LEANARCHITECTURE FRAMEWORK

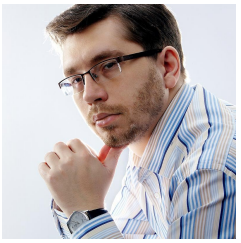
for Continuous Transformation

Version 1.3

Authors

Tomasz Michałek-Czerepak **Tomasz Świerszcz**

Cooperate



Dr. hab. Andrzej Sobczak, prof. SGH

Glossary

LAF uses four basic concepts that have many meanings depending on the context in the IT world. Those 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 expression or the definition of the Application's or Component ability to fulfill an organization core functions' performance. Mostly it is the main role and functionality of the Application or Component.
- **Requirements** - the requirement that the business defines for Applications, Components.

Although the Component is a more general definition, the following 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 it easier to receive this study.

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.

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 consistent form. It does this by appropriately placing the Architect in the software development life cyclce. The architect with the right knowledge can quickly make good decisions.

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 businesses by building agreements between them striving to build a lasting healthy relationship between them. Thanks to this, it allows the personal development of architects working in the organization, which in return has a direct impact on making good decisions regarding the IT environment.

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

The main characteristics of LAF are:

Simple	LAF is easy to understand and short
Pragmatic	LAF concentrates only on activities that provide values 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 complex IT architecture environment, it is full-fledged solution which contains proven guidelines to support it
Enables Architectural Agility	LAF support constat respond to a current situation of organization and adapting to the changing business environment

2. Why good architecture is needed

Every large organization needs to manage IT architecture, especially the Application landscape to stay cost-efficient, flexible and open for a change, not just as a single business area or Application but as a whole organization.

LAF framework is responding to these needs providing full guidelines by defining roles, events and artifacts.

The main goal of implementing LAF in organizations is to:

- Control complexity of IT Application landscape by preventing it from uncontrolled growth and duplication of functional capabilities within the organization,
- Deliver an architectural path enabling a business and technological growth of the organization,
- Keep an Component stack up to date, preventing it from technical degradation and become a legacy system or/as well as technical depth,
- Embed organization goals and mission into software development life cycle providing a better understanding of them across the organization,
- Support organization flexibility for adoption, resiliency to demanding markets and continuous competition,
- Keep cost at the optimal level,
- Protect the organization from a technical bankruptcy.

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

Every organization has their own process of delivering software SDLC, either it is based on Agile ways of working like Scrum,XP, Less, SAFe etc. or more traditional approaches like waterfall, but all of them need architectural guardrails to stay consistent and to be ready for changes.

3. Elements

LAF consists of three main parts:

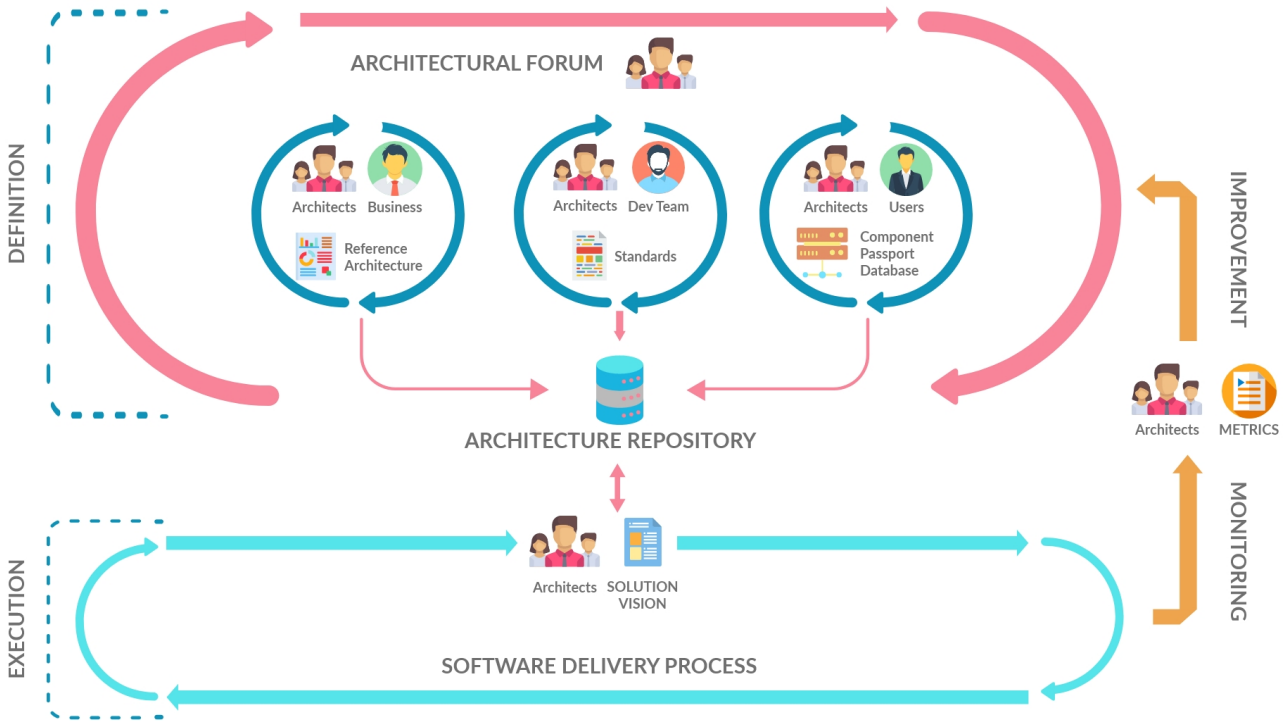
- **DEFINITION** - This is the part in which architects with other people in the organization create collaboratively artefacts that are the basis for the implementation of architecture. In this part are created:

