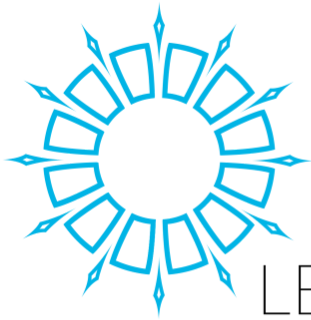




LEANARCHITECTURE FRAMEWORK

Version 1.6

Authors

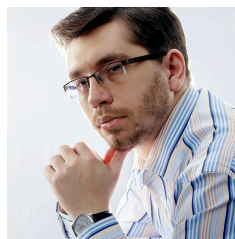


Tomasz Michałek-Czerepak



Tomasz Świerszcz

Cooperate



Dr. hab. Andrzej Sobczak, prof. SGH

© 2020 LAF Institute. Offered under the Attribution Share-Alike license of Creative Commons, accessible at <http://creativecommons.org/licenses/by-sa/4.0/legalcode> and also described in summary form at <http://creativecommons.org/licenses/by-sa/4.0/>. By utilizing this Guide, you acknowledge and agree that you have read and agree to be bound by the terms of the Attribution Share-Alike license of Creative Commons.



Glossary

- LAF uses four basic concepts that have many meanings depending on the context in the IT world. These are:
 - **Application** - a software program or a group of software programs that are designed for the end user.
 - **Component** - a software program or a group of software programs that have an implementation of one or more capabilities. The most common components are Applications, but the definition also includes cloud services, integration services, microservices, and others.
 - **Capability** - the ability of an Application or Component to carry out the fundamental functions required by an organization. These capabilities often represent key business processes, tasks, or data sources depending on the needs of the organization; usually, it is the main role and functionality of the Application or Component.
 - **Requirements** - the requirements that the business defines for Applications and Components.

Although Component is a more general definition, the definitions of Applications and Components are treated interchangeably due to the fact that they participate in the same architectural practices. The use of the word Application makes this document easier to understand.



1. Introduction

The pace, frequency and scope of changes in the world around us force a change in the approach to change management. Since change management has become a new standard, not an emergency, it means that only a continuous transformation of the company will allow it to survive nowadays.

Artificial intelligence is increasingly becoming a central driver of this transformation. AI agents now actively participate across the software development life cycle—from generating code to proposing architectural solutions. LAF addresses this shift explicitly in Section 5, extending its roles and artifacts so that architecture remains effective, cohesive, and well-governed when software development is carried out by both AI and human teams.

The Lean Architecture Framework is a collection of good practices thanks to which the IT environment will respond consistently and quickly to a changing business situation while maintaining its structural consistency. It does this by appropriately placing the Architect in the software development life cycle. An architect with the right knowledge can quickly make good decisions. All LAF elements also work very well in the process of purchasing applications or services. The newly purchased solution will be consistent with the entire environment.

LAF not only contributes to the implementation of continuous transformation; it also actively supports it. LAF does this through constant changes in architectural goals and continually striving to achieve them.

LAF supports the commitment of all architects and business stakeholders by building agreements between them, striving to establish a lasting, healthy relationship. Thanks to this, it fosters the personal development of architects working in the organization, which in turn has a direct impact on making good decisions regarding the IT environment.

Innovation is at the heart of LAF. Day after day, the architect has the opportunity to propose new solutions during direct cooperation with the business. LAF supports the dissemination and implementation of business ideas throughout the organization.

The main characteristics of LAF are:

Simple	LAF is simple, concise, and easy to understand.
Pragmatic	LAF concentrates only on activities that provide value and nothing more.
Adaptable	LAF is easy to adapt to the specifics and needs of the organization.
Complete	LAF is used to manage all aspects of a complex IT architecture environment; it is a full-fledged solution containing proven guidelines to support it.



Enables Architectural Agility

LAF supports a constant response to the current situation of an organization and adaptation to a changing business environment.

2. Why good architecture is needed

Every large organization needs to manage its IT architecture, especially the Application landscape, to stay cost-efficient, flexible, and open to change across the entire organization rather than just within a single business area or Application.

The LAF framework responds to these needs by providing full guidelines that define roles, events, and artifacts.

- The main goals of implementing LAF in organizations are to:
 - Control the complexity of the IT Application landscape by preventing uncontrolled growth and duplication of functional capabilities within the organization,
 - Deliver architectural paths enabling business and technological growth within the organization,
 - Keep the Component stack up to date, preventing technical degradation, legacy systems, and technical debt,
 - Embed organization goals and mission into the software development life cycle, providing a better understanding of them across the organization,
 - Support organizational flexibility for adoption, resiliency to demanding markets, and continuous competition,
 - Keep costs at an optimal level, and
 - Protect the organization from technical bankruptcy.

LAF acknowledges the fact that today's organizations are under constant pressure from demanding markets and competitors. That is why organizations need to undergo continuous transformation instead of relying on big transformational projects that can hardly keep up with a changing market. As a result of properly managed IT architecture, organizations become cost-optimal with a consistent environment ready for further growth and extension, enabling business momentum.

Every organization has its own software delivery process (SDLC)—whether based on Agile methodologies like Scrum, XP, LeSS, SAFe, etc., or more traditional approaches like Waterfall—but all of them need architectural guardrails to stay consistent and be ready for change.

As the pace of building AI-enabled solutions accelerates, a solid architectural description becomes a critical component of their implementation, ensuring a consistent and harmonized environment.



