Axiomatic Design Process for Ecosystem Enterprise Architecture

Application of the Axiomatic Design Process to a Decentralised Organisation

Carlos Trigoso – NHSE Lead Enterprise Architect
May 2025

1. Introduction to Axiomatic Design

- Axiomatic Design, developed by Nam P. Suh, is a structured method for designing complex systems (see slide 30 for references)
- It translates what a system must do (functional requirements, or FRs) into how it will do it (design parameters, or DPs)
- Two core principles guide the process:
    - Independence Axiom: Ensure each part of the system works independently to avoid unintended interactions.
    - Information Axiom: Keep the system simple by minimizing unnecessary data or complexity.
- In enterprise architecture (EA), this approach aligns business needs with technical solutions.
- Goal: Create flexible, scalable, and robust architectures.

2. Why Axiomatic Design for EA?

- Enterprise systems are complex, involving business, data, applications, and technology.
- Axiomatic Design ensures clarity by breaking down needs into manageable parts.
- It promotes modularity, making systems easier to update or maintain.
- Benefits include:
  - Improved agility to adapt to business changes.
  - Reduced risk of system failures.
  - Enhanced performance through simplified designs.
- This method aligns with frameworks like TOGAF for structured EA design.

3. The Four Domains of Axiomatic Design

- **Axiomatic Design uses four domains to structure the design process:**
  - **Customer Domain: Identifies organisational or user needs (e.g., faster processes).**
  - **Functional Domain: Defines what the system must do (e.g., reduce processing time).**
  - **Physical Domain: Specifies how the system will achieve it (e.g., standardized workflows).**
  - **Process Domain: Details the processes to implement solutions (e.g., process changes)**
- **These domains guide the transformation from needs to technical specifications.**
- **In EA, they map to layers generally described as: Business, Data, Application, and Technology.**

I&AM Architecture and Four Domains of Axiomatic Design

I&AM sub domains	Customer Domain CAs	Functional Domain FRs	Physical Domain DPs	Process Domain PVs
1. Data Services	User lifecycle	Applications	Delivery	Parameters
2. Data Workflows “Provisioning”	User and management services	Authentication, Authorisation	Programmes and projects	User management
3. Data Management	Attributes of the System	Functional Requirements	Components, Sub-components	System processes
4. Data Control	Return, Efficiency	Business Goals	Business Structure	Human, Financial Resources
	Cas = Customer Attributes, FRs = Functional Requirements, DPs = Design Parameters, PVs = Process Variables			

4. Axiom of Independence in EA

- The Independence Axiom requires system components to work autonomously.
- In EA, this means designing business functions that operate independently but connect seamlessly.
- Example: A services delivery unit should function without relying heavily on other units.
- Benefits:
  - Flexibility to modify one part without affecting others.
  - Reduced risk of widespread system issues.
- Achieved by breaking the “enterprise” into modular units like units, regions, departments or services.

5. Axiom of Information Minimisation

- **The Information Axiom focuses on simplicity by reducing unnecessary data or processes.**
- **In EA, this means designing systems with only the essential information for tasks.**
- **Example: A reporting system should process only relevant data to avoid complexity.**
- **Benefits:**
  - **Faster decision-making due to streamlined data.**
  - **Easier system maintenance and user understanding.**
  - **Simplifies management and boosts operational efficiency.**

6. Applying Axiomatic Design to TOGAF Layers

- **TOGAF divides EA into four layers: Business, Data, Application, and Technology.**
- **Axiomatic Design maps FRs and DPs across these layers:**
  - **Business Layer: Define business needs (e.g., improve services) and solutions (e.g., optimise workflows).**
  - **Data Layer: Ensure data supports business needs (e.g., accurate data).**
  - **Application Layer: Deploy software to meet business needs (e.g., patient facing apps).**
  - **Technology Layer: Provide infrastructure (e.g., data services).**
- **This ensures alignment from business goals to technical implementation.**

7. Hierarchical Decomposition in EA

- **Axiomatic Design breaks high-level needs into detailed solutions step-by-step.**
- **Process:**
  - **Start with broad business goals (e.g., improve productivity).**
  - **Break them into specific needs for data, applications, and technology.**
- **Example:**
  - **Business Goal: Faster patient processing.**
  - **Data Need: Accurate patient data.**
  - **Application Need: Real-time tracking software.**
  - **Technology Need: Scalable application services.**
- **Ensures every level supports the overall goal. This is supported by "decoupling" the FR and DP matrices.**

8. The “Zigzagging” Method

- The zigzagging method links FRs and DPs across layers.
- Process:
- Define business needs and solutions.
- Use these solutions to set data needs, then data solutions.
- Continue through application and technology layers.
- Example:
  - Business Need: Enhance services delivery ---- > Solution: Optimize workflows.
  - Data Need: Provide process status ---- > Solution: Use a database.
  - Application Need: Enable tracking ---- > Solution: Deploy software.
- Keeps all layers aligned and consistent.

