SOAR: a general process for Acknowledged SoS Software Architectures

Marcelo Benites Gonçalves (ICMC-Brazil/IRISA-France)

Flavio Oquendo (IRISA-France),

and Elisa Yumi Nakagawa (ICMC-Brazil)

Program

INTRODUCTION

System of Systems (SoS)

A class of systems resulted from the interaction among independent systems that cooperate delivering emergent functionalities and accomplishing a global mission.

- Operational Independence
- Managerial Independence
- Distribution
- Emergent Behavior
- Evolutionary Development

SoS Software Architectures

A structure (or a set of structures) of the SoS

- Constituent Systems: their externally visible properties, and the relationships among them
- Encompasses concepts, properties, specifications, and design decisions
- Essential to promote the success and quality of SoS

Motivation

Architectural construction is a non-trivial activity, even more for complex scenarios of SoS

- Multiple stakeholders, organizations, and interests
- Constituent systems are independently developed and managed
    - How to enable the collaboration?
- The construction of such architectures is currently performed with ad-hoc processes

Proposal

SOAR Process

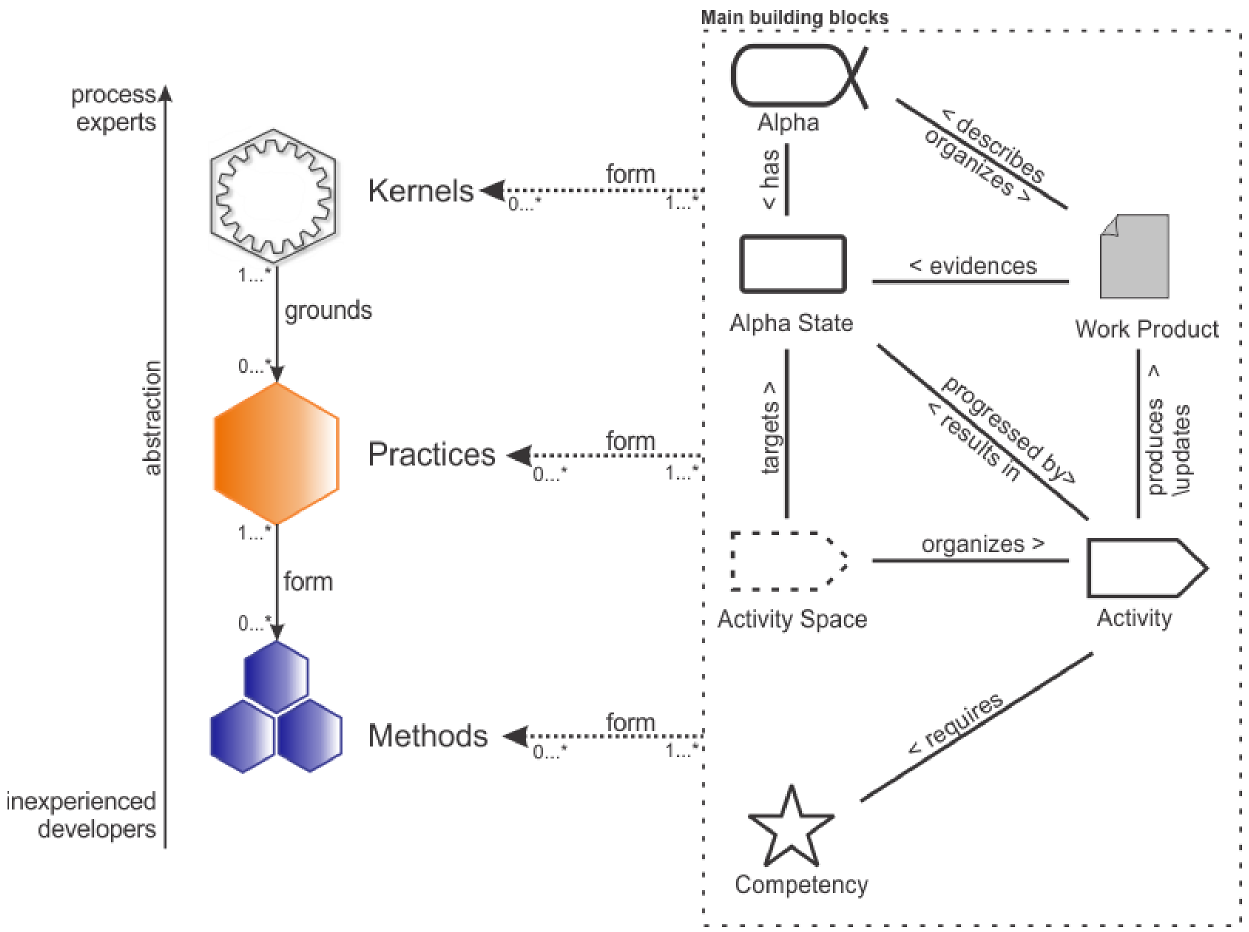
- A general process that supports the instantiation of construction processes for Acknowledged SoS software architectures
- Described with OMG’s Essence Language
- Based on the state of the art of SoS in conjunction with experts knowledge
- For Acknowledged SoS
  - A sub-category in which goals, management, resources, and authority are recognized while the constituent retain their independence

THE ESSENCE LANGUAGE AND KERNEL

Essence Language

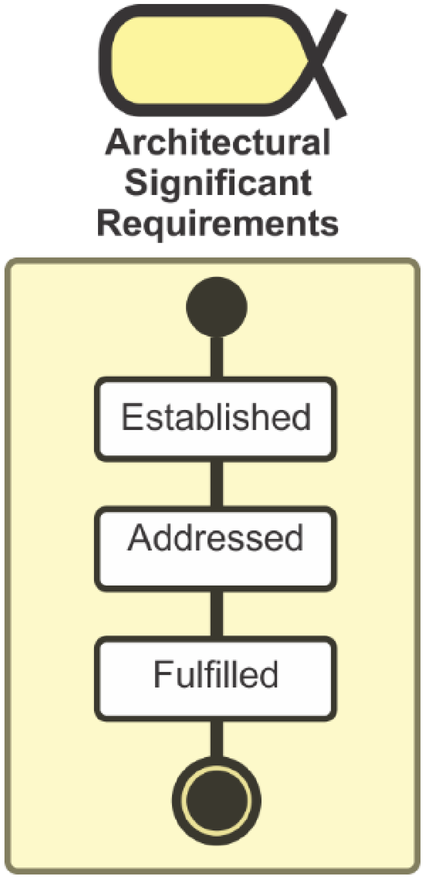
- A domain-specific language for practices and processes on software engineering
- Provides an effective basis for definition, adaptation and use of processes and practices
- Provide both textual and graphical syntax
  - Nobody is constrained to use a graphical notation in situations in which textual notation is easier to handle, and vice-versa.
- It emphasizes intuitive and concrete graphical syntax over formal semantics
  - The focus is on providing a description in a language that can be easily understood by the wide developers community whose interests are to quickly understand and use the language

Essence Language



Alphas *(Essence Language)*

- Determine the "things to work with" in development processes
- Have "states" to be reached
- The "states" are verified with specific checklists
- Are expressed through work products



Activity spaces *(Essence Language)*

Determine the “things to be done” in development processes

Are encompassed by specific activities and tasks

Its well execution is verified through Alpha states to be reached

- EX: in the activity space *ASRs Elicitation* the alpha *ASRs* must reach the state <u>**established**</u>