3. Elements

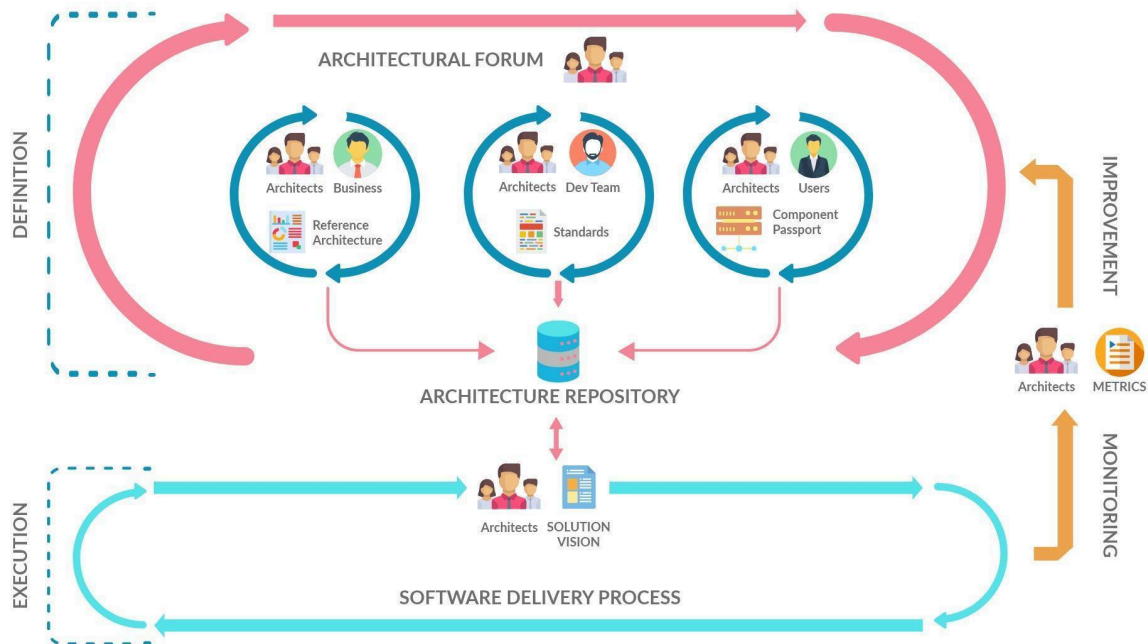
LAF consists of three main parts:

- **DEFINITION** - This is the part in which architects collaboratively create artifacts with other stakeholders in the organization as the foundation for architectural implementation. In this phase, the following are created:
 - **REFERENCE ARCHITECTURE (RA)** - The desired state for the Application portfolio designed to support a changing business.
 - **STANDARDS (ST)** - Standards that maintain the Application environment in a state that enables predictable management.
 - A set of **COMPONENT PASSPORTS (CP)** - This set describes the current state of the Application portfolio, facilitating decisions regarding whether an Application is suitable for further development.
- **EXECUTION** - This part of the architecture is implemented through the active participation of architects in the software development life cycle (SDLC). Architects describe a vision of the solution that responds to business Requirements while ensuring compliance with architectural guidelines.
- **IMPROVEMENT AND MONITORING** - This is the smallest part of LAF, yet it has a significant impact on overall architecture across the organization. Its role is to observe the quality of architecture and architectural processes, providing input for continuous improvement in those areas.

A foundational component common to all elements is the Architecture Repository (REPO)—a central base where individual artifacts are stored. It serves as a key source of information for decision-making when defining solution visions.

The entire architecture is managed and built by a group of architects structured as a formal team working together through regular meetings called the Architectural Forum. Effective teamwork is the single most important factor affecting overall architecture quality. Therefore, LAF also introduces work organization practices.

The overall LAF framework is illustrated in the diagram below.



Lean Architecture Framework Footprint

3.1 Artefacts

3.1.1 REFERENCE ARCHITECTURE

It is a long-term, high-level architectural vision for a particular business area upon which all subsequent solution visions should be created. It serves as a direction and guideline for architectural decisions—a "to be" state that may never be fully finalized. Reference Architecture may adapt over time due to evolving business needs and market demands.

It is a lightweight design of the desired state of the application environment to strive for in the long term. Reference Architecture consists of diagrams illustrating Components, Capabilities, and interactions between them. It should also contain a brief description of each Component and a benefit hypothesis resulting from its implementation. It does not detail intermediate states, implementation roadmaps, specific staffing allocations, or exact budget figures.

It is created in close collaboration with business units and relevant stakeholders, prioritizing their core interests. Clear, understandable language must be established, ensuring Reference Architecture maintains a consistent structure. It incorporates business trends as well as technological developments, aligning with industry standards where available. It unlocks full business potential by addressing key goals and securing alignment with company mission and strategy, setting a solid technical base for organizational growth. Aggregating all reference architectures provides a comprehensive view of the enterprise architecture as a whole.



Reference Architecture defines the goal capabilities of the Application environment for future architectural changes. Each Application implementation should bring the overall environment closer to achieving this goals. Reference Architecture continuously evolves alongside market shifts and organizational changes. Continuous adaptation in pursuit of a dynamic goal is at the heart of Continuous Transformation. This model is illustrated in the diagram below.

