Impacts of agility, resilience and Control in Network Enabled Capability on SoS Architecting, modelling and system engineering

Jean-Luc Garnier
THALES LAND & JOINT SYSTEMS
146, boulevard de Valmy
92704 Colombes CEDEX FRANCE
jean-luc.garnier@fr.thalesgroup.com

Dominique Luzeaux
DGA
7-9 allée des Mathurins
92221 Bagneux CEDEX FRANCE
dominique.luzeaux@dga.defense.gouv.fr

Gérard Morganti
THALES LAND & JOINT SYSTEMS
146, boulevard de Valmy
92704 Colombes CEDEX FRANCE
gerard.morganti@external.thalesgroup.com

ABSTRACT

This document presents the SoS Engineering, Architecting and Modelling requirements whose implementation will improve the Agility, the Resilience and the Control in Network Enable Capability (cf. NATO NEC FS Study).

It will also explicit the reasoning schema that allows determining these requirements from the agility, resilience and control needs applied to the distinctive characteristics of an SoS.

1.0 Introduction

1.1 Agility, Resilience and Control definitions

The Gardner group defines Agility as the ability to respond quickly and effectively to rapid change and high uncertainty. Agility can be considered in any activity area; in our case, agility may concern both:

- The activities of the engineering area: requirements, design, development and IVV,
- The activities occurring during the exploitation phase, which include the activities of the operational area (related to the SoS Behaviour itself, including the technical but also the social aspects), of the logistics area and of the system management area.

Resilience can be defined as the ability of a given system to go on processing when a failure occurs. The failure can be due either to a malfunction or to an attack.

So resilience resides in the ability to analyse the failure and the ability then:

- either to make a rollback to a previous correct state and restart the system,
- or to determine an appropriated degraded mode and go on running in that degraded mode.

Thus resilience concerns the activities of SoS system management area and those of the logistics area.

Control is part of the well-known “Command and Control” function; Control occurs during the operational activities in relation to the SoS management activities.

In order to analyse how Agility, Resilience and Control of SoS can be achieved, it seems necessary first to

recall what the existential characteristics of an SoS are.

1.2 SoS definition and characterization

1.2.1 SoS nature issues

The following SoS Definition comes from the DoD document "Systems of systems System engineering guide" of December 22, 2006 Version .9:

> "A system is an integrated composite of people, products and processes that provide a capability to satisfy a stated need or objective (Mil-Std 499B).
>
> A capability is the ability to achieve a desired Effect under specified standards and conditions through combinations of ways and means to perform a set of tasks (CJCSM 3170.01B, May 11, 2005).
>
> SoS is defined as a set of arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities (Defence Acquisition Guide Book ch.4). When integrated, the independent systems can become interdependent, which is a relationship of mutual dependence and benefit between the integrated systems. Both systems and SoS conform to the definition of a system in that each consists of parts, relationships, and a whole that is greater than the sum of the parts; however, although an SoS is a system, not all systems are SoS."

In others terms, like every system, an SoS can be defined as a set of activities, continuous or periodic (from very large to very narrow time period):

- Focused on a pre-determined single mission or on a pre-determined category of missions,
- Offering and consuming a set of services,
- Realized by a given set of human actors playing given roles with the help of automated actors (software and other equipments),
- Using resources like information, energy, materials, etc.
- Interacting through flows of information, of energy, of materials, etc.

This definition offers the advantage that all the activities of a given domain are inside the system automated or not, so that we are able to discuss about the level of the SoS automation.

Categories of SoS may be defined, based on:

- The nature of flow we are considering; we can talk about information-based SoS if we consider SoS implementing an Information System or only the information flows of the SoS,
- The generic character of the SoS: a SoS dedicated to a single mission is more specific than a SoS dedicated to a category of missions.

This definition was already used to define information systems in some methodologies of the 80's like SSADM, IDEF1X, MERISE, etc.

The following characteristics differentiating an SoS from a system, even as complex as it can be, are issued from different existing works of value (In NATO, US, France, etc.):

- Collaborative work in the large, often in the decisional area, achieving the global coherence

through activities orchestration and information sharing,

- Specific focus on:
  - Global efficacy and efficiency,
  - Global Adaptability to the context,
  - Global Reactivity: reactivity of the "information acquisition" process, reactivity of the "determination of actions from the collected information" process and reactivity of the "adaptability to the context" process.
- Like a system, most of the time an SoS is composed of an automated part and of a non automated (or human) part (people are inside the SoS and not outside). However in the case of SoS, the relation between the human part and the automated part of the SoS is much more complex, diffuse and very quickly evolving (even at exploitation time) than in equipment system, even complex, (like a Missile, plane or tank terminal system for example). In fact, an SoS can be assimilated to a system upon others constituent systems, each constituent system being an information system, an equipment system or an SoS (in case of embedded SoS). The Automated part is composed of components:
  - Components are often legacy,
  - Components may be specific or COTS,
  - New components development based on COTS acquisition and reuse of products registered in well-defined repositories, etc.
- At SoS level, constituent systems are viewed as "black boxes" offering and consuming Services,
- Operational independence of constituent systems,
- Management independence of constituent systems,
- Evolutionary and incremental development,
- Emergent behaviour: the SoS functionalities are not only the sum the functionalities of each of the constituent systems,
- Geographic distribution: this does not imply large distance between constituent systems but clear segregation between them in terms of resources and behaviour even if they are physically collocated.

These last five SoS main characteristics have been identified by Mark W. Maier[1]; and refined more recently by John Boardman and Brian Sauser[2].

The following figure presents the SoS composition:

1 Architecting Principles for Systems-of-Systems

2 System of Systems – the meaning of "of"

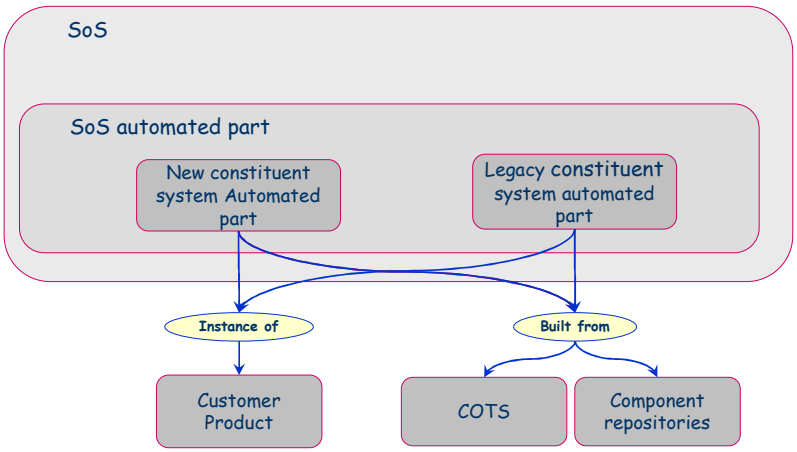


Figure 1 : SoS Composition

1.2.2 SoS examples

In the Defence area:

- Air defence and Control for national security,
- Most of the military operations on a given area,
- The UK program concerning “Area Surveillance”,
- Border surveillance, Maritime and Earth sides,
- Etc.

In other areas:

- International Air traffic management,
- Banking systems interconnection,
- Border surveillance within Schengen area by each national police and customs,
- Security system of large international events like political summit meetings, Olympic games (DAVOS for example), etc.,
- Coordination of production and transport systems,
- Interconnection of the different actors of a national or international Healthcare system,
- Engineering related to civilian or military planes, etc.,
- Etc.

1.2.3 SoS engineering issues

Due to the distinctive characteristics of the SoS, the engineering process of an SoS is also really different from the one of a system.

The following definition comes from the DoD document “Systems of Systems System engineering guide” of December 22, 2006 Version .9:

> “SoS engineering deals with planning, analyzing, organizing and integrating the capabilities of a

mix of existing and new systems into an SoS capability greater than the sum of the capabilities of the constituent parts (Defence acquisition Guidebook, ch 4). Consistent with the DoD transformation vision and enabling Net-Centric Operations (NCO), SoS may deliver capabilities by combining multiple collaborative, autonomous, yet interacting systems. The mix of constituent systems may include existing, partially developed, and yet-to-be-designed independent systems. SoS SE should foster the definition, coordinate development, and interface management and control of these independent systems while providing controls to ensure that the autonomous systems can function within one or more SoS. In most cases, systems are integrated through information exchanges. In the DoD this is accomplished through a set of net centric approaches based on the DoD Net-Centric Data Strategy and DoD 8320.2 that establishes the principles of making data visible, accessible, trustable, and understandable to the enterprise.