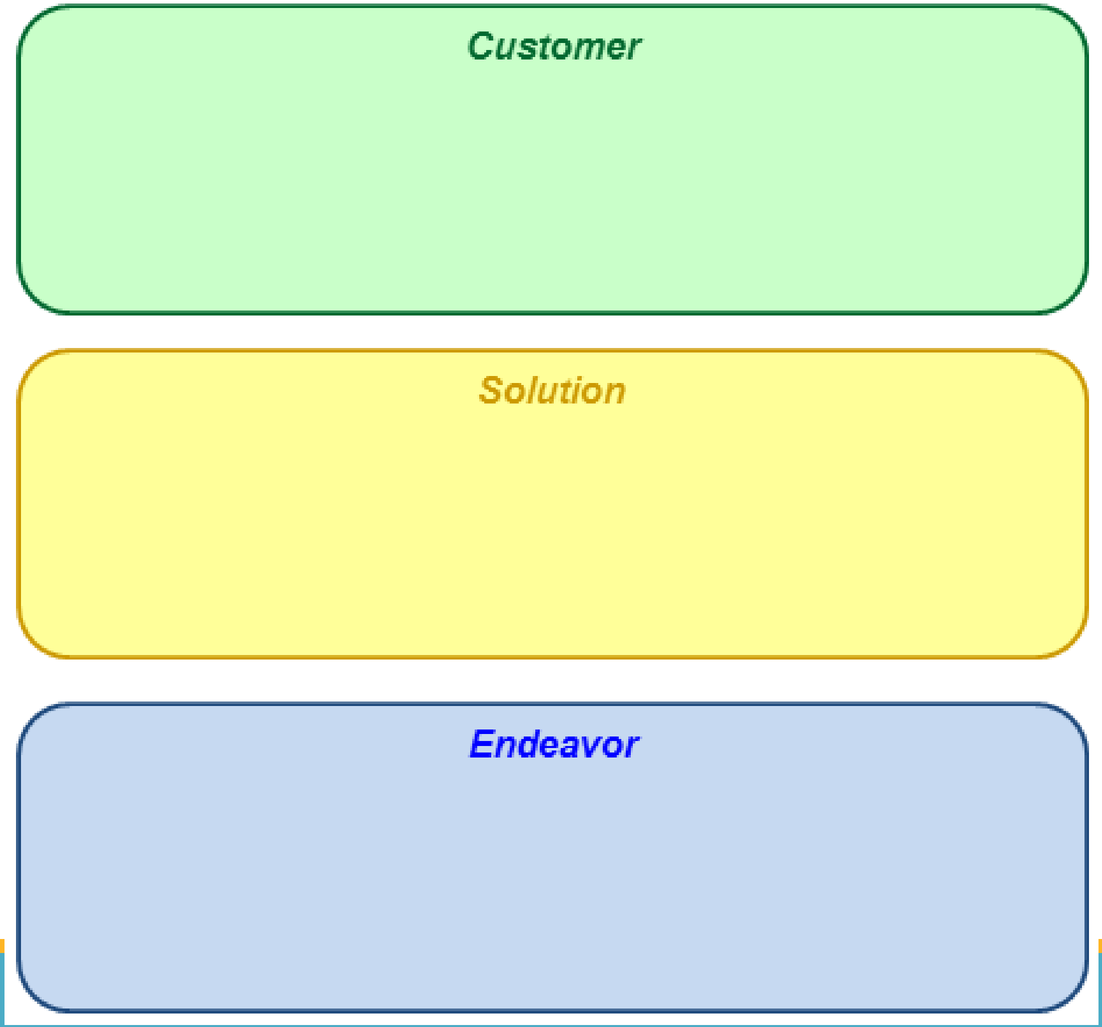
Competencies *(Essence Language)*

- Competencies express the “Required Skills” when constructing SoS software architectures
- **Ex. Stakeholder Representation**: The ability to gather, communicate, and balance the needs of other stakeholders, and to accurately represent their views

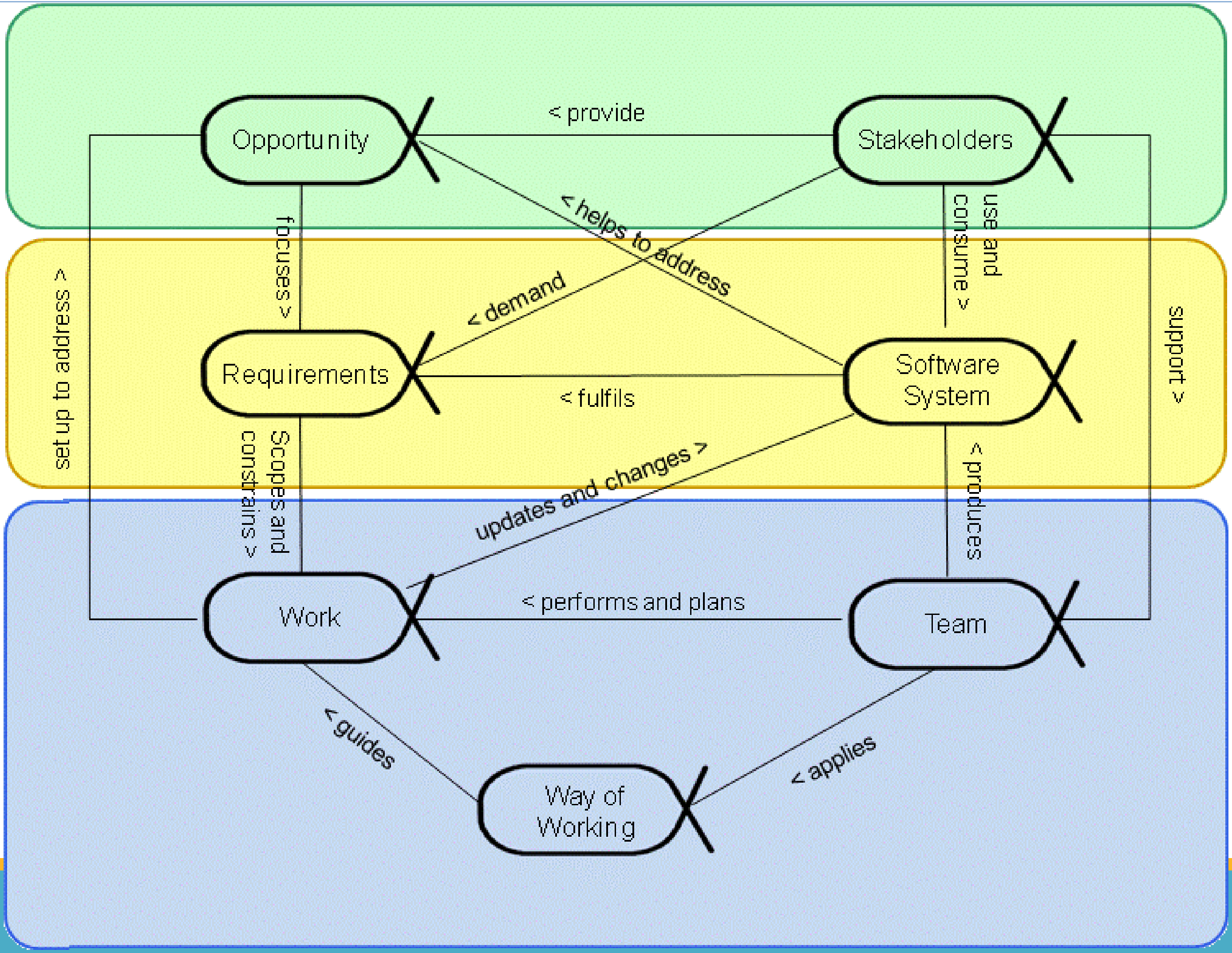
Essence Kernel

- A body of knowledge about software development processes
- Defines a common basis for software development in a <u>scalable</u> and flexible way
  - Different software engineering practices and processes can be composed to different needs
- Can be extended to encompass any recurrent challenge in software engineering