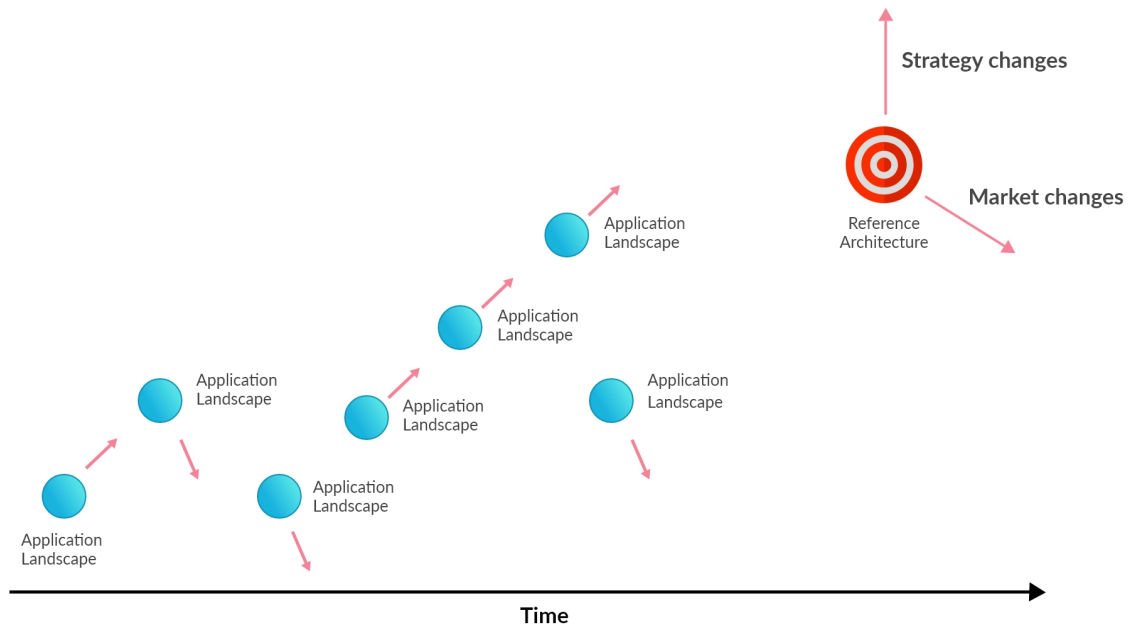


Diagram showing abstract aspirations for reference architecture

Creating a Reference Architecture usually focuses on a specific business unit. An Architect is responsible for developing the Reference Architecture in close cooperation with the business unit, prioritizing their interests by establishing workshops to ensure a thorough understanding of domain needs.

When creating Reference Architecture, architects have the opportunity—and are required—to propose innovative solutions and assist business partners in implementing innovation.

Creating and maintaining reference architecture is an ongoing effort that depends on various factors, such as learnings from implementation, changes in business and technology strategies, shifts in technical layers or legal regulations, and market pressures. For these reasons, Reference Architecture should be adjusted accordingly and reviewed at least once a year to maintain its vitality.

3.1.2 STANDARDS

Efficient and predictable management of an IT environment requires Application architecture to be coherent on multiple levels. Maintaining standardized elements across the Application environment reduces complexity and speeds up decision-making in standardized areas. Standards create a safe environment for development and maintenance teams to innovate.



Order is maintained by establishing formal artifacts—such as published documents or wiki pages—that outline standards, constraints, and required models for implementing functionality within the IT environment.

Standardization should be kept to the minimum level necessary, focusing strictly on essential requirements. The goal is to establish standards that foster a cohesive environment without limiting innovation.

The key types of standards that should be established are:

- technology coherence - limiting technologies within the organization to a defined standard and maintaining version consistency across elements;
- coherence of integration - defining and standardizing cross-system integration best practices;
- observability consistency - unifying logging processes and monitoring system/business events across the application environment;
- consistency in data management - unifying data management practices regarding model design, data consistency, and system ownership; and
- AI usage coherence - standardizing SDLC practices for AI integration, operational guardrails, governance, and compliance with applicable regulations.

Creation

The creation and updating of Standards should occur on a cyclical basis (e.g., annually), where a team of architects drafts the standard and aligns it with relevant stakeholders. For instance, specific technology standards for programming languages should be agreed upon with the development teams using them. Standards should be accessible and clearly communicated to all stakeholders, including external vendors.

3.1.3 COMPONENT PORTFOLIO ASSESSMENT (CPA)

Another essential practice is gathering and evaluating information on the current state of Components, which is recorded in the Component Passport.

COMPONENT PASSPORT (CP) - contains key information about the state of a Component. It serves as a concise summary enabling architects to quickly review fundamental details, particularly brief business descriptions and technical metadata. It is populated and refreshed during each CPA practice across three primary categories: business, technical, and financial. This information determines the future direction of the Component and is stored in its Component Passport (CP).

From a business perspective, the Component is evaluated based on its criticality to business processes. The technical evaluation assesses code quality, delivery pipeline efficiency, adherence to technology standards and trends, as well as fundamental technical details such as software stack and versions.



- The financial evaluation reviews costs related to licensing, infrastructure, maintenance, and development. Financial assessments are grounded in organizational cost averages. Ratings can be represented on a simplified 5-point scale rather than absolute monetary values. Lower overall costs correspond to higher financial ratings.

Ultimately, each Application is assigned to one of four lifecycle categories:

- Investment - Applications with high technical quality and high business value targeted for continued growth and investment.
- Tolerance - Applications with high technical quality but lower business criticality that remain maintained due to low operational costs and stable technical health. These systems represent potential foundations for future expansion.
- Migration - Applications with low technical quality but high business criticality targeted for capability migration to preferred systems.
- Elimination - Applications with poor technical health and redundant or obsolete capabilities targeted for decommissioning.

