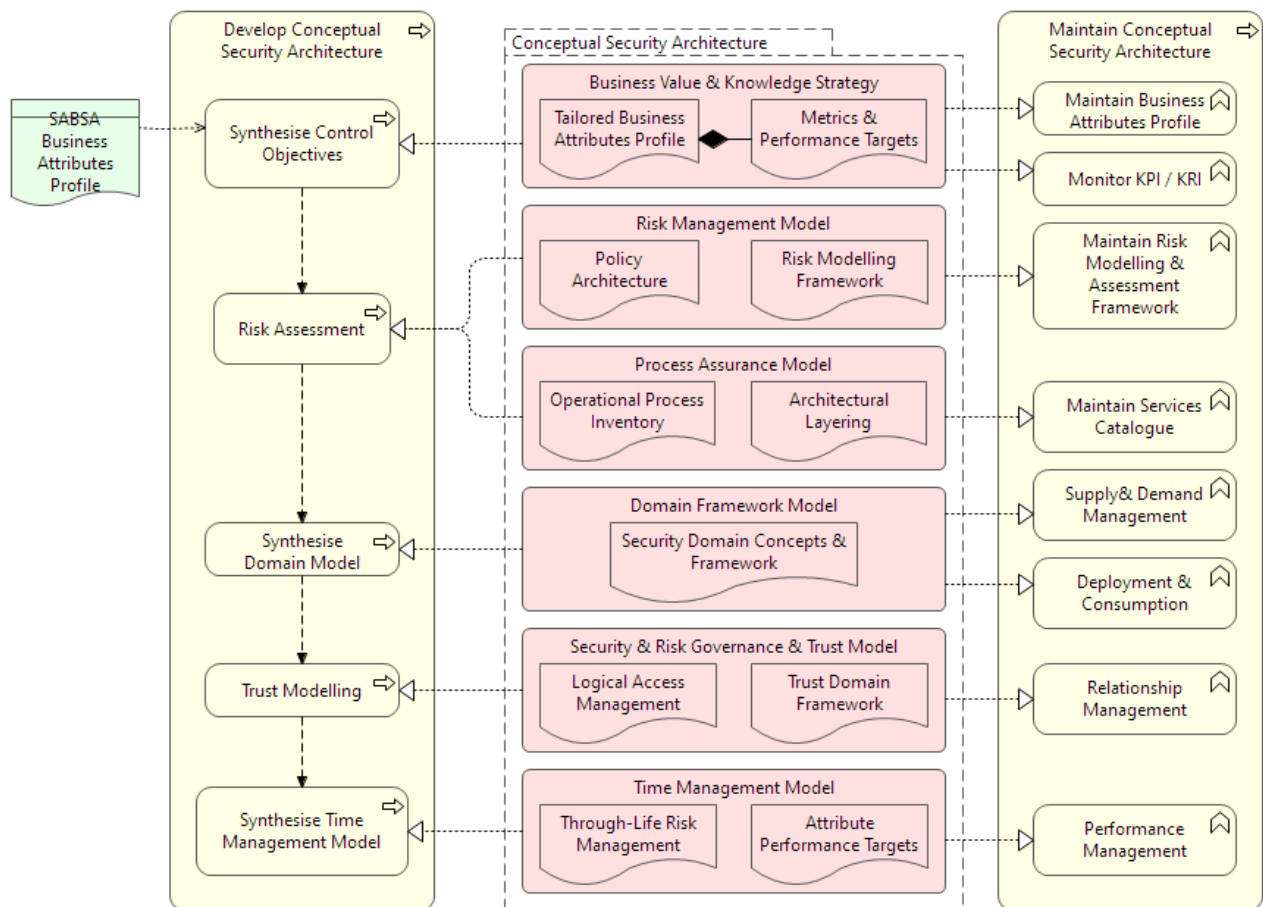


Figure 36: Developing the SABSA Conceptual Security Architecture

7.1 Attribute Profiling

Section 5.3 described how SABSA Attributes can be represented in ArchiMate. To recap the key points:

- SABSA Attributes are at the apex of the motivation pyramid: more abstract than Control Objectives;
- They are global atomic singletons: existing at most once in the model and never appearing, even as copies, more than once in any ArchiMate diagram;
- In models, Attribute profiles are constructed using influence relationships to form tree structures that broadly follow the layered architecture.
- Each Attribute can influence/be influenced by zero or more other Attributes, but the tree hierarchy must hold: Relationships do not influence downwards, so circular paths should not exist in the structure;
- Attributes should be associated to the elements to which they apply and must be influenced by another Motivation element: typically, another Attribute and ultimately by a Control/Compliance Objective.

This section describes how to create such a profile through a worked example of GDPR Article 5, which sets out the principles relating to the processing of personal data. Article 5 §1 clause (a) states:

“Personal data shall be (a) processed lawfully, fairly and in a transparent manner in relation to the data subject (‘lawfulness, fairness and transparency’);”

By identifying the key adjectives in the text, (marked in **bold**) we can select appropriate Attributes from the SABSA Attribute Taxonomy to create the initial Attribute Profile of Figure 37.

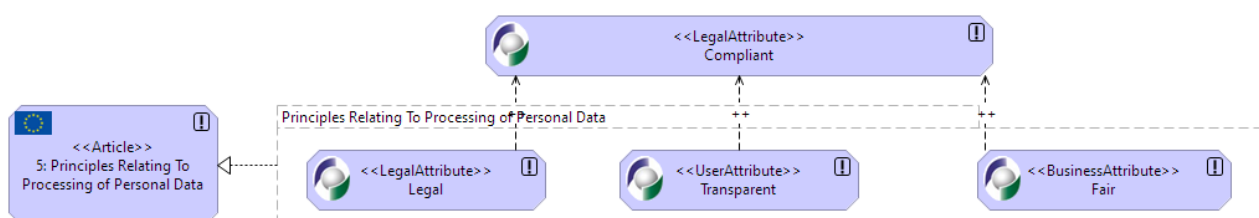


Figure 37: Attribute Profiling (i)

If there had been an explanation of these terms in the article text, it would have been possible to create Compliance Objectives at this point, but since there is none, analysis proceeds to clause (b).

“(b) collected for **specified**, **explicit** and **legitimate** purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes (‘purpose limitation’);”

If we accept ‘**legitimate**’ as a synonym of ‘**legal**’ and that anything ‘**specific**’ is both, by definition, ‘**explicit**’ and contributing to ‘**transparent**’, then we can refine the Profile to that of Figure 38. Note that the caveat to Clause (b) is a candidate <<Exception>> to any Compliance Objectives derived from this profile.

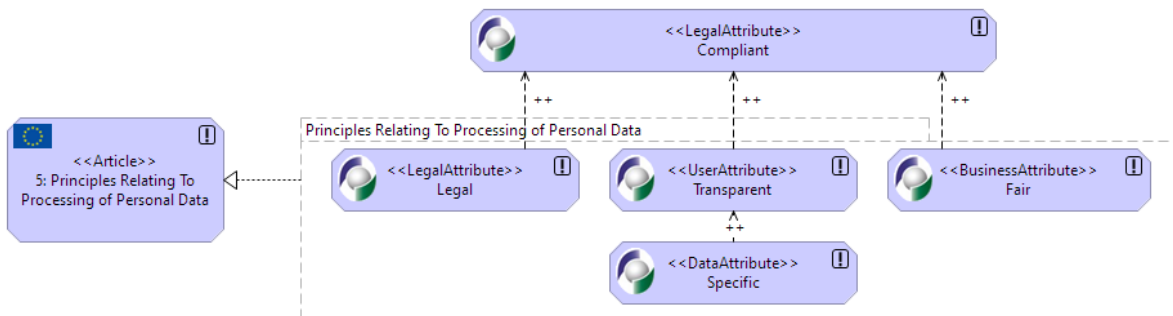


Figure 38: Attribute Profiling (ii)

Continuing this process with Clauses (c) and (d), the profile can now be elaborated to that of Figure 39.

- “(c) **adequate, relevant and limited** to what is necessary in relation to the purposes for which they are processed (‘data minimisation’);
(d) **accurate** and, where necessary, **kept up to date**; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay (‘accuracy’);”

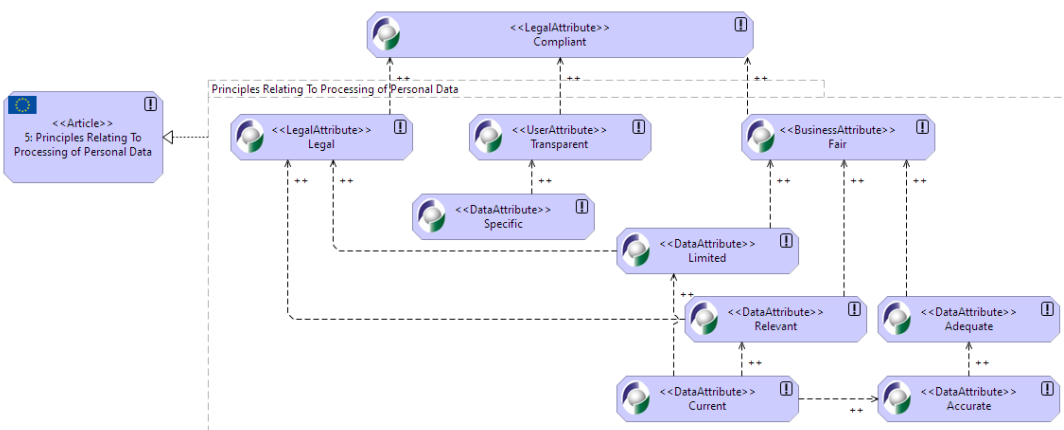


Figure 39: Attribute Profiling (iii)

By similarly analysing clause (e) and (f) and §2, we arrive at the completed profile shown in Figure 40.

- “(e) kept in a form which permits **identification** of data subjects for no longer than is necessary for the purposes for which the personal data are processed;
(f) processed in a manner that ensures appropriate security of the personal data, including **protection** against unauthorised or unlawful processing and against **accidental loss, destruction or damage**, using appropriate technical or organisational measures (‘**integrity and confidentiality**’).
2.The Controller shall be **responsible** (sic²⁷) for, and be able to demonstrate compliance with, § 1 (‘**accountability**’);”

²⁷ The Controller is actually *accountable* for demonstrating compliance using RACI terminology.

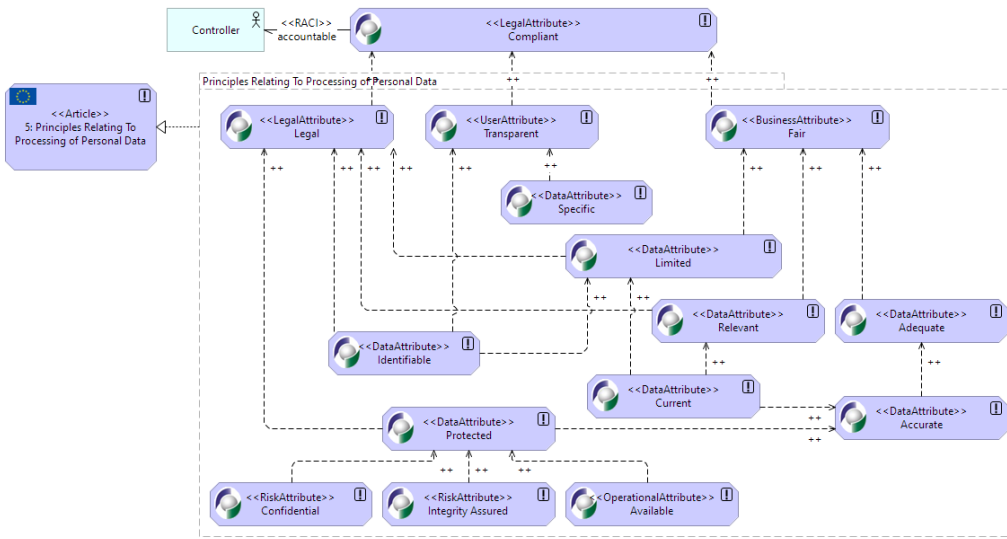


Figure 40: Completed Attribute Profile

The final Attribute Profile captures the essence of Article 5: the leaf nodes [Confidential, Integrity-Assured, Identifiable, Specific, Current & Available] for which metrics and Compliance Objectives shall be defined.

7.2 Risk Management & Strategy

Risk and its constituent factors (threat, vulnerability and impact) are modelled as stereotypes of Assessment (adhering to the guidance in Sections 4.2 & 5.5). The Security Overlay provides high-level representations of these concepts that can be adapted to suit an organisation's preferred risk methodology. The schema for these elements has already been presented in Table 15: Risk Element Properties.

7.2.1 Risk Management

The Motivation aspect of the SABSA Matrix also addresses the process of risk management. Recalling the discussion of architectural planes in Section 4.1, a risk analysis process would be modelled in the secondary architecture (Management Processes) and look something like that shown in Figure 41.

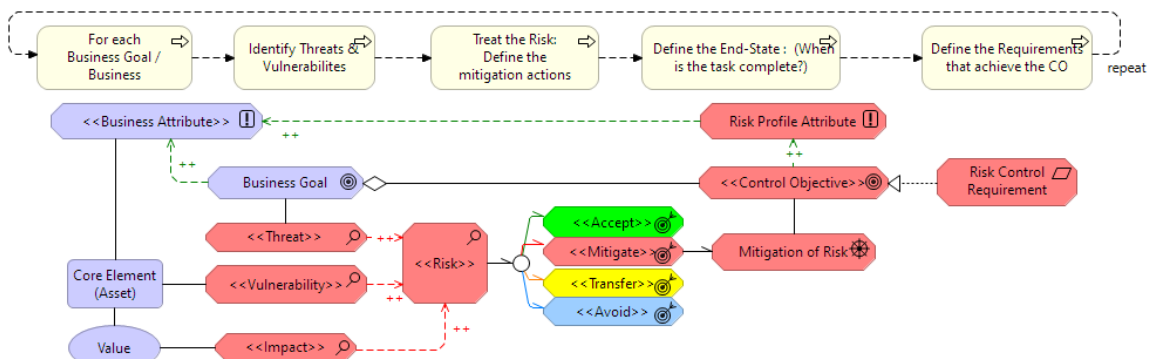


Figure 41: Business Risk Analysis Process

The first step identifies assets of significant value and the SABSA Attributes in which that value is enshrined. For each Attribute, a set of concrete Business Goals are defined that describe the desired future state. The risk analysis process takes each Business Goal in turn and performs:

- Threat Analysis to identify threats and uncertainties affecting the Business Goal;
- Vulnerability Analysis to identify weak points in attack vectors that could expose Attributes to this threat;
- Impact Analysis to assess potential loss of Asset value, should the threat materialise.

The risk assessment considers the severity of the threat, the exposure of the vulnerability, and the value of the Asset to determine the nature and scale of the risk. An influences relation is proposed here to reflect directionality and inherent subjectivity by which these factors influence risk analysis.

In the risk treatment phase, the Owner of a Business Goal must decide to accept, mitigate, transfer, or avoid the risk (modelled as stereotyped **Outcomes** of the Treatment process). The latter two options will most likely result in some redesign of the Contextual model, so the need to model them explicitly may be transient.

A <<Mitigate>> outcome creates a **Driver** for change. Ideally, a **triggers** relationship would be available here, but directional association must suffice as the specification forbids motivational elements from participating in dynamic relationships. Mitigation must be addressed by *Control Objectives*, aggregated into the Business Goal. Aggregation best reflects the many-to-many nature of the Goal – Control Objective relationship.

Of the alternative relations, composition implies that a Control Objective is inherent and exclusive to a single Business Goal. Realisation presents each Control Objective as a choice, each sufficient to achieve the Goal.

7.2.2 Policy Architecture

The Policy Architecture is a framework of codified statements of regulatory compliance controls or those documented in the organisation's policies and standards. As discussed in Section 5.8, policies, standards and regulations are modelled as Representation stereotypes containing a structure of Groupings, Articles/Mandates, Compliance Objectives and Requirements.

There are significant differences between controls mandated by policy and those derived from risk:

- Policies are predicated on Statements of Applicability that define the policy's scope and the conditions under which its controls apply. If the System of Interest is within scope and meets the criteria, the control statements must be applied, regardless of whether the organisation is willing to accept the risk or not;
- Emphasis on the traceability/audit of legal, compliance or certification requirements to controls.

In order to create models that can help manage the organisation's compliance posture, the Security Overlay provides elements that support the architectural pattern shown in the metamodel of Figure 42.