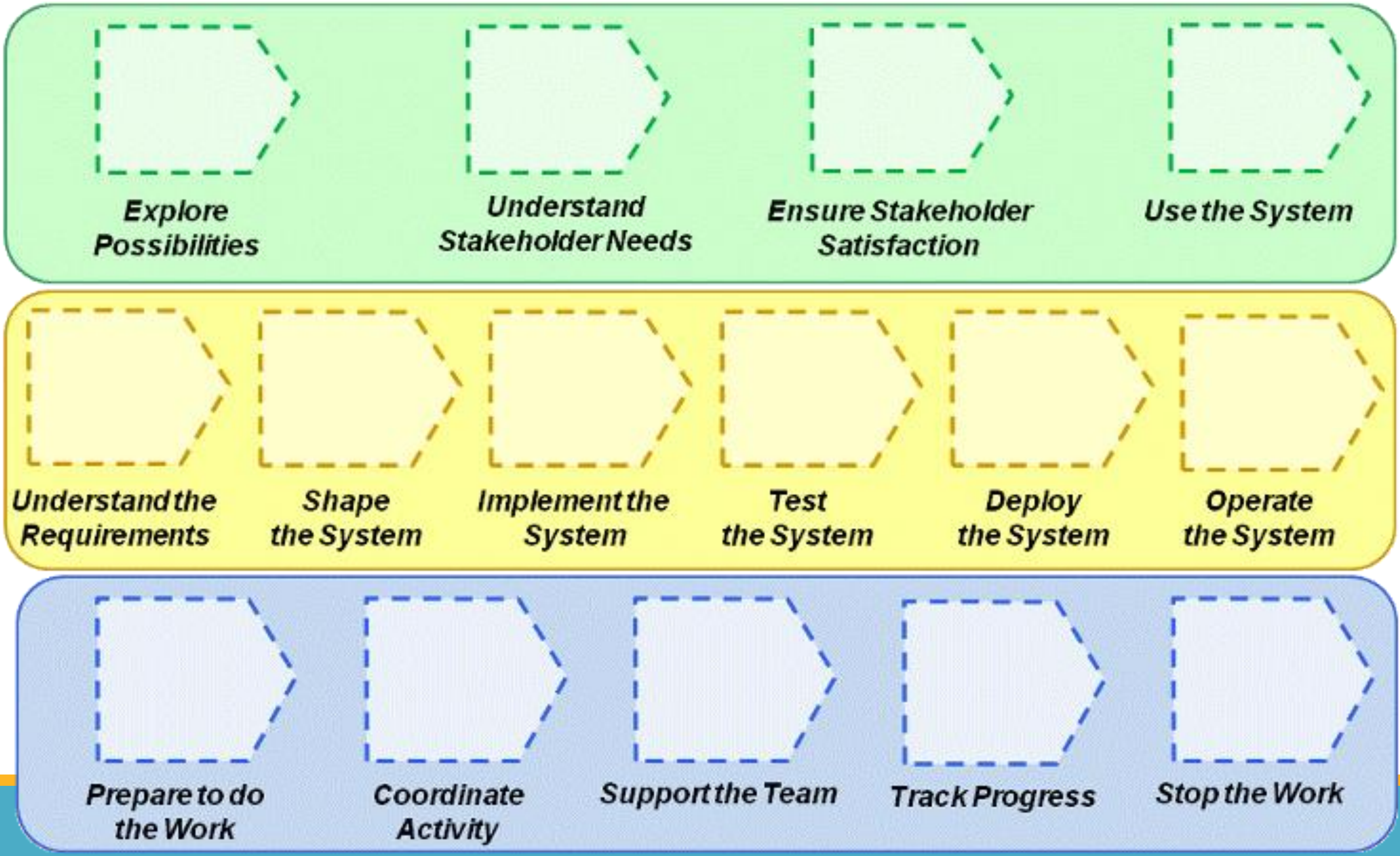
Essence Kernel Areas of Concern



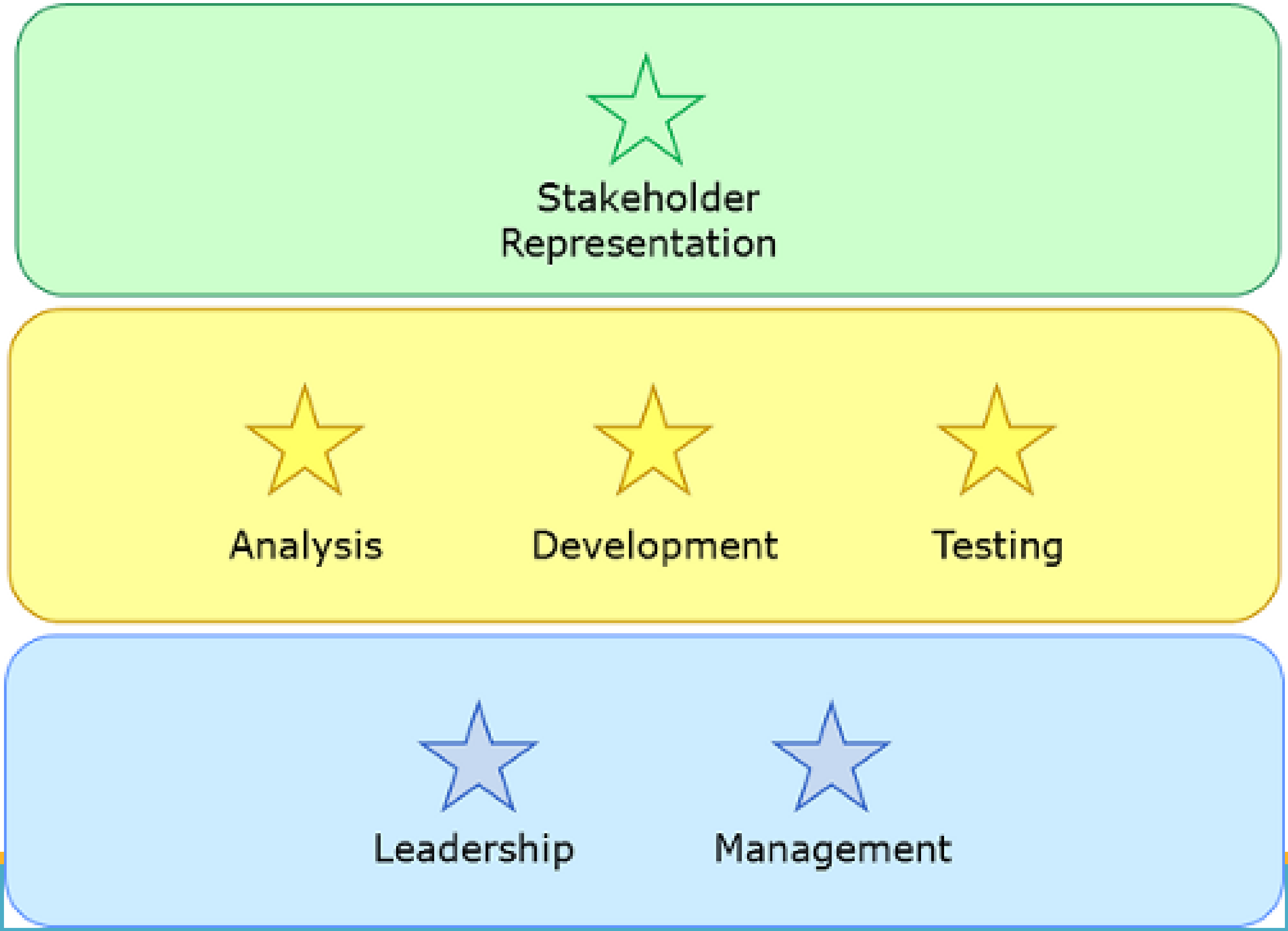
Essence Kernel Alphas



Essence Kernel Actv. Spaces



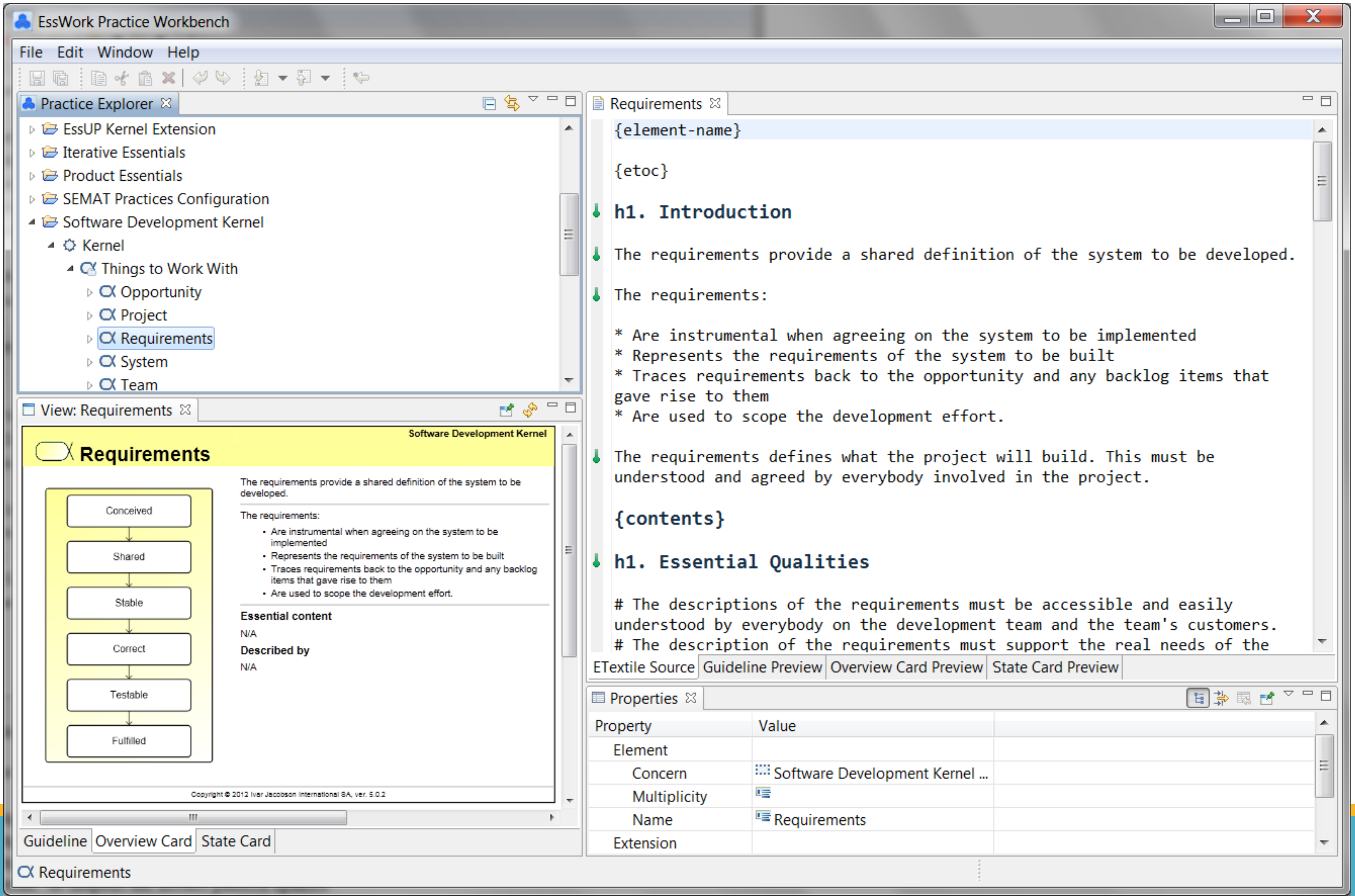
Essence Kernel Competencies



EssWork

- An Integrated Practice Development Environment based on Eclipse
- The main goal is to provide a comprehensive development environment for processes, methods, and practices authoring
- Easy and intuitive to develop and deploy practices based on a the convenient kernel, mainly for Essence Kernel

EssWork



```
{element-name}

{etoc}

h1. Introduction

The requirements provide a shared definition of the system to be developed.

The requirements:

* Are instrumental when agreeing on the system to be implemented
* Represents the requirements of the system to be built
* Traces requirements back to the opportunity and any backlog items that
gave rise to them
* Are used to scope the development effort.

The requirements defines what the project will build. This must be