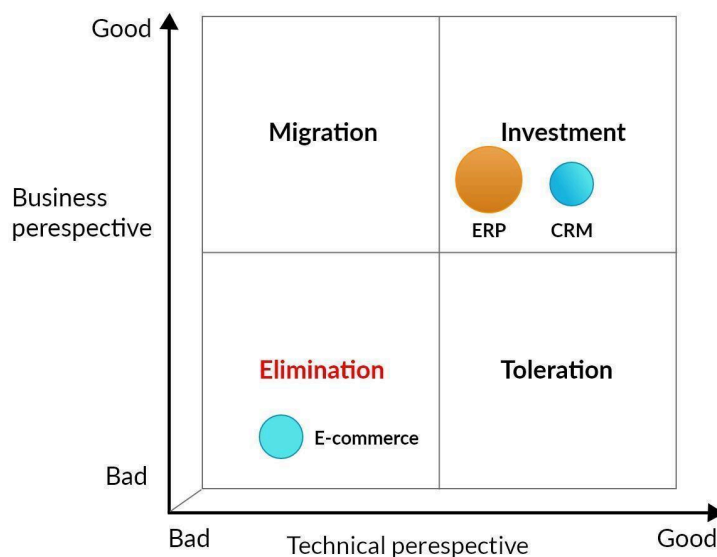


Diagram showing portfolio allocation for Components. Point size reflects financial evaluation (larger points represent higher financial ratings).

Portfolio categorization is critical when deciding which system to prioritize for development when redundant capabilities exist across Components.

Updating the Component Passport should be a recurring practice. It is recommended to include direct end-user feedback obtained through survey methodologies such as Net Promoter Score (NPS).



Final evaluations for business-critical Components should be thoroughly reviewed, agreed upon, and approved by business owners. Technical attributes of Components should be routinely updated by development teams.

- The assessment process follows these steps:
 - technical evaluation of the Component by the architect;
 - user feedback collection;
 - financial evaluation;
 - business rating by the business owner;
 - Component categorization; and
 - communication of status changes, where applicable.

AI FOOTPRINT - As AI-assisted development becomes part of the SDLC, CPA should also track the proportion of a Component's code or logic generated by AI. This metric supports evaluating long-term maintainability and vendor risk alongside traditional business, technical, and financial perspectives.

3.1.4 SOLUTION VISION

SOLUTION VISION (SV) - A high-level architectural design addressing specific business needs. Architectural updates are embedded directly into business requirements; changes to architecture are never pursued in isolation. SV represents an incremental step within continuous transformation toward the Reference Architecture. Depending on scope, an SV may introduce new Components, reallocate Capabilities across systems, or decommission outdated Applications and technologies.

Involving software engineers in drafting the SV is critical. Strong collaboration between architects, developers, DevOps engineers, and system administrators ensures high architecture quality and successful implementation. Architects foster shared understanding by actively engaging in day-to-day engineering workflows.

Architects work with business partners to define non-functional requirements (NFRs), including security specifications, ensuring that proposed solutions are built on robust technical foundations.

SV implementation:

Creation of a Solution Vision is integrated into the organizational SDLC and triggered by business Requirements. It transforms business initiatives into architectural designs, defining logical components, capabilities, interaction patterns, infrastructure changes, and key NFRs.

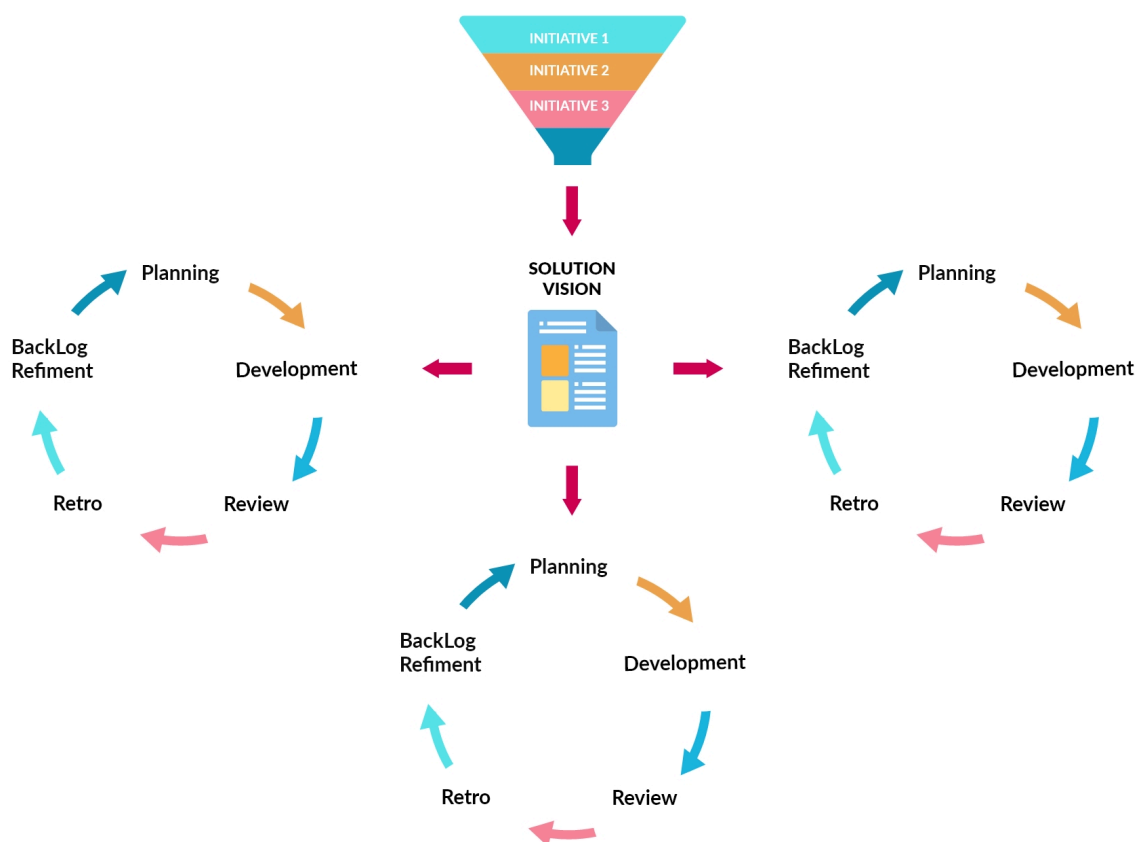
Successful implementation relies on delivering incremental architecture updates (Components, interfaces, capabilities) aligned with the Reference Architecture, Component



