Extending SON To Clouds And Things

GOSSONoT: A Generalized Open-Source Self Organizing Network of Things Platform

Article Format Of Presentation

Document #PLPC-180052
Version 0.5
February 12, 2019

This Document is Available on-line at:
http://www.by-star.net/PLPC/180052

Mohsen BANAN
Email: http://mohsen.1.banan.byname.net/contact

Contents

List of Figures

Part I

Overview

1 Summary

Lessons Learned From SON In Telecom's Context:

Our experience with Self-Organizing Networks (SON) in Telecom's context over the past decade has demonstrated that very large networks can be successfully managed when:

- Interfaces to network elements are well defined (OSS and MOs).
- Concept Of SON-Modules Is Widely Well Understood And Considered Central.
- Proper SON-Platforms Are Deployed Through Out The Network.
- Systems Management Efforts Are Focused On Consistent SON-Modules Development.

Extending SON:

These Lessons Can Be Applied To Managing Other Large Networks (Clouds).

Notes:

2 Strategy

Pillars:

- Generalize the concept of SON-Modules such that **All** Systems Management activities can be implemented as SON-Modules.
- Use Generalized SON-Modules To Also Implement Network-Element-Adapters.
- Provide An Open-Ended Framework For Development And Execution Of SON-Modules

Realization/Implementation:

Our Interactive Commands Module (ICM) model allows for any type of Systems Management processing.

GOSSONoT is a powerful open-ended modules execution framework.

Notes:

3 Key Differentiators

Unified, Converged, Simplified And Open-Sourced

Unlike Most Other Cloud Management Approaches, GOSSONoT:

- Is based on the real experience of SON.
- Is purely based on Python-ICMs which are cohesive and unified.

In Contrast:

Other approaches to Cloud Management usually bloat, diverge and implode.

Notes:

4 Related Documents

Interactive Command Modules (ICM) and Players A Framework For Cohesive Generalized Scripting http://www.by-star.net/PLPC/180050 — [4]

Remote Operations Interactive Command Modules (RO-ICM) Best Current (2019) Practices For Web Services Development http://www.by-star.net/PLPC/180056 — [3]

A Generalized Swagger (OpenAPI) Centered Web Services Invocations And Testing Framework http://www.by-star.net/PLPC/180057 — [1]

Extending SON To Clouds And Things GOSSONoT: A Generalized Open-Source Self Organizing Network of Things Platform http://www.by-star.net/PLPC/180052 — [2]

Notes:

5 Part Of A Much Bigger Picture – ByStar and BISOS

GOSSONoT is Part Of A Much Bigger Picture.

GOSSONoT Is Part Of: The Libre-Halaal ByStar Digital Ecosystem And Part Of: BISOS: ByStar Internet Services OS

GOSSONoT is primarily being used and developed in that context.

Notes:

6 About This Document (Presentation/Podcast)

You can obtain this document at its access page: http://www.by-star.net/PLPC/180052
where it is available in multiple forms and multiple formats:

- **Article/Book Form**: Best suited for cover-to-cover reading (pdf).
  - **Pdf Format**: Best suited for printing and cover-to-cover reading.
  - **HTML/Web Format**: Best suited for Web reading and cross referencing.
- **Presentation Form**: Best suited for quick scan – with live URLs –(pdf).
  - **Screencast**: A slide oriented voice-over narrated presentation (Reveal.js Based)
  - **PDF Slides**: Best suited for printing of the slides (Beamer Generated)
  - **HTML Slides And Notes**: Slide and notes in html format (Beamer+HaVeA Generated)
  - **PDF Slides and Notes**: Best suited for printing of presentation notes (Beamer Generated)

We can benefit from your feedback. Please let us know your thoughts. You can send us your comments, corrections and criticisms to `mailto:feedback@mohsen.1.banan.byname.net`

7 Document Outline

- Generalizing SON For Clouds And Things
- Overview Of GOSSONoT
- GOSSONoT Software Architecture – Installation And Usage
- GOSSONoT-Modules And Interactive Command Modules
- GOSSONoT-Things-Interfaces And GOSSONoT-Things-Lists
- GOSSONoT-Modules Execution User Interfaces And Environments
- Use Case Examples

Notes:

Part II

Generalizing SON For Clouds And Things

8 Obvious Desires – Self Organizing Networks

Obvious Desires

Self Organizing Networks

Any Operator Of Any Network Wants Her Network To Be:

- Self-Configuring
- Self-Optimizing
- Self-Healing

Wishes Vs Reality

But, that is mostly fantasy and usually involves more work than imagined. Is it reasonable to abstract a solution that spans multiple network types?

Can SON Be Extended?

The concept of Self Organizing Networks (SON) originated in the well structured and standardized Cellular-Mobile Networks. In that scope, SON is very real. Are those same concepts and models applicable to Clouds?

Notes:

9 About SON In The Telecom Context

In The Telecom Context, SON Is Very Real:

The idea and concepts of Self Organizing Networks (SON) started to be formalized in 3GPP at around 2006. First generation of SON products started to appear in 2009.

All major Telecom equipment manufacturers (Nokia, Ericsson, Huawei) have a SON product offering. Cisco also have a strong product offering. SON products are usually Multi-Technology/Multi-Layer (2G/3G/4G/5G) and Multi-Vendor with respect to OSS infrastructure interfaces (Nokia, Ericsson, Huawei).

Every carrier (ATT, T-Mobile, Orange, Verizon) has a SON Solution.