The Net-Centric Data Strategy establishes the use of communities of interest to work toward common vocabularies to accomplish these principles. This is a key evolution in SoS thinking away from engineering point-a-point interfaces toward exposing your data to the enterprise in a common vocabulary, resulting in a one to many interface that solves the integration problem not only for the engineered solution, but for unanticipated uses as well".

The following points will achieve a more complete characterization of the SoS engineering process.

SOA

It is a specifically well suited architectural approach for SoS; SOA brings two essential concepts:

- The Intermediation concept: the components do not directly interact ; they interact through a special component implementing a function of Intermediation,
- The Service concept: the functionalities of a system are viewed through a given set of Services : the way to access to the service is specified once through a Service contract (Service Level Agreement) which covers functional and non functional requirements (interoperability aspect) but may have several implementations (portability aspect).

The practice of SOA has an impact on the way to represent the SoS requirements and specifications, the way to develop the SoS and the way to realize the Integration Verification and Validation of the SoS.

SoS Lifecycle

A "single mission"-dedicated SoS is built at the beginning of the mission and disappear at the end of the mission; it is specific to the mission; its birth and death are the same as those of the mission.

A "mission category"- dedicated SoS is build to provide capabilities covering all the missions of the category, just before the occurrence of any of the missions : its birth and death are not tied to those of the missions; when a mission occurs, the SoS has to be "configured" in order to better fit the mission (the complexity of this configuration depends on the scope of the category).

SoS engineering process

So, the engineering process is globally an integration process (the term "integration" covers activities occurring before and after the development ones) of existing constituent system and / or new constituent system, created from COTS or developed by reusing some products in accordance to an integration schema and an evolution management process of the SoS integration schema:

- Requirements definition, i.e. Definition of the SoS expected capabilities to fulfil the SoS

objectives (or effects): Effect Based Operation and Capability Based Development principles.

- SoS definition : in particular, a definition of the interoperability rules and data; plus mediation specifications to cover:
  - data adaptation according to constituent system understanding (semantic, procedural (protocols) and syntactical levels)
  - and service orchestration according to resource management and behaviour mainly.
- IVV activities.

Legacy issue

The automated part of the SoS is built from many legacy autonomous constituent systems; most of the time, it is not possible to make these constituent systems evolving, in the same time, in order to fit the requirements of the SoS, because of the programmatic independence of some constituent system and / or the fact that some constituent system is a product used in different SoS, the required evolution not being applicable to its different uses; So, an dedicated adaptor is developed - implementing the intermediation function previously mentioned–in order to bind up the legacy component to the SoS use. The concept of Service will make easier the practice of such interface adaptation.

Product Line issue

All the characteristics already mentioned leads to a situation where same constituent systems will be integrated in several SoS; these constituent systems can be exactly the same or rather the same.

This context offers to the provider of these constituent systems the opportunity to consider these shared constituent systems as products and, so, to manage lines of products.

The feasibility to share products depends on the compatibility of the non-functional requirements (like performance and resilience ones) applying on different constituent systems having the same functional requirements.

The interest to share products depends on the cost of managing different constituents systems compared to the cost of managing a single product plus its integration into the different SoS.

Otherwise, this product management implies to be able to ensure the ascendant compatibility of them through interfaces able to take into account the evolution of the products versions.

Reuse issue (different from the previous one)

A constituent system may be developed by reusing some reusable components available in appropriate repositories; the difference between these components and the above products are the following ones:

- The size: an example of a product is a “tactical editor”, an example of component is “random number delivering”,
- The property : a product is the property of SoS constituent systems, a component is “off the shelf” free or not.

CD&E approach and Multi-provider issue

The constituent systems of a SOS may have been developed by different providers; so:

- How to deal with the industrial property?
- How to deal with the respective responsibilities of different providers? Does the Customer play the role of Lead Capability Integrator (**LCI**), alone or not? of Lead System Integrator (**LSI**), alone or not?
- How the LCI will be able to proceed to the acceptance of a constituent system of a given provider and then proceed to the deliver of this constituent system to LSI? What level of confidence may a LSI get about such a constituent system?

The following figure presents an overview of an engineering process including a **CD&E approach** with the new roles and a more iterative collaborative schema between the actors of the SoS engineering process:

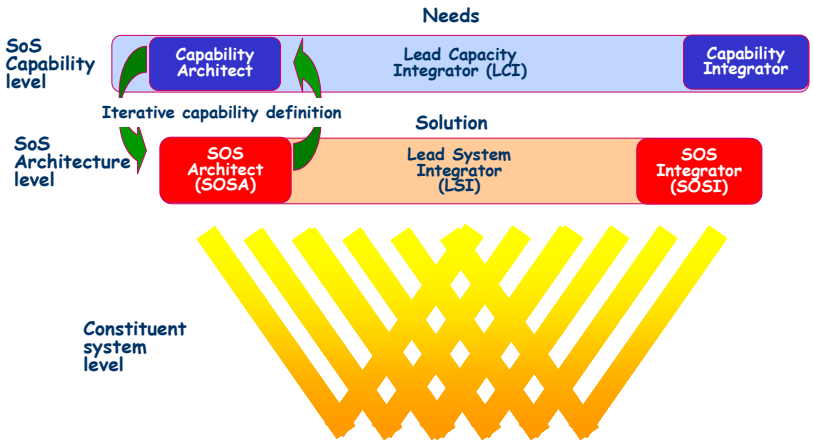


Figure 2 : SoS Engineering Process new roles and collaborative schema

1.2.4 Conclusion on SoS definition

The SOS System Engineering process is different than the System one. The following distinctive characteristics are:

- Essential role of the architects,
- News roles: SOSA, SOSI, etc.,
- Need to consider the whole life cycle at SoS level (Trough life Capability Management),
- Need to consider system management, logistics activities, and other non-functional purpose activities during engineering process.

2.0 Requirements required by SoS Agility, Resilience and Control

This section identifies the main SoS Engineering, Architecting and Modelling Requirements required by Agility, Resilience and Control, with regard to the SoS differentiating characteristics.

2.1 Engineering Requirements

2.1.1 Urbanization Requirement

Urbanization is the transformation process defining how to progressively transform the current state of the SoS toward the target state compatible of the full SoS definition trough several intermediate states so that:

- The target state of the SoS will offer a “plug-in / plug-out” capability based on:
  - better modularity capabilities: all the activities of the system are partitioned into activities subsets having so no more redundancies between them and packaged in constituent systems,
  - standard-based interoperability capabilities between these constituent systems,
- the intermediate states will offer the necessary cohabitation of legacy and new constituent systems through the implementation of a specific architecture pattern called “Intermediation”.

In its target state, the SoS is architected as a set of interacting constituent systems, legacy and new ones.

This urbanization process is quite suited to SoS engineering; it can be represented by the following figure:

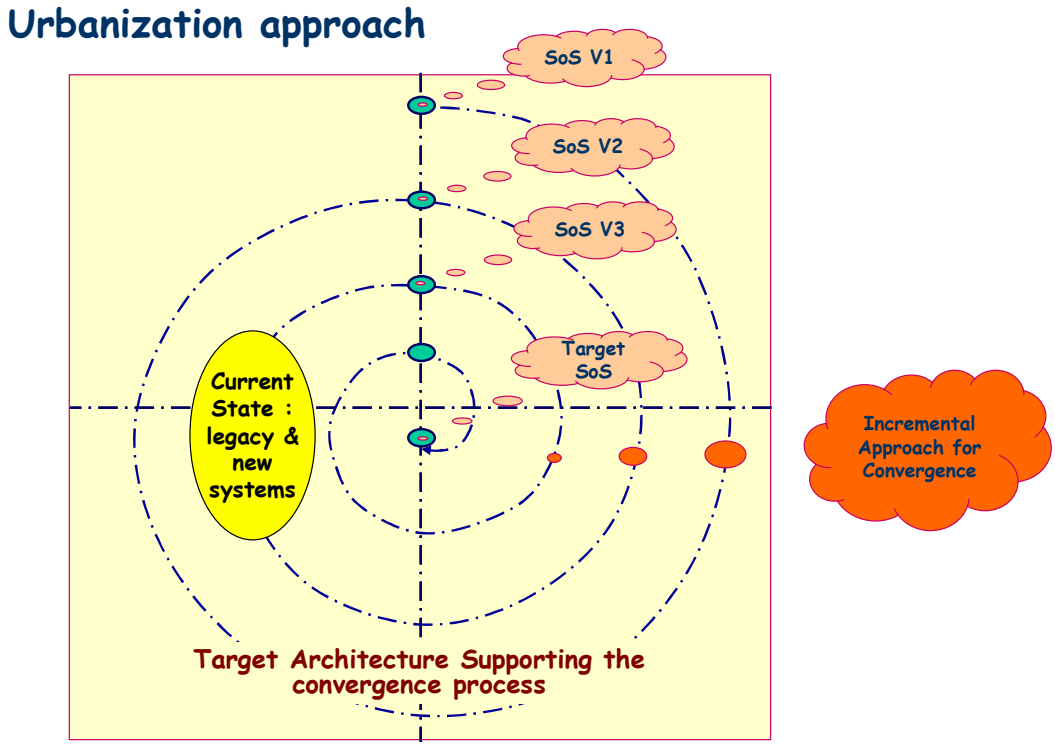


Figure 3 : Urbanization approach

The progressive transformation of the SoS express the convergence process towards the target state where SoS will offer the “plug-in / plug-out” capability.

This “plug-in / plug out” capability is essential to fit the Agility need because it will be much more easy both to build the SoS ant then to make it evolving ; It is also important as regards to the Resilience need because the SoS modularity will facilitate its dynamic re configuration needed to be resilient.

2.1.2 Global Automation Analysis Requirement

Integrated **L**ogistics **S**upport (**ILS**) approach have already focused on the need to make in parallel the engineering of a system and the engineering of its support system (including system management and

logistics activities) based on the fact that the global objective is to optimize the system availability and the global cost of the system (including the support during its whole life).

The Urbanization requirement enforces the need to globally analyse the operational activities (or process), the system management activities (or process) and the logistics activities (or process) because the modularity aspect of the SoS target state must concern these three processes and, for each of them, both the automated and the non automated parts.

The Autonomic computing concept initiated by IBM concerns the automation of the system management activities : self-configuring, self-optimizing, self-healing and self-protecting. The automation of the three operational, system management and logistics processes plus the automation of the engineering process itself will have a very positive impact on the SoS Agility; Self-configuring (and reconfiguring) and Self-healing will have a particular positive impact on SoS Resilience.

This point is summarized by the following figure:

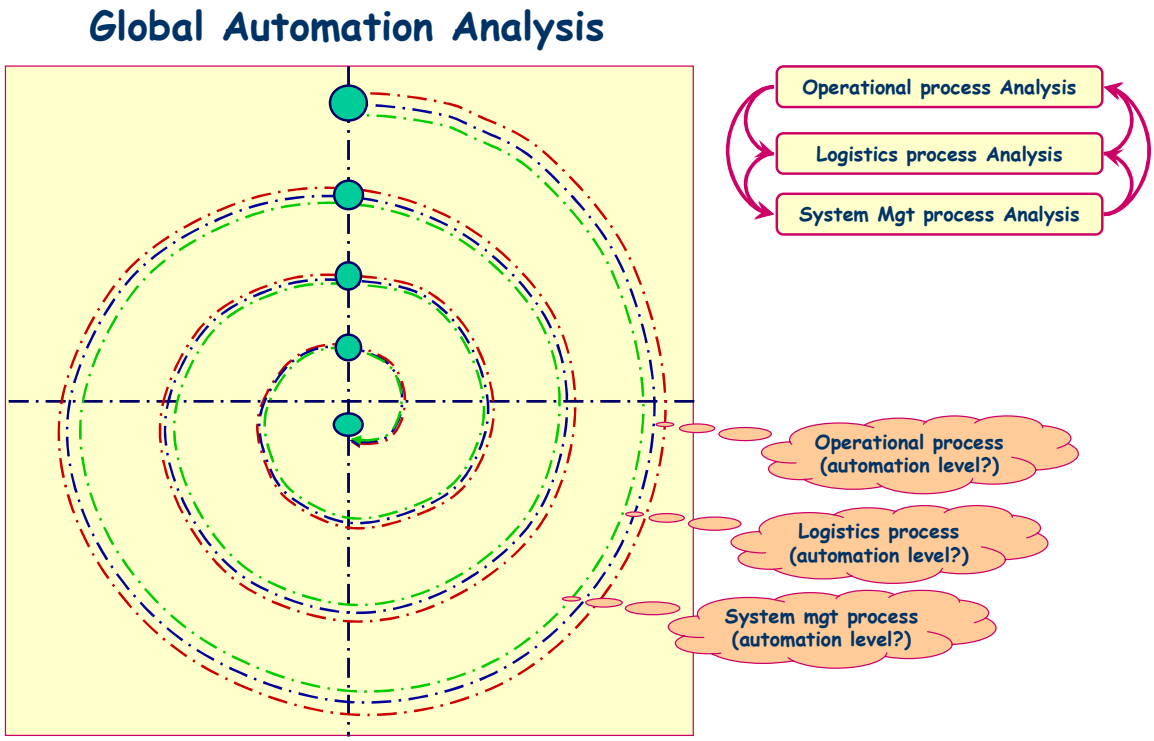


Figure 4 : Global automation analysis requirement

2.1.3 Process-based global value Analysis Requirement

Business **P**rocess **M**anagement (**BPM**) brought two essential ideas of interest for SoS:

- The first idea is that the efforts affected to the automated part of the SoS must be justified by the operational enhancements which can be expressed within the process models,
- The second idea is that the monitoring of processes can be partly automated since the processes are described by models that can be interpreted by a specific component usually called “Workflow engine” or “Orchestration engine”.

In such a way, the value analysis can be based on the definition of Processes enhancements which are then translated into functional and non functional requirements applying on the SoS automated part, as follows in the following figure:

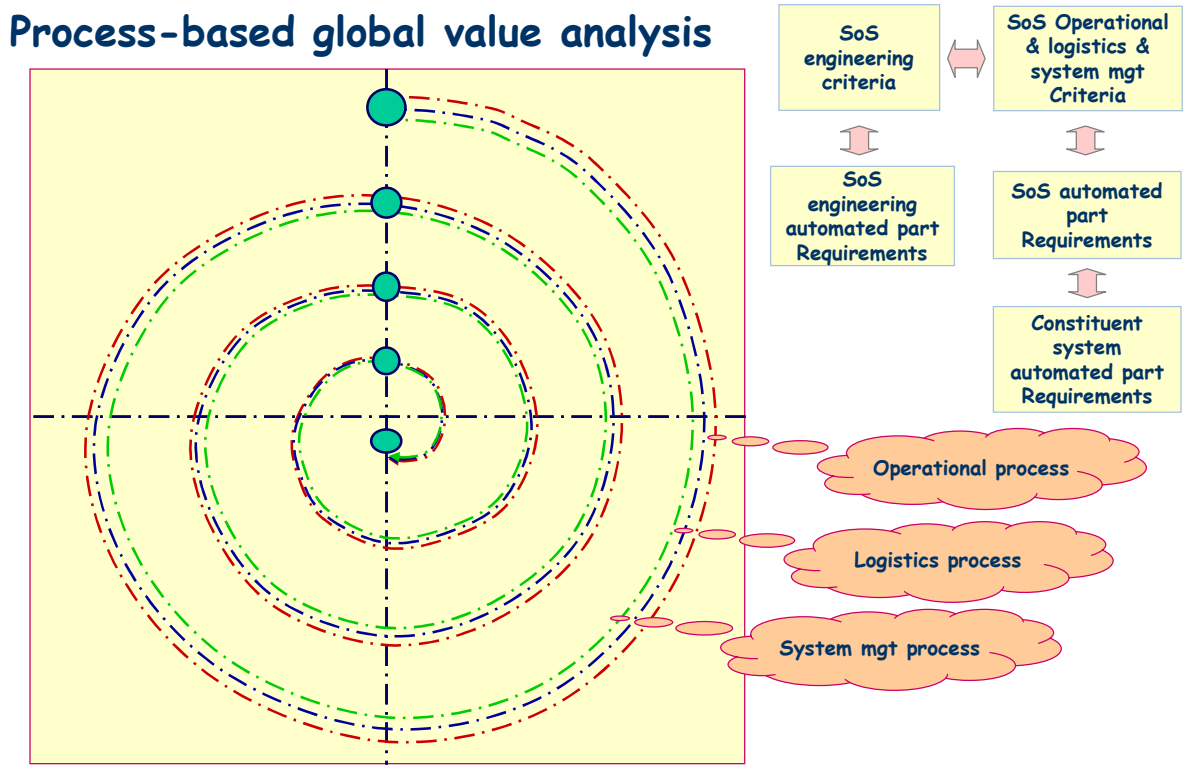


Figure 5 : Process - based value analysis

This value analysis must be leaded according to the stakeholders view points and must concerns at least the three previous processes (operational, system management and logistics ones) plus the engineering process itself.