Passports, and Standards. This iterative approach delivers faster business value, validates benefit hypotheses early, and avoids the risks and costs of large-scale, disruptive transformations ("big bang" projects).

Architecture changes are valuable only when they deliver clear organizational outcomes. Consequently, changes should be limited to what is necessary for economic value and technical growth, staying closely tied to the enterprise SDLC.

For example, within a Scrum-based development framework, the Solution Vision acts as an input for engineering teams. The architect creates the overall solution vision for an initiative, which is then refined and executed across application teams, as illustrated below.



An abstract illustration of positioning the Solution Vision within the SDLC

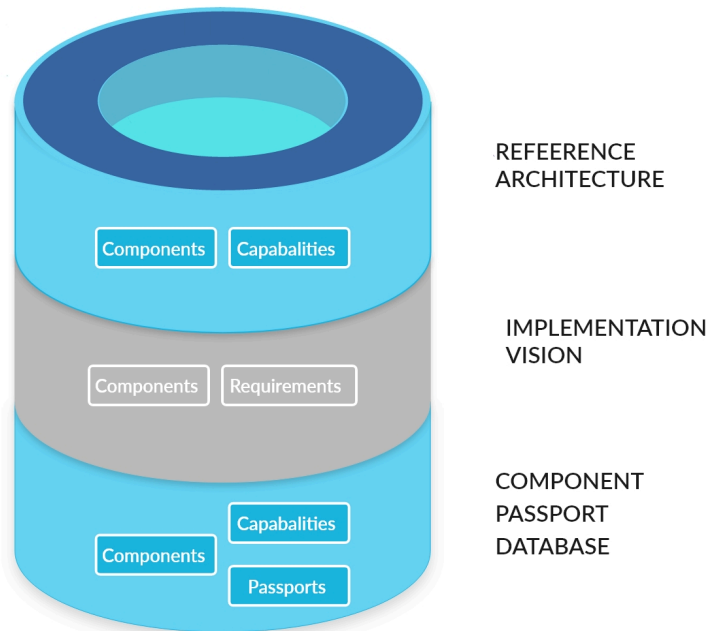
3.1.5 REPOSITORY

All LAF artifacts are stored in a central Architecture Repository. This database captures details regarding:

- future state - the Reference Architecture along with expected capabilities;
- delivered capabilities - active requirements and related Solution Visions;
- current status - Component attributes recorded in Component Passports.



ARCHITECTURE REPOSITORY



Ref. Graphic abstraction REPOSITORY

The shape of the repository, including the list of Application properties, presentation of capabilities and Solution Vision depends on the organization and should be optimized by the continuous improvement process. The repository should be built in a way that allows flexible changes in it. The content of the repository should contain a minimum set of data needed to classify the components or re-design the reference architecture. This is necessary due to the need to have current data in REP and the too expensive cost of obtaining them will have a huge impact on their quality.

3.1.6 METRICS

While the real transformation towards reference architecture is made in an incremental model by “Solution Vision”, there is a need for metrics to make sure that the organization stays on the right track. It is extremely important, especially in the continuous transformation company development mode.

Tracking architectural indicators allows organizations to:

- verify that tangible architectural changes are occurring;
- understand how business decisions impact technical strategy; and
- make deliberate decisions regarding trade-offs between speed, business value, and architectural health.



Metrics provide key data for retrospective reviews and are typically evaluated quarterly, subject to organizational needs.

Thresholds should be defined for each metric. Threshold breaches should prompt review, retrospective analysis, and corrective action.

Reference Implementation Metric (RIM) - Solution Visions should align with standards, CPA insights, and Reference Architecture targets. When temporary trade-offs occur due to urgent business timelines or regulatory constraints, compliance status is recorded using three categories:

- "Compliant" - The implementation fully aligns with Reference Architecture targets, incrementally advancing system components, capabilities, and interfaces toward desired states.
- "Neutral" - The change does not alter overall architecture, core interfaces, or standards (e.g., minor functional patches or UI tweaks).
- "Not compliant" - The implementation deviates from Reference Architecture, standards, such as extending unapproved legacy components, introducing non-standard technologies, or creating unapproved component dependencies.