Notes:

9.1 Telecom SON Environment: Clean And Standardized Managed Objects

---

Telecom SON Environment:

Clean And Standardized Managed Object Definitions Have Been In Place

Telecom's SON Builds On Formalized Definitions Of Managed Objects(MOs):

X.700 – Common Management Information Protocol (CMIP) – Started to define the Telecom's Network Management model with formal Managed Objects in 1988 (blue-books).

3GPP has kept that formal standardized tradition for 3G, 4G and 5G in the well protected TelePhants walled-garden environment.

Operations Support Systems interoperability initiative (OSSii) is the foundation of SON.

MOs As SON Enablers

It is this formal definition of Managed Objects that has made SON successful.

Notes:

9.2 Nokia's SON Product: EdenNet

Nokia SON Product:

EdenNet

`https://networks.nokia.com/solutions/edennet`

- Multi-Technology: 2G/3G/4G/5G
- Multi-Vendor OSS Interfaces: Nokia, Ericsson, Huawei
- Python Based
- SON-Modules-Platform Model
- Large Library Of Proprietary SON-Modules

Notes:

9.3 Ericsson's SON Optimization Manager

https://www.ericsson.com/us/ourportfolio/network-management/son-optimization-manager

- Multi-Technology: 2G/3G/4G/5G
- Multi-Vendor OSS Interfaces
- "Use-Case" Paradigm

Notes:

9.4 Huawei's SON Product: SingleSON

Huawei SON Product:
SingleSON
http://carrier.huawei.com/en/products/wireless-network/subsolution-singleoss/singleson

- Multi-Technology: 2G/3G/4G/5G
- Multi-Vendor OSS Interfaces

Notes:

9.5 Cisco SON Suite

https://www.cisco.com/c/en/us/products/wireless/son-suite/index.html

- Multi-Technology: 2G/3G/4G/5G
- Multi-Vendor OSS Interfaces
- Model: SON-Apps – Modules – Use Cases
- Large Library Of SON Applications
- Also supports Packet core, ANDSF.

Notes:

10 The Equivalent Of SON In Clouds Context

- A Whole Lot Of Standalone And Non-Integrated Open-Source Packages (“Management Components”) That Are Not Made To Fit Together.
- Each cloud provider tries to integrate these components.
- Lack of standardization at Managed Objects level.
- Only basic commonality and standardization at Linux and distros level
- No equivalent to SON Modules

Notes:

11 SON Functions

SON functionalities are commonly divided into three groups:

- Self-configuration functions: Network elements and systems are to conform to the “plug-and-play” paradigm.
- Self-optimization functions: Network elements and systems are to be “monitored” and “adjusted” towards optimum performance.
- Self-healing functions: When network elements and systems become inoperative or mis-perform, fault-management and self-healing mechanisms aim at reducing the impacts from the failure. For example, by re-routing traffic and re-adjusting load balancers. Identifying failures in a timely manner is primary goad of Self-healing functions.

Notes:

12 Typical Anatomy Of SON Platforms

SON Platforms typically have a number of common characteristics and features:

- A unified processing language – Often Python.
- A consistent set of network elements interfaces and systems interfaces – Often abstracted as Things-Adapters / Things-Interfaces.
- A “SON-Modules-Development Framework” with which monitoring and adjusting functionality can be implemented – using Things-Adapters.

- A “SON-Modules-Dispatch Framework” functioning as a user-interface for triggering execution of SON-Modules.
- A “SON-Modules-Execution Framework” through which large scale parallel execution of SON-Modules is managed. For Audit-Control purposes full information about each instance of execution is kept.
- A “SON-Modules-Results-Analysis Framework” through which visualization of results is addressed.

Notes:

13 Use Of SON Modules In Conjunction With Machine Learning

SON Platforms are often used in conjunction with specialized machine-learning engines.

- SON-Modules Are Used To Monitor Network Elements And Systems And Extract Relevant Information
- The Extracted Information Is Fed To The “Big Data” Platform
- Machine-Learning Engines process the SON Extracted Information And Identify Improvements.
- SON-Modules Are Used To Apply The “Adjustments”.

Notes:

14 Extending SON To Clouds And Things

SON (Self-Organizing Network) has thus far:

- Been limited to the realm of TelePhants (Telecom Elephants)

  TelePhants Operators: Verizon, AT&T, T-Mobile, Sprint, Orange

  TelePhants Suppliers: Nokia, Ericsson, Huawei, Cisco

It is possible to extend SON such that its “Managed Objects” are “Abstract Things” which include Cloud’s network elements and systems and IoT entities.

Notes:

14.1 Culture Of TelePhants Vs Culture Of Cloud Operators

Culture Of TelePhants and Culture Of Cloud Operators often stand separate and distinct, even when an organization has both.

Notes:

Culture Of TelePhants (Caricatured)

Culture Of TelePhants – Caricatured

- Old School – TelePhants Operators Remain Dumb, Fat And Happy – TelePhant Suppliers provide the technology and do much of the work under contract. A Convenient Milk and Be-Milked Arrangement.
- Co-Opetition – Through 3GPP things are well standardized and remain inside the proprietary collective walled-garden. Little is re-invented technology moves forward as a collective.

Notes:

Culture Of Cloud Operators (Caricatured)

Culture Of Cloud Operators – Caricatured

- New School – Many Cloud Owners are both Cloud Operators and Cloud Technology Suppliers. Dynamics are: trendy, chaotic, fast-moving, re-inventive, unorganized and inconsistent.
- Private Walled Gardens: Google, Facebook, Amazon, Microsoft, etc; keep re-inventing their own infrastructures. Much Open-Source is bastardized. Late and little Open-Source is given back. After the fact standardization happens at IETF. Things move fast but often go side ways.