understood and agreed by everybody involved in the project.

{contents}

h1. Essential Qualities

# The descriptions of the requirements must be accessible and easily
understood by everybody on the development team and the team's customers.
# The description of the requirements must support the real needs of the
```

Software Development Kernel

Requirements

Conceived → Shared → Stable → Correct → Testable → Fulfilled

The requirements provide a shared definition of the system to be developed.

The requirements:

- Are instrumental when agreeing on the system to be implemented
- Represents the requirements of the system to be built
- Traces requirements back to the opportunity and any backlog items that gave rise to them
- Are used to scope the development effort.

Essential content

N/A

Described by

N/A

Copyright © 2012 Ivar Jacobson International SA, ver. 5.0.2

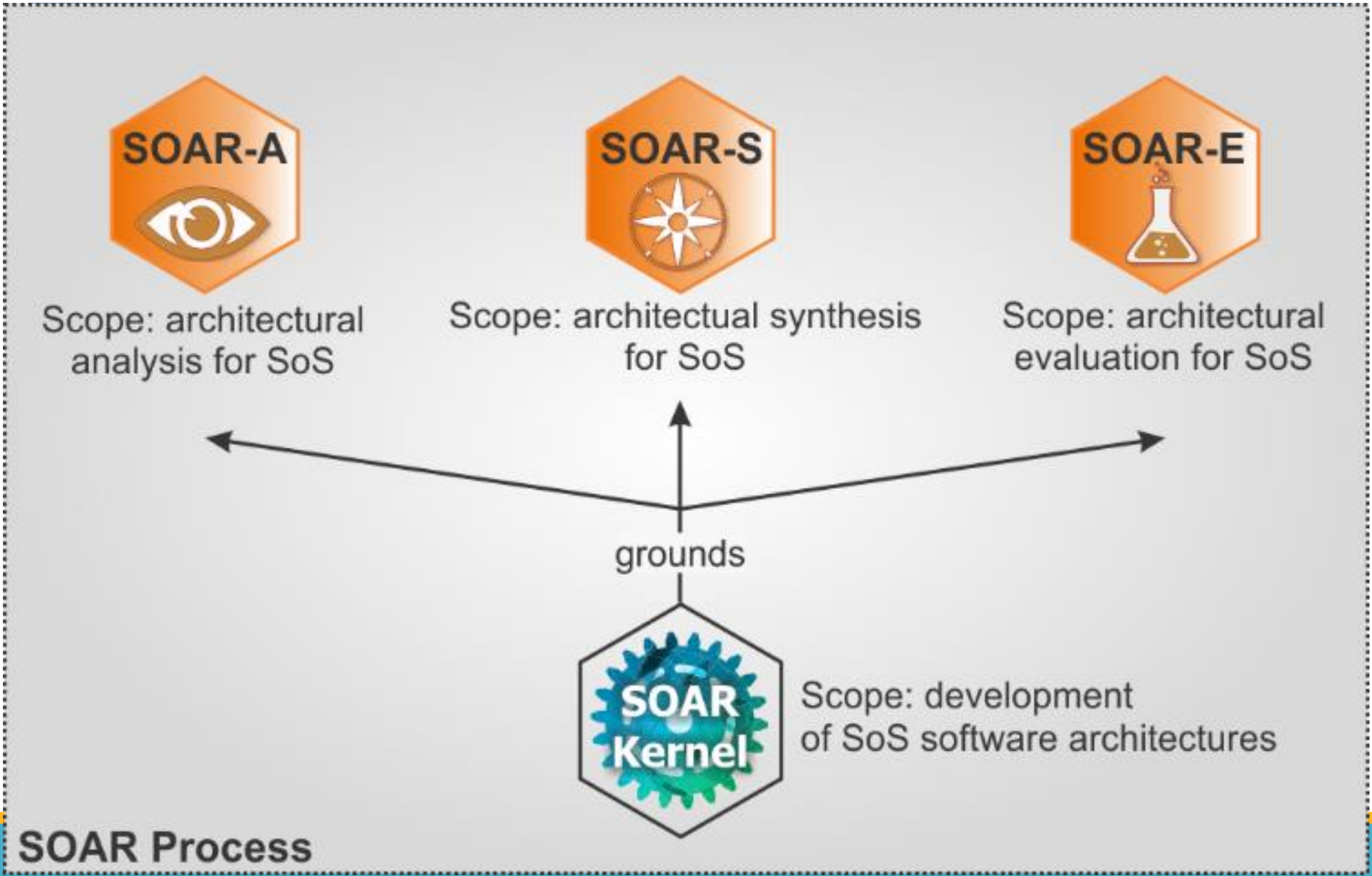
Property	Value
Element	
Concern	Software Development Kernel ...
Multiplicity	
Name	Requirements
Extension	