    - **REFERENCE ARCHITECTURE (RA)** - The desired shape of the portfolio of Applications that are to support a changing business.
    - **STANDARDS (ST)** - Standards that allow you to maintain the Application environment in a state that allows predictable management
    - **COMPONENT PASSPORT DATABASE (CPD)** - Current state of the Application portfolio that facilitates making decisions about whether the Application is suitable for further development.
- **EXECUTION** - this part of the architecture is implemented through the participation of Architects in the software development process (SDLC). Architects describe a vision of the solution that responds to business Requirements and at the same time complies with the architectural guidelines.
- **IMPROVEMENT and MONITORING** is the smallest part of LAF with significant impact on the whole architecture in the organization. Its task is to observe the quality of architecture and architectural processes and should be used as an input to continuous improvement in those areas.

A common part for all elements is Architecture Repository (REPO) - the base in which individual products are placed. It is also a source of information for making decisions when creating a vision of a solution.

The whole architecture is managed and built by a group of architects gathered in a formal team that works together through regular meetings called the Architectural Forum. The effective work of this team is the most important factor affecting the quality of architecture. Therefore, LAF also introduces work organization practices.

The whole of LAF 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 of a particular business area, based on which all the further solution visions should be created. It's a direction and a guideline for architectural decisions, a "to be" state which most likely will be never achieved. Reference Architecture might be changed before its final implementation due to the changing business and demanding markets.

It is a lightweight design of the desired state of the application environment that will be strived for in the long term. Reference Architecture consists of diagrams with 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 cover any of the states in between or information about how it will be achieved and does not cover which resource will be involved or how big budget should be allocated.

It is created in close collaboration with a business unit and other relevant stakeholders, taking their interest in the first place. An easily understandable language needs to be established and Reference Architecture should have a consistent form. It takes into account business trends as well as trends in technology and can be referenced to industry architecture if available. It enables a full business potential through addressing its goals and needs securing

alignment with company mission and strategy. It sets a technical base for further growth of an organization. Aggregation of all reference architectures creates a view of architecture of the whole organization.

Reference Architecture as a set of capabilities of the Application environment is the goal for architectural changes. Each Application implementation should bring the whole environment closer to achieving this goal. Reference Architecture is changing due to market and internal organization changes. Constant changes in the pursuit of a changing goal are at the heart of Continuous Transformation. The whole is illustrated in the diagram below.