Notes:

14.2 Best Of Both Worlds (For TelePhants And Cloud Operators)

There are many lessons that Cloud Operators can learn from TelePhants:

Build On SON's Proven Success

- Identify SON's model as a universal foundation for Cloud Management.

There are many lessons that TelePhants can learn from Cloud Operators:

Recognize The Extended Scope Of SON

- TelePhant Operators can do a whole lot on their own with the right Open-Source Platforms.
- GOSSONoT allows for SON to be applied to their entire network – if Radio-Heads could see beyond RAN.

Notes:

Part III

Overview Of GOSSONoT

15 Our Goals And Motivations For Extending SON To Clouds And Things

---

Our Goals And Motivations

For Extending SON To Clouds And Things

Based on the lessons learned from the experience of the past decade with SON in the Telecom's context and the availability of large and potent relevant Open-Source components, we want to:

- Build GOSSONoT: A Modules Oriented Open-Source SON-Platform
- Develop A Large Collection Of Things-Adapters and Things-Agents (Things-Interfaces) For Network-Elements And Systems Within Clouds and IoT.
- Develop A Rich Library Of SON-Modules That Use Things-Interfaces to Monitor, Optimize and Heal Things.
- Develop A Set Of SON-Modules That Can Feed Corresponding Machine-Learning Engines.
- Develop A Set Of SON-Modules That Can Act On Behalf Of Corresponding Machine-Learning Engines To Adjust Things.

Notes:

16 About GOSSONoT

About GOSSONoT

Open-Source + SON + Cloud + IoT

GOSSONoT (Generalized Open-Source Self-Organizing Network of Things) Is A Platform That Is:

- Purely Implemented In Python.
- Purely Based On Free and Open Source Software And Services (FOSSS).
- Implements The SON-Modules Model Based On SON's Telecom Experience.
- Is Designed For Web-Scale.
- Can Be Used To Manage Cellular-Mobile Entities and Cloud Entities And IoT Entities in an expandable model.

Notes:

17 GOSSONoT's Hour Glass Model

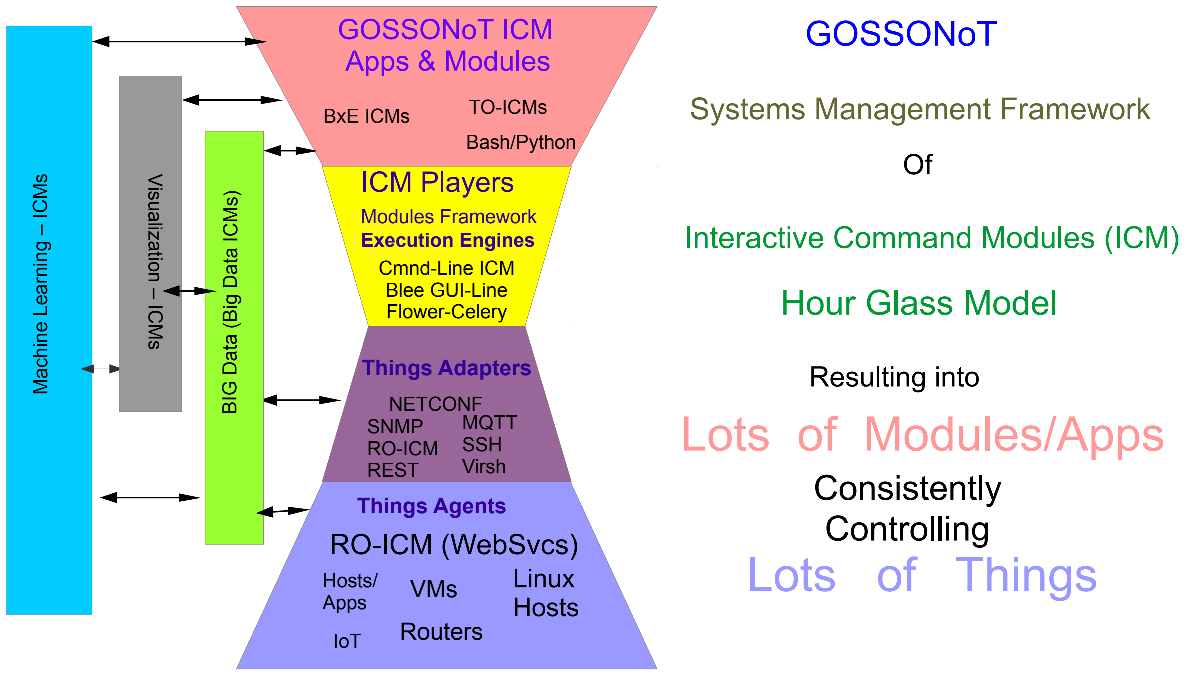

Figure 1: GOSSONoT's Hour-Glass Model

Notes:

18 Scope And Scale Of GOSSONoT

Scope:

- All Linux Based Network Elements And Systems Within A Cloud
- All Management Aspects: Configuration, Optimization, Fault Detection and Healing

Scale:

SON's model, architecture and implementations have proven to scale in largest Telecom operator's networks.

Notes:

19 GOSSONoT As Cloud's Management Convergence Point

Scope and scale of GOSSONoT presents it as a "Convergence Point" for all systems management activities of a Cloud.