THE SOAR PROCESS

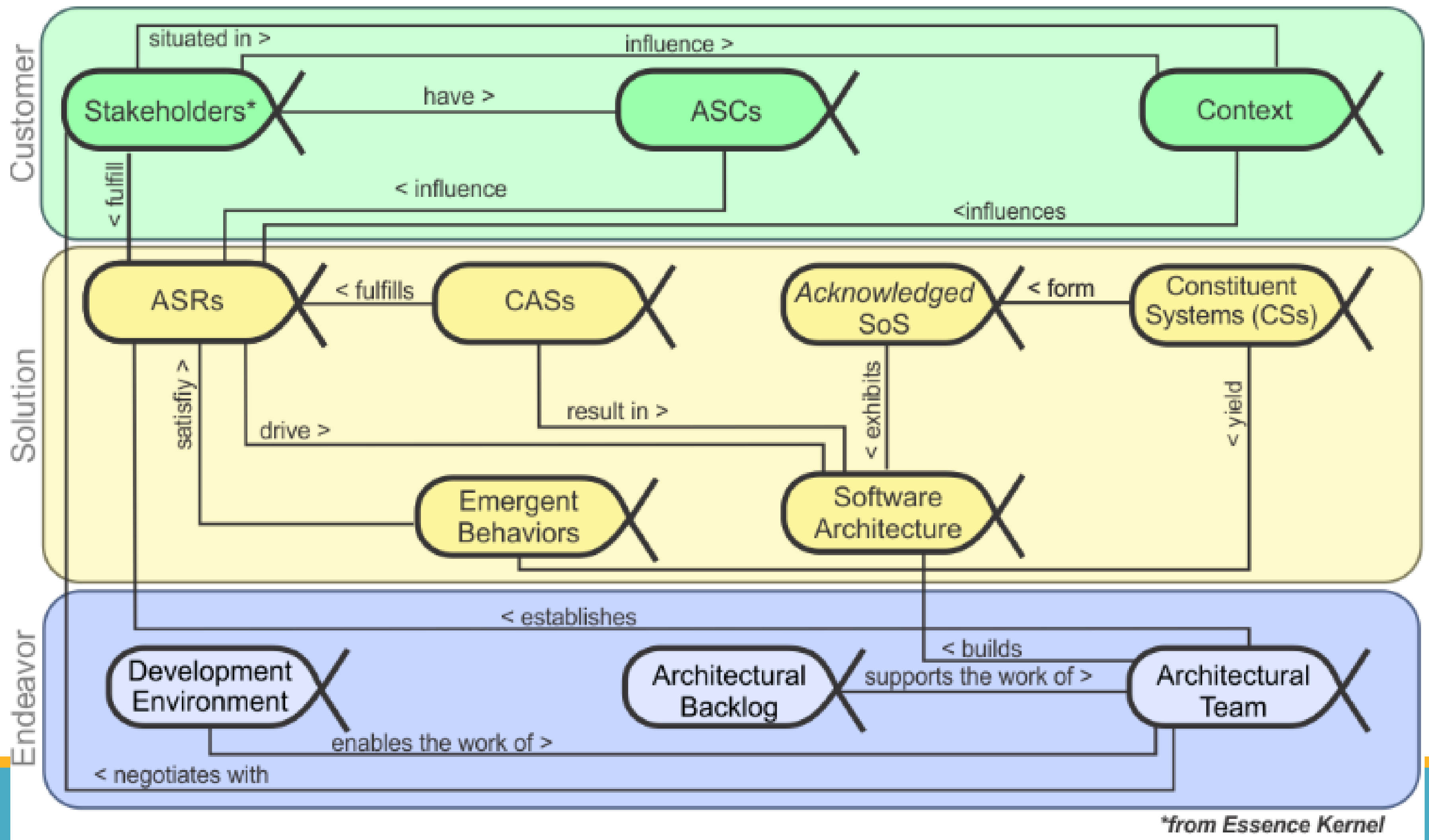
SOAR Process

- Not attached to any specific application domain or technology
  - e.g., architectural style or architectural pattern
- Incremental and evolutionary approach
  - Flexible and dynamic application
- Represented with Essence Language
  - EssWork

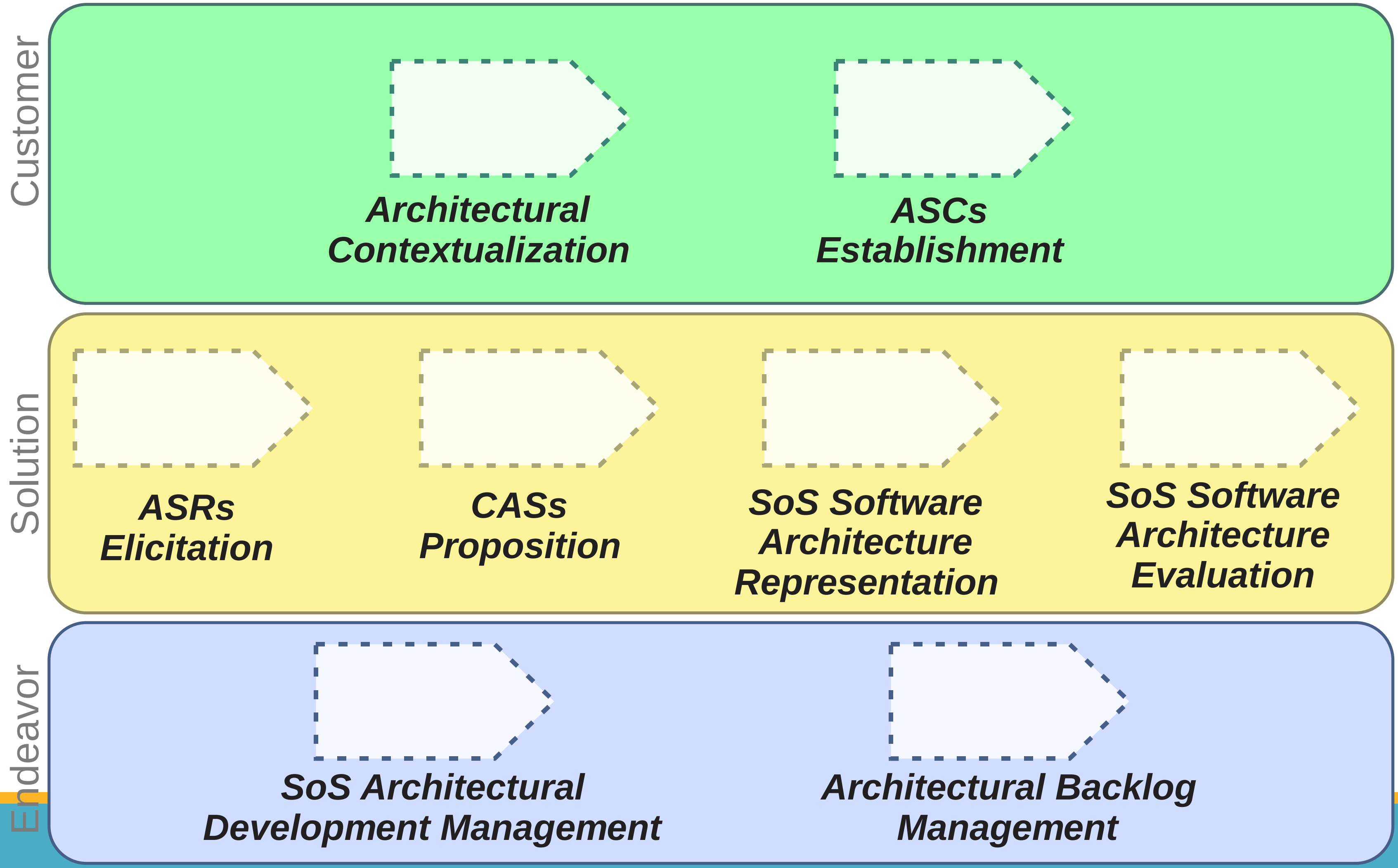
SOAR Process


SOAR-A
Scope: architectural analysis for SoS
SOAR-S
Scope: architectual synthesis for SoS
SOAR-E
Scope: architectural evaluation for SoS
grounds
SOAR Kernel
Scope: development of SoS software architectures
SOAR Process