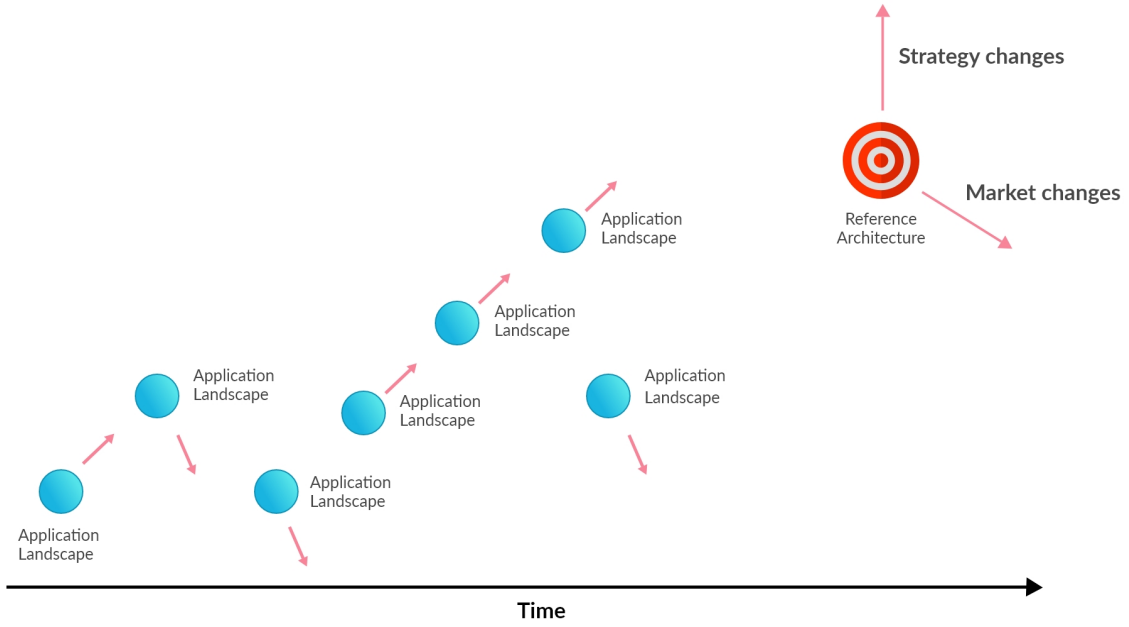


Diagram showing abstract aspirations for reference architecture

Creating a Reference Architecture is usually focused on a particular business unit. An Architect is responsible for creating Reference Architecture with close cooperation with a business unit, taking their interest in the first place by setting a series of workshops to be sure of achieving a full understanding of business domain needs.

When creating Reference Architecture, architects have the opportunity and even more are required to propose innovative solutions and support business in implementing innovations.

Creating and maintaining reference architecture is a continuous work and may depend on various factors , such as learning from implementation, changes in business and technological strategies, changes in the technological layer or legal regulations, the market pressure etc. That is why Reference Architecture should be adjusted accordingly, however it should be at least reviewed once a year to maintain its vitality.

3.1.2 STANDARDS

Efficient and predictable management of the IT environment requires that the Application architecture should be coherent on many levels. Maintaining unified elements in the Application environment allows not only to reduce chaos in architecture but also to speed up decision making in areas covered by standardization. Standards create a safe space for experiments for teams involved in software development and maintenance

The way to maintain this order is creating formal artifacts e.g. published documents or wiki pages describing these standards, restrictions and required models for implementing functionality in the IT environment.

This standardization should be carried out at the minimum level needed and limited to the most important requirements. The goal is to find a set of standards that will create a coherent environment while not reducing innovation.

The most important types standards which should be created are:

- technology coherence - limiting the amount of technology in an organization to a defined list and maintaining consistency as to the version of all its elements
- coherence of integration - standardizing and defining good cross-system integration practices,
- monitoring and logging consistency - unifying the logging process and monitoring system and business events in the application environment
- consistency in data management - unification of the way data is managed in terms of building models, their consistency and the responsibility of systems for this.

Creation

The creation and updating of Standards should be based on a cyclical periods eg. once a year in which a team of architects prepares the content of the standard, which is later agreed with interested parties. For example, details of technology standards for the used programming language should be agreed with development teams that use the technology. Standards should be available and communicated to all interested parties including suppliers.

3.1.3 COMPONENT PASSPORT DATABASE (CPD)

Another very important artifact is a database containing basic information about the current state of Components. Collected in 3 basic categories: business, technical, financial. This information is gathered to determine what the future of the Component should be.

In the business perspective, the Component is assessed for its importance in business processes. The technical category assesses the quality of the code, the quality of delivery processes and the timeliness of technology with standards.

The financial perspective is an assessment of the cost of Components in the areas of licenses, infrastructure, maintenance, development costs. The financial context of the assessment should be based on the average values of the organization. When calculating the rating, the

result can be based on, for example, a 5-point scale, not necessarily based on real values. The lower the costs are the better the app is.

In the end, each Application should reach to a specific of 4 groups of destination:

- Investment - Applications rated technically high and having great business significance are intended for further development and investments.
- Tolerance - Applications rated technically high but with little business significance, due to their low cost and good technical condition, should remain in the environment until their business significance and technical rate is maintained. They could be potentially good base for further development and investments
- Migration - Applications rated technically low but with high business significance are intended for a transfer of capabilities to Applications form a previous groups
- Elimination - Applications in a poor technical condition with functionalities that are irrelevant or other Applications cover them should be intended for termination

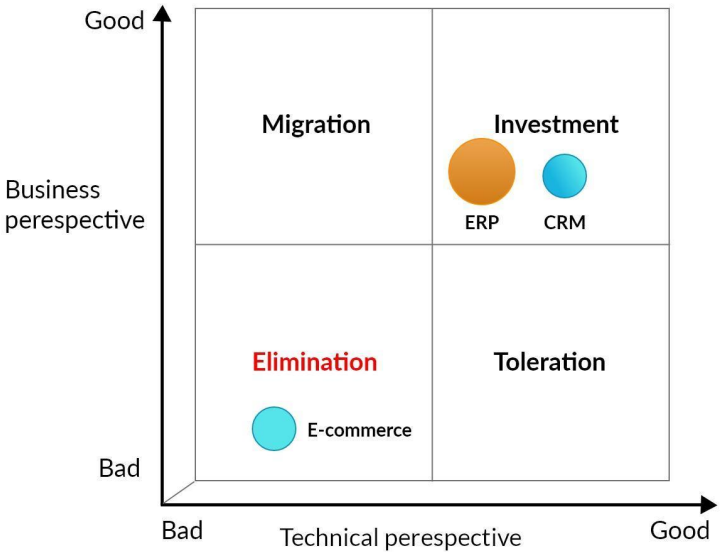


Diagram showing how to allocate Components to portfolios. The point size illustrates the financial perspective of each Component.(The larger the point is the better the Component is)

The assignment to the category is important when comparing which of the two Components development should occur if the two Components have redundant Capability.