Over time all ad-hoc scripts and isolated management functions can be brought to become GOSSONoT-Modules and GOSSONoT-Apps.

Notes:

20 An Overview Of GOSSONoT Architecture

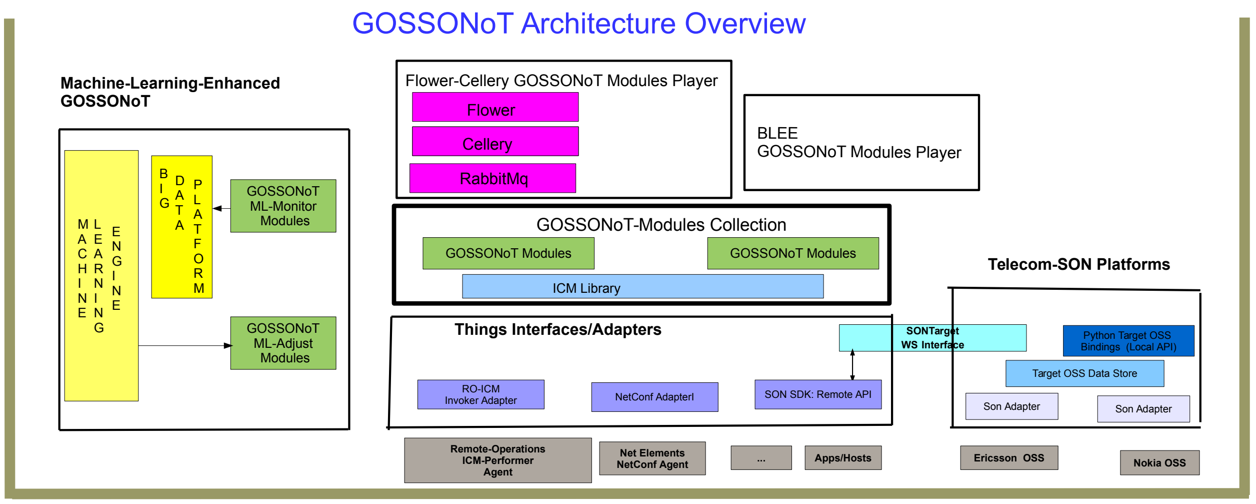
GOSSONoT Architecture Overview
Machine-Learning-Enhanced GOSSONoT
MACHINE LEARNING ENGINE
BIG DATA PLATFORM
GOSSONoT ML-Monitor Modules
GOSSONoT ML-Adjust Modules
Flower-Cellery GOSSONoT Modules Player
Flower
Cellery
RabbitMq
BLEE GOSSONoT Modules Player
GOSSONoT-Modules Collection
GOSSONoT Modules
GOSSONoT Modules
ICM Library
Things Interfaces/Adapters
RO-ICM Invoker Adapter
NetConf AdapterI
SON SDK: Remote API
SONTarget WS Interface
Telecom-SON Platforms
Python Target OSS Bindings (Local API)
Target OSS Data Store
Son Adapter
Son Adapter
Remote-Operations ICM-Performer Agent
Net Elements NetConf Agent
...
Apps/Hosts
Ericsson OSS
Nokia OSS

Figure 2: GOSSONoT Architecture Overview

Notes:

20.1 Main Ingridients Of GOSSONoT Architecture

As shown in the preview figure, GOSSONoT architecture consists of:

- GOSSONoT-Modules
- GOSSONoT-Modules-Player
- GOSSONoT-Apps

- GOSSONoT-Things-Adapters
- GOSSONoT-Things-Agents
- GOSSONoT-Things-Proxies
- Machine-Learning-Enhanced-GOSSONoT

Notes:

Part IV

GOSSONoT Software Architecture – Installation And Usage

21 GOSSONoT Software Components

- Modules Dispatch Sub-System
- Remote Operations – Web Services Sub-System
- Modules (ICM and GOSSONoT) Framework – Module Players And Development Environment
- GOSSONoT-Modules Library
- Things-Interfaces Collection – Things-Adapters and Things-Agents

---

Notes:

21.1 Modules Dispatch – Software Ingredients

GOSSONoT-Modules Dispatch (1 of 2)

Software Components

Flower: Celery monitoring tool

Celery Flower is a tool for monitoring celery tasks and workers.

https://flower.readthedocs.io/en/latest/ – https://github.com/mher/flower

```
pip install flower
```

Celery: Distributed Task Queue

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.

http://www.celeryproject.org/

```
pip install celery
```

Notes:

GOSSONoT-Modules Dispatch (2 of 2)

Software Components

RabbitMQ: Message Broker

RabbitMQ is an intermediary for messaging. It gives your applications a common platform to send and receive messages.

https://www.rabbitmq.com/

```
sudo apt-get install rabbitmq-server
```

Notes:

21.2 Remote Operations – Web Services

Remote Operations – Web Services – (1 of 2)

Software Components

Swagger – OpenAPI: Automating And Formalizing REST APIs Creation And Consumption

Swagger is a set of tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs.

https://swagger.io/docs/specification/about/

```
git clone https://github.com/swagger-api/swagger-codegen
```

Bravado: Automated REST Client

Bravado is a python client library for Swagger 2.0 services. It aims to be a complete replacement to swagger codegen for invokers.

https://github.com/Yelp/bravado

```
pip install bravado
```

Notes:

Remote Operations – Web Services – (2 of 2)

Software Components

Authlib: Python library For building OAuth

JWS, JWK, JWA, JWT are supported.

https://authlib.org/

```
pip install Authlib
```

Notes:

21.3 ICM (Interactive Command Modules) – Software Components

ICM (Interactive Command Modules) (1 of 2)

Software Components

ICM: Interactive Command Modules Unified Model

A Framework For Cohesive Generalized Scripting. A Model For GUI-Line User Experience.

http://www.by-star.net/PLPC/180050

```
pip install unisos.icm
```

RO-ICM: Remote Operations Interactive Command Modules

ICMs can be auto-converted to become invokable as web services.

http://www.by-star.net/PLPC/180056

```
pip install unisos.mmwsIcm
pip install roPerf
```

Notes:

21.4 GOSSONoT-Modules Library And Things Adapters Collection – Software Components

---

ICM (Interactive Command Modules) (1 of 2)

Software Components

ICM: Interactive Command Modules Unified Model

A Framework For Cohesive Generalized Scripting. A Model For GUI-Line User Experience.

http://www.by-star.net/PLPC/180052

```
pip install bisos.gossonot
```

Notes:

22 Integrated Software – Installation

There are several different ways of installing GOSSONoT.

The most convenient way is use bisos.bootstrap to create a fresh VM with all components in place and integrated.

Notes:

23 Current Status Of GOSSONoT Software

GOSSONoT's proof-of-concept and prototyping date back to 2010

First alpha public release of GOSSONoT was in 2017.

GOSSONoT is being currently used and developed in The Libre-Halaal ByStar Digital Ecosystem.

At this time GOSSONoT should only be viewed as an early alpha release. Incremental public release will be made publicly available.

Notes:

24 GOSSONoT's Organic Model – Not A Monolithic Paradigm

GOSSONoT is architected to be a set of collaborative and loosely tied components. We avoid the monolithic paradigm.

What ties everything together are the following:

- The Pure Python Strategy
- Use Of Only Open-Source Core Components
- ICM Centered And ICMs Everywhere Strategy
- Unix Philosophy

GOSSONoT is designed to be ever growing.

Notes:

25 Growth Dynamic Of GOSSONoT

GOSSONoT is based on an open-ended design. We anticipate that it will be used in ways that we can not foresee. Obvious growth areas include:

- GOSSONoT-Modules – ICMs
- Things-Interfaces Pairs: Things-Adapters and Things-Agents – And Particularly Remote-Operations-ICM based Things-Adapters
- GOSSONoT-Modules and ICM Players and GOSSONoT-Apps
- Interfaces and Integrations With Machine-Learning Enhancements

---

Notes:

Part V

GOSSONoT-Modules And Interactive Command Modules (ICMs)

26 GOSSONoT-Modules Are Specializations Of Interactive Command Modules (ICMs)

GOSSONoT's Generecities And Universalities Are Based On ICMs

ICMs are general purpose "Commands" that contain within themselves full information about the format and structure of their inputs and outputs.

On demand, ICMs can report their input and output structures.

ICMs contain a set of (usually related) Commands that are only limited by Python and available libraries.

Notes:

27 ICM Software And Documentation

ICM Software

https://github.com/unisos-pip/icm

```
pip install unisos.icm
```

ICM Documentation

> **Unified Python Interactive Command Modules (ICM) and ICM-Players A Framework For Development Of Expectations-Complete Commands A Model For GUI-Line User Experience** http://www.by-star.net/PLPC/180050 — [4]

Notes:

28 ICM Framework, Modules And Players

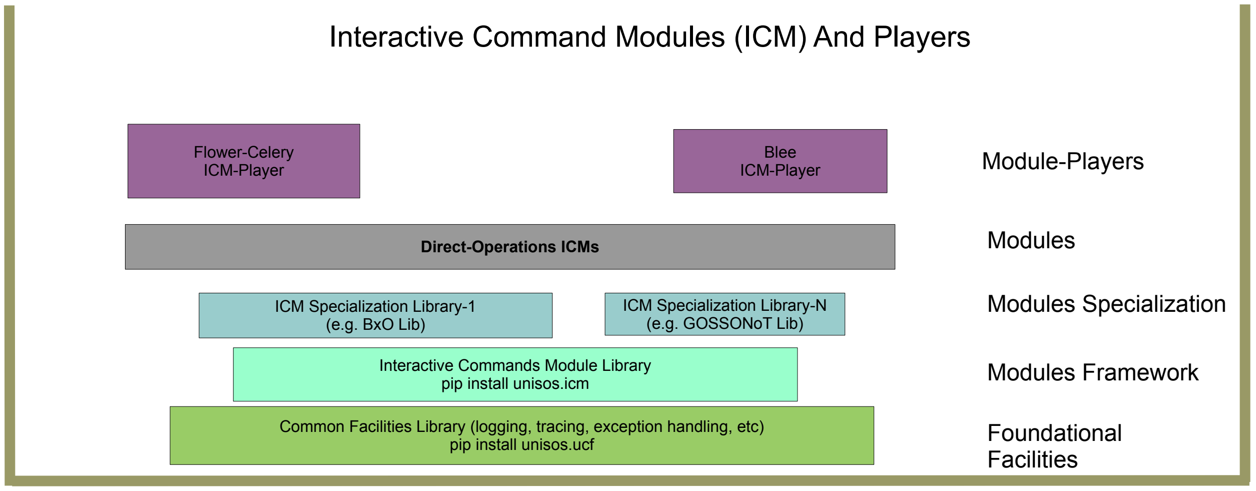
Interactive Command Modules (ICM) And Players
Flower-Celery ICM-Player
Blee ICM-Player
Module-Players
Direct-Operations ICMs
Modules
ICM Specialization Library-1 (e.g. BxO Lib)
ICM Specialization Library-N (e.g. GOSSONoT Lib)
Modules Specialization
Interactive Commands Module Library pip install unisos.icm
Modules Framework
Common Facilities Library (logging, tracing, exception handling, etc) pip install unisos.ucf
Foundational Facilities