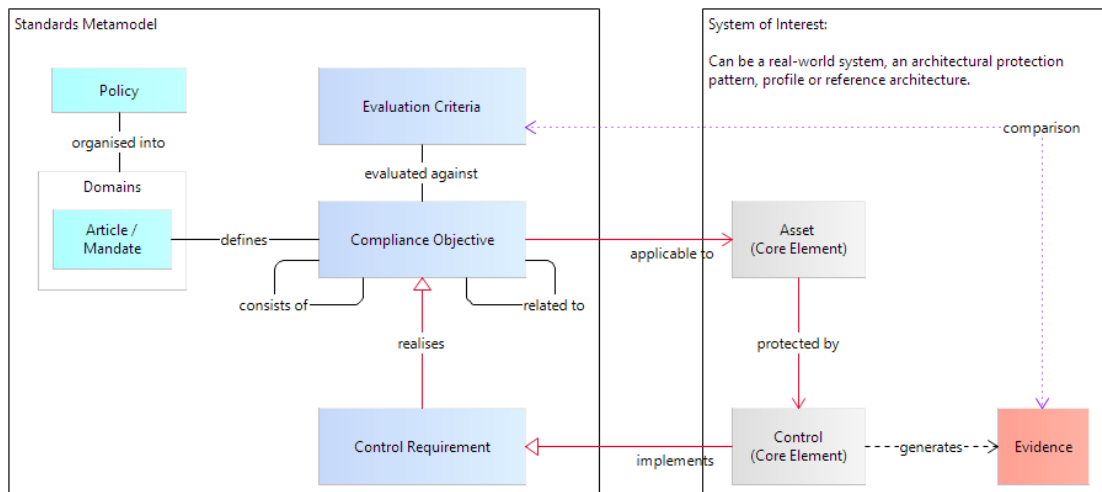


Figure 42: Compliance Metamodel

The structure of the compliance document is shown on the left, the target architecture on the right. Two kinds of relationships link them: one denoting the applicability (applicable to) of Compliance Objectives to Assets (core layer elements) and the other implementation (implements) of Control Requirements (and Constraints) by real-world Controls.

An acceptable compliance posture is demonstrated if, and only if, the following can be established:

- that each Asset to which an Objective applies is protected by a Control that implements each of the Requirements necessary to realise that Objective: i.e. there exists the closed path, marked in red;
- that all Controls on these paths produce evidence of effectiveness that meet minimum criteria.

The former, necessary but not sufficient, may be established by path analysis of the model.

The extent to which the latter can be automated varies on a control-by-control basis. In any event, such analysis would be performed outside the architectural model (unless you are contemplating a daily import of Production log files into your EA tools !!!). The model should at least reference the locations where the necessary evidence can be found.

Figure 43 shows how this might be applied to model compliance with NIST SP800-53r5 Objective, AC-1.

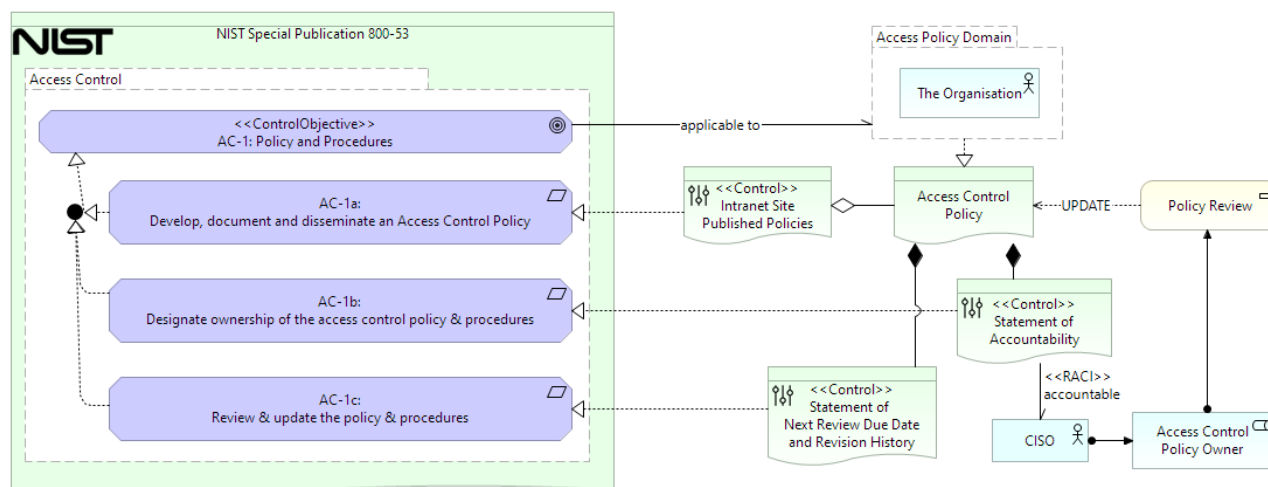


Figure 43: An Example Compliance Model

Given their sheer quantity, the ability to trace policy statements through to the realisation of controls is perhaps one of the most valuable of model validation capabilities.

7.2.3 Multi-Regulatory Compliance

A typical policy, standard or regulation contains several hundred compliance statements and requirements. The task of tracing them to effective control measures via manual methods presents a significant challenge. Unfortunately, most organisations operate in a multi-regulatory environment in which they are subject to multiple, simultaneous control frameworks (imposed by regulators, service providers, key clients or market demand). Moreover, the digital economy demands compliance with ever more standards, with increasing rigour, more frequently and held to ever high levels of scrutiny. A strategy of addressing each framework in isolation would not only require a huge commitment of resources, but it would also be wasteful, given the overlap and duplication in their scope and objectives. Consequently, an aim of modelling should be to refactor these controls, drawn from disparate risk and compliance sources, into a minimal, consolidated control set.

The modelling approach outlined in this paper can help achieve this consolidation. This section explains how. We have seen in Sections 0 and 7.1 that we can build an Attribute profile from a multi-tiered risk analysis or compliance mandate that produces a structure that resembles either the right or left side of Figure 44.

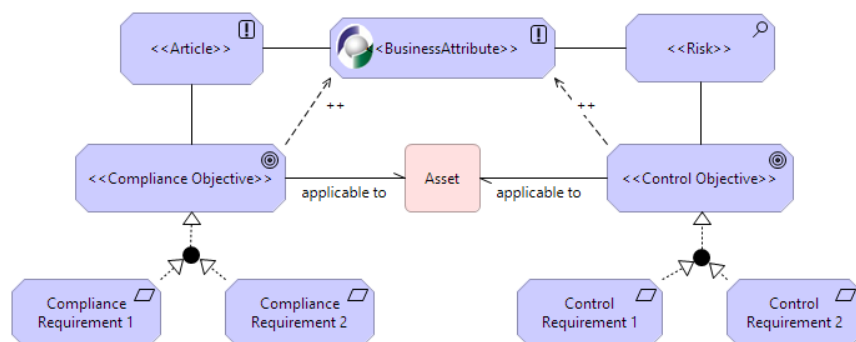


Figure 44: Control Consolidation

After investing a significant effort in profiling all assets in the architecture, the model may easily contain several thousand control elements (Compliance & Control Objectives and Requirements). Most modelling tools provide a means to generate a view from the perspective of any element, so it is relatively easy to enumerate all the control statements that apply to any given Asset.

Unfortunately, because these control statements have been derived from different sources at different times, their titles and descriptions are expressed in free-form text, making it difficult to detect potential duplicates. However, because these statements have been linked to an Attributes Profile and that SABSA Attributes are atomic singletons, we can be fairly confident that control statements that influence the same SABSA Attribute are good candidates for consolidation. By re-wording the control statement title and description, combining similar statements into one is often possible. Even for control statements too complex to merge easily, the fact that they influence the same Attribute means that rationalisation should be possible at the sub-goal or requirement level.

Figure 45 shows this in practice using a generated view of the SABSA Attribute: 'Available'. It shows this Attribute being influenced by four Objectives: two from GDPR and two from ISO 27001.

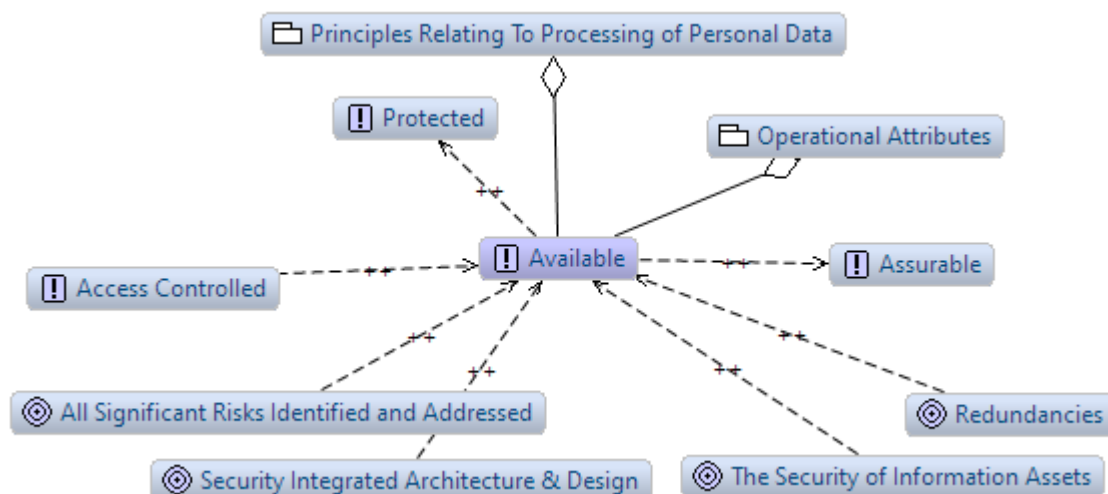


Figure 45: Possible Duplicate Objectives, coupled through Attribute Profile

7.3 Conceptual Security Services

The Process cell of the SABSA Matrix is the place to consider the “How” of Conceptual Layer Services.

Section 4.2 describes how Control Requirements must eventually be implemented by concrete measures in the core layers, shown in Figure 8 as Security Services and later extended in Section 5.10 by the Security Overlay’s <<Control>> stereotype for non-service-oriented mechanisms. However, two important security services must be represented at the conceptual level because they mediate the use of Logical layer resources by Business Layer Actors. We speak, of course, about Authentication and Access Control.

So how should we represent these services in our models? Figure 46 shows some possible options.

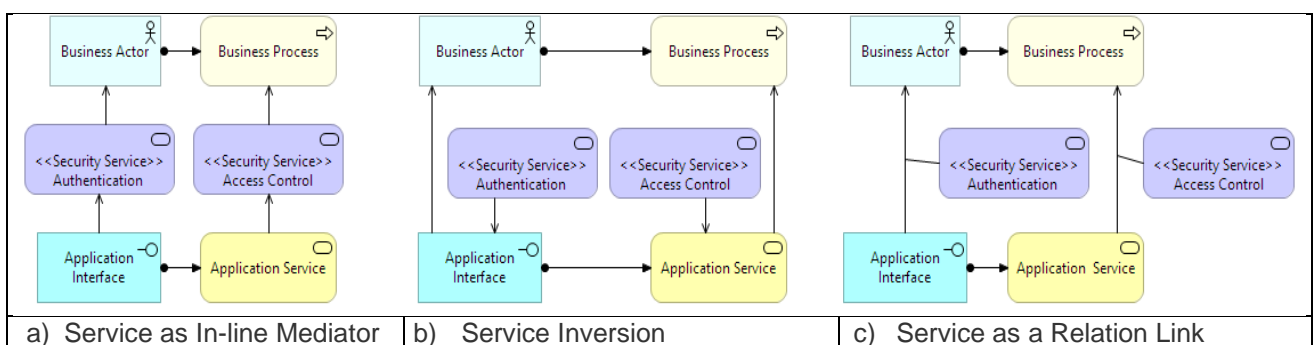


Figure 46: Security Services in the Conceptual Layer

There are two important things to note:

- Conceptual services are Singletons. The idea of an Authentication Service, for example, is unique, no matter how many ways it might be implemented;
- A convention on service placement: Access control mediates access to services; Actors authenticate to interfaces.

In Figure 46 (a), conceptual services are inserted in-line between business and application layer elements. Visually, the role of mediation is clear. However, shared services (e.g. Single Sign-on) become a Singularity: a hub in the underlying model, linking all Actors and Interfaces, all Processes to all Services and losing the analytical ability to determine which Business nodes use which Applications.

Figure 46 (b) restores the direct serving relationships from application to business layer but loses the sense of security services acting as mediator. The pattern also runs counter to an architectural design principle that services offered should not be invoked from a lower layer. The pattern is not particularly accurate either: authentication, in particular, is often intercepted by application containers, i.e. at the technology level, rather than invoked explicitly by the applications themselves.

Figure 46 (c) provides a pattern that resolves these issues: the direct link between business process and application is maintained, the security service is visually represented as guarding the service call, and there is no inversion of the directionality of service invocation between layers. It also retains the flexibility to show where the service implementations are deployed invoked in the application or technology layers.

At a practical level, the Security Overlay proposes Security Services as Singleton elements, created as stereotypes of Application Services, but rendered the same visual style as Motivation elements.

7.4 Identity and Trust

The “People” cell of the SABSA Conceptual Layer considers how business entities (Organisational Actors and Roles) can be mapped to their Logical counterparts (Accounts and Application Roles), an important topic, given the emphasis placed on Logical Access Management in security governance.

Similarly, the trust that exists between real-world actors needs to be made explicit in our models if it is to be adequately protected. The concepts introduced in this Section will explain how this can be achieved.

7.4.1 Identity and Access Rights

Logical Access Management (LAM) is a Logical layer process that provides Contextual entities (Actors in Business Roles) with suitably provisioned accounts, principally to access applications and technology. A few Conceptual elements (Principal, Access Rights, Credentials) are required to establish this mapping from Contextual to Logical elements. Currently, none of the concepts mentioned, nor any representation of Accounts and Application Roles, exist in ArchiMate.

What the Security Overlay would like to express is shown in Figure 47(a). Business Actors are represented by one or more Principals in the Conceptual layer²⁸. Each Principal is issued with Credentials and granted a set of organisation-wide access rights (Authorisations), commensurate with the scope and nature of their Role.

Note that not all Principals are derived from Business Actors. The concept also represents machines: robots, batch processes, applications, servers, or logical entities that participate in authenticated transactions.

At the Logical Layer, Credentials are verified by an Authentication Service to bind Principals to Accounts. The LAM system will have pre-provisioned each account with sets of application-specific access rights that map to Application Roles defined in and recognised by Application Components and System Software.

This pattern forms part of an ‘Administration time’ process (secondary architecture) and is supported by Management Matrix services such as identification, enrolment, authorisation, provisioning and credential management. Credentials and Application Role Claims are presented to Authentication and Access Control.

²⁸ This construct is often used to ensure that Single Sign-On does not enable an Actor to move seamlessly between Production and non-Production environments, or that an individual who is both an employee and a customer of an organisation is issued with accounts that are logically isolated.

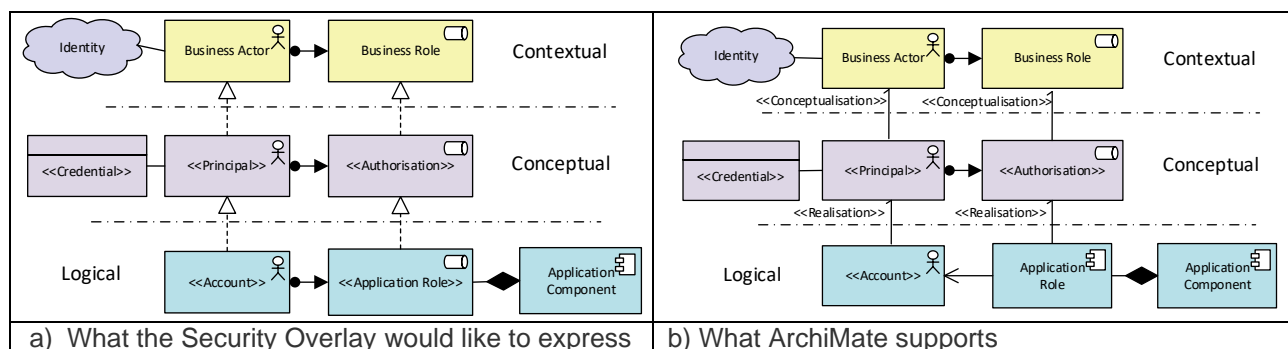


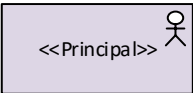
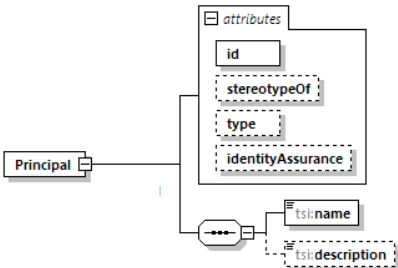
Figure 47: Conceptual Entity Metamodel

As ArchiMate lacks native elements to represent these security concepts, the Security Overlay uses stereotypes of Business layer elements. Ideally, the Security Overlay would have kept the precedent of using the **realisation** of similar entities across layers to achieve a consistent structure from Business Actor (*Contextual*), via Principal (*Conceptual*) to Account (*Logical*) and from a Business Role (*Contextual*), via Authorisation (*Conceptual*) to an Application Role (*Logical*). It would also have been intuitive to use composition from Application Component to Application Role as the definition and implementation of role-based access is coded into software.