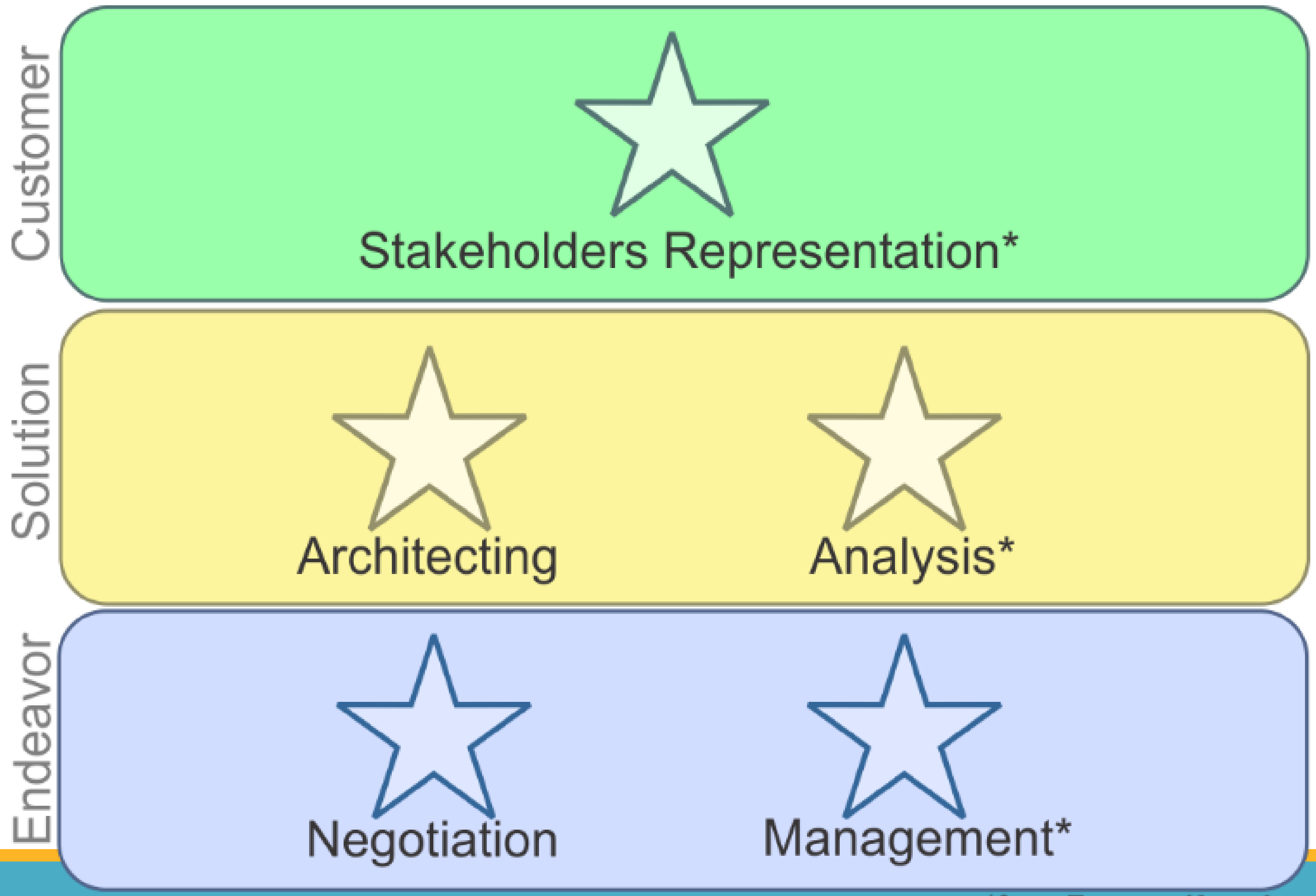
SOAR Kernel - Alphas



SOAR Kernel – Activity Spaces


Customer
Architectural Contextualization
ASCs Establishment
Solution
ASRs Elicitation
CASs Proposition
SoS Software Architecture Representation
SoS Software Architecture Evaluation
Endeavor
SoS Architectural Development Management
Architectural Backlog Management