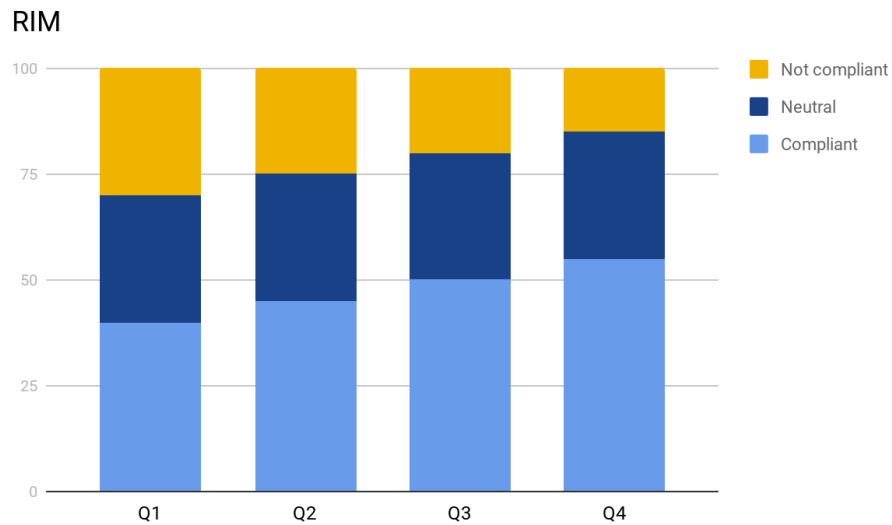
The formula for calculating RIM is defined as:

$$RIM = \frac{CV}{AV}$$

CV = Number of compliant Solution Visions

AV = Total number of Solution Visions

RIM tracks architectural alignment over defined timeframes (e.g., quarterly). Monitoring these trends enables leadership to adjust transformation investments toward fully compliant initiatives.



An example of reporting the status of the RIM metric

Application Reference Metric (ARM) - Measures ratio of total applications to strategic applications within the CPA portfolio.

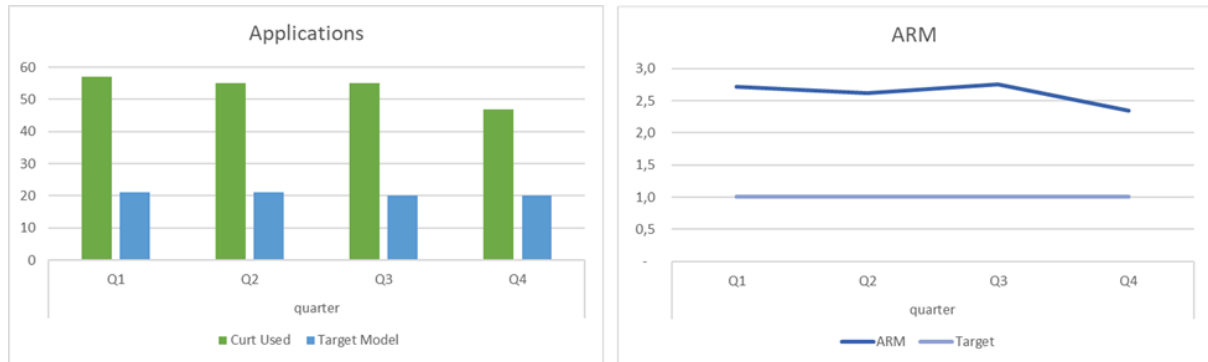
ARM tracks component portfolio health, evaluating whether capital is directed toward strategic applications while legacy or low-value systems are systematically decommissioned.

$$ARM = \frac{AP}{IA}$$

AP = Total number of applications in CPA

IA = Number of "Investment" and "Tolerance" applications

For example, if an enterprise maintains 100 total applications and 50 are classified as "Investment" or "Tolerance", the resulting ARM ratio is 2.0. Over time, this metric should converge toward 1.0.



An example of reporting the status of the ARM metric

Implemented Capabilities Redundancy Metric (ICRM) - Evaluates capability redundancy by comparing active capability instances against target capability counts defined in the Reference Architecture.

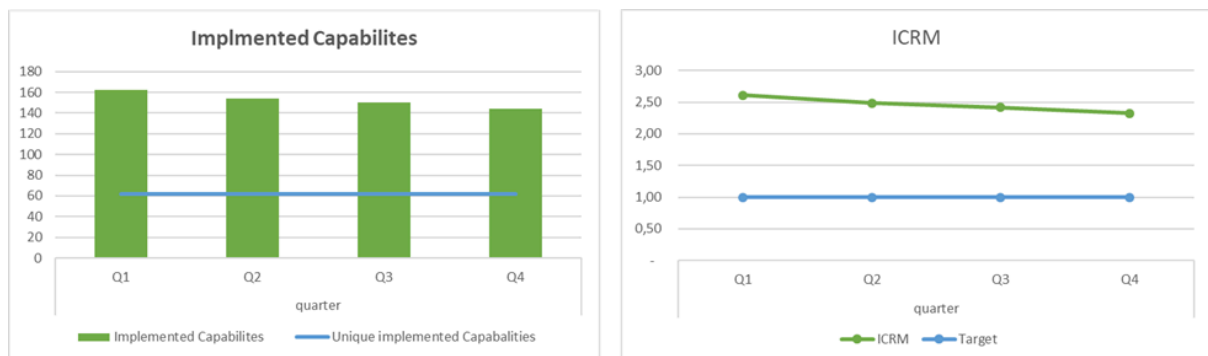
ICRM ensures duplicate functional implementations remain within controlled parameters. While target ratios often aim for 1.0, controlled redundancy may be accepted for specific architectural requirements.

$$ICRM = \frac{IC}{TC}$$

IC = Number of implemented capability instances

TC = Target capability count

For example, if 6 instances of an "SMS Notification" capability exist across systems when the target architecture requires 2, the resulting ICRM coefficient is 3.0. Over time, the ratio should approach 1.0.