9. Handling Multiple Organisations

- Many enterprises involve multiple organisations with unique needs. This is particularly the case in the NHS and its ecosystem.
- Each organisation has its own set of FRs (e.g., different workflows).
- Axiomatic Design adapts by:
  - Defining FRs for each organisation separately.
  - Mapping them to solutions while ensuring independence.
- Example: Two organisational units might need:
  - Unit 1: Better reporting ----> Standardised tools.
  - Unit 2: Secure data ----> Validation rules.
- Solutions are tailored but integrated. Standardisation through FR consolidation.

10. Multiple Organisation Design Process

- **For multiple organisations or units, the process involves:**
  - **Step 1: Identify each organisation's needs (FRs).**
  - **Step 2: Define shared needs (e.g., common data standards) and unique needs.**
  - **Step 3: Create shared solutions (e.g., a standard architecture) and specific solutions (e.g., tailored applications).**
  - **Step 4: Break down solutions across data, application, and technology layers. Separate data from technology implementation.**
- **Ensures all organisations' needs are met while maintaining system coherence.**
- **Governance aligns unique and shared solutions.**

11. Decentralised Organisations

- In decentralized setups (local, regional, national), needs vary by level.
- Process:
  - Local Level: Define specific needs (e.g., local service targets).
  - Regional Level: Combine local needs with regional goals (e.g., resource allocation).
  - National Level: Set overarching goals (e.g., unified policies).
- Solutions at higher levels guide lower levels, while local needs influence higher goals.
- Balances local autonomy with enterprise-wide consistency.
- Resolves apparent conflicts and strategy disparity across levels.

12. Local Level Design in Axiomatic Design Processes

- Local entities (e.g., clinics, branches) have specific needs, like timely service delivery.
- Define local FRs: e.g., "FR_L1: Provide timely care" ----> Solution: "DP_L1: Hire staff."
- Expressed as: Local needs = [Local Design Structure] × Local Solutions.
- Bottom-up: Local FRs inform regional needs (e.g., resource demands).
- Top-down: Regional solutions (e.g., funding) guide local FRs but don’t replace them.
- Example: A clinic’s need for staff shapes regional budgets, while regional policies set staffing guidelines.
- Ensures local autonomy within broader constraints.

13. Regional Level Design and Aggregation

- Regional entities aggregate local needs and add region-specific goals, like equitable resource distribution.
- Define regional FRs: e.g., "FR_R1: Allocate resources" ----> Solution: "DP_R1: Regional funding."
- Expressed as: Regional needs = [Regional Design Structure] × Regional Solutions.
- Bottom-up: Local FRs (e.g., staff needs) shape regional FRs (e.g., budget allocation).
- Top-down: National solutions (e.g., policies) guide regional FRs, preserving regional uniqueness.
- Example: Regional funding reflects local clinic needs and follows national health policies.
- Balances local inputs with national guidance.

14. National Level Design and Guidance

- National level sets overarching goals, like improving staff productivity and system efficiency.
- Define national FRs: e.g., "FR_N1: Improve outcomes" ---- > Solution: "DP_N1: Unified policy."
- Expressed as: National needs = [National Design Structure] × National Solutions.
- Bottom-up: Regional FRs (e.g., resource needs) inform national FRs (e.g., policy scope).
- Top-down: National DPs guide regional and local solutions without overriding them.
- Example: A national health policy incorporates regional inputs and sets guidelines for local clinics.
- Ensures enterprise-wide coherence with flexibility.

15. Best Practices for Axiomatic Design in EA

- Involve stakeholders from all layers (business, planning, delivery, technology, etc.).
- Clearly define needs at each level, making them measurable.
- Aim for independent designs to avoid complex interactions.
- Choose the simplest solutions to meet needs.
- Validate designs through prototypes and reviews.
- Conduct workshops to align business and technical teams.
- Regularly revisit needs to adapt to changes.

