DRAFT - IT Project Guidance

On Social Cohorts

Version:

0.1

## Purpose & Audience

To provide to decision makers understanding of Cohorts as a dynamic means of grouping Persons

## Synopsis

TODO

## Contents

[Purpose 1](#_Toc195791704)

[Synopsis 1](#_Toc195791705)

[Contents 2](#_Toc195791706)

[Introduction 3](#_Toc195791707)

[Background of this work 3](#_Toc195791708)

[Context 3](#_Toc195791709)

[Background of issue 3](#_Toc195791710)

[Problem Statement 3](#_Toc195791711)

[H3 4](#_Toc195791712)

[H4 4](#_Toc195791713)

[Appendices 5](#_Toc195791714)

[Appendix A - Document Information 5](#_Toc195791715)

[Versions 5](#_Toc195791716)

[Images 5](#_Toc195791717)

[Tables 5](#_Toc195791718)

[References 5](#_Toc195791719)

[Review Distribution 5](#_Toc195791720)

[Audience 5](#_Toc195791721)

[Structure 5](#_Toc195791722)

[Diagrams 6](#_Toc195791723)

[Acronyms 6](#_Toc195791724)

[Terms 6](#_Toc195791725)

# Background

Within the architecture of a modern information system, the **Social Domain** is recognised as a primary domain, positioned alongside the **System Domain** and **Media Domain**. While the System Domain governs the operational and infrastructural aspects of the platform, and the Media Domain manages artefacts and representations, the Social Domain encodes the human-centric constructs that underpin identity, affiliation, and interaction.

The Social Domain is responsible for modelling concepts such as Persons, Groups, Roles, and the Relationships that bind them.

These constructs are foundational to any system that seeks to represent real-world social structures—whether educational, organisational, familial, or community-based. The domain supports enduring entities (e.g. a School or Organisation), transient roles (e.g. Student or Member), and the relationships that allow metadata to be attached to a Person’s involvement in a Group (e.g. enrolment date, role type, or status).

One area of increasing relevance, but insufficiently addressed in prior guidance, is the concept of **Cohorts**. Unlike static Groups, which are explicitly defined and persist as named entities, Cohorts represent dynamically defined sets of Persons based on shared characteristics. These characteristics may be demographic (e.g. birth date range), behavioural (e.g. service usage), or contextual (e.g. location at a point in time). A common example is the "BabyBoomer" cohort, defined by a date-of-birth range rather than membership in a named Group.

Cohorts differ from Groups in a critical architectural respect: they do not inherently possess a persistent relationship object between the Person and the Cohort. In static Group models, the relationship object (e.g. PersonGroupRole) serves as the anchor for metadata, enabling systems to record and manage attributes of the relationship itself. This includes operational data such as the date of joining, the role held, or flags indicating status or eligibility.

In contrast, Cohorts are typically defined by a **CohortDescriptor**, which encodes the filtering logic used to identify members. The resulting set of Persons is derived, not declared. While this supports flexibility and dynamic querying, it introduces a significant limitation: there is no enduring object representing the relationship between a Person and the Cohort. Consequently, metadata cannot be attached to the relationship, and operational clarity suffers.

Some systems attempt to mitigate this by introducing a **CohortMembership** catalogue, which records instances of Persons being associated with a Cohort. However, without a persistent Cohort entity distinct from the descriptor, this approach remains fragile. It lacks the semantic clarity and lifecycle management afforded by traditional Group relationships. Moreover, it complicates auditability, versioning, and metadata enrichment—functions that are increasingly critical in systems supporting eligibility, entitlements, or longitudinal analysis.

**Issues**

The absence of a persistent relationship object in Cohort models leads to several architectural and operational issues:

* **No metadata anchoring**: Without a relationship object, metadata such as inclusion date, source of inclusion, or cohort-specific flags cannot be reliably stored.
* **Limited auditability**: Dynamic inclusion based on descriptors cannot be audited in the same way as declared relationships.
* **Operational ambiguity**: Systems cannot easily distinguish between current and historical cohort membership, nor can they support freezing membership at a point in time.
* **Versioning challenges**: Changes to cohort definitions are difficult to track and manage without a persistent entity to version against.
* **Entanglement risk**: Encoding operational logic directly into descriptors undermines maintainability and introduces coupling between business rules and data access logic.

# Design Guidance

To resolve the limitations inherent in current Cohort modelling, the Social Domain must evolve to treat Cohorts as first-class entities, structurally analogous to Groups but semantically and operationally distinct. This distinction is necessary to preserve clarity in system behaviour, maintainability, and metadata management.