Completing the database should be a cyclical event, it is recommended that one of the Application assessment component is an input from the end-users perspective made through surveys regarding the Application. A good way is to use the Net Promoter Score (NPS)

The final assessment of business-critical Components should be discussed in detail, agreed and approved by the business unit. Technical features of the Component should be updated on a regular basis by the development team

The steps should be performed:

- technical evaluation of the Component by the architect
- Application user rating
- financial evaluation
- business Component rating by the business owner
- Component categorization
- category change communication, if applies

3.1.4 SOLUTION VISION

SOLUTION VISION (SV) - It is a high-level architectural design which responds to current business needs. The changes in the architectural layer are built into those Requirements. It always comes with business value, never alone as changes in architecture are never the target by itself. It is a part of business change, step by step moving towards a Reference Architecture. It is a true heart of incremental continuous transformation of organization in the architectural landscape. In some cases it might contain changes like a new Component, a shift of capabilities from one Component to another, decommissioning Application/Components/technology/capability and others like migrations etc.

One of the most important elements of a good SV is involvement of developers in its creation. Effective cooperation between the architect and developers/ devops/admins is crucial to ensure high-quality architecture and effective implementation. Understanding the assumptions presented by the architect should be built through the participation of the architect in the daily life of the development team.

In the context of building solutions, the architect with business is responsible for the creation of high-quality non-functional requirements, including safety requirements. The architect should be sure that the solution he is building is based on good NFR

SV implementation:

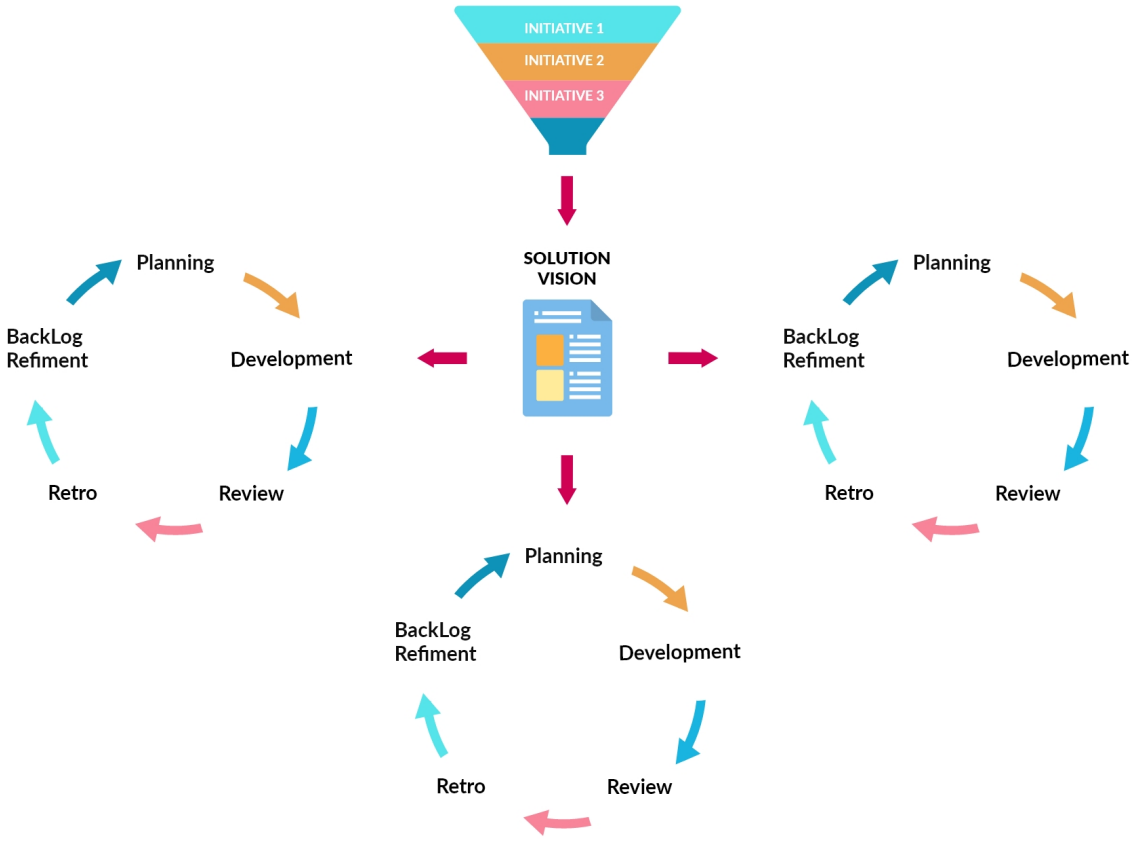
Creation of Solution Vision is built into organization SDLC and it is always triggered by current business Requirements. It transforms business initiative into an architectural design of one or more applications showing them as logical components with their capabilities and interaction between them. It should cover all changes in the hardware layer and define Non Functional Requirements for business initiative as they are major inputs for a good architectural design.