16. Benefits of Axiomatic Design in EA

- **A structured approach simplifies complex system design.**
  - **Aligns technical solutions with business goals.**
  - **Enhances collaboration across business and IT teams.**
  - **Reduces risks by identifying issues early.**
  - **Improves system performance and scalability.**
  - **Frees local and regional initiative to address specific problems.**
  - **Supports innovation by allowing flexible updates.**
  - **Case Study Example: A retailer reduced order delays by standardizing workflows and deploying cloud infrastructure.**

17. Challenges and Solutions

- **Challenges:**
  - Complex systems with many interdependencies.
  - Balancing flexibility with structured design.
  - Legacy systems and ad-hoc solutions may need rationalisation.
  - Resolving conflicts between stakeholders.
- **Solutions:**
  - Use iterative design to refine solutions.
  - Validate each layer’s design before proceeding.
  - Implement governance to manage conflicts.
  - Link consolidation targets to collaborative strategy.
  - Formal requirements analysis resolve uncertainties.
  - Ensures streamlined solutions and service delivery.

18. Steps to Implement Axiomatic Design

- Develop a collaborative, joined-up Business Architecture process.
- Establish Architecture Governance to maintain consistency.
- Form work groups with experts from each EA layers.
- Identify business needs through workshops.
- Develop Requirements Analysis and consolidation at each layer of the Ecosystem.
- Formalise business needs through the Axiomatic Design Process.
- Break these into data, application, and technology needs and solutions.
- Use the zigzagging method to ensure alignment.
- Prototype and test designs at each stage.
- Re-evaluate regularly to adapt to new needs.

19. Conclusion

- Axiomatic Design provides a clear, structured way to design enterprise systems.
- Its principles of independence and simplicity ensure robust, flexible architectures.
- By aligning business needs with technical solutions, it supports strategic goals.
- Applicable to single or multiple organisations, including decentralized setups.
- Encourages collaboration, reduces risks, and enhances scalability.
- Adopt Axiomatic Design to build agile, effective enterprise architectures.

19. Conclusion

- Axiomatic Design provides a clear, structured way to design enterprise systems.
- Its principles of independence and simplicity ensure robust, flexible architectures.
- By aligning business needs with technical solutions, it supports strategic goals.
- Applicable to single or multiple organisations, including decentralized setups.
- Encourages collaboration, reduces risks, and enhances scalability.
- Adopt Axiomatic Design to build agile, effective enterprise architectures.

Annex

Formalisation of the Axiomatic Design Process

- **Axiomatic Design maps what systems must do (FRs) to how they do it (DPs).**
- **Expressed as: `{FR} = [A]{DP}`, where `[A]` links FRs to DPs.**
- **Challenge: Multiple organisations have unique FRs, which may overlap or conflict.**
- **Solution: Combine all FRs into one system while keeping them independent.**
- **Each organisation's FRs (e.g., `{FR_1}, {FR_2}`) are defined separately.**
- **Shared needs (e.g., security) and unique needs (e.g., workflows) are identified.**
- **Goal: Create a unified system that meets all needs efficiently.**
- **Uses a structured matrix to ensure clarity and modularity.**

B. Combining FRs and DPs

- Each organisation’s FRs form a vector, e.g., `{FR_i}` for organisation `i`.
- Combine into a single FR vector: `{FR} = [{FR_1}, {FR_2}, ..., {FR_N}]`.
- DPs include shared solutions (`{DP_common}`) and specific solutions (`{DP_i}`).
- Total DP vector: `{DP} = [{DP_common}, {DP_1}, ..., {DP_N}]`.
- The matrix `[A]` connects FRs to DPs: `{FR} = [A]{DP}`.
- `[A]` has blocks: shared DPs affect all FRs; specific DPs affect only their FRs.
- Zero blocks ensure specific DPs don’t impact other organisations’ FRs.
- This structure maintains independence, reducing unintended interactions.

C. Hierarchical Design Across Levels

- organisations may operate at local, regional, and national levels.
- Local FRs (e.g., `{FR_L,i}`) reflect specific needs, like service delivery.
- Regional FRs (`{FR_R,r}`) combine local needs with regional goals.
- National FRs (`{FR_N}`) include regional inputs and overarching goals.
- Bottom-up: Local needs shape regional FRs, which inform national FRs.
- Top-down: National DPs (e.g., policies) guide regional and local DPs.
- Each level uses: `{FR_level} = [A_level]{DP_level}` to map needs to solutions.
- Iterative process ensures alignment across all levels.