SOAR Kernel - Competencies

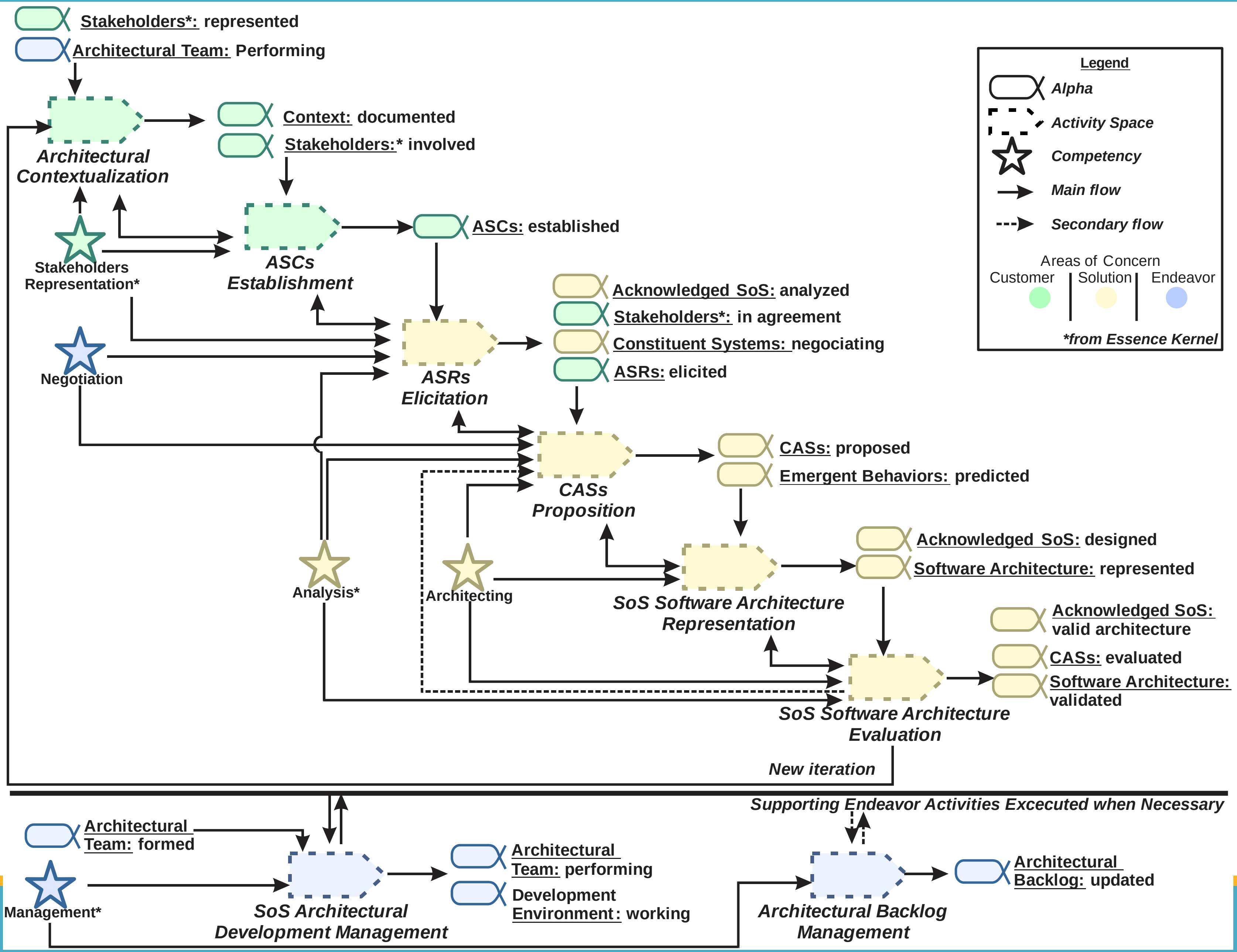


*from Essence Kernel

SOAR Kernel - Competencies

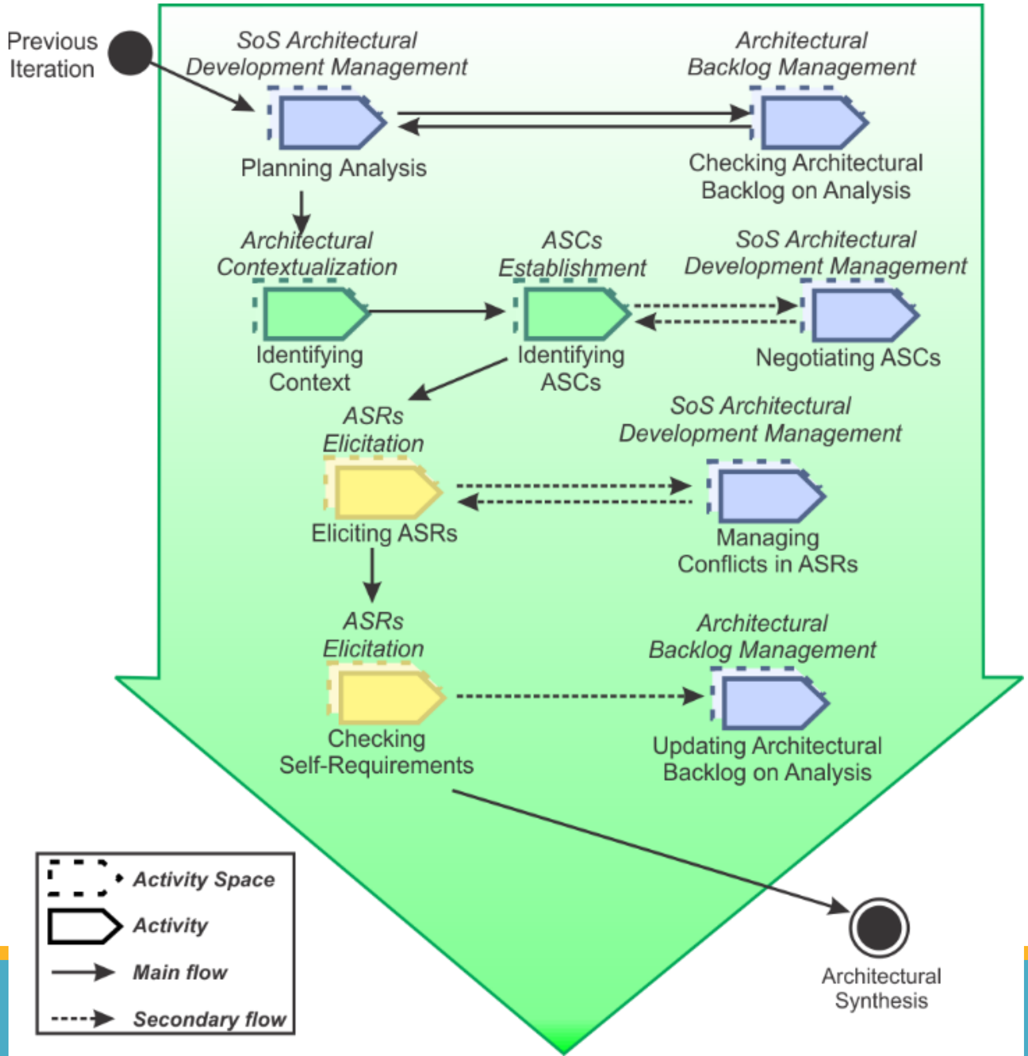
- **Architecting:** the ability to design and represent effective software architectures following the standards and norms agreed by the architectural team.
  - Exploit all the knowledge about the Context, ASCs, and ASRs and balance them by creating appropriate CASs
  - A combination of talent, experience, knowledge, and design skills to develop and maintain the software architecture as expected
- **Negotiation:** the ability to negotiate and reach agreement among different stakeholders in the heterogeneous environment of SoS.
  - It is a combination of technical and personal skills to avoid and manage conflicts reaching the best agreements to SoS development

SOAR Kernel Workflow


Stakeholders*: represented
Architectural Team: Performing
Architectural Contextualization
Context: documented
Stakeholders:* involved
Stakeholders Representation*
ASCs Establishment
ASCs: established
Negotiation
ASRs Elicitation
Acknowledged SoS: analyzed
Stakeholders*: in agreement
Constituent Systems: negociating
ASRs: elicited
CASs Proposition
CASs: proposed
Emergent Behaviors: predicted
Analysis*
Architecting
SoS Software Architecture Representation
Acknowledged SoS: designed
Software Architecture: represented
SoS Software Architecture Evaluation
Acknowledged SoS: valid architecture
CASs: evaluated
Software Architecture: validated
New iteration
Legend
Alpha
Activity Space
Competency
Main flow
Secondary flow
Areas of Concern
Customer
Solution
Endeavor
*from Essence Kernel
Supporting Endeavor Activities Excecuted when Necessary
Architectural Team: formed
Management*
SoS Architectural Development Management
Architectural Team: performing
Development Environment: working
Architectural Backlog Management
Architectural Backlog: updated