The key for a proper implementation of it is to do incremental changes in the architectural landscape providing new Components, interactions, capabilities step by step with a full compliance to a Reference Architecture, information form CPD, Standards. Using this approach organizations have a faster beneficial outcome and a chance to test a benefit

hypothesis which stood behind implementation, having a possibility for faster adoption and learnings, saving money compared to a big transformation project (big bum projects).

The target of an enterprise is not to make changes in the architecture layer by itself , but to make changes which bring value for the organization. That is why we should only make as many of them as they would be sufficient for a proper economic outcome and secure technological growth of the enterprise. That is why it has to be tightly combined into enterprise SDLC.

For example, Solution Vision can be introduced into a software development environment based on the scrum methodology in the model in which this document is input for the operation of individual teams. The architect builds a vision of the solution for the entire initiative and it is transmitted to individual application teams. This is illustrated in the diagram below.

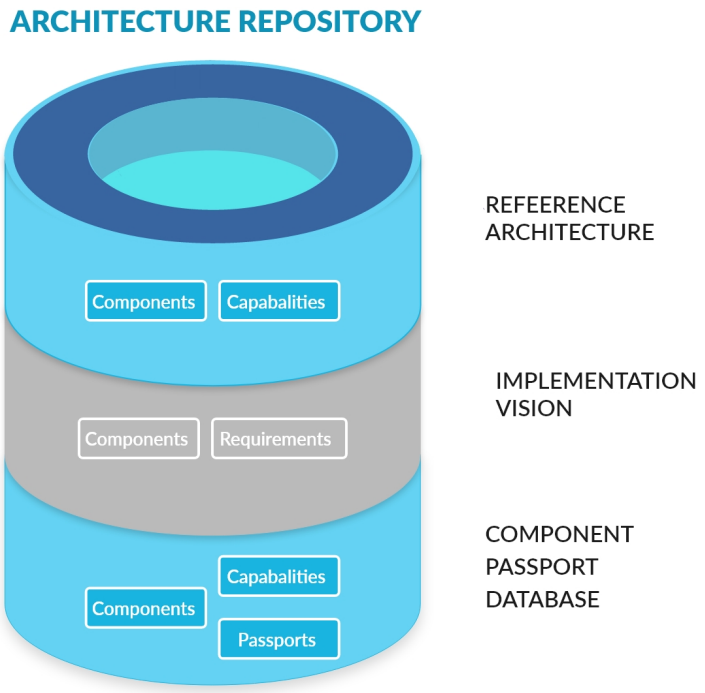


An abstract illustration of placing Solution Vision *in SDLC*

3.1.5 REPOSITORY

Products of all LAF practices should be in one central architecture repository. This database should contain information about the Application in the context of:

- the future - a place in Reference Architecture along with the expected Capabilities
- capabilities it provides - a set of requirements and develop Solution Visions related to this Application
- its current status - Application properties collected in the form of a passport



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.

Monitoring of architectural indicators will enable us to:

- Have a knowledge if organization really changing in architectural layer
- Understand on how business decisions affect architecture
- Make conscious decision on tradeoffs between speed, business value and architecture changes

Proposed metrics are a good base to monitor that progress and a great source for organization and architect retrospections. Usually these metrics are collected quarterly. Time might be adjusted depending on the needs of the organization.

Each of the proposed metrics should have a set threshold for better progress monitoring, any breach of set threshold should lead to an open discussion, retrospections and adjustments.

Reference Implementation Metric (RIM)- each Solution Vision should be aligned and compliant with standards, an outcome of CPD and reference architecture moving towards its fulfillment, but in some cases it cannot be achieved due to time-critical projects, legal regulation etc. That is why each Solution Vision should contain attributes which specify its overall compliance with Reference Architecture. This attribute usually can take one of three options:

- "Compliant" - means "implementation design" which is fully compatible with Reference Architecture and implementation will incrementally transform application landscape towards its desired state designed in a Reference Architecture, especially in terms of used Components, its capabilities and interactions between them.
- "Neutral" - implementation does not influence the application landscape, especially in terms of standards, used Components, interaction between them and their capabilities. It is usually a small change or UX changes.
- "Not compliant" - its when "implementation design" can't fulfill the requirement of being compliant with Reference Architecture. The situation happens especially when some of the legacy Components are developed or updated, not welcome technology are used, new Components or extra interactions are added or any others breaches against Reference Architecture.