D. Bottom-Up and Top-Down Processes

- **Bottom-up aggregation: Local FRs influence regional, then national FRs.**
- **Example: Local need for staff informs regional budget, shaping national policy**
  - **Top-down decomposition: National DPs set regional FRs, which guide local FRs.**
- **Example: National policy defines regional funding, guiding local staffing.**
  - **At each level, DPs satisfy FRs while respecting higher-level guidance.**
  - **Regional FRs: `{FR_R,r} = {FR_R,r^derived} + {FR_R,r^direct}`, combining national and regional needs.**
  - **Local FRs follow similarly, ensuring unique needs are preserved.**
  - **Zigzagging method aligns levels, maintaining independence and simplicity.**

D. A three-Level Organisation/Ecosystem

- **For a three-layer organisation, the Axiomatic Design process uses hierarchical block-triangular matrices to map DPs to FRs at each level while enforcing sequential decoupling. Aggregation rules formalise how lover-level outcomes contribute to higher-level goals. This structure aligns with large-scale systems engineering, where independence is maintained through layered decomposition and controlled cross-level dependencies.**

- **To avoid coupling in the hierarchical design matrices, national-level FRs and DPs must be systematically propagated downward to regional and local levels during the design process, ensuring alignment and minimizing conflicts. By defining national FRs/DPs first and iteratively refining them into regional and local requirements (e.g. via decomposition rules or constraints), lower-level units can adjust their FRs/DPs before finalising their design matrices, ensuring that regional and local solutions inherently comply with national objectives.**

- **This feedback loop, enabled by iterative validation of lower levels FR-DP mappings against national requirements presents unintended couplings by resolving mismatches early, preserving the block-triangular structure of the design matrix hierarchy, where higher level DP constrain but do not directly interfere with lower-level independence.**

F. Definition of Uncoupled, Decoupled and Coupled Design Matrices

Uncoupled Design		
A11	0	0
0	A22	0
0	0	A33

Decoupled Design		
A11	0	0
A21	A22	0
A31	A32	A33

Coupled Design		
A11	A12	A13
A21	A22	A23
A31	A32	A33

E. Example: Mix of Uncoupled, Decouples and Coupled Design Matrices

NHS England

WHY: Which Processes are needed to Achieve the Business Goals (business strategy)	HOW: How processes are correlated across units, services, entities (models, frameworks)	WHO: Which Roles are responsible for each process (organisations, people)	WHAT : What are the results of products of the processes (technology, systems)

A11	0	0
0	A22	0
0	0	A33

A11	0	0
A21	A22	0
A31	A32	A33

A11	A12	A13
A21	A22	A23
A31	A32	A33

A11	A12	A13
A21	A22	A23
A31	A32	A33

Bibliography

- 1. Suh, N. P. (1990). The principles of design. Oxford University Press. This foundational book introduces Axiomatic Design, outlining its two core axioms (Independence and Information) and a matrix-based methodology for systematically mapping functional requirements to design parameters
- 2. Suh, N. P. (2001). Axiomatic design: Advances and applications. Oxford University Press. This book extends the methodology to complex systems, including organisational applications.
- 3. Farid, A. M., & Suh, N. P. (Eds.). (2016). Axiomatic design in large systems: Complex products, buildings and manufacturing systems. Springer. Applies Axiomatic Design to large-scale systems, including complex organisational structures.
- 4. Gungor, Z., & Sarihan, H. I. (2023). A systematic approach to enterprise architecture using axiomatic design. Procedia CIRP, 119, 455-460. https://doi.org/10.1016/j.procir.2023.02.122 This paper proposes a methodology to apply Axiomatic Design to EA, using the two axioms to systematically analyse enterprise capabilities and map requirements across EA layers (business, data, application, technology). It emphasizes aligning strategic goals with technical solutions.
- 5. van der Zee, B., & de Vries, B. (2024). Applying the V model and axiomatic design in the domain of IT architecture practice. Procedia CIRP, 132, 104-109. https://doi.org/10.1016/j.procir.2024.09.050 Focuses on healthcare IT architecture. Integrates Axiomatic Design with the V model to manage change in EA. Presents case studies using Axiomatic Design's domains (customer, functional, physical) to model strategic and technical requirements.