Unfortunately, these cases extend ArchiMate in ways for which it was not designed: **realisation** between elements of the same type and Role **composition** into Application are not legal. The closest approximation supported by the ArchiMate 3.1 specification is shown in Figure 47(b), with a new *Conceptualisation* relationship used to derive Conceptual layer elements from Business ones and an improvised *Realisation* to derive Logical layer entities from Conceptual ones. Application Roles can be modelled as components of the Application by modelling them as Components themselves. This downside can be offset if your tool supports custom iconography. The significant upside of this is the ease of assigning Application Roles to Application Functions.

Table 31 summarise the new Conceptual layer elements. Logical layer elements are addressed in Section 8.

Table 31: Elements used in Logical Access Management

Element	Properties	Schema
	<p>Principal is the conceptual representation of an Entity capable of holding an Account in the target system, including TECHNICAL entities.</p> <p>Principals are created through an identification process with a given level of identityAssurance.</p>	

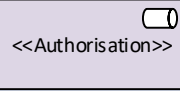
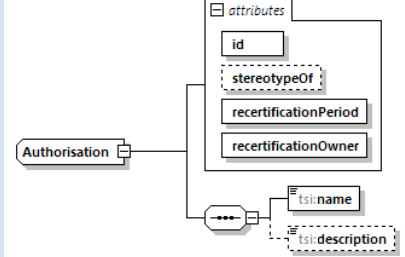
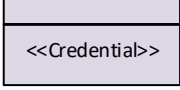
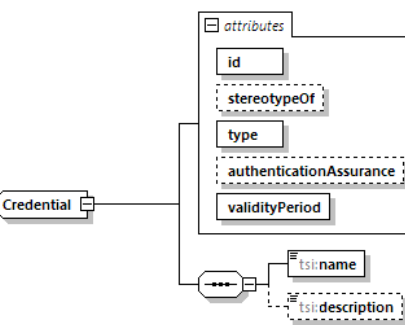
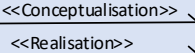
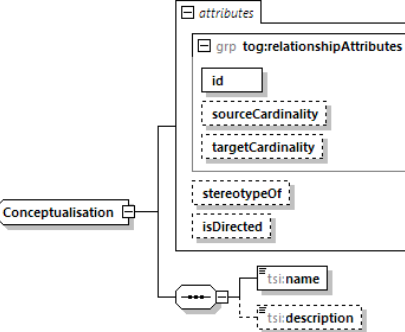
	<p>Authorisation is a conceptual representation of actions that a Principal is authorised to perform in a Business Role.</p> <p>As it decouples Business Roles from Application Roles, it models the type of mappings used in central access management systems, e.g. Active Directory Groups. Its properties capture the period and responsibility for the recertification cycle.</p>	
	<p>Credential is the conceptual representation of a proof of identity that enables a Principal to authenticate to a given application or authentication service.</p> <p>The type property is used to enumerate common types of credential: PIN, PASSWORD, SOFTTOKEN, BIOMETRIC etc.</p> <p>The authenticationAssurance property denotes the associated authentication strength – taking into account the technology, processes of generation, distribution etc, expiry etc.</p> <p>The validityPeriod property is used to express the credential's expiry lifetime.</p>	

Table 32: Conceptual Layer Relationships

Relationship	Properties	Schema
	<p>Conceptualisation is a stereotype of Association, used to derive Conceptual elements from Business elements.</p> <p>Realisation, also expressed as a stereotype of Association, is a pure workaround for the lack of support for native realisation between elements of these types.</p> <p>For both, isDirected is fixed to TRUE and oriented towards the element in the higher architectural layer.</p>	

Linking these elements to the Logical layer, Account represents an authenticated Principal on the target system from which access to a restricted set of resources (e.g. the individual's own email) can be controlled. Application Role represents roles defined in the application code to support, for example, role-based access control (RBAC) mechanisms. These elements will be discussed further in the next section.

By modelling this structure, the following examples of security validation become possible:

- Transparency in onboarding: the ability to detect a single individual enrol in the system under multiple Principals, thereby potentially by-passing segregation of duty controls;
- To assure that human Principals are not assigned machine accounts and cannot access interfaces and services intended for machine accounts (and vice-versa);
- Highlighting areas of risk caused by sensitive functions being accessed by an Active Directory group that has no identified Owner or has not been recertified within the period required by policy;
- Every instance of a Credential appearing outside the LAM domain (or the domain of its creation) should trigger requirements for secure distribution, storage, access control etc.

7.4.2 Roles and Responsibilities

The SABSA Conceptual People cell addresses the concepts of role and responsibility. At the business layer, roles are defined in terms of mediating an Actor's engagement in behaviour (e.g. processes). The task of the Conceptual model is not just to map these role assignments from the physical to the virtual realm but to do so in a way that is manageable, transparent, efficient, maintainable, scalable and auditable. This section establishes this conceptual model.

Figure 48 shows how the concepts established earlier can be used to illustrate the relationship between business layer entities (An employee engaging in Business Process Y un the Role X). To perform Process Y, the employee needs to perform three functions (A, B and C) using Application Z.

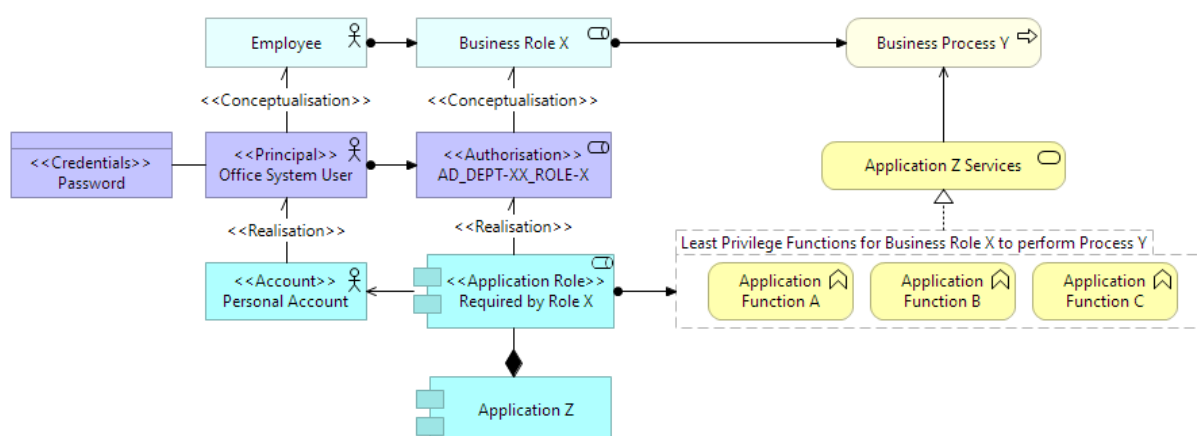


Figure 48: Modelling Identity & Role Concepts

In the Conceptual layer, the employee is instantiated as a Principal with an Office System User profile, assigned to an Authorisation profile appropriate for staff of Department XX, in the Business Role X.

This Authorisation profile (also known as a Group Profile) is a wide-ranging set of authorisations, defining the employee's full access rights to the office network domain, multiple applications, infrastructure such as remote log-in, badge-entry systems, etc. The employee is also issued with a credential – a conceptual proof of authentication – that enables the person to claim the digital identity enshrined in the Principal.

When that authenticated Principal is presented to a given target application, such as Application Z, the application binds the Principal, by its identifier, with a local account that has been assigned a specific Application Role, whose profile includes access to Functions A, B and C.

Key to this layered model is the recognition that an Application knows nothing of its external context. It can differentiate between different accounts, knows which accounts have been assigned to which local roles, and has been programmed to allow certain roles to access certain functionality. That's all.

At the Conceptual layer, the User Directory manages the enrollment of employees as Principals, allocated as members of various Groups. It also maintains a mapping from each Group to the Application-Roles that Group Members must perform in their business role. It sees nothing of the functional capabilities that lie behind those Application-Role titles.

7.4.3 Trust

The final aspect of the Conceptual People cell is that of the trust that exists between Business Actors. Trust is essential to business transactions yet rarely made explicit in conventional Architectural Descriptions.

Trust causes business partners to accept risks that would otherwise be declined, based on intangibles such as reputation, personal history or the expectation of a long-term relationship. It is essentially a human quality that, though it cannot be reproduced outside the context of the business relationship, must be protected as soon as any part of that transaction is delegated to technology (e.g. an Agent, a process or an IT system).

The analysis of signals crossing a security domain boundary provides a means to identify any implicit trust in the context of the interaction. Trust must be represented explicitly in the model to derive the necessary protection requirements. Examples of inter-domain signals are shown in Figure 49.

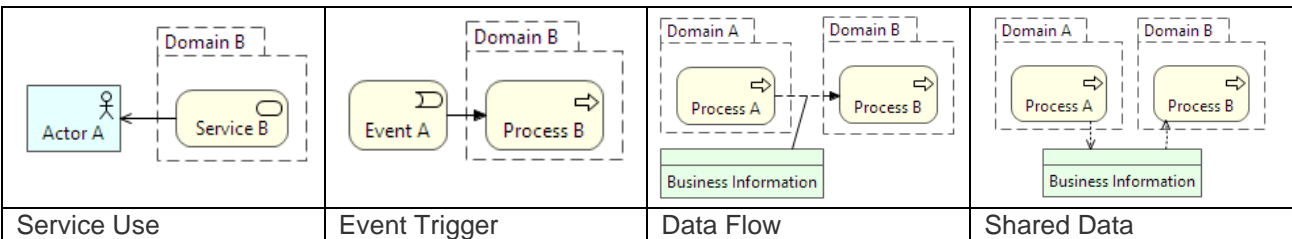


Figure 49: Common examples of Signals crossing Domain Boundaries.

While active structure and behaviour elements should be placed in the domain where an activity occurs (both visually and via structural relationships), the modelling guidance for passive structure²⁹ elements is more nuanced. Information is generally copied and distributed in the course of processing and comes to exist in multiple domains.

A modeller may opt to assign an Object to all domains in which it is used or “free” it by not assigning it at all. The choice will often be determined by the sensitivity of the information: objects subject to regulation, for example, should be assigned to a security domain if access from outside that domain is a concern.

Another characteristic of a valid trust model is that, while any element may be a “trusted” entity, only an active structure (or behaviour assigned to an active structure) can be a ‘trusting’ entity. In other words, information may be the object of a trust relationship (be it trusted or untrusted) but is incapable of offering trust itself.

To model trust in ArchiMate, a stereotyped flow relationship indicating that “A trusts B” seems most intuitive³⁰. It can also convey a direction of trust that is independent of that of the underlying signal or show opposing flows to indicate bi-directional trust.

Like SABSA Business Attributes, Trust is modelled as a stereotype of Principle that, like Attributes, should only be *influenced* in the model through the realisation of Control Objectives. A simple example of trust concepts expressed using ArchiMate in this way is shown in Figure 50.

²⁹ except Artefacts and Materials (in Technical & Physical layers) which represent physical objects.

³⁰ While deliberately avoiding the question of what actually flows from one entity to the other.

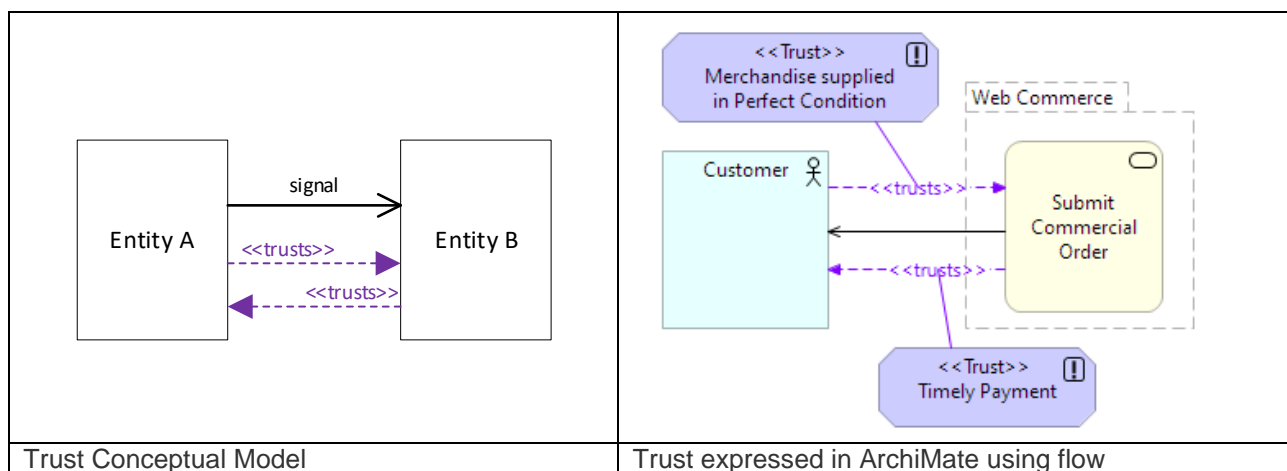


Figure 50: Simple Trust Relationships using Flow

This approach offers simplicity but, despite being a good first attempt, it has a couple of issues:

Firstly, the trust relationships are not linked to the underlying signal. Whenever more than one signal is drawn between a pair of entities, analysis cannot contextualise which trust belongs with which signal³¹.

Second, the ArchiMate grammar prevents flow relationships from connecting to a passive structure. Trust in an Object, as in “I trust every email that comes from my bank”, cannot be expressed using this approach.

To overcome these limitations, the Security Overlay offers an alternative representation using Trust stereotypes alone. Figure 51 illustrates how this is used to apply trust expressions to the patterns introduced in Figure 49.

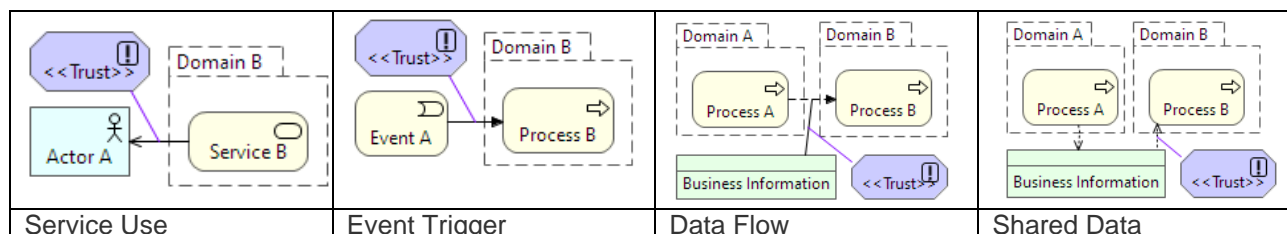
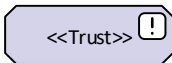
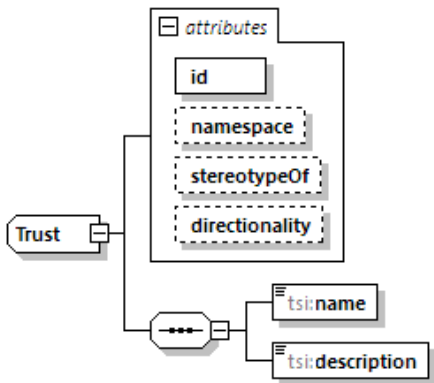

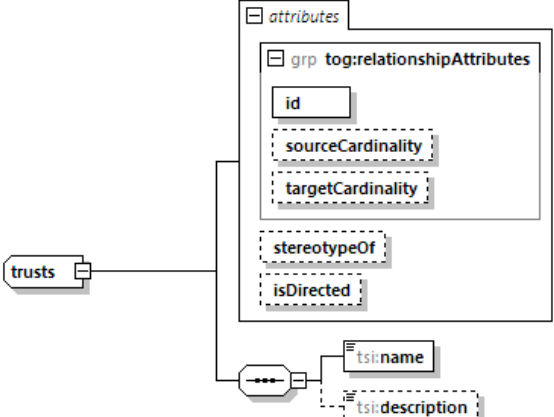


Figure 51: Trust Attributes associated with inter-Domain Signals.

This alternative representation lacks a visual representation of the direction of trust, which is not necessarily the same as that of the signal, that can be partly offset by good naming of the Trust element and elaborated by its textual description. For automated analysis, the Overlay provides a **directionality** property that sets the direction of the trust relative to the associated signal.

