SAL/M

Jump to navigation Jump to search

Contents

- 1 Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Objectives
  - 1.4 Definitions, Acronyms, Abbreviations
  - 1.5 References
- 2 Proposed System
  - 2.1 Overview
  - 2.2 Subsystem Decomposition
  - 2.3 Software/Hardware Mapping
    - 2.3.1 System Performance
      - 2.3.1.1 General System Performance
      - 2.3.1.2 Input/Output Performance
      - 2.3.1.3 Processor Allocation
      - 2.3.1.4 Memory Allocation
      - 2.3.1.5 Memory Allocation
    - 2.3.2 Connectivity
    - 2.3.3 Network Architecture
  - 2.4 Persistent Data Management
  - 2.5 Access control and security
  - 2.6 Global Software Control
  - 2.7 Boundary Conditions
- 3 Subsystem Architectures
- 4 Glossary
- 5 Legal

Introduction

Purpose

SAL/M is a solution that manages the Software and Systems Architecture

Lifecycle.

This document serves to bootstrap the lifecycle of the solution, which will later manage its own design, including the system definition document, within itself using features designed for this.

Scope

SAL/M should cover high level aspects of solution design using only artifacts created/represented inside of its portfolio.

SAL/M is not intended to supplement skill gaps in solution design. It is to faciliate the solution design process, generate the artifacts of that process, and nothing more.

Objectives

- Facilitate an intact distributed systems design process.
- Generate artifacts of that process.
- Represent all available components in a software ecosystem to those ends.
- Represent relationships between those components.
- Generate a compiled artifact on demand of the current state of a solution design.

Definitions, Acronyms, Abbreviations

- **Design** - Representation of the structure and function of all aspects of a system.
- **Component** - A piece of a system that has a function or purpose in that system.
- **Ecosystem** - A collection of systems encompassed within an organizational boundary.
- **System** - A collection of components that serve a collective function.
- **Distributed System** - A system that is distributed across systems.

References

http://www.cs.cmu.edu/afs/cs/project/classes-bob-1/owl-f1996/group/architecture/SDD/Control.html

Proposed System

Overview

SAL/M is a content management system oriented purely for solutions architects to design and represent distributed system components,

distributed systems, and ecosystems collaboratively.

It is intended for all sizes of organizations that wish to design their solutions in a trackable way and generate the design artifacts for the current state at will.

While highly recommended for similar to be used in organizations, the solutions design process, or software architecture lifecycle, is not a business process. All distributed systems must be designed. SAL/M strives to facilitate that.

Subsystem Decomposition

The top level subsystems comprising the system are as a 3-tier web application:

> A database, which acts as the data store for an API. The database selected for the alpha version is postgres.
> An API which facilitates high and low level interaction with the database. The API selected for the alpha version is Flask. Flask is a python API management system.
> A web-based UI, which consumes the API to provide the user with a content management system for managing the design process.

The API tier will have three layers:

> **_DBIO** - One layer will consist of low level, raw I/O with the database.
> **LOGIC** - A more abstract layer will provide the interface for the UI to consume that is oriented to high level SAL/M features.
> **_SECX** - A layer that consumes a separate API from another project, called IDM/A, will provide the security layer.

The UI tier will consist of static html and javascript. No server-side scripting is intended for this tier. This is deliberate to partition responsibility and scope cleanly across all the top level subsystems.

Software/Hardware Mapping

All pieces will rest upon runtimes available on most Linux systems. As SAL/M is a project under development, it has no existing hardware or software systems. The top level subsystems will provide demarcation capability between hosts if desired, but could run entirely on a single bare metal host running linux, on one or multiple VMs distributed by layer, or, with the exception of the data layer (due to being stateful), in linux containers managed by a container orchestration system.

The database layer, as this is a postgres installation, should be run on a semi-permanent host or as otherwise recommended by official postgres documentation.

System Performance
General System Performance
Input/Output Performance
Processor Allocation
Memory Allocation
Memory Allocation
Connectivity
Network Architecture
Persistent Data Management
Access control and security
Global Software Control
Boundary Conditions

Subsystem Architectures

Glossary

Legal

Design, spec, system definition, all communications relevant to SAL/M and all pieces, representations, implementations relevant to SAL/M belong exclusively to SILO GROUP, LLC. and Chris Punches except where explicitly noted by license inclusion in any source repositories. All rights reserved and all rights intended to be enforced.

Retrieved from "https://wiki.silogroup.org/index.php?title=SAL/M&oldid=238"

This page was last edited on 26 July 2020, at 17:16.

Content is available under Attribution-NonCommercial-NoDerivatives 4.0 International unless otherwise noted.