Figure 3: ICM Framework, Modules And Players

Notes:

Part VI

GOSSONoT-Things-Interfaces And GOSSONoT-Things-Lists

29 Abstraction Of Things

- Manageable Things with Things-Interfaces
  - Things-Adapters (RO-ICM-Invoker)
  - Things-Agent (RO-ICM-Performer)
- Things-Lists
  - Things-Interfaces-List
  - Things-Interfaces-Parameters

Notes:

30 Things-Interfaces: Primary Things-Adapters And Things-Agents Protocols

---

- Web-Services ICMs – (Swagger Based RO-ICM-Invoker RO-ICM-Performer)
- SNMP
- NETCONF
- SSH
- MQTT (IoT)

Notes:

30.1 Web Services – Remote Operations Interactive Command Modules (RO-ICM)

Direct Operations Interactive Command Modules (DO-ICM)

We call an ICM that invokes local operations (DO-ICM)

Remote Operations Interactive Command Modules (RO-ICM)

When desired a DO-ICM can be auto-converted to a Remote Operations ICM. Both sides (Performer and Invoker) are auto-generated.

Notes:

30.1.1 Web Services – Remote Operations Interactive Command Modules (RO-ICM)

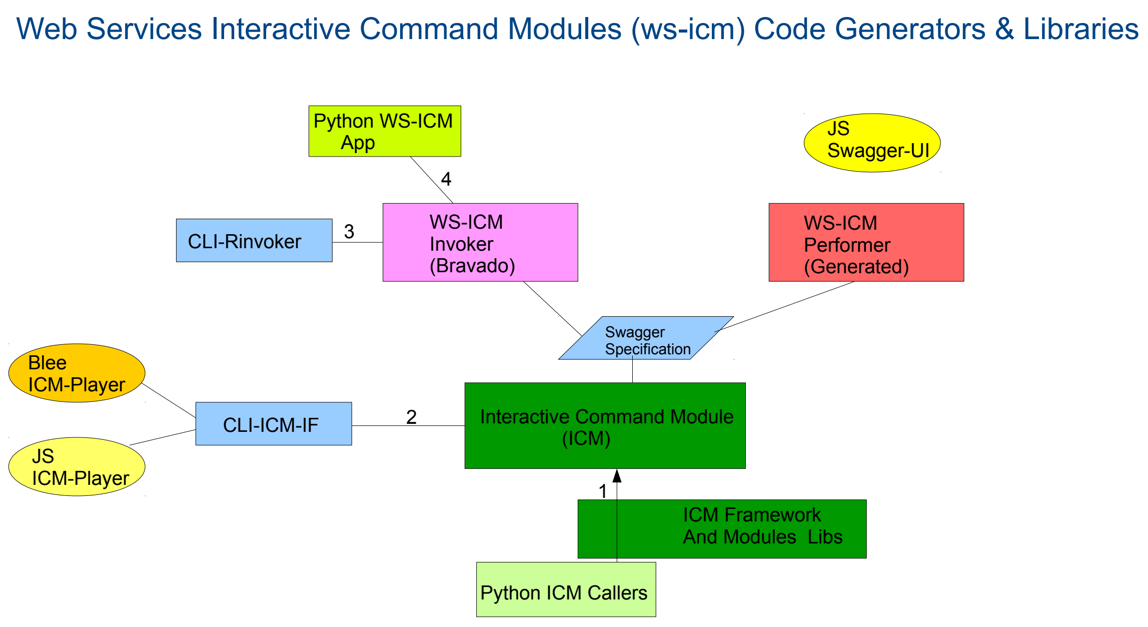
Web Services Interactive Command Modules (ws-icm) Code Generators & Libraries
Python WS-ICM App
JS Swagger-UI
4
CLI-Rinvoker
3
WS-ICM Invoker (Bravado)
WS-ICM Performer (Generated)
Swagger Specification
Blee ICM-Player
CLI-ICM-IF
2
Interactive Command Module (ICM)
JS ICM-Player
1
ICM Framework And Modules Libs
Python ICM Callers

Figure 4: Web Services Interactive Command Module (WS-ICM) Using Swagger Code Generators

---

Notes:

30.1.2 RO-ICM-Performers As Things-Agents

RO-ICM Performer Software

https://github.com/bisos-pip/mmwsIcm

```
pip install bisos.mmwsIcm
```

RO-ICM Performer Documentation

> **Remote Operations Interactive Command Modules (RO-ICM) Best Current (2019) Practices For Web Services Development** http://www.by-star.net/PLPC/180056 — [3]

Notes:

30.1.3 RO-ICM-Invokers As Things-Adapters

RO-ICM Invoker Software

`https://github.com/bisos-pip/mmwsIcm`

```
pip install bisos.mmwsIcm
```

RO-ICM Invoker Documentation

> **A Generalized Swagger (OpenAPI) Centered Web Services Invocations And Testing Framework** http://www.by-star.net/PLPC/180057 — [1]

Notes:

Part VII

Module-Players: GOSSONoT-Modules Execution User Interfaces And Environments

31 Execution Modes Of GOSSONoT-Modules

There are 3 different models for executing GOSSONoT-Modules:

- Ephemera Execution Model – Development And Testing
- Audit Trailed Execution Model
- Parallel Audit Trailed Execution Model

Notes:

32 Module-Players: User Interface For Execution Of GOSSONoT-Modules

---

GOSSONoT-Modules (ICM-Modules) are designed to self contain all user-interface related information. At this time, three types of Module-Players have been implemented

- Command-Line Players
- Blee-Player
- Flower-Celery

Notes:

Part VIII

Integrated Modules Development Environments – Emacs-Blee

33 GOSSONoT-Modules Development Environments

Poly-SON-Modules

GOSSONoT-Modules Development Environments

GOSSONoT-Modules are Python Code.