³¹ Although associations can be shown visually via selective diagrams ArchiMateSE aims to enable automated analysis.

Table 33: Elements & Relationships used in Trust Modelling

Element	Properties	Schema
	<p>Trust is modelled as a stereotype of the Principle element that can be associated with any relationship to make any trust underpinning the transaction explicit.</p> <p>A directionality property defaulted to FORWARD declares the direction of the trust.</p>	
Relationship		
 <p>A trusts B</p>	<p>Simple trust between entities that arise from the general context rather than a specific transaction is modelled by a stereotyped data flow: the arrow indicating the direction of trust and its nature elaborated using a Trust element.</p>	

The use of Security Domains and Trust Models support the following validations:

- Interactions that cross security domain boundaries are identifiable as prime candidates for trust analysis. Traditional architectural descriptions focussed primarily on technical aspects overlook the significance that trust plays in the business's risk appetite. Making this trust explicit in the model is a critical success factor for establishing a complete set of system protection requirements.
- A model declaring a serving relationship that crosses a trust boundary should also have a data or control flow (trigger) defined to be considered complete. This rule will identify any sensitive data (confidentiality, integrity, authenticity etc.) crossing the boundary in the context of that service.
- Many patterns, such as access to sensitive or private data from outside the domain, can be matched as a policy violation if the Trust element is not protected through Control Objectives realised by Controls.

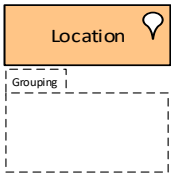
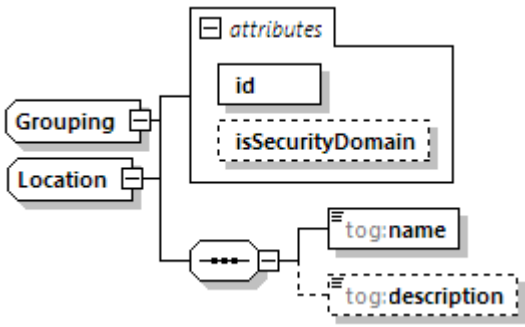
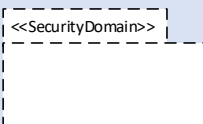
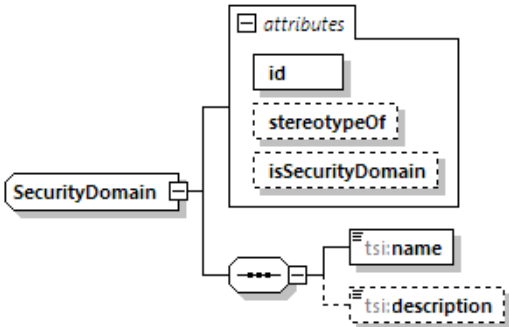
7.5 Domain Framework Model

The Conceptual Domain Framework denotes security domains in the System of Interest. The Security Overlay models them using a <<SecurityDomain>> stereotype of Grouping. These domains are an essential basis for creating Trust Models: the analysis of security concerns that arise from signals (service calls, system accesses, event triggers and data flows) that are enacted across domain boundaries.

Where a SecurityDomain perfectly aligns with an existing bounded space (physical, logical, organisational, jurisdictional, network zone) defined by **Grouping** or **Location**, the Security Overlay offers an **isSecurityDomain** property, set FALSE by default, as a means of marking them as security domains without duplication.

The relevant ArchiMate elements for domain modelling are shown in Table 34. ArchiMate's syntax allows elements to be placed into Groupings using *composition*, *aggregation* or *assignment*. Composition is generally too strong due to its implied exclusivity: in practice, elements are often subject to several domains simultaneously. *Aggregation* is usually the best choice for Grouping domains, *assignment* for Location.

Table 34: Security Domain Mapping

Element	Properties	Schema
	Any Location or Grouping element can be designated as a Security Domain by setting its isSecurityDomain property (FALSE by default) to TRUE.	
	SecurityDomain is a stereotype of grouping that makes the security domain property visually explicit. The isSecurityDomain property is fixed to TRUE.	

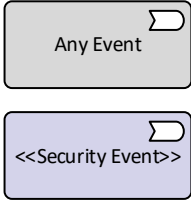
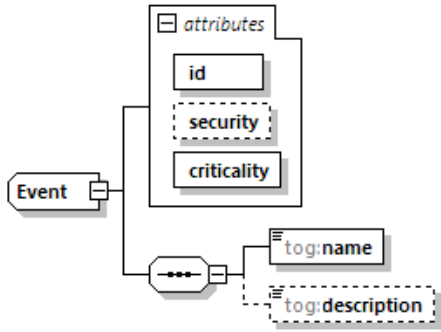
By assigning elements to security domains, analysis can determine whether a signal traverses a domain boundary: a condition that prompts the need for Trust Modelling.

7.6 Security Events

Security events are expressed using a similar approach to that outlined above.

The Security Overlay defines a security flag property for all Event elements, set FALSE by default. Marking an existing event as a security event is just a matter of setting this flag TRUE. A <<SecurityEvent>> stereotype can be created from any Business, Logical, Technology, or Implementation Event, as appropriate, to introduce a new (purely conceptual) security event, the.

Table 35: Security Events

Element	Properties	Schema
	<p>Security-significant conditions occurring in the system context, triggering or triggered by active structure and behaviour elements</p> <p>Modelled by a stereotyped Security Event or by annotating an existing regular Event by setting the security property TRUE</p> <p>The criticality property indicates the significance of the event.</p>	

8 Modelling the Logical Security Architecture

The Logical Architecture documents the Solution Architect's view of the system. In this section, conceptual elements described in the previous section are applied to and guide the Application and Data architecture.

It also proposes security properties for ArchiMate's logical layer elements that support security analysis.

Table 36: SABSA Logical Architecture

		Assets (What)	Motivation (Why)	Process (How)	People (Who)	Location (Where)	Time (When)
Logical	Artefacts	Information Assets	Risk Management Policies	Process Maps & Services	Trust Relationships	Domain Maps	Calendar & Timetable
		Information Asset Register Business Information Model	Risk Models; Domain Policies; Assurance Criteria & Framework	Information Flows; Functional Transformations; Service-Oriented Architecture; Services Catalogue; Application Functionality & Services	Domain Authorities; Entity Schema; Privilege Profiles; Trust Relationship Models	Domain Definition; Inter-Domain Associations & Interactions	Start Times, Lifetimes & Deadlines
	Activities	Logical Asset Management	Policy Management	Delivery Management	Enterprise-wide User Management	Service Catalogue Management	Evaluation Management
		Knowledge Management; Release & Deployment Management	Risk Modelling; Management of Policy Development & Maintenance; Policy Publication Compliance Management	SLA Management; Supply Chain Management; BCM; Financial Management; Transition Management	Trust Modelling; Identity & Access Management; Account Administration & Provisioning	Configuration Management; Capacity Planning; Availability Management	Monitoring & Reporting KPIs & KRIs

This outline enables us to visualise the set of artefacts and activities of the Logical Architecture, expressed in ArchiMate in Figure 52

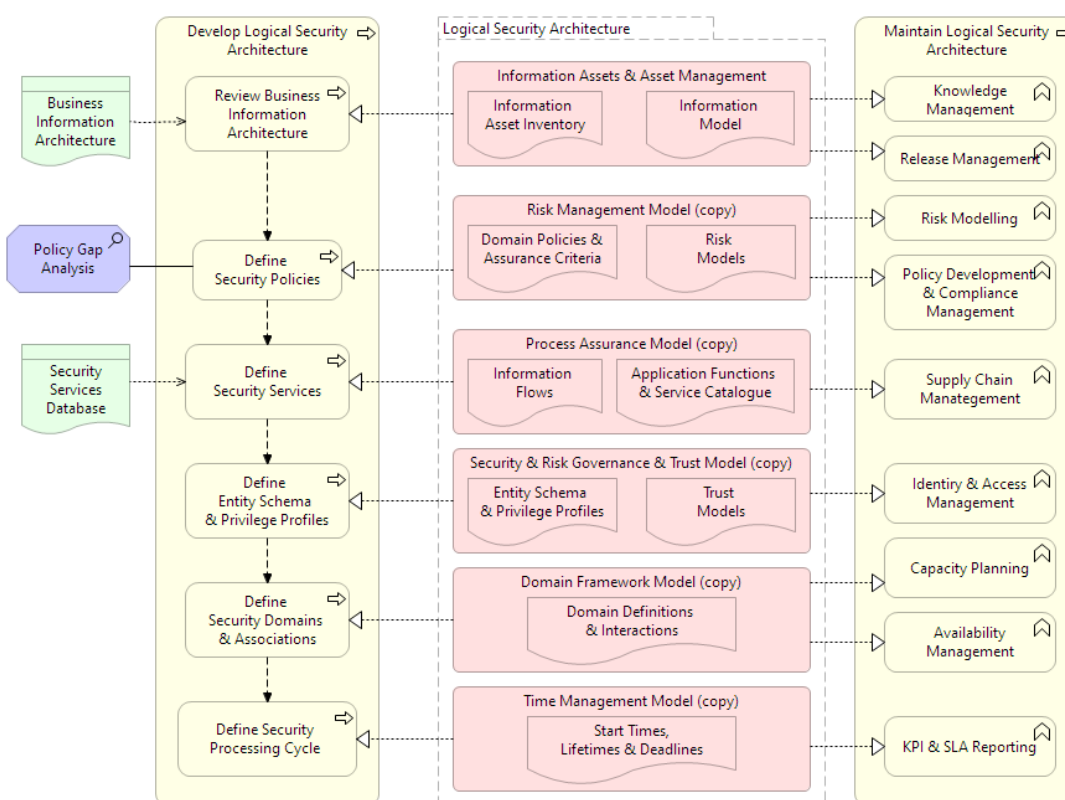


Figure 52: Developing the SABSA Logical Security Architecture

8.1 Information Assets

The principal information assets in this layer are software assets (applications) and data.

8.1.1 Application Components

Most organisations maintain a register of the applications deployed in their IT environment. The primary drivers for this register are operational: patch management, vendor support, licence management etc. Because these concerns are focussed on software products rather than logical building blocks, discussion is deferred to Section 9: Modelling the Physical Security Architecture.

The kind of inventory relevant at this layer is the identification of *critical* applications: the software upon which the organisation is disproportionately reliant for achieving its primary mission. Organisations perform a regular 'Critical Application Review' to identify these dependencies. Applications on the critical list are then prioritised for risk assessment, penetration testing, monitoring, business continuity planning, audit etc.

But how should critical applications be identified? The most common approach is simply to request a rating recertification from the Application Owner. This approach has a few issues:

- In a service-oriented architecture, the use of application services can be highly dynamic. Even if the Application Owner is aware of who is using the application (a big ask), they may not know for what – neither at the time that access was granted, how it is currently used, or will be used in the future;
- A tendency for subjective bias. Applications are disproportionately important to their owners. The application's availability, for example, might be a factor in the measurement of their performance;

A better solution can be found using architectural models, as shown in Figure 53.

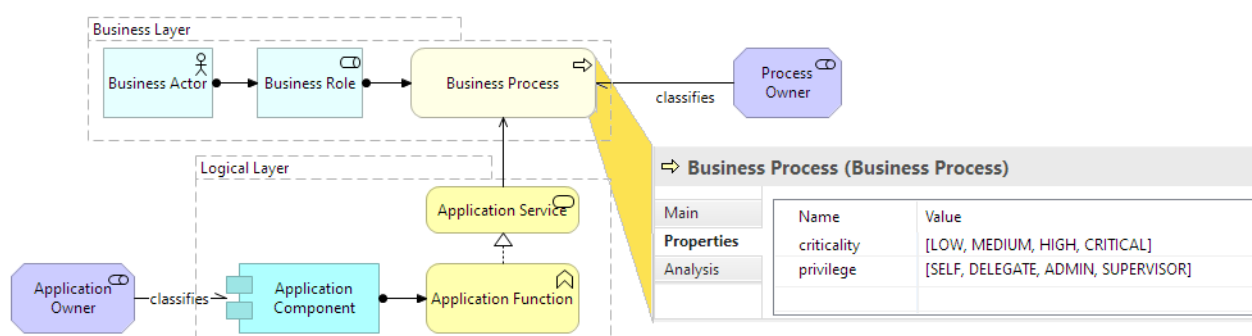



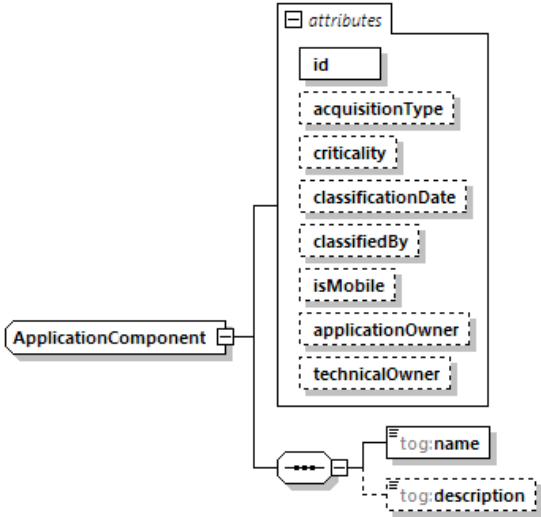
Figure 53: Critical Applications

A better definition of a critical application is 'one that supports mission-critical capabilities and value streams'. Analysis performed at the contextual level has enabled the model to trace this value to business processes. The Security Overlay provides a **criticality** property to identify critical business behaviour elements.

Figure 53 can be analysed to identify Application Components that provide functionality to Business Processes rated 'CRITICAL'. All such Applications are candidates for the critical list. Their candidature can be confirmed by inspecting the **criticality** property of the *serving* relationship.

The criticality of an application can be determined as the highest common denominator of critical process-centric analysis. Any Application that provides 'HIGH' or 'CRITICAL' services to a 'CRITICAL' process should rank highly in any Critical Application Register.

Table 37: Application Component Properties

Element	Properties	Schema
	<p>acquisitionType identifies the means of acquisition: [UNCLASSIFIED, CUSTOM, OPENSOURCE, COTS, GOTS]. Its value can indicate intent at development time or derived from an Artefact post-deployment;</p> <p>The properties criticality, classificationDate and classifiedBy capture the certification lifecycle;</p> <p>isMobile is a flag that identifies mobile code, defaulting to FALSE;</p> <p>applicationOwner is responsible for authorising access and performing periodic recertification;</p> <p>technicalOwner is responsible for defining a secure configuration;</p>	

8.1.2 Security Configuration

Security Configuration is a stereotype of Data Object that declares any configuration settings that differ from the baseline (least common functionality) configuration. It is appropriate at the logical layer because different instances of the same building block may have different configurations in different contexts. Properties match those of the baseline configuration but with values toggled between ENABLED and DISABLED.

8.1.3 Software Defects and Malware

As Defects are associated with specific software releases, discussion is deferred to the Technical Layer. Nevertheless, being modelled as a <<Vulnerability>> (Motivation layer elements can be placed anywhere), it is often useful to associate functional defects with application behaviour elements directly.

The Security Overlay includes a Malware stereotype of Application Component that can represent non-human agents or malware threat. It provides a type property that enumerates a controlled vocabulary of threat types with a default set of values drawn from the STIX³² taxonomy.

8.1.4 Data Assets

A means of tracking critical data assets is a prerequisite for their effective protection. When those assets are subject to regulation, as is the case with personal or medical data, it is also mandated by compliance.

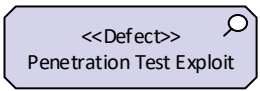
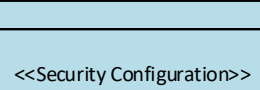

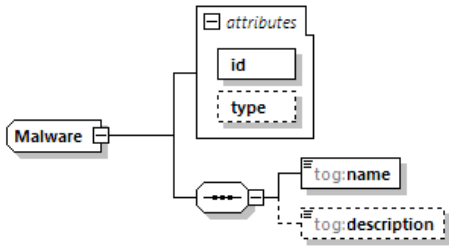
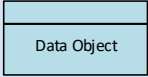
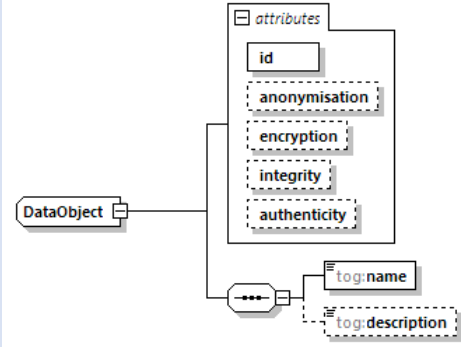
³² Structured Threat Information Expression (STIX™) is a language used to express cyber threat intelligence.