An example of reporting the status of the ICRM metric



Capability Reference Metric (CRM) - Measures progress toward target capability coverage by comparing currently utilized capabilities against target capabilities proposed in the Reference Architecture.

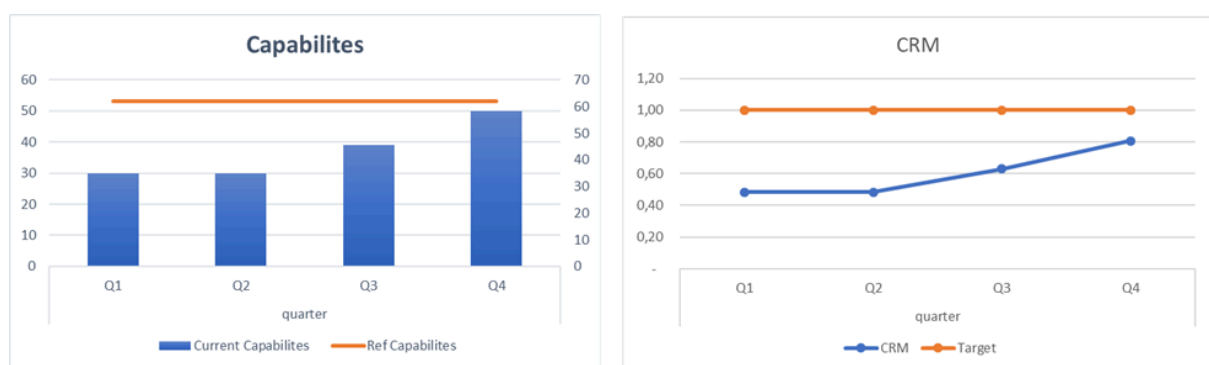
CRM verifies whether active initiatives introduce required strategic capabilities into the environment.

$$CRM = \frac{UC}{TC}$$

UC = Number of utilized capabilities in current environment

TC = Total capabilities proposed in Reference Architecture

For example, if 46 of 100 target capabilities are implemented, the CRM metric is 0.46. Measured quarterly, this indicator should gradually approach 1.0 as the environment matures.



An example of reporting the status of the CRM metric

3.2 Architectural Forum

Effective architecture requires seamless communication across architectural roles. To maintain autonomy, architects must understand how local decisions impact the broader enterprise context. LAF establishes the Architectural Forum as the primary collaboration platform. Held weekly for 2 to 4 hours, this forum allows architects to present findings, align on Architecture Repository schemas, review Reference Architecture drafts, approve new Components and capabilities, evaluate Solution Visions, and update organizational standards.

The Forum also serves as a planning platform for governance tasks, such as maintaining the Repository, facilitating CPA reviews, managing Reference Architecture coverage, and updating enterprise standards. Solution Vision drafting is managed directly within the enterprise SDLC rather than scheduled during Forum sessions. Overall, the Forum strengthens cross-domain awareness and promotes reuse of Components, capabilities, services, and technologies.



3.3 THE TEAM

The Architects team has a flat structure and consists of people with deep technological knowledge and a strong understanding of the business context. Team members may have different roles related to technology and business domains. The team is self-organizing, which helps optimize its efficiency and maintain a high level of communication. Team members have significant autonomy and are fully accountable for delivering LAF artifacts. They share the same norms, rules, and standards.

The team is supported by a Master Architect who ensures that the team follows LAF practices and rules. The Master Architect facilitates collaboration, creates an environment for open discussion, supports knowledge exchange, helps remove impediments, and maintains the quality of architectural practices, rules, and the Repository. The role is analogous to that of a Scrum Master.

In organizations where AI agents participate in the SDLC, the Architects team is responsible for maintaining the quality and currency of the machine-readable representations of the Reference Architecture and Standards used as context for those agents.

The Master Architect ensures that the organization adheres to LAF principles through coaching and training. The Master Architect also helps the organization position the Architects team appropriately so that all artifacts and standards are respected.

The master architect should also pay special attention to the cooperation with other units in particular development teams and business units, ensuring that this cooperation is based on mutual respect, common goals and partnership.

Architecture team sizing depends on organizational scale, domain complexity, and SDLC maturity. The objective is to maintain a lean structure that supports efficient software delivery.

In large enterprises with more than 8 architects, the team should be subdivided into domain-focused sub-teams. These groups conduct localized architectural reviews and improvement sessions. Sub-teams select representatives, including their Master Architect, to participate in enterprise-wide forum meetings. In such setups, a Lead Master Architect is appointed to coordinate across sub-teams.

4. LAF setup

4.1 Best practices

4.1.1 Common language



A critical success factor for LAF implementation is establishing a shared vocabulary across business leaders, developers, and architects. Clear common language enables productive architectural discussions across non-technical and technical stakeholders.

This vocabulary should remain concise, grounded in plain definitions of Applications, Capabilities, and domain targets.

Domain-Driven Design (DDD) concepts are strongly recommended as a foundation for establishing a ubiquitous domain language.

4.1.2 Division

Large enterprises operate complex business models with diverse product and service lines. In such environments, Reference Architecture should be segmented into domain-specific reference architectures mapped along value streams or organizational units.

4.1.3 Architecture Log

Organizational turnover and evolving business models require a persistent decision history. Maintaining an architectural knowledge base ensures continuity and strategic alignment over time.

The Architecture Log acts as a decision repository tracking key events, such as:

- procurement decisions for new technologies, Applications, or Components;
- approved exception to the Reference Architecture or Standard;
- publications of new Reference Architecture versions or Standards; and
- decommissioning approvals for Applications and Components.