Any Python IDE (Interactive Development Environment such as: Emacs, pyCharm, sublime, eclips, Visual Studio Code, etc. can be used to develop GOSSONoT-Modules/ICMs.

We have enhanced Emacs's python development environment to be fully aware of GOSSONoT-Modules. We call that flavor of Emacs python development environment: Blee.

Notes:

34 Blee: An Emacs Based Integrated GOSSONoT-Modules Development Environments

---

Blee is a GOSSONoT-Modules/ICMs Integrated Development Environment that supports:

- A rich ICMs Templates Inclusion Library – based on yasnippet `https://www.emacswiki.org/emacs/Yasnippet`
- A rich ICMs macro support library – based on org-mode dblock `https://orgmode.org/manual/Dynamic-blocks.html`
- Blee ICM-Player – Allows for interactive specification of Things-Lists, ICM-Params, ICM-Args for ICM execution.

Notes:

Part IX

Machine-Learning Enhanced GOSSONoT (MLE-GOSSONoT)

35 MLE-GOSSONoT

GOSSONoT can be enhanced by Machine-Learning capabilities.

The interactions between GOSSONoT and Machine-Learning engines are accommodated by two classes of GOSSONoT-Modules.

- GOSSONoT-Machine-Learning-Monitor-Modules
- GOSSONoT-Machine-Learning-Adjustment-Modules

Notes:

36 Structure Of Machine-Learning Enhanced SON Platforms

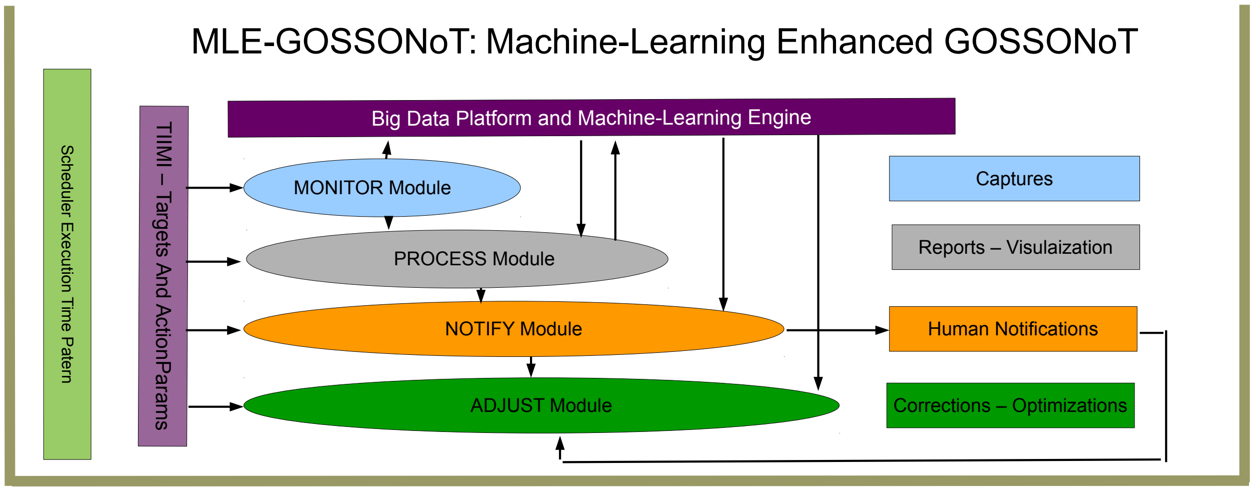
MLE-GOSSONoT: Machine-Learning Enhanced GOSSONoT
Scheduler Execution Time Patern
TIIMI – Targets And ActionParams
Big Data Platform and Machine-Learning Engine
MONITOR Module
PROCESS Module
NOTIFY Module
ADJUST Module
Captures
Reports – Visulaization
Human Notifications
Corrections – Optimizations

Figure 5: Machine-Learning Enhanced GOSSONoT (MLE-GOSSONoT)

Notes:

Part X

Poly-SON-Modules – GOSSONoT-ICMs And Proprietary SON-Modules

37 GOSSONoT-Modules Can Support Secondary SON Platforms

Poly-SON-Modules

GOSSONoT-Modules Can Support Secondary SON Platforms

For Python Based Modules Oriented SON Platforms, GOSSONoT-Modules can be enhanced to also run in Secondary SON Platforms.

In such conditions, we call that module “A Poly-SON-Module”.

Developing SON-Modules as Poly-SON-Modules provide many benefits where the strength and special features of both platforms can be used.

Such an approach also provides additional strategic options to module developers for transitioning from one platform to another.