As with applications, the criticality of data stems from the criticality of information in the contextual layer. As seen in Section 6.1.2, the Security Overlay enables Business Objects to be annotated with properties that convey their security classifications (confidentiality, privacy, integrity, authenticity and retention period), enabling all logical representations of that information to inherit these protective markings.

Modelling provides a means to bind data objects to their business significance and show its provenance (where it comes from), how it is processed (by which application functions having what read/write/modify capability), where it goes (flow relationships, sometimes traversing domain boundaries), and who has access to it (the business actors and roles to whom it is served). By tracing down to the Technology layer, as will be shown in Section 9.1.1.2, we can also determine its storage location.

The security properties that enable a Data Object to contribute to the protection of information include the ability to implement data-level controls for confidentiality, privacy and integrity, using various techniques of encryption, anonymisation (tokenisation, redaction,) and integrity protection (checksum, HMAC, signature).

Table 38: Data Object Properties

Element	Properties	Schema
	Vulnerability elements represent defects published in CVE lists or discovered in the software development lifecycle.	Discussed in Technology Layer
	Security Configuration declares functionality that has been enabled in addition to the baseline (least common functionality) configuration. Properties match those of the baseline configuration but with values toggled between <code>ENABLED</code> and <code>DISABLED</code> .	An open schema in which to declare the security configuration of the application as deployed in a specific application context.
	Malware is used to model the threat of malicious code. <ul style="list-style-type: none"> The type property is based on an open vocabulary that includes values such as: ADWARE, BOT, DOWNLOADER, KEYLOGGER, RANSOMWARE, ROOTKIT, SPYWARE, TROJAN, VIRUS, and WORM. 	
	Object properties declare controls applied at data level: anonymisation for privacy protection: <code>[NONE, MASK, REDACT, TOKEN, HASH, OBSFUCATE]</code> ; encryption for confidentiality protection: <code>[NONE, 3DES, AES, AES256, RSA, ECC]</code> ; integrity for tamper detection: <code>[NONE, CHECKSUM, MD5, HMAC, SHA1, SHA2, SIGNATURE]</code> ; and authenticity : <code>[NONE, SELF, ASSERTION, SHARED_SECRET, SIGNED]</code> .	

8.2 Risk Modelling

8.2.1 Risk Modelling

Sections 4.2, 5.5 and 7.2 described the Security Overlay's risk elements and risk management strategy without being prescriptive about any particular methodology that an organisation may choose to adopt.

These high-level definitions can be adapted to suit an organisation's chosen risk methodology. To illustrate how the Logical layer artefacts of SABSA's Motivation apply these concepts, this section models the [OpenFAIR Risk Analysis Example Guide](#), which contains a qualitative and quantitative analysis.

8.2.2 The Scenario

The example scenario considers a situation in which an HR Executive commits the security sin of posting their log-in credentials on a sticky note attached to their workstation. The Executive's office is accessible to 5 groups of people, each with differing motivations, capabilities and competencies, as shown in Figure 54.

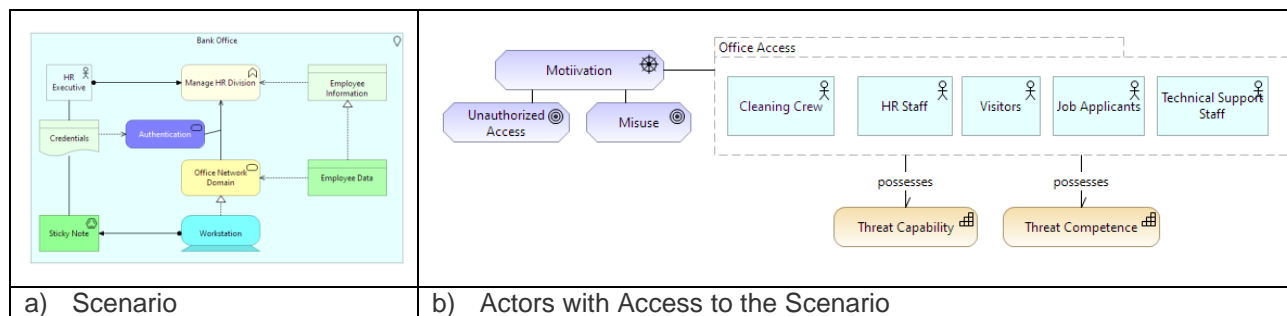


Figure 54: OpenFAIR Example Scenario

The analysis described in the guide focuses on the risk posed by one of these groups: the Cleaning Crew.

The OpenFAIR methodology uses a factored analysis technique to assess this risk, modelled in Figure 55. The assessment of any factor in the analysis tree is determined by a qualitative (risk matrix) or quantitative (mathematical) combination of its constituent factors.

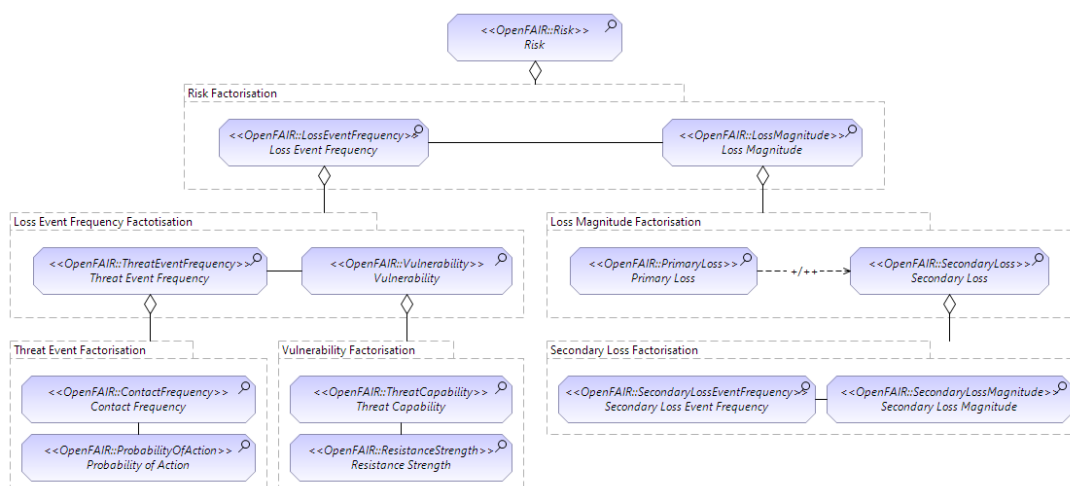


Figure 55: Factored Risk Analysis in OpenFAIR

The qualitative analysis of the example scenario is shown in Figure 56, the quantitative in Figure 57.

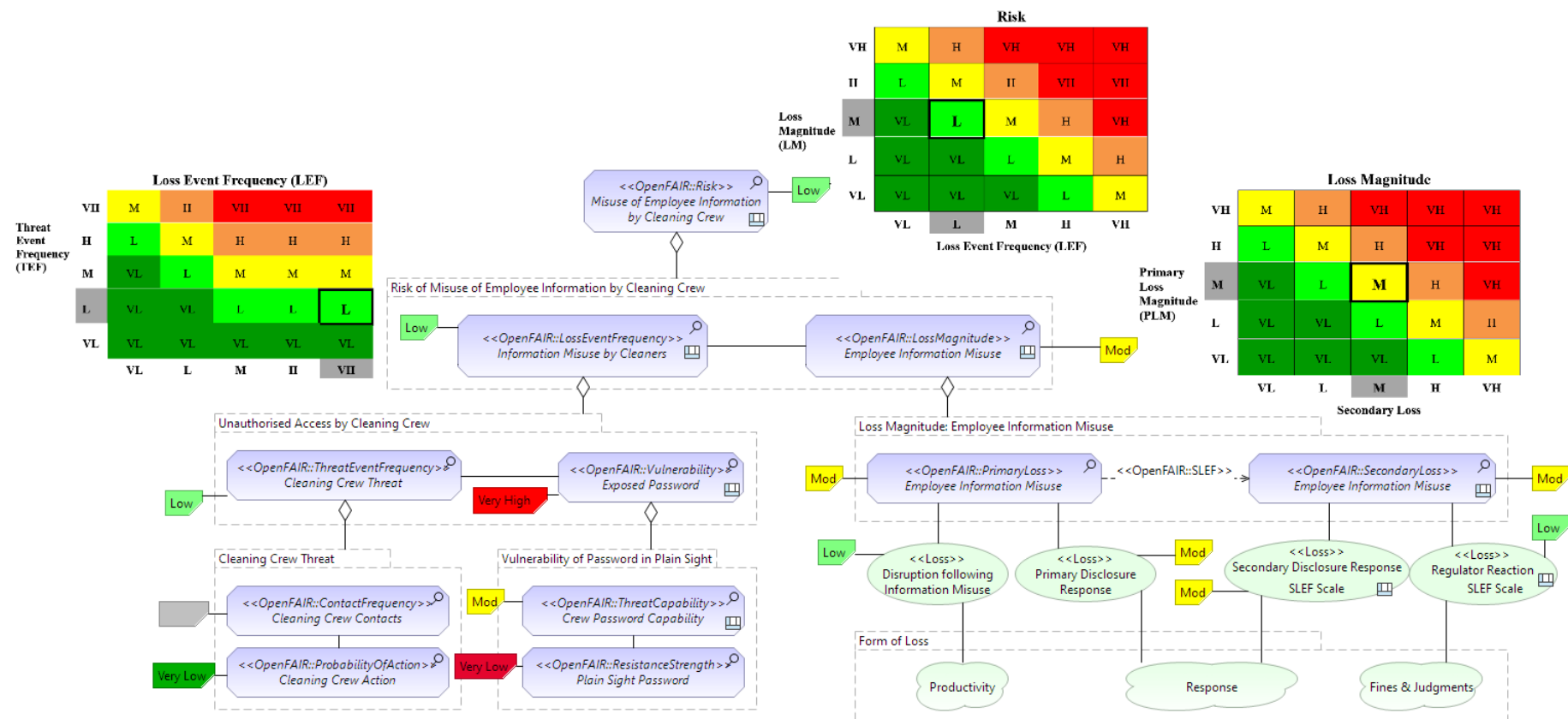


Figure 56: OpenFAIR Example - Qualitative Analysis

Figure 56 shows risk matrices combining factors using qualitative scales using Security Overlay risk elements, specialised and extended for OpenFAIR.

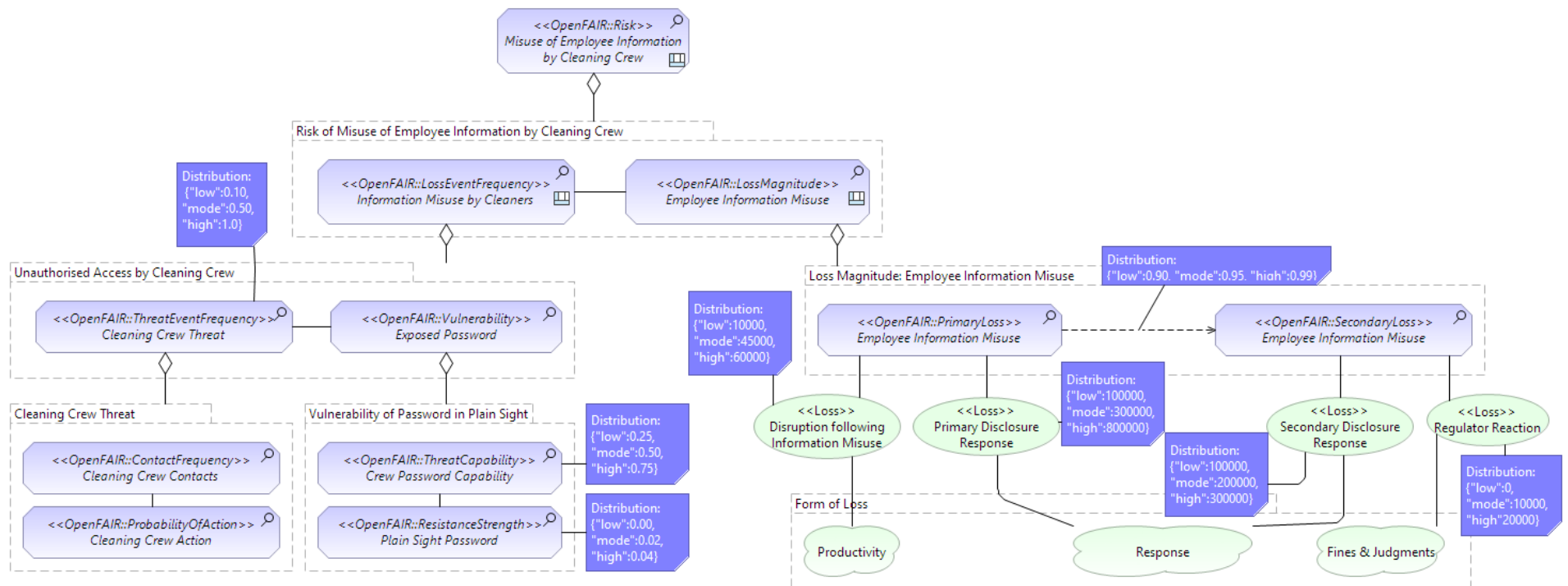


Figure 57: OpenFAIR Example - Quantitative Analysis

Figure 57 shows the analysis of the same scenario using quantitative methods in which estimates are based on numerical ranges in the form of a 90% confidence interval (low, high) and the estimated most likely value (mode).

Of course, the ability to define qualitative or quantitative calculations is beyond the expressive capability of the ArchiMate notation. Still, many tools offer scripting or programming support that could implement model-driven risk analysis based on input parameters structured in this way.

8.3 Application Functionality and Services

The Process cell of the SABSA Matrix Logical architecture addresses application functionality and services. Services are key to both availability and access control. The focus of this section is placed on availability. Access control will be discussed in the next section, the 'People' cell, which continues the discussion of the design and implementation of Logical Access Management.

The previous sections enabled the model to trace the value of assets and mitigate risks from the information specified in the Contextual layer. The model also provides a means of ensuring that logical layer services can provide the necessary availability of and delivery to committed SLAs.

In terms of what we might call 'supply-side availability', the Security Overlay is based on a simple premise that the most important characteristic of internal behaviour elements (processes, functions and interactions) is their contribution to fulfilling the business mission. As discussed earlier, criticality can be determined through traceability to Value Streams or by an enumerated criticality property with qualitative range of values.

For services, it is their availability: human-oriented services being presented as having a service window; technical services by an enumerated availability ranges (e.g. HIGH = 99.99 - 99.999%), supported by recovery targets, the recovery time (RTO), and recovery point (RPO) objectives.

On the 'demand side', availability is also affected by the resources available to handle the rate of incoming requests. Recall the Security Overlay's annotation of properties for Actor population size, Business Service execution frequency, SLA reaction and response times, the cardinality on servicing relationships.

When traced from the Business layer to the Application layer, these factors provide information that can help determine whether Application functions and services are capable of meeting peak demand within the terms of the SLA. Diagrams like the example of Figure 58 might be used to identify critical business services reliant on applications not operated in high availability configurations.