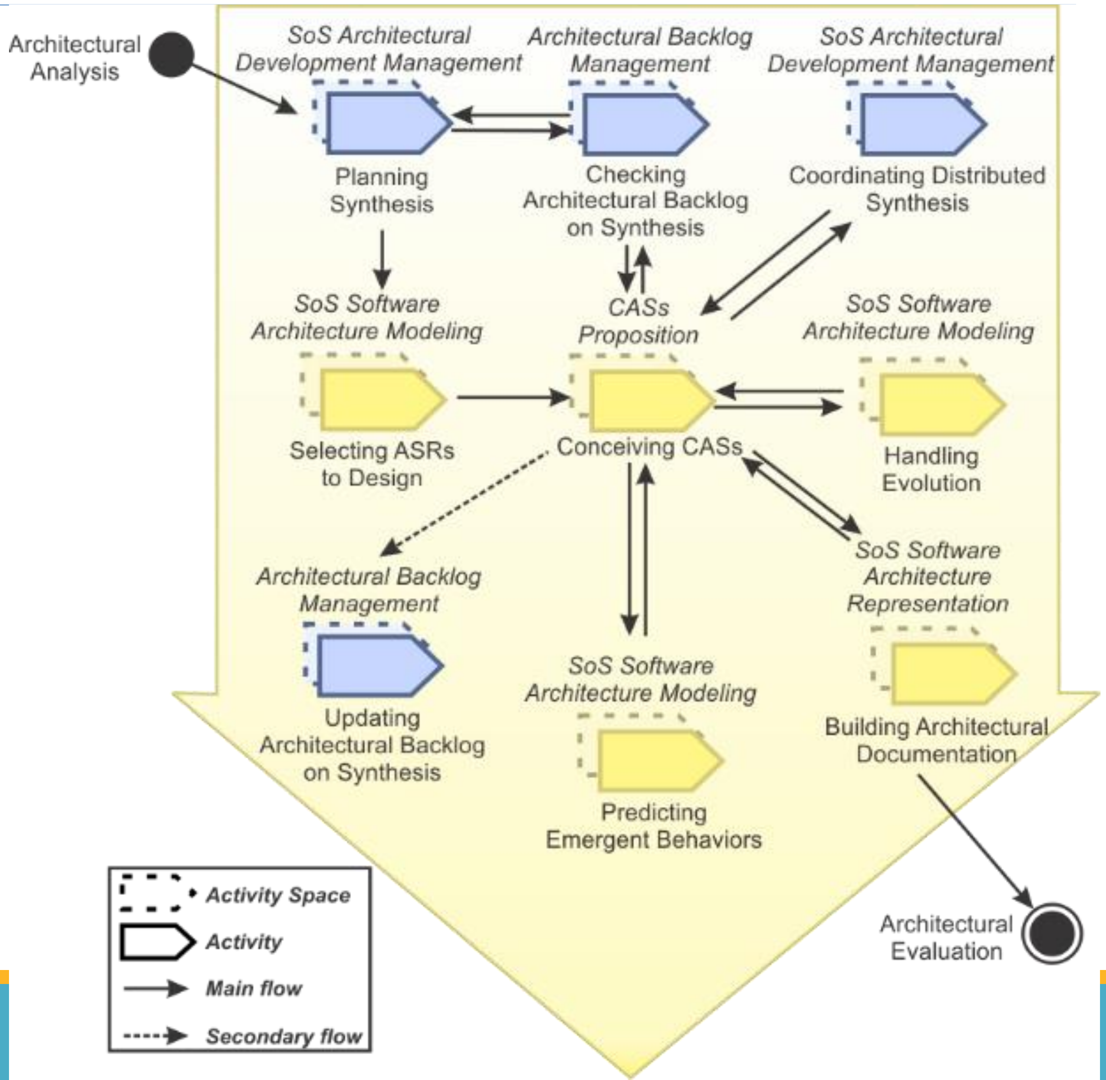
The SOAR-A Practice

- Describes the "how to" to perform architectural analysis in acknowledged SoS


Previous Iteration
SoS Architectural Development Management
Planning Analysis
Architectural Backlog Management
Checking Architectural Backlog on Analysis
Architectural Contextualization
Identifying Context
ASCs Establishment
Identifying ASCs
SoS Architectural Development Management
Negotiating ASCs
ASRs Elicitation
Eliciting ASRs
SoS Architectural Development Management
Managing Conflicts in ASRs
ASRs Elicitation
Checking Self-Requirements
Architectural Backlog Management
Updating Architectural Backlog on Analysis
Architectural Synthesis
Activity Space
Activity
Main flow
Secondary flow

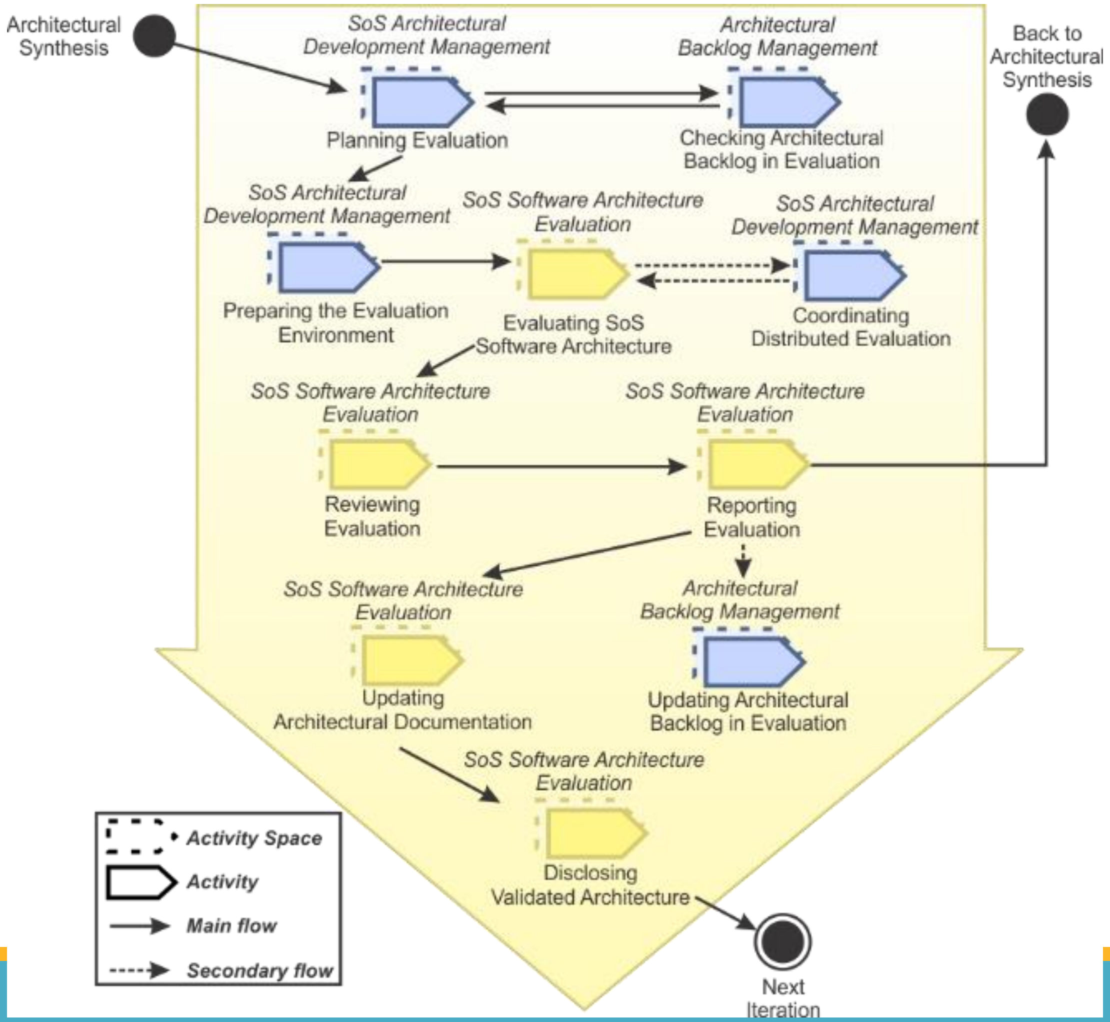
The SOAR-S Practice

- Describes the “how to” to perform architectural synthesis in acknowledged SoS


Architectural Analysis
SoS Architectural Development Management
Architectural Backlog Management
SoS Architectural Development Management
Planning Synthesis
Checking Architectural Backlog on Synthesis
Coordinating Distributed Synthesis
SoS Software Architecture Modeling
CASs Proposition
SoS Software Architecture Modeling
Selecting ASRs to Design
Conceiving CASs
Handling Evolution
Architectural Backlog Management
SoS Software Architecture Representation
Updating Architectural Backlog on Synthesis
SoS Software Architecture Modeling
Building Architectural Documentation
Predicting Emergent Behaviors
Activity Space
Activity
Main flow
Secondary flow
Architectural Evaluation

The SOAR-E Practice

- Describes the "how to" to perform architectural synthesis in acknowledged SoS


Architectural Synthesis
SoS Architectural Development Management
Planning Evaluation
Architectural Backlog Management
Checking Architectural Backlog in Evaluation
Back to Architectural Synthesis
SoS Architectural Development Management
Preparing the Evaluation Environment
SoS Software Architecture Evaluation
Evaluating SoS Software Architecture
SoS Architectural Development Management
Coordinating Distributed Evaluation
SoS Software Architecture Evaluation
Reviewing Evaluation
SoS Software Architecture Evaluation
Reporting Evaluation
SoS Software Architecture Evaluation
Updating Architectural Documentation
Architectural Backlog Management
Updating Architectural Backlog in Evaluation
SoS Software Architecture Evaluation
Disclosing Validated Architecture
Next Iteration
Activity Space
Activity
Main flow
Secondary flow

CONCLUSION AND FUTURE WORK

Conclusions

The SOAR process was built to deliver a comprehensive support for architecting processes of Acknowledged SoS

- Has the adequate level of abstraction to complex development environments of SoS
- A contribution to the development projects of the new, important class of SoS software systems

Future Research Outline

1. Development of specialized versions of SOAR
   - Focus on specific application domains (e.g., smart cities)
   - To integrate new approaches
2. Develop additional elements to be integrated on SOAR (e.g., a consensual quality model for SoS)
   - With the maturation of standard solutions for SoS, SOAR can also evolve enhancing its power of support
3. Extend SOAR in order to encompass other SoS subcategories (i.e., virtual, collaborative, and directed)

SOAR: a general process for Acknowledged SoS Software Architectures

Marcelo Benites Gonçalves (ICMC-Brazil/IRISA-France)

Flavio Oquendo (IRISA-France),

and Elisa Yumi Nakagawa (ICMC-Brazil)