Groups are *declared* entities. They are created explicitly and persist as named constructs within the system. Their membership is managed through deliberate actions—such as enrolment, assignment, or invitation—and the Group itself often has governance, identity, and a lifecycle independent of its members. The relationship between a Person and a Group is mediated by a persistent object (e.g. PersonGroupRole), which supports metadata, auditability, and operational logic.

Cohorts, by contrast, are *derived* entities. They are defined by a rule or filter applied to the Person population, and their membership is inferred from attributes rather than declared. A Cohort does not possess governance or agency; it is a conceptual grouping based on shared characteristics. Treating Cohorts as Groups risks conflating declared and derived membership, which introduces operational ambiguity and undermines the integrity of relationship modelling.

To address this, the system should introduce a persistent Cohort entity. This entity represents the conceptual group (e.g. BabyBoomers, Service-Eligible Youth) and provides a stable anchor for system logic, metadata, and lifecycle management. It is distinct from the CohortDescriptor, which encodes the filtering logic used to derive membership. The descriptor may change over time, be versioned, or be frozen for operational purposes, but the Cohort entity remains stable and identifiable.

The second component is the CohortMembership object. This object links a Person to a Cohort and serves as the relationship container on which metadata can be hung. This includes attributes such as the date of inclusion, the method of inclusion (e.g. derived vs declared), flags indicating eligibility or status, and any operational annotations. This pattern mirrors the use of PersonGroupRole in static Group models, ensuring consistency across the domain.

This approach supports both dynamic and declared cohort membership. In dynamic mode, membership is derived from the descriptor and recomputed as needed. In declared mode, membership is explicitly recorded, allowing the system to freeze the cohort at a point in time or to support auditability. Both modes rely on the Cohort entity as the anchor, and the CohortMembership object as the relationship.

Operational clarity is further enhanced by supporting versioning of CohortDescriptors. This allows the system to track changes in cohort definitions over time, and to associate Persons with specific versions of a cohort. This is critical in contexts where eligibility or inclusion criteria evolve, and where historical analysis or compliance requires traceability.

Finally, metadata must not be encoded solely within the descriptor logic. Doing so undermines maintainability and introduces coupling between business rules and data access. Instead, metadata should be attached to the Cohort entity or the CohortMembership object, depending on whether it pertains to the group as a whole or to the individual relationship.

Appendices

Appendix A - Document Information

Authors & Collaborators

* Sky Sigal, Solution Architect

### Versions

* 1. Initial Draft

### Images

[Figure 1: TODO Image 2](#_Toc144995112)

### Tables

[Table 1: TODO Table 3](#_Toc145048484)

[Table 2: TODO Table 2 3](#_Toc145048485)

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
|  |  |
|  |  |
|  |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Acronyms

API

: [Application Programming Interface](#Term_ApplicationProgrammingInterface).

DDD

: Domain Driven Design

GUI

: [Graphical User Interface](#Term_ApplicationProgrammingInterface). A form of [UI](#Acronym_UI).

ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. [IT](#Acronym_IT) is a subset of ICT.

IT

: acronym for Information, using Technology to automate and facilitate its management.

UI

: User Interface. Contrast with [API](#Acronym_API).

### Terms

Refer to the project’s Glossary.

Application Programming Interface

: an Interface provided for other systems to invoke (as opposed to User Interfaces).

Capability

: a capability is what an organisation or system must be able to achieve to meet its goals. Each capability belongs to a domain and is realised through one or more functions that, together, deliver the intended outcome within that area of concern.

Domain

: a domain is a defined area of knowledge, responsibility, or activity within an organisation or system. It groups related capabilities, entities, and functions that collectively serve a common purpose. Each capability belongs to a domain, and each function operates within one.

Entity

: an entity is a core object of interest within a domain, usually representing a person, place, thing, or event that holds information and can change over time, such as a Student, School, or Enrolment.

Function

: a function is a specific task or operation performed by a system, process, or person. Functions work together to enable a capability to be carried out. Each function operates within a domain and supports the delivery of one or more capabilities.

Person

: a physical person, who has one or more Personas. Not necessarily a system User.

Persona

: a facet that a Person presents to a Group of some kind.

Quality

: a quality is a measurable or observable attribute of a system or outcome that indicates how well it meets expectations. Examples include reliability, usability, and performance. Refer to the ISO-25000 SQuaRE series of standards.

User

: a human user of a system via its UIs.

User Interface

: a system interface intended for use by system users. Most computer system UIs are Graphics User Interfaces ([GUI](#Acronym_GUI)) or Text/Console User Interfaces (TUI).