This is what the formula for calculating the indicator may look like:

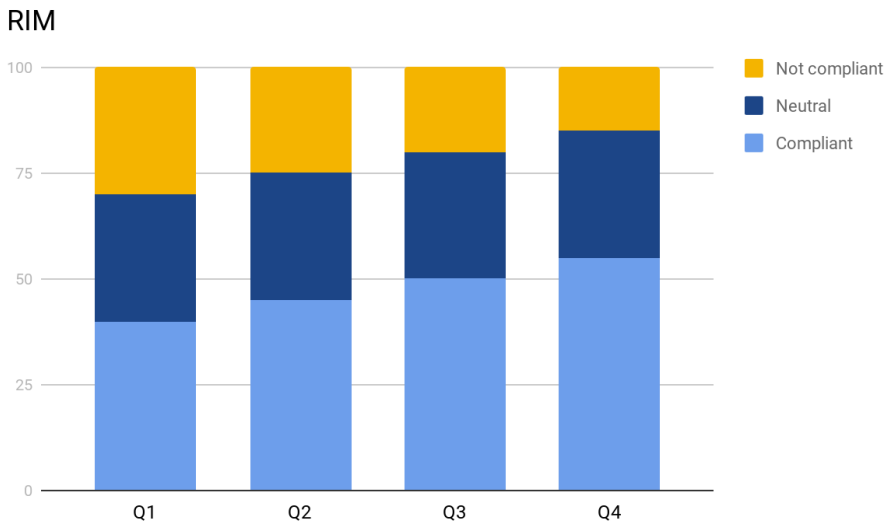
$$RIM = \frac{CV}{AV}$$

CV= Number of complaint Solution Vision

AV= number of all Solution Vision

RIM is one of the key metrics used to track change directions and is tracked as a percentage of each attribute value on the set period of time e.g. once per quarter. You can read from the

graph below a trend of developed changes. Based on it, the organization might adjust the pace of transformation investing more time and funds on “Compliant” Solution Vision .



An example of reporting the status of the RIM metric

Application Reference Metric (ARM) - it is an indicator of the Applications used in the current Application environment of the organization for Applications in CPD with the status "Investment" and "Tolerance".

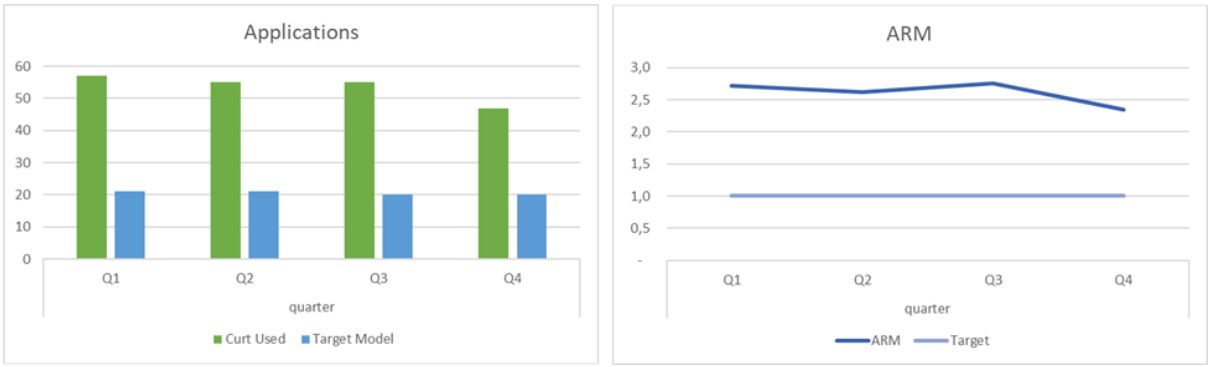
Thanks to the use of this metric, we are able to check the quality of the component in our portfolio. We can check whether applications are created that are worth the investment and those that are weak in terms of business and technology are eliminated.

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

AP =Number of All Application in CPD

IA = Number of invest and Tolerate Applications

For example, if all Applications are 100 in the environment and those with the status "Investment" and "Tolerance" 50 then the ARM indicator is 2. In the long run, this indicator should strive to the value of 1.



An example of reporting the status of the ARM metric

Implemented Capabilities Redundancy Metric (ICRM) - this is a measure of the redundancy of implemented capabilities. The measure is the number of occurrences of capabilities in existing Applications, relative to the target number of capabilities.