The ability to do this process-based global value analysis will be of significant contribution in order to fit the Agility need because it provides the means to rapidly evaluate the interest of SoS evolutions.

2.1.4 LCI / LSI Requirement

As part of the **Through Life Capability Management** (TLCM), the SoS engineering process must clearly distinguish:

- Activities related to the SoS **Capacity definition** from the required objectives or effects,
- Activities related to the **Capacity Implementation.**

The Capability definition process will be under the responsibility of a **L**ead **C**apability **I**ntegrator (**LCI), the Capability Implementation** one will be under the responsibility of a **L**ead **S**ystem **I**ntegrator (**LSI).**

When the constituent systems of the SoS come from different providers, the LCI and the LSI have to:

- Take care of their own ability to held the job: own competence, ability to use the providers

competences, etc.

- Define what are the conditions of the involvement of the providers in such a way their interests are preserved: property, information privacy, etc..

The following figure mentions the two different roles, LCI and LSI, as well as the multi-provider characteristic which have an important impact on the way to make these providers participate to LCI and LSI roles:

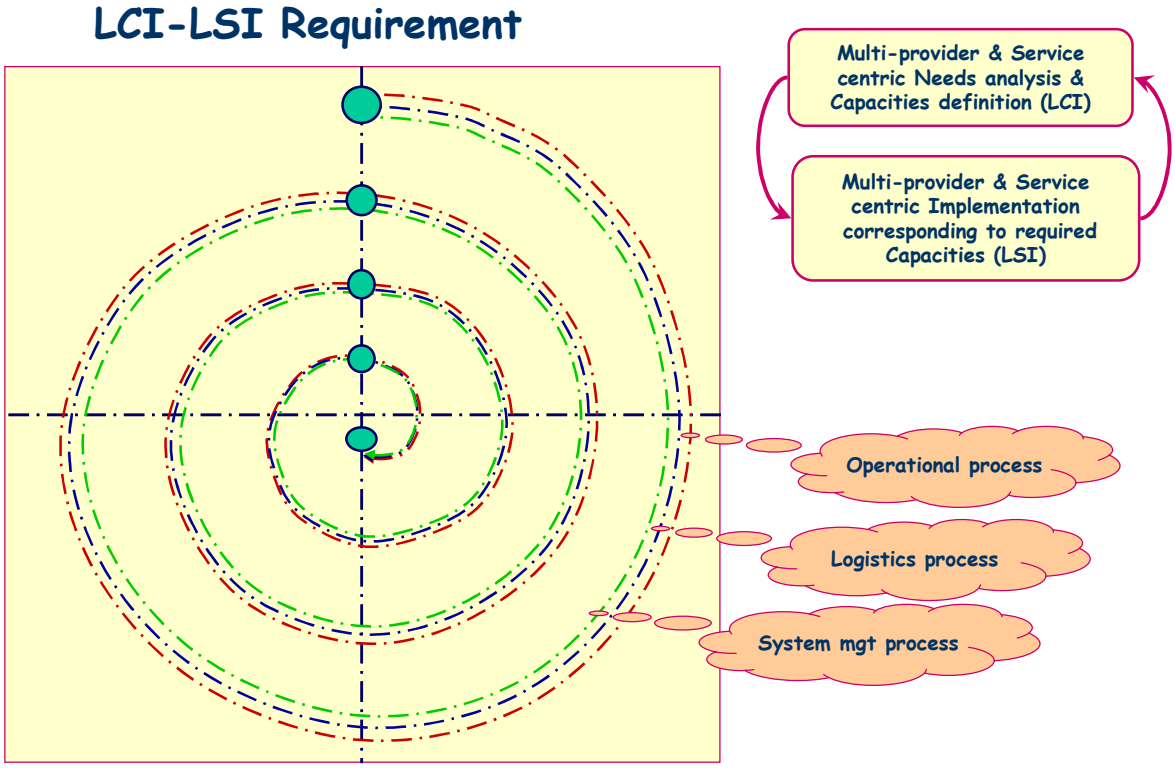


Figure 6 : LCI – LSI distinction requirement

This distinction between LCI and LSI roles is essential to provide the necessary seamless engineering process in order to contribute to the overall Agility need.

2.1.5 Service-Function centric organization duality requirement

As a consequence of the occurrence of the “Service” Concept, a SoS is viewed from the customer side as a set of services; these services are distributed among the constituent systems of the SoS. But a Service correspond to some use case of the constituent system process and it is realized by a given set of the constituent system functions; functions can be totally or partially automated, and the automated parts of these functions will be implemented and materialized by software, equipments and so on.

The Capabilities definition process can be seen as a two layers process:

- the top layer will be structured per Service,
- the bottom one will be structured by Function,

Service and Function objects being in a “Many-to-Many” relationships.

The following figure illustrates these layers for the SoS engineering process:

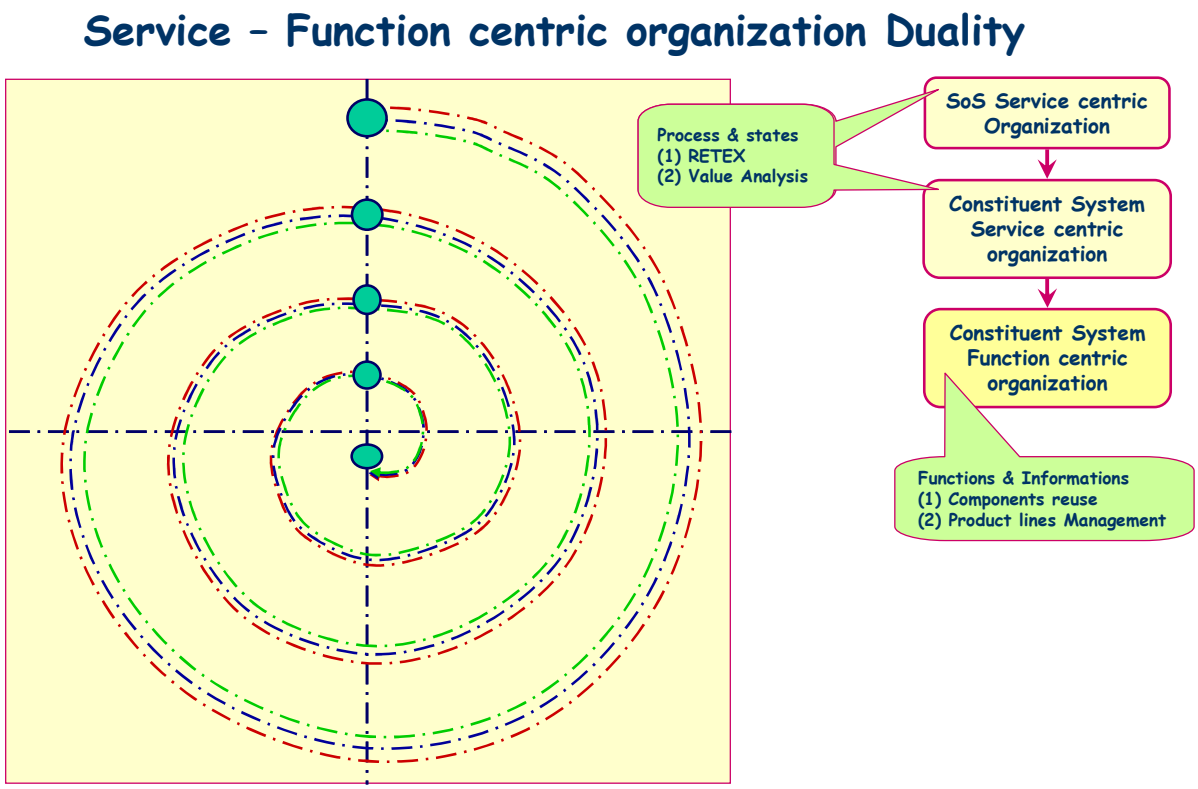


Figure 7 : Service – Function centric organization duality

This distinction between engineering activities organized per SoS Service but built on top of engineering activities organized per SoS constituent system function contribute to the overall Agility need because it allows to reuse products and/or components and so to reduce the overall delay for delivering the SoS.