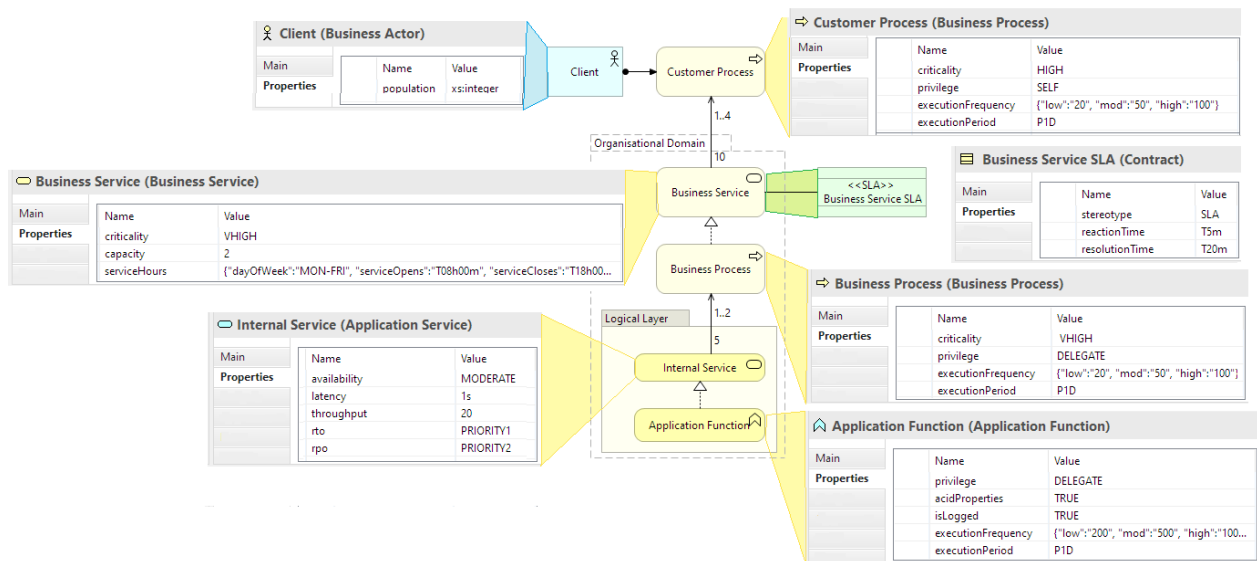

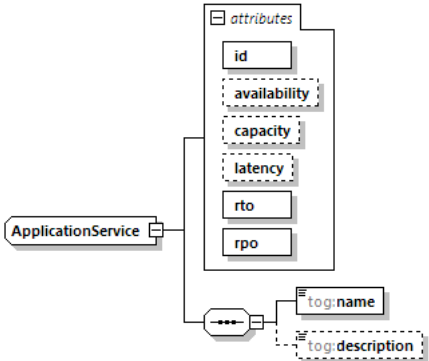
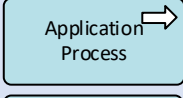

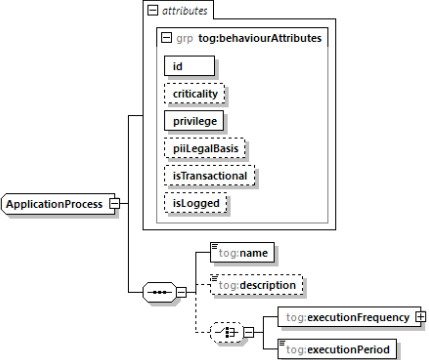


Figure 58: Application Services

The properties of the Application Behaviour elements are shown in Table 39. Because service elements play a key role in both availability and access control, only the availability properties are explained in this section.

Table 39: Application Behaviour Availability Properties

Element	Properties	Schema
	<p>Application Services offer functionality and availability. The availability properties include:</p> <ul style="list-style-type: none"> an availability requirement: enumerated ranges based on % up-time: [NA, UNCLASSIFIED, VLOW, LOW, MODERATE, HIGH, VHIGH]; capacity indicates the maximum supported limit for in-flight service requests; a latency target for maximum acceptable response time; Recovery Time & Point Objectives (rto and rpo) for recovery targets following any outage. 	
 	<p>Application behaviour element properties most relevant to availability are:</p> <ul style="list-style-type: none"> criticality captures the highest common denominator of business activity supported: [NA, UNCLASSIFIED, VLOW, LOW, MODERATE, HIGH, VHIGH]; isTransactional flags whether the activity exhibits ACID qualities; isLogged flags whether the activity is logged. executionFrequency and executionPeriod are used to express, as a confidence range, how often the process is performed. <p>Remaining properties are discussed in Section 8.4.4.</p>	

8.4 Logical Access Management

A key security issue for Application Services is the ability to enforce access control, predicated in turn on the Service's ability to establish a minimum level of assurance of the digital identity presented by the user. On the premise that on the Internet, “*nobody knows you’re a dog*”³³, the Security Overlay must be able to express that while “*being a dog*” might not be a concern for some low-risk services, others will demand a degree of confidence that the Principal accessing the service is a legitimate proxy of the identified, real-world Actor.

To model this, the Security Overlay proposes properties that reflect the 3-level assurance criteria identified in NIST’s Digital Identity Guidelines [Ref. 8]: The default schema proposes as a 3-tier classification, *STANDARD*, *ENHANCED*, *ASSURED*, that maps easily to the assurance levels described in NIST, but this can be adapted to other schemes such as ISO/IEC 29115’s 4-level scheme or a proprietary scheme (e.g. numeric range), defined by the implementing organisation.

Access control is increasingly implemented via external services (policy administration, information and decision points), but enforcement is predominantly local and predicated on the Principal having authenticated to an account that has been provisioned with a set of access rights.

Credentials generally materialise in the logical level as some form of data (e.g. a passcode, dactylogram or retina scan) that must be compared against the credential on record. These may or may not be associated with the target system account, depending on whether authentication is performed locally or represented by a trusted token from a centralised Authentication Service, as shown in Figure 59Figure 48.

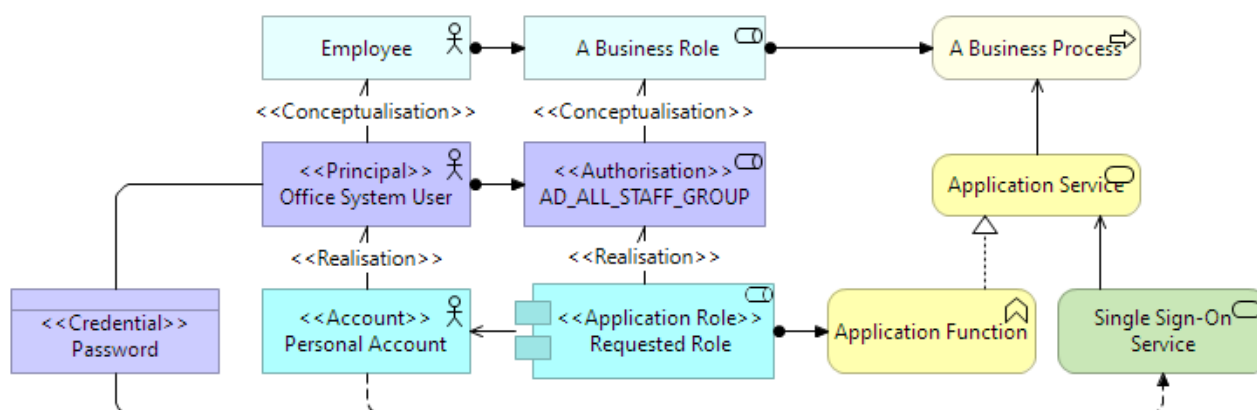


Figure 59: Modelling Authentication in the Primary Architecture

In the example of Figure 59, a Service Provider initiated Single Sign-on Authentication has been modelled as an Infrastructure service because that is how it looks from the Application’s perspective: the paradigm of the externally-visible service redirecting an unauthenticated session to an authentication service.

The foremost reference on practical ArchiMate [Ref. 7] advocates the structuring of models in 3 architectural planes: a primary architecture (describing run-time use), a secondary architecture (system development,

³³ Peter Steiner: The New Yorker, July 5, 1993

administration and operations) and a tertiary architecture (ownership and governance). While authentication is a run-time process in the primary architecture, the service is best modelled as part of a LAM capability in the secondary architecture, as shown in Figure 60.

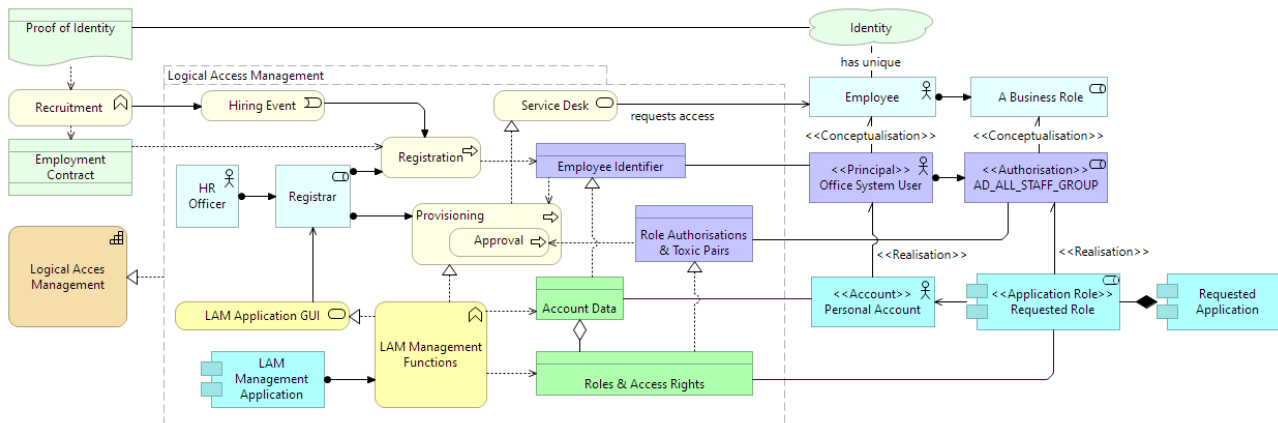


Figure 60: Example of Registration & Provisioning in the Secondary Architecture

8.4.1 Account

Account is the logical layer representation of a Principal from the perspective of an application or target system. A Principal may hold multiple Accounts on multiple systems.

The principal security properties of an Account are:

- **type**: indicates whether the account is intended for use by human individuals (NOMINAL), shared among a team (FUNCTIONAL) or machines (TECHNICAL) or a temporary disposable account used for testing, guest wifi etc. (DISPOSABLE).
- **domain**: identifies the policy domain that defines account namespaces, naming conventions, the assignment or right to claim a particular account name etc.

8.4.2 Application Role

Application Role represents the roles (and their associated set of access rights) defined in the application code or configurable on the target system. In the Security Overlay, Application Roles are fashioned from a stereotype of Application Component. They serve Accounts and can be assigned to Application functions, as shown in Figure 59 and Figure 60.

8.4.3 Application Service

Application Services were discussed earlier in relation to SLA's and Availability. In terms of their ability to assert access control, three additional properties are provided:

- **identityProof**: enrolment and credential issuance processes bind the applicant to a real identity;
- **authenticationStrength**: the assurance strength of the authentication mechanism itself;
- **authenticatorTrust**: assurance in the credential being presented, considering aspects such as bearer-type, one-time use, freshness, challenge/response, trust in the credential issuer etc.

8.4.4 Application Process and Function

The key security-related property of an Application Function (and other internal behaviour elements) is the **authorisationContext** in which the function is executed. It is important to declare when the application is performing the function using its own access rights or is acting as a proxy for the caller, under some form of impersonation or constrained delegation. The former indicates a highly privileged technical account that merits high assurance controls.

8.4.5 Application Interface

Application Interfaces bind Application Services to an endpoint or channel through which they may be consumed. Interfaces can be categorised into two types: human (e.g. command line, GUI; voice-operated) and machine interfaces (e.g. remote procedure call, message-driven, REST or SOAP), each of which is likely to be subject to different constraints or policies.

For example, an organisation may decide that all machine-to-machine authentication must use certificates rather than passwords or that machine credentials should not be valid when proffered to human interfaces.

The Security Overlay provides a `type` property to indicate interfaces intended for human or machine use.

Each interface via which a service is exposed must satisfy the minimum **authenticationStrength** and **authenticatorTrust** requirements defined by the Application Service using a type-appropriate mechanism.

An **authentication** property declares the authentication mechanism supported by the interface, set to one from a list of predefined values: password, password hash, ticket, OTP, public key, certificate, SAML etc.

Table 40 presents a summary of overloaded elements discussed so far.

Table 40: Security Properties of Elements used in Logical Layer


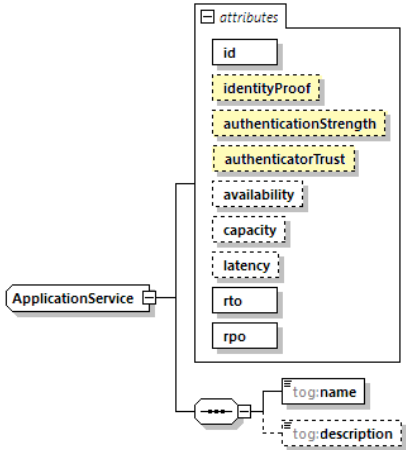
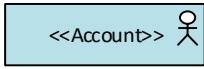
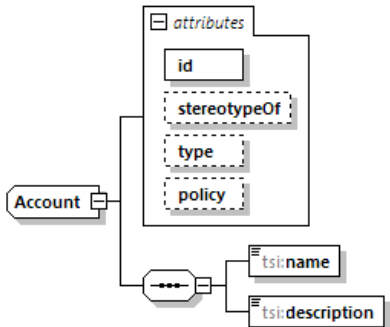

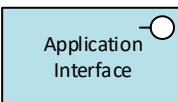
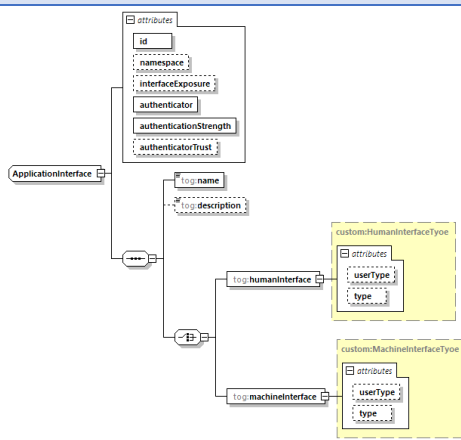
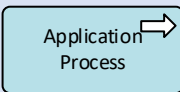

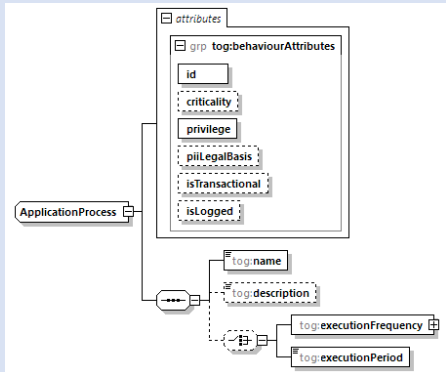
Element	Properties	Schema
	<p>Application Service availability properties were discussed in Section 8.3</p> <p>The access control properties, identityProof, authenticationStrength & authenticatorTrust are enumerated types that denote the required assurance.</p>	

Table 40: Security Properties of Elements used in Logical Layer

Element	Properties	Schema
	<p>Account is the logical layer representation of a Principal from the perspective of an application or target system</p> <ul style="list-style-type: none"> type: indicates whether the account is intended for use by human individuals (NOMINAL), shared among a team (FUNCTIONAL) or machines (TECHNICAL) or a temporary disposable account used for testing, guest wifi, etc. (DISPOSABLE). policy: identifies the policy domain that defines account namespaces, naming conventions, the assignment or right to claim a particular account name etc 	
	<p>An Application Role is modelled as a stereotype of Application Component. It defines a set of access rights on a single target system.</p>	<p>An open schema in which to declare a user-defined set of access rights</p>
	<p>The interfaceType property distinguishes interfaces used by humans from those intended for machines. Each option has its own set of valid enumerated values</p> <p>The interfaceExposure property marks the interface as being exposed locally or remotely.</p> <p>The authenticator property declares the mechanism implemented at the interface, e.g. PASSWORD or BIOMETRIC and its associated assurance strength and trust (a reflection of the rigour of the user registration process)</p>	
 	<p>Application behaviour element properties include:</p> <ul style="list-style-type: none"> criticality captures the highest common denominator of business activity supported: [NA, UNCLASSIFIED, VLOW, LOW, MODERATE, HIGH, VHIGH]; privilege indicates the access profile used in execution; e.g. [SELF, USER, ADMIN, ROOT]; piiLegalBasis declares the justification for any personal data processed; isTransactional flags whether the activity exhibits ACID qualities; isLogged flags whether the activity is logged. executionFrequency and executionPeriod express as a confidence range how often the process is performed. 	

8.5 Logical Domains

Logical domains can represent notional boundaries in the Logical Layer, such as schema belonging to a particular namespace, web applications deployed under an Internet domain name, or the authorised usage (functions) covered by a PKI Signing Certificate.

As with other core layers, logical domains can be expressed by Grouping elements (Location being intended to represent only physical or conceptual space).

Pure logical space is modelled as a Grouping unadorned. A means of marking a Logical Domain as a security domain (conceptual space) is discussed in Section 7.5: Domain Framework Model.

8.6 Timing and Events

8.6.1 Application Security Events

These elements can be used to model security-significant errors and exceptions thrown by application behaviour, e.g. an exception encountered while validating a digital signature. The schema is the same as that presented in Section 7.6.

9 Modelling the Physical Security Architecture

The Physical Security Architecture provides a security perspective of the Builder's view of the target system. The entities to be considered in this layer include the physical forms of persisted information (software & data), physical infrastructure (hardware, system software & network components) and human-machine interfaces.

This layer of the SABSA Matrix is concerned with how these components are structured and managed holistically to provide an operational environment of technological platforms and physical space. The security aspects of each component taken individually are addressed in the SABSA Component Architecture.

When it comes to modelling, it is difficult to show the organisation of components without the components themselves. We must also contend with the fact that ArchiMate has no direct equivalent of the Component Architecture: its Physical layer representing the world of industrial robots, IoT, facilities and distribution networks. For these reasons, this section discusses the Physical and Component layers together.