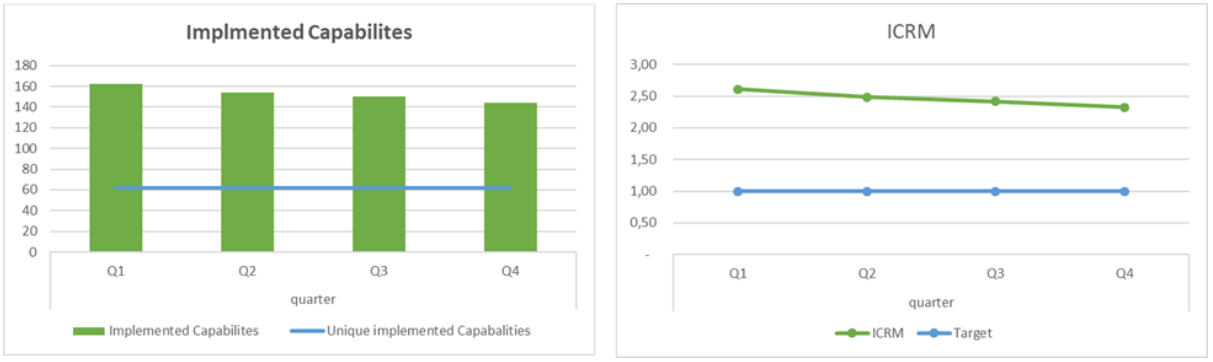
This metric is used to monitor that functionalities are not built redundantly above the expected level. The target value should not always be one. Sometimes it makes sense to have a few capability instances.

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

EC= Number of occurrences of capabilities in existing Applications,

TC= Relative to the target number of capabilities

For example, if 6 simplified "send SMS to Customer" capabilities are implemented in our simplified environment and it should be only 2 times, the ICRM coefficient for such a theoretical environment is 3. The value of the coefficient should go to 1.



An example of reporting the status of the ICRM metric

Capability Reference Metric (CRM) - it is an indicator of the capabilities used in the organization to those proposed in the organization's reference architecture (AR).

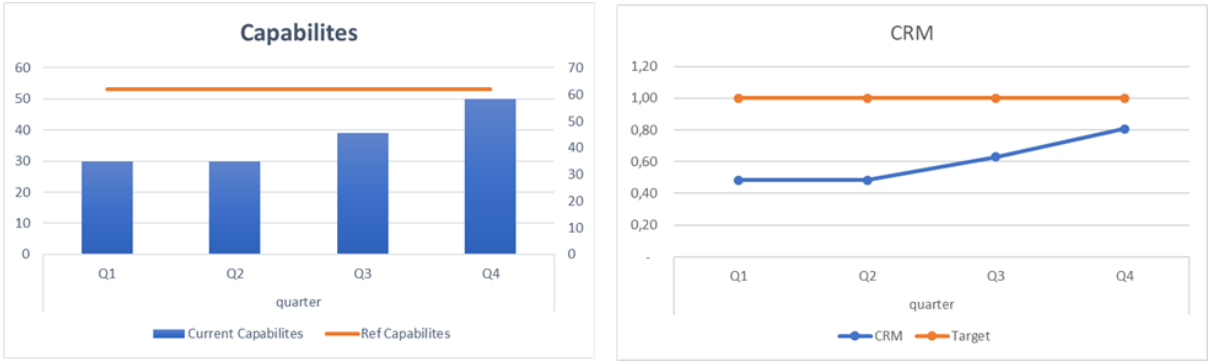
This metric is used to monitor whether the organization is aiming for the expected capabilities landscape. We can regularly check whether the projects / initiatives provide new capabilities

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

UC= Number capabilities used in the organization's current Application environment

TC= Number capabilities proposed in the organization's reference architecture (AR)

For example, if 46 capabilities are implemented in the organization and there are 100 in AR, then the CRM factor should be 0.46. The metric should be measured at a specific time interval, e.g. once every quarter. In the long run, the number of possessed capabilities seek to level with the reference architecture and the indicator to level 1.



An example of reporting the status of the CRM metric

3.2 Architectural Forum

For effective architecture LAF requires great communication between all the architects. To stay independent they need to know the influence of their job on other architects and their areas as well as understand the holistic picture of the whole enterprise. That is why LAF introduced the Architectural Forum as a major communication platform. This meeting should be held at least once per week and take from 2 to 4 hours. The meeting is for architects to present major discoveries made by them, to agree on used language and shape of Architecture Repository, discuss the output of Reference Architecture, agree on a new Components and new capabilities introduced into organization, bring and discuss their doubts regarding

Solution Vision, introduce and modify used standards within an organization and any other major issues which will appear on development phase.

The meeting is also used as a planning session for an architectural task like: maintaining the Repository, conducting APM, managing areas to be covered with Reference Architecture, refreshing an enterprise Standards or introducing new. Architectural Forum is not designed to schedule any work regarding Solution Vision as it is triggered by Enterprise SDLC. The meeting leads to a better understanding of the whole organization by particular architects and leads to the re-use of Components, capabilities, services, technologies.

3.3 THE TEAM

The Architects team has a flat structure, consisting of people with a deep knowledge of technology as well as a deep understanding of the business context, architects should and may have different roles related to the technological and business division. It's a self-organized team which optimizes its efficiency and ensures a high level of communication as it is a key success factor. The team members have high autonomy with full accountability for delivering LAF artifacts. They share the same norms, rules and standards.