2.2 Architecting requirements

2.2.1 Intermediation and Modularity Requirement

The modularity properties explained here after must be applied to:

- The operational non-automated and automated activities, packaged into constituent system (without redundancy between them),
- The system management non-automated and automated activities associated to the software tools and their corresponding resources (without redundancy between constituent systems),
- The logistics non-automated and automated activities associated to the military tools and their corresponding resources (without redundancy between constituent systems).

The modularity properties are:

- within a constituent system, the activities and the information are gathered in categories, each category being a partition (in the sense of the set theory) : an “information” category is produced

by only one “activity” category and a pair (“activity” category, “information” category) is under the responsibility of only one function,

- resources as information, material, energy, etc, are exchanged between constituent systems through standardized interfaces,
- a constituent system can be committed and rolled back.

The concept of Community of Interest (**COI)** is a very interesting one if it can be mapped on the operational concept associated to the **SoS constituent system**; in such a way, the COI conforms to the functional system definition, completed by the above modularity properties.

The COI is then the element which offers and consumes services, self-orchestrated, or orchestrated by another special COI ensuring the “intermediation” function of the SoS.

Assuming this definition, process model, organization model and data model can be associated to the COI; Therefore, the SoS is independent from the organization and of the running of their components (systems).

Every interaction between constituent systems is under the control of a special constituent system implementing the intermediation function, being responsible:

- to ensure the service orchestration, covering all the types of interaction between constituent systems: information distribution (push/pull, publish/subscribe, synchronous/asynchronous, etc.) and service distribution,
- to manage the services accessibility, availability, concurrence, exclusivity, and arbitration, i.e. **service directory,**
- to ensure, through interfaces, the ascendant compatibility of the services evolution,
- etc.

The modular architecture of an SoS based on the principle that every constituent system interaction is under the control of a specific “Intermediation” constituent system may be illustrated by the following figure:

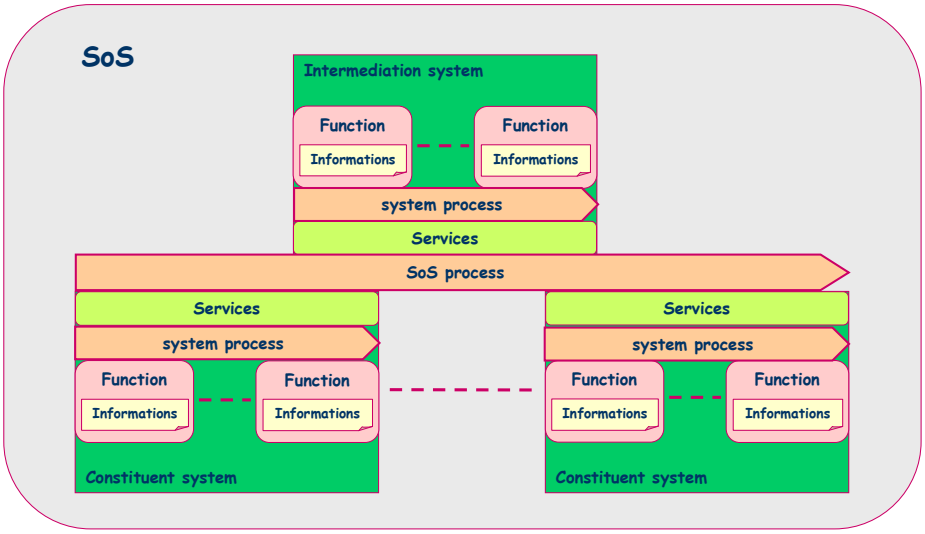


Figure 8 : SoS "intermediation - modularity" requirement

The corresponding modular architecture of an SoS including its automated part may be illustrated by the following figure:

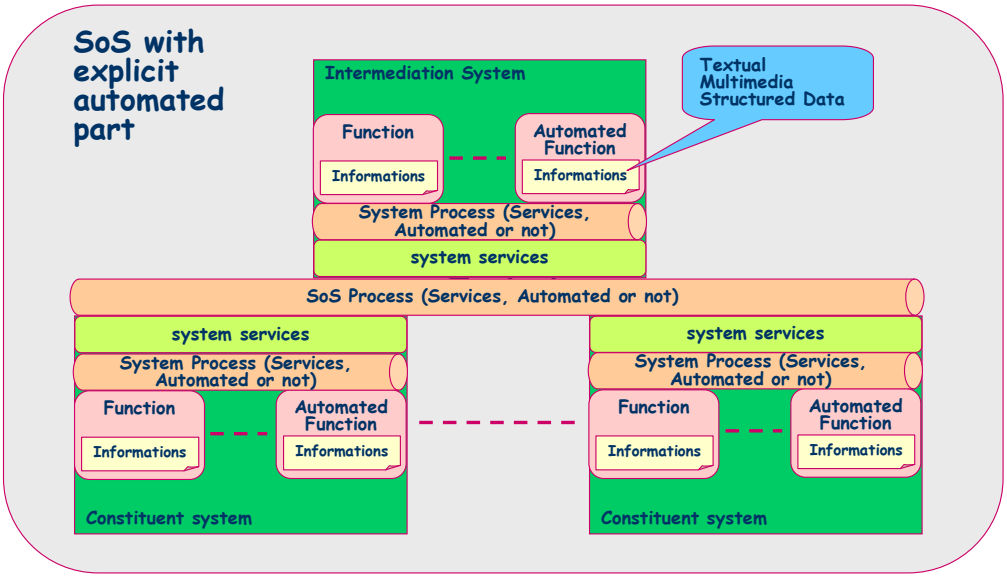


Figure 9 : SoS automated part "intermediation – modularity" requirement

Moreover, this modularity requirement implies the ability to allow the cohabitation of legacy systems with new ones through a legacy system adaptor, avoiding having to adapt the legacy system itself, as illustrated by the following figure:

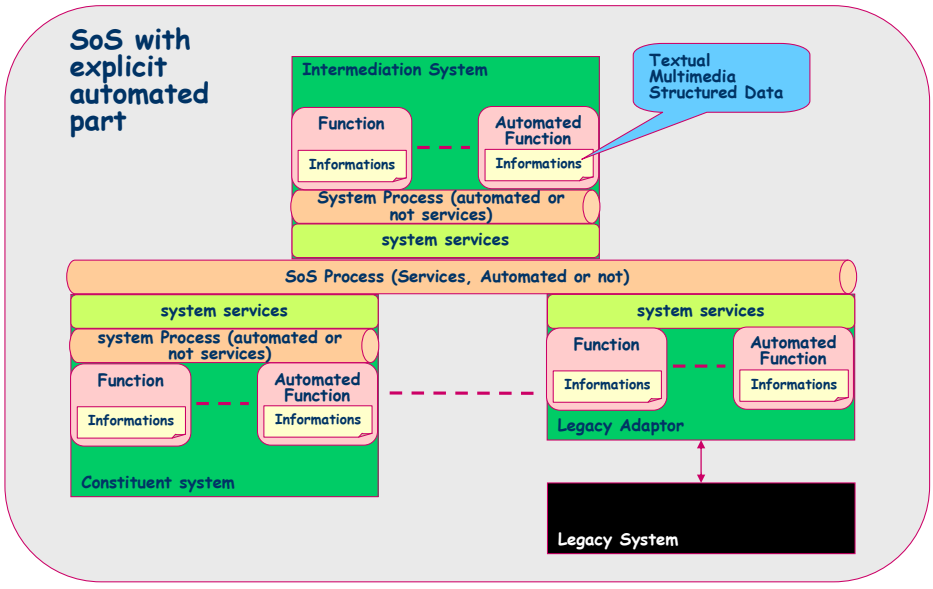


Figure 10 : "new and legacy systems cohabitation" requirement

2.2.2 Service localization transparency Requirement

Here, the question is: where Services will be deployed and then geographically localized? The requirement is that the whole services (non automated and automated parts) will be **geographically distributed** (provided that the performance requirements are satisfied) ant that the human and automated actors of the SoS will not know their localization (**geographic distribution transparency**).