The artefacts of the SABSA Physical and Component layers are shown in Figure 42 and Figure 43.

Table 41: SABSA Physical Architecture

Physical	Artefacts	Assets (What)	Motivation (Why)	Process (How)	People (Who)	Location (Where)	Time (When)
		Data Assets	Risk Management Practices	Process Mechanisms	Human Interface	Infrastructure	Processing Schedule
	Activities	Data Dictionary & Data Storage Devices Inventory	Risk Management Rules & Procedures; Risk Metadata	Working Procedures; Application Software; Middleware; Systems; Security Mechanisms Process Control Points	User Interface to Business Systems; Identity & Access Control Systems	Workspaces; Host Platforms, Layout of Devices & Networks	Timing & Sequencing of Processes & Sessions
		Physical Asset Management	Risk Data Management	Operations Management	User Support	Resources Management	Performance Data Collection
		Change Management; Platform & Data Storage Management	Risk Procedure Management; Risk Metadata Management	Job, Incident, Event & Disaster Recovery Management	Service Desk, Problem & Request Management	Physical & Environmental Security Management; Real Estate & Facilities Management	Business Systems Monitoring; Procedure Management

Table 42: SABSA Component Architecture

Component	Artefacts	Assets (What)	Motivation (Why)	Process (How)	People (Who)	Location (Where)	Time (When)
		Components & Standards					
	Activities	Component Assets	Risk Management	Process	Human Entities:	Locator	Step Timing Sequencing
		Products and Tools, including Data Repositories & Processors	Risk Analysis Tools; Risk Registers; Risk Monitoring & Reporting Tools	Tools & Protocols for Process Delivery; Application Products;	Identities, Job Descriptions; Roles; Functions; Actions & Access Control Lists	Nodes, Addresses & other Locators; Component Configuration	Time Schedules; Clocks, Timers & Interrupts
		Delivery and Continuity Management	Operational Risk Management	Process Delivery Management	Governance Relationship & Personnel Management	Environment Management	Time & Performance Management
		Assurance of Operational Excellence & Continuity	Risk Assessment; Risk Monitoring & Reporting; Risk Treatment	Management & Support of Systems, Applications & Services	Management & Support of Enterprise-wide & Extended Enterprise Relationships	Management of Buildings, Sites, Platforms & Networks	Management of Calendar & Timetable

9.1 Data and Technology Assets

The principal assets in this cell are the persisted formats of information/knowledge (applications and data) and IT infrastructure (hardware, system software and network components). The 'bricks & mortar' type of physical assets are discussed in Section 9.5.

9.1.1 Artefact

In standard ArchiMate, Artefact serves as the universal passive structure element of the Technology Layer. It is overloaded to represent any kind of data object in the file system: executables, scripts, data & configuration files, databases, documents, specifications: everything in fact, except System Software.

The Security Overlay refines Artefact into sub-types that better reflect its different purposes and provides appropriate properties. These stereotypes are Executable, Data and Configuration. Having distinct stereotypes facilitates the creation and maintenance of separate software and data inventories.

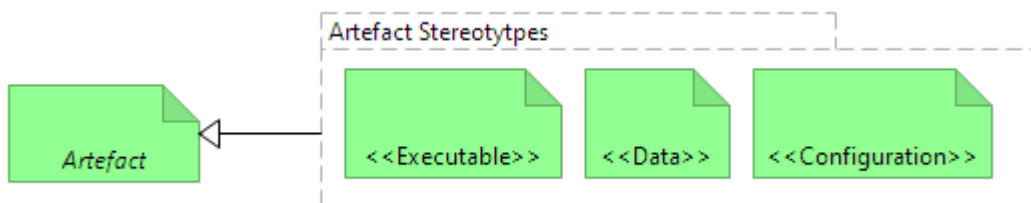


Figure 61: Stereotypes of Artefact

The consequence of this in the model is that an Artefact element's **isAbstract** property is TRUE by default, and turned FALSE only when one of these stereotypes is selected.

9.1.1.1 Executable

Executables are annotated with the same version control and file integrity properties as System Software. Commercial software can also be associated with CVE vulnerabilities (Software Defects), whereas bespoke software vulnerabilities may be associated with internally-managed defect reports arising from static analysis, penetration testing, bug reports etc.

9.1.1.2 Data

Data Artefacts realise Data objects from the Logical Layer. Their security properties also reflect those of the Data Object, but here **encryption** and **integrity** refer to controls applied at the file/database level rather than those managed by the application. A **type** property is used to distinguish between artefacts stored in the file system and those that are part of a database.

9.1.1.3 Configuration

A configuration file associated with an Executable or System Software element in the Technology Layer represents a Baseline Configuration that should include all relevant security settings (secure by default).

Different instances of this component in the Logical Layer may opt to enable or relax these settings to achieve a Least Privilege configuration in a specific application context. Figure 62 illustrates this principle.

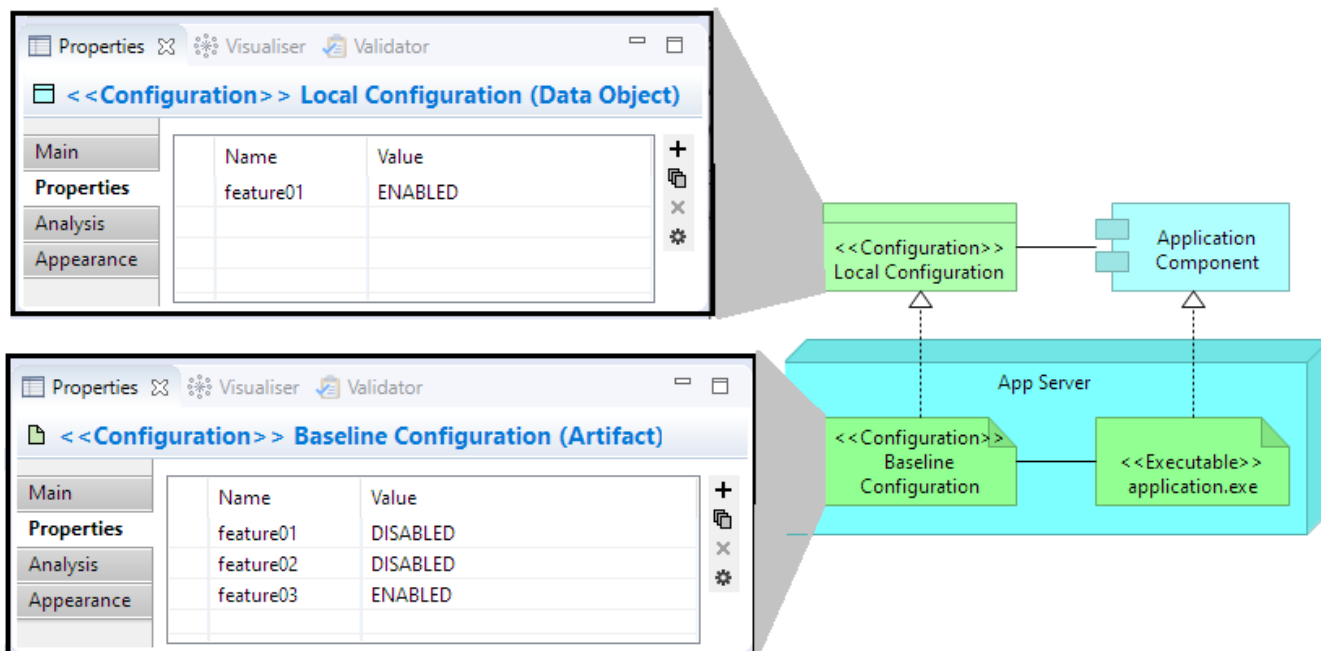


Figure 62: Configuration Files

9.1.2 Device and Node

Nodes represent a computational or physical resource that hosts, manipulates or interacts with other computational or physical resources. Devices model physical IT resources (i.e. hardware) upon which system software and Artefacts may be stored or deployed for execution.

When modelling security infrastructure, it is sometimes useful to model nodes and devices having a specific security purpose as distinct stereotypes so that role-specific properties can be assigned.

For example, Figure 63 shows a Hardware Security Module (HSM) being modelled as a stereotyped Device with properties that reflect FIPS-140-2 assurance levels and a VPN Gateway as a stereotyped Node with properties that reflect the Common Criteria Evaluation Assurance Levels (EAL) for the protection profile of a type of product.

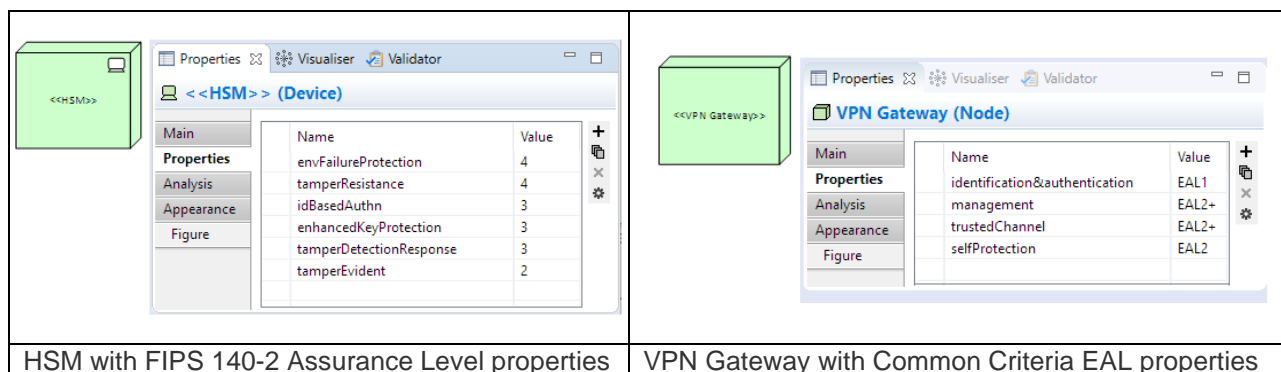


Figure 63: Stereotyping Security Nodes & Devices

9.2 Risk Management Practices

9.2.1 Defect

Software Defects are either published in product threat advisories, CVE lists or discovered in custom code. Because architectural models are mostly concerned with deployed code (either in the current or some future state), defects are more likely to be associated with Production code than those still in Development.

The main exception to this is an organisation's process for approving deviations. Often there is legitimate business pressure for software to be deployed into Production despite defects: a planned security control may not have been completed in time, or a penetration test has uncovered a problem at the eleventh hour. In such circumstances, a decision must be made on risk. Models can be analysed as 'what if' scenarios to inform these risk decisions. Figure 64 illustrates how highly relevant risk information is readily at hand.

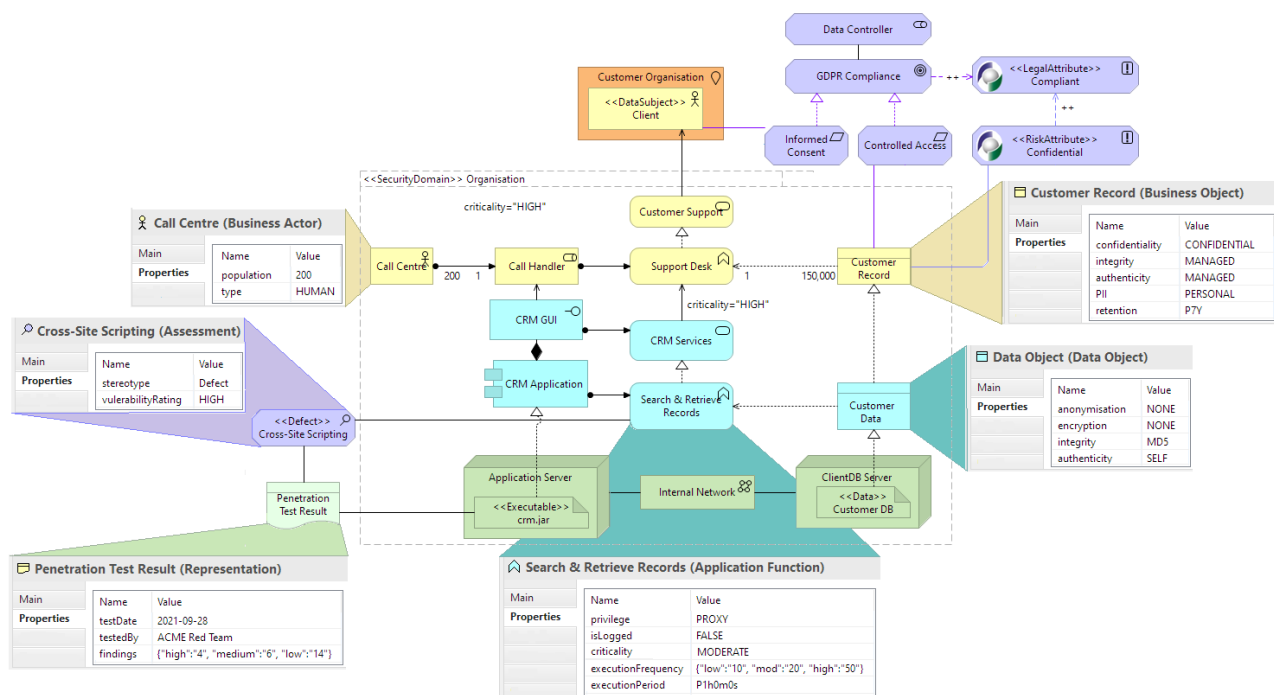


Figure 64: Risk Management Practices

In the Security Overlay, defects are modelled as specialisations of Vulnerability. The XSS Vulnerability in Figure 64 evaluates the severity of the technical findings of a penetration test recently performed on a CRM application. The report identifies 24 defects, 4 of which are rated HIGH. One of these is manifest in the application's 'Search & Retrieve' function. This function has read access to Customer Data, information classified as CONFIDENTIAL PERSONAL and openly identifiable (no anonymisation). The 'Search & Retrieve' function is considered of MODERATE criticality because the CRM Support Desk (itself a moderately critical Business Process) is highly reliant on it. The function runs as a proxy with the access rights of the End User. As Search and Retrieve actions are not logged, there is no audit trail of records that have been accessed. Finally, we can see that the vulnerable function is used by 200 Call Centre accounts, each with access to 150,000 Customer records. Any breach would be a compliance issue, requiring the Controller to notify the GDPR Supervisory Authority.

9.3 Process Mechanisms

From a security perspective, Technology Services fall into two main categories: those that realise the security services identified in the Conceptual layer and those that provide general platform services to Applications.

Figure 65 illustrates the pattern by which a conceptual security service, authentication, is realised by two distinct Technology Services: one which provides a single sign-on capability for the organisation domain; the other, the local authentication mechanism of a cloud application.

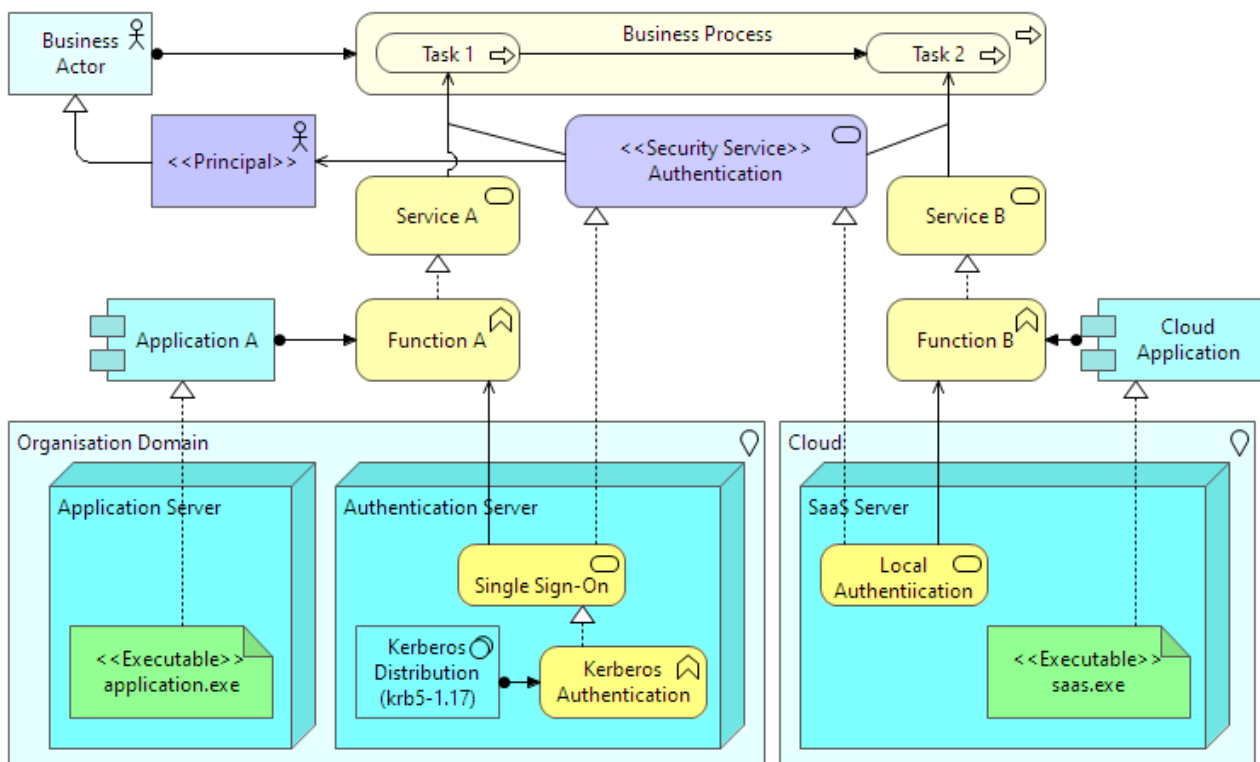


Figure 65: Realisation of Security Services

Similar patterns can be deployed for other security services provided by the infrastructure: Security Incident Event Management (SIEM), Privileged Access Management, Virus Scan, Data Leak Prevention and so on.