They are supported by a Master Architect responsible for ensuring that the architectural team follows LAF practices and rules. Facilitates a collaboration between team members, provides an open environment for an open discussion, leading to exchange of knowledge and concerns, helps to remove impediments and provide a proper quality of architectural practices, rules and repository. Playing a special role in difficult moments such as lack of communication in the team or the impossibility of finding quick compromise as often needed.The Master Architect role is analogous to the Scrum Master role.

Master architect ensures that organization adheres to the LAF principles through coaching and teaching about the framework.

It helps the organization in the appropriate location of the team of architects so 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.

The size of the architectural team depends on the size of the organization, its complexity and maturity of the software delivery process. The goal is to achieve such a minimum composition that will ensure the smooth implementation of the software delivery process.

In large organizations, where the number of architects is greater than 8, it is good to separate smaller area teams. These teams have their own architectural meeting and improvement sessions. Each team should select its representatives including Master Architect to participate

in the meeting about the whole architecture. With such a division, it is worth appointing a Lead Master Architect responsible for the cooperation of architectural teams.

4. LAF setup

4.1 Best practices

4.1.1 Common language

One of the most important factors for the successful implementation of LAF and consequently achieving a good quality architecture is the development of a common language together with business, programmers and other relevant stakeholders. Appropriately built language allows discussions about architecture between business representatives with each other or with people from the IT department.

This language should be simple and based on a simple definition of what an Application and Capabilities are, and how we can define the future of a given business area.

We recommend using the DDD (Domain driven design) concept as a method to develop an appropriate communication language

4.1.2 Division

Most large organizations have a complex business model based on a multitude of products, services or processes. In such organizations, it is worth making reference architecture prepared for a given business area. The entire reference architecture should preferably be divided according to the organizational structure or across value streams and each part should be developed separately. This creates domain reference architecture.

4.1.3 Architecture Log

Due to the fact that the organization is changing in particular, there is a rotation of employees, it is very important that there is a knowledge base about the decisions taken. This allows us to maintain business continuity and pursue the goal consistently.

Architect Log is a database in which important decisions, such as:

- a decision on choosing the purchase of a new technology / Application / Component
- acceptance of an exception to the Reference Architecture or Standard
- AR publication or standards
- elimination of Applications/ Components from the environment

4.1.4 Continuous improvement sessions

Every 2-4 week should be held a cadence based continuous improvement session for Architects. The meeting should take anywhere between 1.5 h to 2h. The retrospection session is well known in Agile Ways Of Working and already very well described in publicly available materials and that is why a particular technique should be adopted form that world and it will not be covered in this material. If an organization already uses an agile approach within an organization, we highly recommend to reach and adapt the techniques which are well known in the organization. The important thing is to notice that architects take part in SDLC retrospections independently from Architects continuous improvement sessions and bring any global discoveries into this meeting. An example session result may be: a change in the component evaluation model, a change in the SV template.

4.1.5 Fast Tracks

Not every initiative should be covered by Solution Vision. Initiatives that are within one existing Application and do not change its capabilities should be implemented along a fast path.

4.2. Anti Pattern

4.2.1 Target Architecture

The main problem and, consequently the failure to implement LAF is to misunderstand what Reference Architecture is. Reference architecture is interpreted as a big vision of transformational change is often called a program with a schedule, budget estimation and other design elements. It becomes a very distant goal, mostly for many years. It entails all the consequences of designing the Application environment in a simultaneously changing business. In most cases, the transformation program is abandoned at the first big problem in the delivery process. Then all architecture is abandoned. LAF supports the gradual pursuit of reference architecture which is changing with the changing business situation.

4.2.2 Depp 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 Architect's 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

If architects are not included in the delivery process and do not work with the business and developers, they are consequently isolated from the knowledge that affects the Application environment and from the state of the environment. In practice, it is not possible then to track and implement the reference architecture. Application teams will shape the environment in a way appropriate to the Application they create without seeing a wider context. A business

working with specific Requirements will focus on achieving short-term goals. Architects must be a part of SDLC to be able to implement changes them in long-term perspective

4.2.4 Architecture as a bottleneck

Too extensive documentation of processes and elements and as a consequence a huge amount of work for architects will make architecture a bottleneck. This will lead to bypassing the architectural aspects very soon. All repositories should be built at the level of minimum compliance with requirements and contain only information needed by stakeholders of architecture and SDLC

4.3 Tools

LAF is a set of good practices that are independent of the tools or lack of them. LAF can be implemented using only a spreadsheet. However, a very good idea is to automate some steps and store data in repositories with greater analysis capabilities. We recommend building / owning:

- A repository in the form of a central database giving the opportunity to work for many people in this application team at the same time. You can start with a spreadsheet in a shared folder
- Having an architecture portal containing information about standards, architecture log, reference architecture
- Applications that automate the collection of surveys from end-users of Applications
- A tool for automating the collection of information about Applications, e.g. from static code analysis systems or bug trackers
- Tools for automating the collection of metrics

Appendix . Changes between version 1.3 and 1.2

Changes related to feedback from implementations:

- Small updates on the description of practices and rules
- Updates on Master Architect role