This geographic distribution transparency will facilitate the SoS dynamic reconfiguration during the SoS System management activities; it can be symbolized, on the following figure, by the representation of a different geographic area for each SoS constituent system; for example:

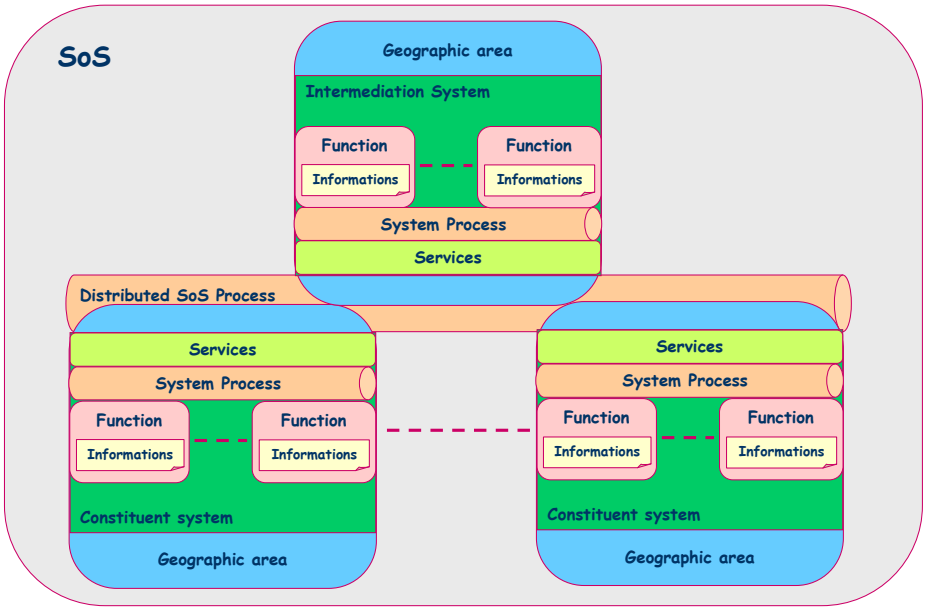


Figure 11 : Service localization transparency requirement

The **Reach Back** concept is based on the geographic distribution SoS capability.

2.2.3 "SoS as a Service" Requirement

In the classical approach, the provider develops a product and delivers it to the customer who then uses it to realize some operational activities as well as the system management corresponding ones.

In a "Software as a service" approach, as described by the literature, the provider develops a product but, instead of delivering it to the customer, he installs a platform including all the computerized resources necessary to run the product and he runs it and realizes itself the system management corresponding activities. Finally, the result of the engineering is provided as a service to be directly usable.

Using the same reasoning “System as a service and “SoS as a Service” approaches can be expressed, on the following figure, by the allocation the different geographic areas to different providers, the customer keeping as least the Intermediation constituent system; for example:

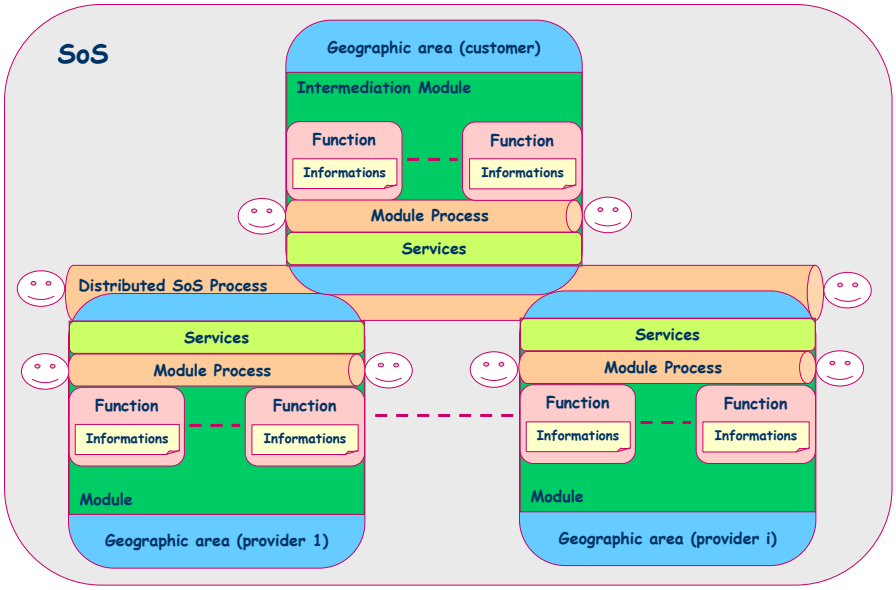


Figure 12 : “SoS as a Service” requirement

Implementing this approach (provided that the performance requirements can be satisfied) will enforce the resilience of the SoS.

2.2.4 Service Portability Requirement

The ability to distinguish between the specification and the implementation of a service (i.e. several implementations of the same service), allows the provider independence from the customer but also enforces the agility of the SoS engineering activities because the occurrence of a service implementation change does not impact the applications using that service.

The following figure illustrates the portability requirement; the services of a given application are implemented in two different technical environments, i.e. .NET and J2EE:

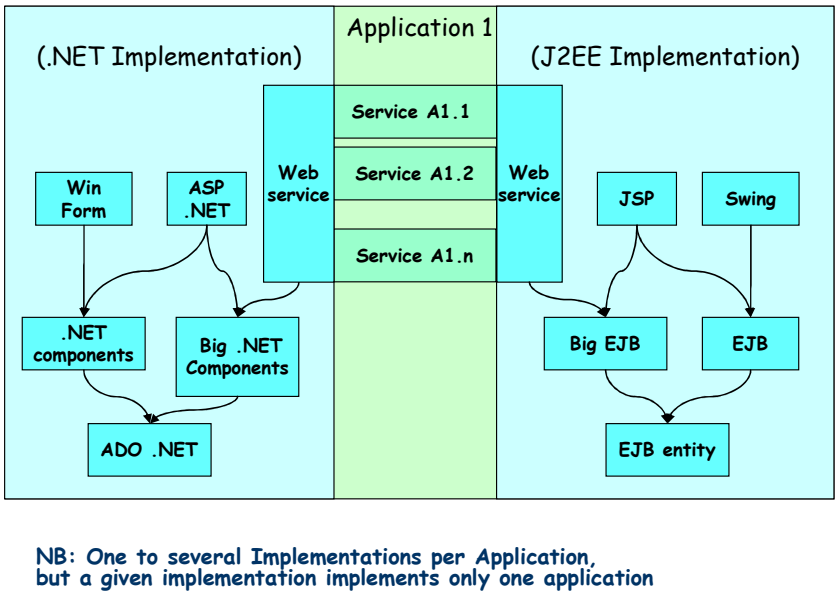


Figure 13 : Service portability requirement

2.3 Modelling requirements

The previous architecting requirements have major impacts to the SoS modelling activities, mainly intermediation, modularity and interoperability, concerning:

- The architecture models,
- The process and state models,
- The data models,
- the relationships between the models types.

2.3.1 Models Architecture and Architecture modelling Requirements

The architecture model must cover the whole SoS, non-automated and automated parts of it. The engineering of the SoS automated part must distinguish the specification level from the implementation level (due to portability requirement).

An SoS constituent system model must identify the services (offered and consumed), the processes, the states, the flows, the functions and the information contents of the constituent system.