The closed loops, including the invocation of the conceptual Authentication service, its technical implementations and the services being protected, means that the model can not only verify that assurance strength requirements required by each Application Service are met but also trace any Compliance requirements by identifying the closed loops described in Section 7.2.2: Policy Architecture.

Turning to Technology Services in general, while the ability to enforce access control is as important as the earlier analysis of Application Services, it is often mitigated by being co-located on the same Node, local network or network domain. Priority should focus therefore, on service invocations that originate outside trust domain boundaries.

9.3.1 Technology Functions and Services

Technology functions and services are modelled in much the same way as their counterparts at the Application level (Section 8.3). They share the same security concerns of availability and access control, so it should come as no surprise that their Security Overlay properties are the same.

9.3.2 System Software

The most important property of system software is its release version. This information is key to determining its patch status and any CVEs³⁴ (Software Defects) to which it may be vulnerable.

ArchiMateSE proposes two properties for this purpose: **version**, which declares the release version and a derived property, and a derived property, **patchStatus**, which makes explicit whether the release version is current. If the executable code carries some form of integrity protection, e.g. a hash or code signing, this can also be denoted using the **integrity** property.

9.4 Human-Machine Interfaces

9.4.1 Technology Interface

Technology Interfaces represent endpoints that specify how Technology Services can be accessed by other active structure elements in the environment. These may be other Technology Nodes, Application Components, or Actors/Roles (e.g. relevant to a Management Interface).

The properties of the Technology Interface replicate those of the Application Interface, but in practice, are more likely to be technical APIs serving Application Components. GUIs tend to be used when Operators require intervention.

9.5 Physical Environment

The Location element is perfectly suited to representing physical workspaces or the place where IT equipment is located (e.g. Data Centres). Network zones and other logical or policy-governed boundaries are better represented by Groupings, marked with the **security** property set TRUE when coinciding with a security domain, as described in Section 7.5: Domain Framework Model.

9.6 Timing and Interrupts

9.6.1 Technology Security Events

These elements can be used to model security-significant errors and exceptions thrown at the infrastructure level, e.g. an alert thrown by an Intrusion Detection System or Data Leak Prevention. The schema is the same as that presented in Section 7.6.

³⁴ A list of [Common Vulnerabilities & Exposures](#) maintained by mitre.org

Table 43 presents a summary of Technology elements.

Table 43: Security Properties of Elements used in Technical Layer


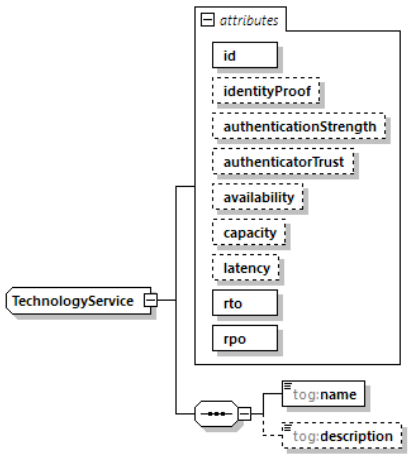
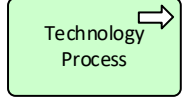
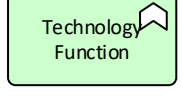
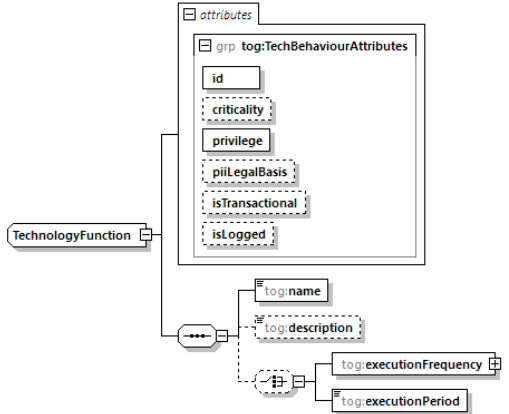
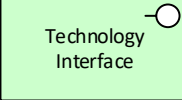
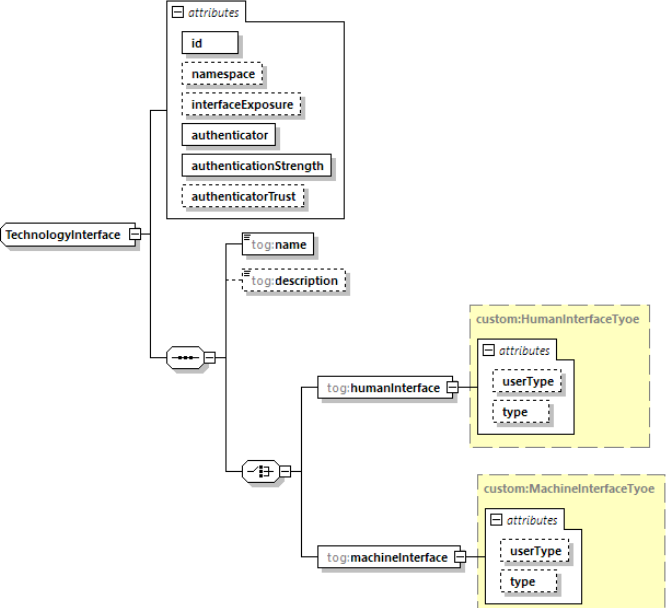
Element	Properties	Schema
 <p>Technology Service</p>	<p>Infrastructure Service properties mirror those defined for Application Services.</p>	 <p>The schema for TechnologyService shows a container element with a group of attributes: id, identityProof, authenticationStrength, authenticatorTrust, availability, capacity, latency, rto, and rpo. It also has two optional attributes: tog:name and tog:description.</p>
 <p>Technology Process</p>  <p>Technology Function</p>	<p>Technology behaviour properties mirror those defined for Application behaviour.</p>	 <p>The schema for TechnologyFunction shows a container element with a group of attributes: id, criticality, privilege, piiLegalBasis, isTransactional, and isLogged. It also has two optional attributes: tog:name and tog:description. Additionally, it has two optional attributes: tog:executionFrequency and tog:executionPeriod.</p>
 <p>Technology Interface</p>	<p>Technology Interface properties mirror those defined for Application Interface.</p>	 <p>The schema for TechnologyInterface shows a container element with a group of attributes: id, namespace, interfaceExposure, authenticator, authenticationStrength, and authenticatorTrust. It also has two optional attributes: tog:name and tog:description. Additionally, it has two optional attributes: tog:humanInterface and tog:machineInterface. The tog:humanInterface attribute is further detailed with a custom:HumanInterfaceTypeoe element, which has attributes userType and type. The tog:machineInterface attribute is further detailed with a custom:MachineInterfaceTypeoe element, which has attributes userType and type.</p>

Table 43: Security Properties of Elements used in Technical Layer

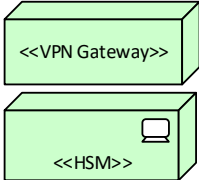
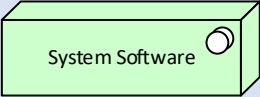
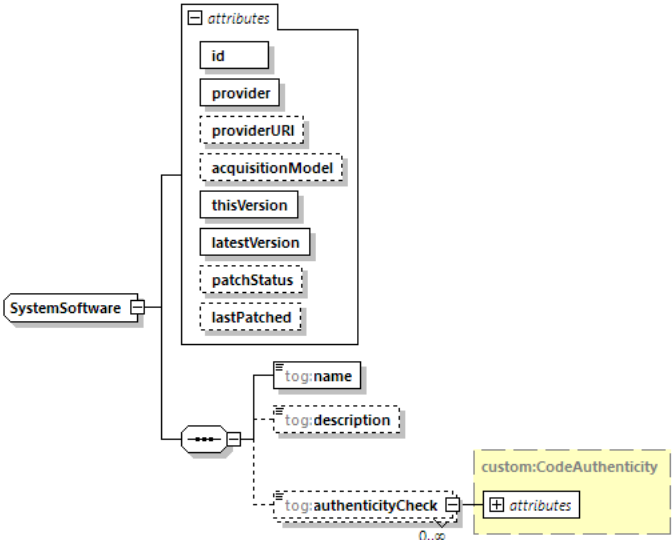
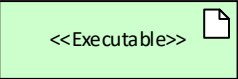
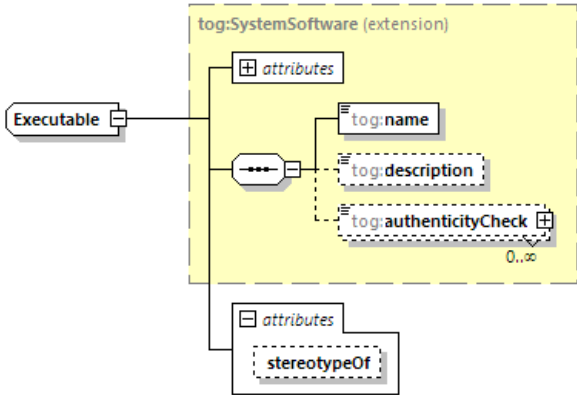
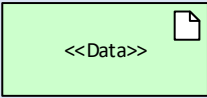
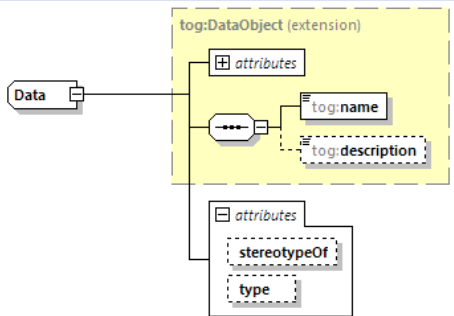


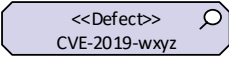
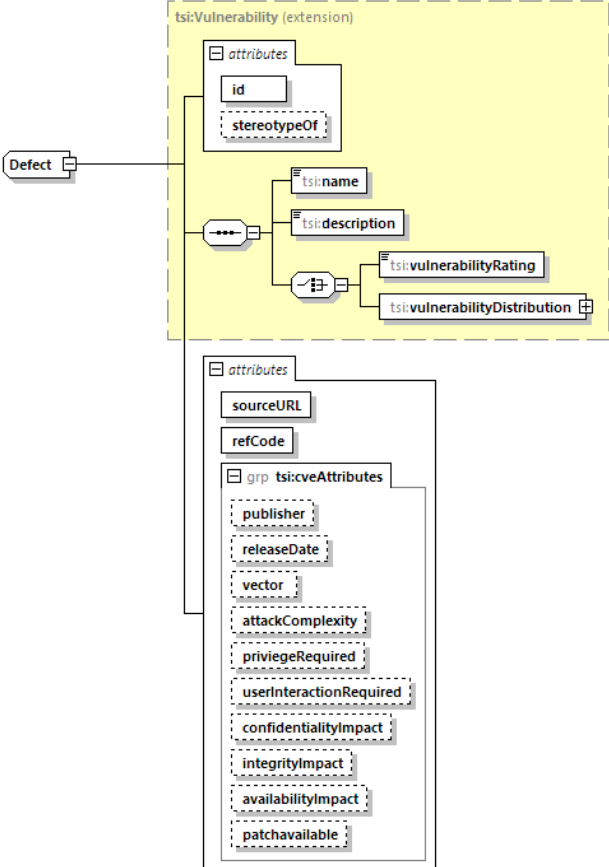
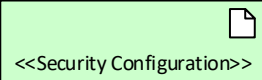
Element	Properties	Schema
	Nodes and Devices with a specific security purpose can be modelled as distinct stereotypes so that role-specific properties can be assigned.	Element Specific
	System Software properties declare the patch release version , date and status (derived), and any code integrity protection. The authenticityCheck is used to verify signed code.	
	Executable is modelled as a stereotype of Artefact but with the same properties as System Software.	
	Executable is modelled as a stereotype of Artefact but with the same properties as DataObject plus a type property that indicates the storage medium, e.g. FILE, DB_TABLE.	

Table 43: Security Properties of Elements used in Technical Layer

Element	Properties	Schema
  	<p>Defect is modelled as a specialisation of Vulnerability, adding properties that reference an external document such as a CVE or Test Report.</p>	
	<p>Security Configuration declares functionality that has been enabled in the baseline (least common functionality) configuration. Properties are user-defined but set to either ENABLED or DISABLED</p>	<p>Element Specific</p>

10 Conclusion

This paper proposes a viable and versatile Security Overlay to support the SABSA approach with the ArchiMate EA modelling language. The expected benefit of such an alignment is that ESA concepts can be incorporated into a unified EA model that is shared with other architects and reflects the reality of the enterprise's single architecture. Security becomes an integral part of the design.

The ability to generate consistent views of a single model will benefit all who rely upon them: the more accurate, consistent and complete the model, the better the analysis, decision-making, change planning and execution is likely to be. Should a rudimentary alignment have proved impossible, the prospect of creating distinct EA/ESA models, defined in different notations, by different tools, by different teams would be a death knell. Synchronisation would likely be so complex and resource-intensive as to be unviable.

Equally important as what security brings to the EA model is what modelling brings to security.

Firstly, visualisation: security models are a complex mesh of concepts and relationships that are far better suited to representation by a simple yet expressive notation of symbolic nodes and lines than as checklists and matrices. Because diagrams are much more intuitive to create, understand, verify and review, the result is more usable and of superior quality.

Secondly, the economics of modern IT drive projects to deliver more value earlier and repeatedly on ever shorter cycles. These "Agile" approaches shift emphasis heavily towards functionality, too often to the detriment of good structure and the planning, blueprints and documentation overhead that go with it. In this world, architecture has been defined as the opposite of Agile: *"the decisions that need to be made correctly at the start of the project because they are difficult to change afterwards"*³⁵.

SABSA teaches us that security architecture is not another architectural layer but a holistic concern, affecting all EA layers. Risks arising from a de-emphasis on architecture and design are especially exposed by Agile. How can we perform an effective security design review when "the code is the design"?

It's not that Agile is implacably opposed to documentation – only that it should be an enabler and not a drag on the project. To this end, scaled Agile methodologies are turning towards Model-Based System Engineering (MBSE) as a means to keep the artefact set complete, up-to-date, consistent and tailored to specific stakeholder views.

By generating these artefacts as views of a single underlying model, this overhead is vastly reduced. Reliable documentation can be maintained and published easily and regularly.

Re-use becomes possible on many levels: elements, patterns and even analysis. As these models expand beyond the logical design (UML) to full system models (ArchiMate), this Security Overlay will enable SABSA

³⁵ Martin Fowler: "Who Needs an Architect?" :
(<http://files.catwell.info/misc/mirror/2003-martin-fowler-who-needs-an-architect.pdf>)

to be lighter, faster, and better adapted to the short-cycle, agile, minimum viable approach of modern projects.

Lastly, ArchiMate models are machine-readable because they are defined in a semi-formal notation that can be exported in a standardised XML Exchange Format. The Security Overlay outlined in this paper has maintained an eye on its potential use, not only for documentation but also to query and analyse them via automated means. Should a security notation such as the one described here become a standardised extension to the language, the prospect of tools to support, for example, policy compliance checking or computer-assisted requirements generation, would become a realistic proposition.

In demonstrating this alignment, relatively few fundamental issues have been encountered. The most significant of these are the different concepts of an architectural Principle, the lack of a means of demonstrating the precise implementation of Controls and the absence of the elements necessary to populate the Conceptual layer. These challenges have been resolved by using the language's inbuilt extension mechanisms without breaking the grammar while retaining a degree of elegance.

Although there is still a significant amount to learn in this area, there is a basis here to work towards viable security patterns in ArchiMate that can lay the groundwork for a standardised security extension to the language in the future.