4.1.4 Continuous Improvement Sessions

Architects should hold bi-weekly or monthly continuous improvement sessions lasting 1.5 to 2 hours. Retrospective methodologies widely adopted in Agile frameworks should be adapted for these sessions. Architects also participate in standard SDLC retrospectives, bringing key engineering insights back to the architectural continuous improvement meetings. Example outcomes include updating component evaluation or refining SV document templates.

4.1.5 Fast Tracks

Not every initiative requires a full Solution Vision. Minor updates confined to a single Application that do not alter core capabilities should follow a fast-track approval process.

4.2 Anti-Patterns

4.2.1 Rigid Target Architecture

A common pitfall in LAF adoption is misinterpreting Reference Architecture as a rigid, multi-year transformation master plan complete with fixed schedules and budget estimates. Such rigid plans struggle in dynamic market conditions and are frequently abandoned at the



first delivery obstacle, jeopardizing overall governance. LAF advocates for incremental progress toward an adaptable reference architecture that evolves with business conditions.

4.2.2 Deep design

Architects who are not developers of that Application and do not work directly on implementation should not design the implementation of functionality in a given Application. Teams of programmers are responsible for the technical design. The teams agree on specific functionalities with business. Architects should build the basis for implementing functionality in the form of API guidelines, integration methods, and major technological changes. The most efficient model for implementing architecture is the participation of architects in the life of development teams. This participation depends on the type of project and its stage. At the beginning of the project, we recommend the Architects should closely collaborate with the development team. In the later stages, the Architect should support the team as needed.

4.2.3 Architects out of SDLC

When architects operate in isolation from development pipelines and business units, they lose visibility into system states and real-world delivery constraints. Consequently, application teams implement localized solutions without broader context, prioritizing short-term fixes over long-term stability. Architects must remain embedded in the SDLC to guide sustainable technical strategy.

4.2.4 Architecture as a bottleneck

Excessive documentation standards and bureaucratic approval processes turn architecture into an operational bottleneck, leading teams to bypass governance. Repositories and artifacts should adhere to lean data principles, capturing only essential details required by stakeholders.

4.2.5 Architecture blind to AI

If Reference Architecture and Standards exist only as documents intended for people, while development teams rely on AI to generate code, the architecture effectively stops being applied - because an AI agent does not consult a wiki or a document intended only for human consumption while generating code. Warning signs of this anti-pattern should be observed in Metrics.

4.3 Tools

LAF principles are tool-agnostic and can be managed using basic spreadsheets. However, automating workflows and centralizing repository storage improves analytical capabilities. Recommended tooling includes:

- a centralized repository database supporting multi-user concurrent access especially AI agents
- an architecture portal hosting standards, logs, and reference architectures;
- automated survey tools for capturing end-user feedback;
- automated integrations with static analysis tools and issue tracking systems; and



- automated metric aggregation platforms.

5. LAF for AI-augmented SDLC

This section addresses the architectural implications of AI participation in the SDLC. It does not define organizational governance for AI models, AI services, or other technologies; these remain outside the scope of LAF.

LAF assumes that traditional transformation bottlenecks stem from human decision-making velocity, focusing on optimizing interactions between architects, business units, and developers. When AI agents execute substantial portions of the SDLC, the primary bottleneck shifts: human-readable artifacts (RA, ST, CP) are ignored by automated coding tools unless transformed into machine-consumable context. The following additions extend LAF to maintain effective governance in AI-assisted SDLC environments.

5.1 Machine-Readable Architecture Context (MRAC)

Alongside human-readable RA, ST, and CP artifacts, organizations using AI-assisted SDLCs should maintain structured, machine-readable counterparts (e.g., structured JSON/YAML schemas) to inject into AI developer agents as system context. MRAC includes condensed Reference Architecture mappings, machine-validatable Standards rules, and key Component Passport data. Human-readable documents remain the primary source of truth, while MRAC serves as a synchronized derivative.

5.2 AI-Assisted Solution Vision

Using MRAC and business requirements, AI agents can generate initial drafts of Solution Visions. The architect's role evolves from primary author to reviewer and approver. Within Fast-Track pipelines, AI-generated changes confined to single components that do not change its capabilities can be automatically validated against Standards and Reference Architecture, with RIM attribute computed programmatically. Non-compliant or ambiguous cases should be escalated to the Architectural Forum.

Appendix. Changes between versions 1.5 and 1.6

Version 1.6 extends LAF to support software development lifecycles where significant tasks are performed by AI. All additions listed below are proposed updates tracked against version 1.5 for individual review.

- **New section:**



- "5. LAF for AI-augmented SDLC" - introduces Machine-Readable Architecture Context (MRAC) and AI-Assisted Solution Vision (5.1–5.2).
- **Changes to existing sections:**
 - 1. Introduction - noted that AI agents increasingly participate in the SDLC, as detailed in Section 5;
 - 2. Why good architecture is needed - added a paragraph explaining why architecture serves as explicit context for AI agents rather than solely an after-the-fact checkpoint;
 - 3.1.2 Standards - added "AI usage coherence" as a fifth standard type covering approved AI models/agents, required MRAC context, and human review requirements;
 - 3.1.3 Component Portfolio Assessment - added "AI Footprint" as an assessment perspective alongside business, technical, and financial views;
 - 3.3 The Team - extended Master Architect responsibilities to include maintaining MRAC quality and currency; and
 - 4.2 Anti-Patterns - added "4.2.5 Architecture blind to AI".
- **Grammatical, linguistic, and orthographic corrections throughout the document**