So, an architecture model will link to process models, transition – state models, structured data models and "exchanges structures data models, as follows by the following figure:

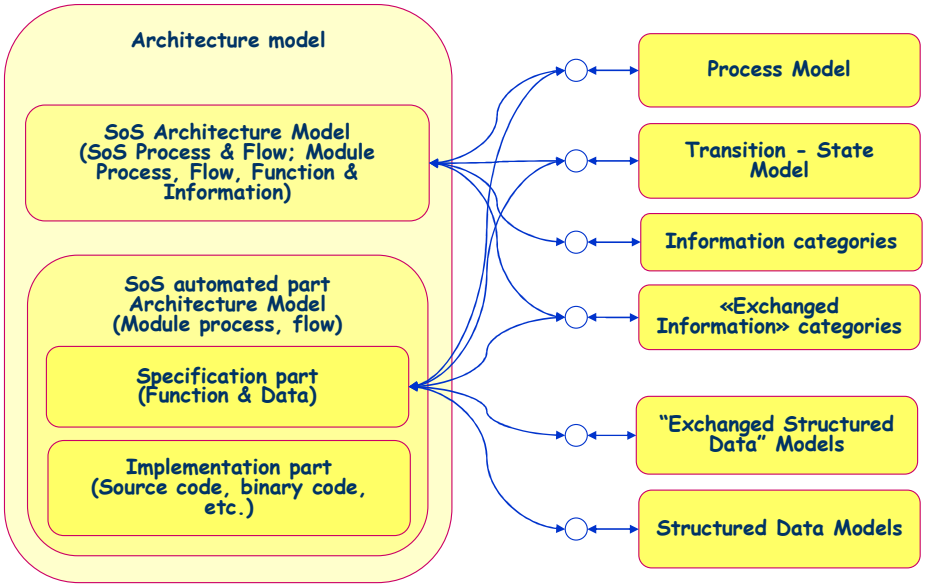


Figure 14 : SoS models Architecture

2.3.2 Process and State modelling requirements

At SoS level, each constituent system is perceived as an actor (automated, partially or not) receiving a service demand, executing the service et sending the result of the service performance.

So an occurrence of SOS can be modelled as a sequence of constituent system occurrences realizing a SoS-level services, each constituent system occurrence realizing a constituent system-level service.

Synchronisation points on these sequences represent stable states of constituent system and of the SoS itself. These points must be explicit (by a COMMIT operation expressed in a process modelling language like Business Process Modelling Language, **BPML**) in order be able to restart the SoS or some of the constituent systems when occurring a failure (by some **compensation** operations also expressed in BPML).

A transaction at SoS level may take a long time and is composed of lower transactions at constituent system level : it is a long nested transaction (covering automated and non automated activities); some of these activities can not be rolled back, you must forecast some compensation activities; because of state combinatory complexity, the restart process often implies people.

The following figure expresses the long nested transaction structure of an SoS:

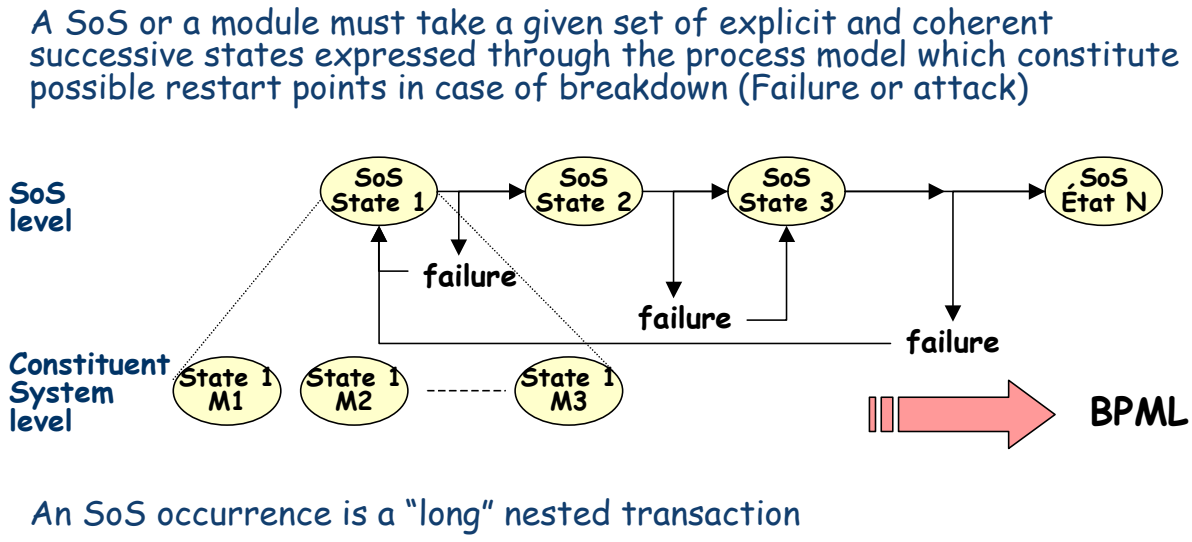


Figure 15 : Process and State modelling requirement

2.3.3 Data modelling requirements

The architecting modularity requirement implies that functions encapsulate information; so, it is necessary to define information categories in order to be able to model the responsibility relationships between Functions and Information. Moreover, these information categories will be referenced by information flows in order to characterize the semantic contents of such flows.

The following figure expresses the dependencies between the information categories referenced on flows at SoS and Constituent system levels:

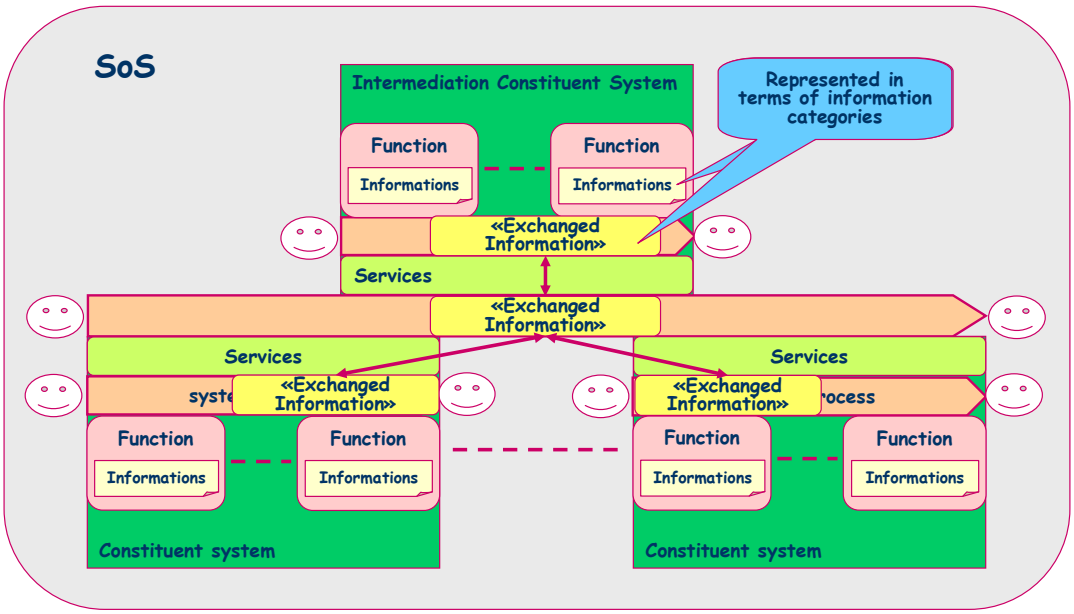


Figure 16 : Data modelling requirement (1)

Information will be implemented in the automated part of the SoS as text, multimedia or structured data; due to the intermediation approach of the services exchanges between constituent systems, all these data or just the structured ones, will then be modelled using the following method:

- For each constituent system, make a data model per function and a “pivot” data model covering all exchanged (or exchangeable) data and document with the correspondences between each function data model and the “pivot” data model,
- At SoS level, make a “pivot” data model covering all exchanged (or exchangeable) data at SoS level (between constituent systems) and document with the correspondences between each constituent system “pivot” data model and the SoS “pivot” data model,
- These different correspondences are used to implement the data transformations which must occur when exchanging them.

For each “pivot” data model (at SoS and constituent system levels), a “pivot” exchanged data model will be derived from the “pivot data model for each information flow.

The following figure expresses the different data models (excepted the exchanged data ones) and their relationships:

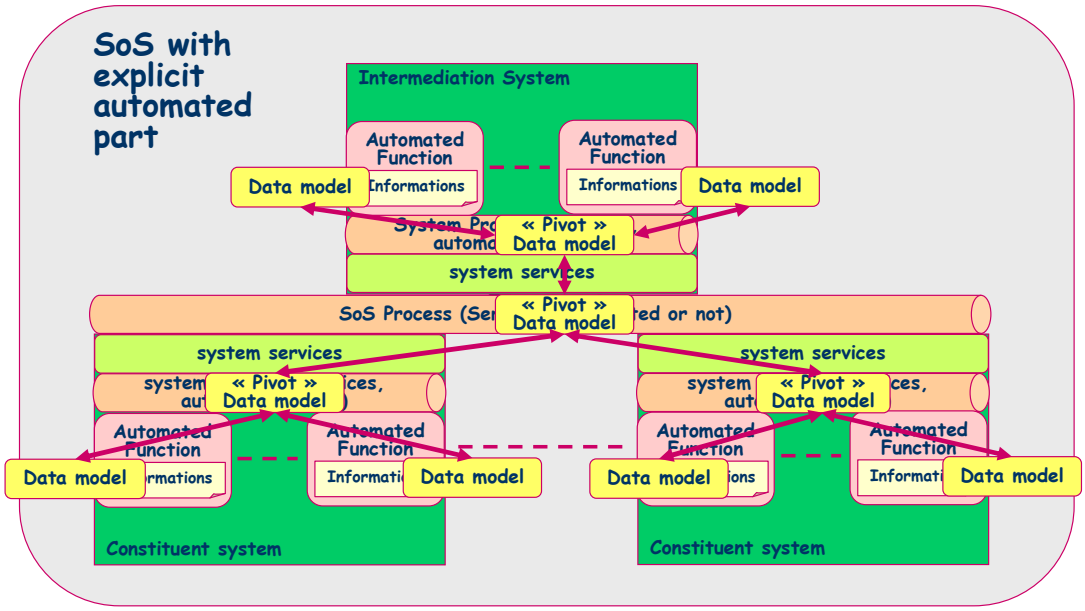


Figure 17 : Data modelling requirement (2)

2.3.4 Value analysis modelling Requirements

For both value analysis and requirements engineering: measures needed to evaluate the value, functional and non functional requirements and their relation with the previous measures.

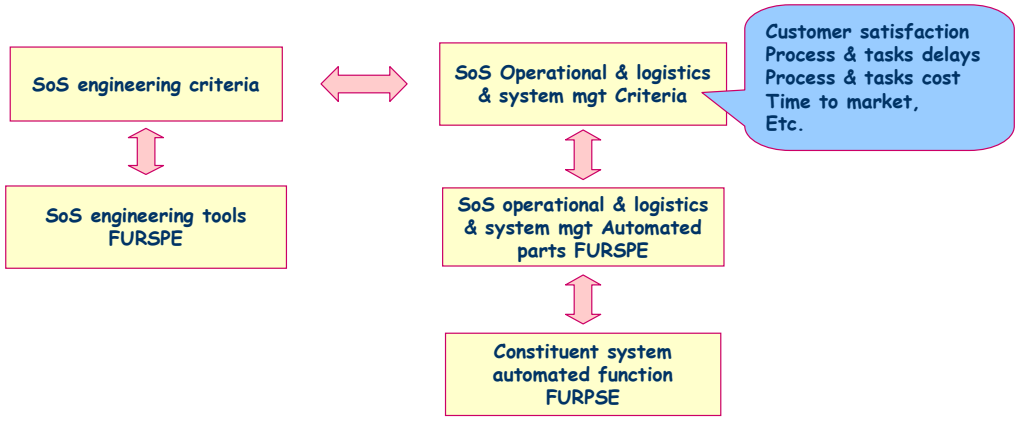
Due to the process-based global value analysis requirement, it is necessary to define the target process enhancements criteria at SoS level; these criteria will then be transformed in functional and non functional requirements at SoS level.

The criteria and corresponding requirements at SoS level will then be derived at constituent system level.

This value analysis process has to be realized for each activity domain:

- Engineering activities,
- Operational activities,
- Logistics activities,
- System management activities.

The following figure expresses the different sets of information (criteria on one hand and of functional and non-functional requirements on the other hand) and their relationships that will allow to conduct the SoS Value analysis:



- ISO 9126 (used as requirements classification)
  - F (functionality Category) : Functional adequacy, Security, Interoperability
  - U (Usability Category) : IHM Ergonomics, ease of learning - training, ease of operating
  - R (Reliability Category) : Fault tolerance, Restart, Maturity
  - P (performance Category) : effectiveness (response time, …), efficiency,
  - S (Serviceability Category) : ability and ease of testing, breakdown analysis, fault confinement, modifying
  - E (Upgradability Category) : Adaptability & Evolution, Portability

Figure 18 : Value analysis modelling requirement

3.0 CONCLUSION

SOS Agility mainly relies on:

- The ability to implement an engineering process having the following properties:
  - The process-based value analysis conferring the ability to quickly evaluate what are the impacts of some evolutions,
  - The LCI-LSI distinction and complementarity conferring the seamless engineering process,
  - The service-function centric organization duality conferring the ability to conciliate the external view of the SoS through the services, the internal view through the functions, and the ability to reuse,

- The adoption of a **Service Oriented Architecture** which may offer:
  - The SoS modularity and the loose coupling of the constituent systems (due to the existence of a special constituent system called “intermediation”), both conferring the plug in / plug out ability,
  - The concept of service and the corresponding Service Level Specification **(SLS)**, Service level Agreement **(SLA)** and the Service level Management **(SLM)**: the provider commits to the SLS, the LSI structures the SLA and the system management follows up the SLM,
  - The portability of SoS services implementation as long as service exhibition is done independently from the service provisioning.

Note: All these architectural properties have impacts on the way of making SoS modelling as explained in section 2.3.

SoS Resilience and Control rely on:

- The ability to make a global analysis during SoS engineering and to implement the architectural plug-in / plug-out ability already mentioned. Both of them allow to dynamically reconfigure the SoS when occurring some events produced by Control or by system management activities,
- The ability to implement Service localization transparency and ”SoS as a service” approach for some of the SoS services.

The following figure summarizes the relationships between the Agility, Resilience and Control needs and the SoS engineering, architecting and Modelling requirements:

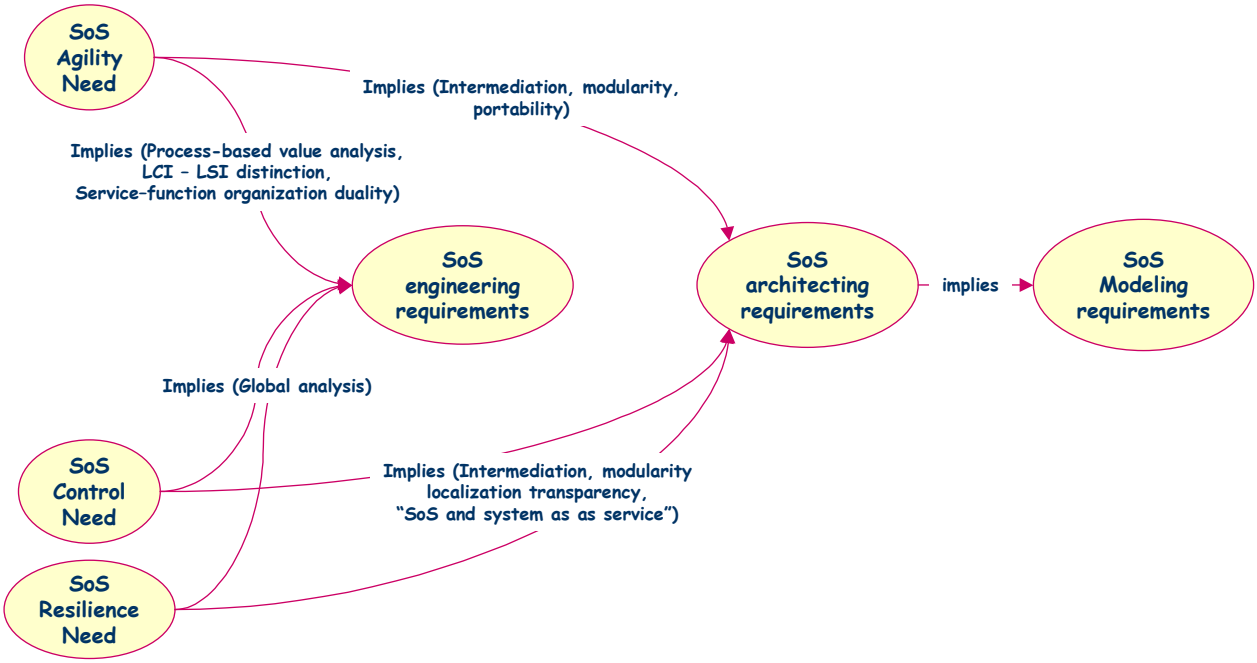


Figure 19 : “Dependencies between requirements and needs” Diagram