Notes:

38 Example Of A Poly-SON Module Running In Two SON Platforms

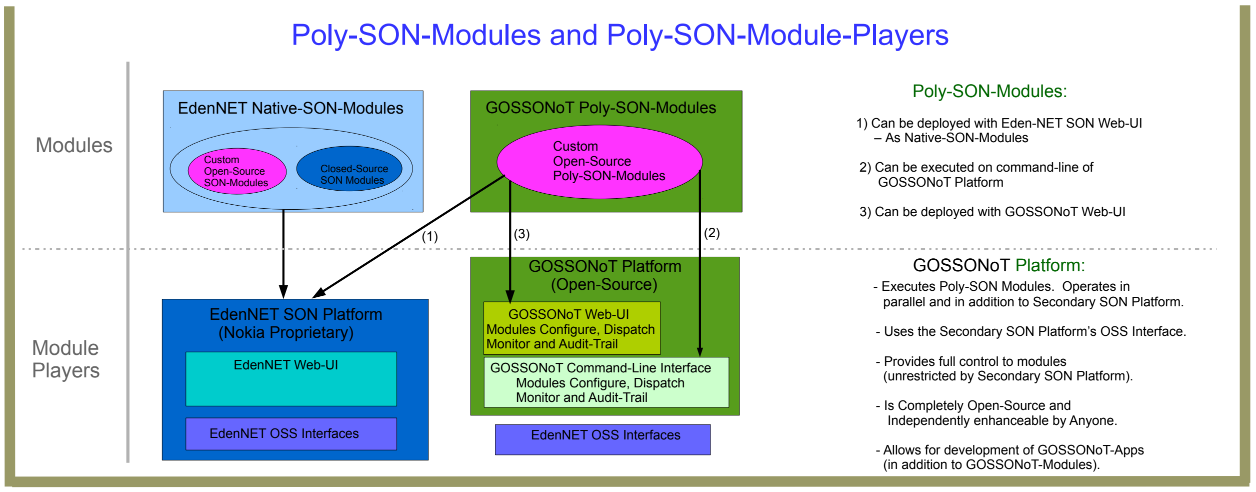
Poly-SON-Modules and Poly-SON-Module-Players
Modules
Module Players
EdenNET Native-SON-Modules
Custom Open-Source SON-Modules
Closed-Source SON Modules
GOSSONoT Poly-SON-Modules
Custom Open-Source Poly-SON-Modules
(1)
(3)
(2)
EdenNET SON Platform (Nokia Proprietary)
EdenNET Web-UI
EdenNET OSS Interfaces
GOSSONoT Platform (Open-Source)
GOSSONoT Web-UI Modules Configure, Dispatch Monitor and Audit-Trail
GOSSONoT Command-Line Interface Modules Configure, Dispatch Monitor and Audit-Trail
EdenNET OSS Interfaces
Poly-SON-Modules:
1) Can be deployed with Eden-NET SON Web-UI – As Native-SON-Modules
2) Can be executed on command-line of GOSSONoT Platform
3) Can be deployed with GOSSONoT Web-UI
GOSSONoT Platform:
- Executes Poly-SON Modules. Operates in parallel and in addition to Secondary SON Platform.
- Uses the Secondary SON Platform's OSS Interface.
- Provides full control to modules (unrestricted by Secondary SON Platform).
- Is Completely Open-Source and Independently enhanceable by Anyone.
- Allows for development of GOSSONoT-Apps (in addition to GOSSONoT-Modules).

Figure 6: Poly-SON-Modules and Poly-SON-Module Players

Notes:

Part XI

Use Case Examples

39 Overview Of Scenario Examples

Main functional areas of SON are:

- Self-Configuring: Configurations Management
- Self-Optimizing: KPI Monitoring, Parameter Adjustment
- Self-Healing: Monitor, Process, Notify, Adjust

As examples we now apply these to GOSSONoT's very different things (VMs, IoTs, Networks).

- Self-Configuring – After VM creations, verify/set consistent passwords on large number of VMs.
- Self-Optimizing – Before coming home, the owner of the house indicates that he is on his way home.
- Self-Healing – Layer 3 information indicates failures, other network interfaces are used for access and routing purposes.

It is the consistency and cohesion of these different example scenarios that demonstrates the power and value of GOSSONoT.

Notes:

40 Self-Configuring: VM Passwords

- List Of VMs to be subjected to configurations is specified as Things-List – as an example see: `https://github.com/bisos-pip/gossonot/blob/master/dev/bisos/gossonot-base/ts-librecenter-localhostIcm.py`
- List of Parameters to be configured and the rules for configuration are – as an example see: `https://github.com/bisos-pip/gossonot/blob/master/dev/bisos/gossonot-base/bxpUsageParamsIcm.py`
- The appropriate GOSSONoT Configuration Module is invoked with the selected Things-List and Things-Params-List

---

Notes:

41 Self-Optimizing: IoT – The Home Owner Comes Home

- Home Owner Signals To His GOSSONoT's "Home-Management-Module" That He Is On His Way Home.
- Home Owner's Home Arrival Time is estimated.
- Current Home Temperature And Temperature Adjustment Rates And Desired Temperature Are Determined.
- Home-Management-GOSSONOT-Module determines when to turn on the furnace.
- When the Home-Owner's Lactation is determined to be close enough to the house by the Home-Management-GOSSONoT-Module, additional driveway lights are turned on and the Garage Door is opened.

Such a prototype of a Home-Management-GOSSONoT-Module exists. It can be considered autonomous and privacy-oriented as the Home-Owner "owns" the Home-Management-GOSSONoT-Module as well as his house and the things in his house.

Notes:

42 Self-Healing: Network Performance Monitoring – Links Adjustment

---

- A large number of hosts are being instrumented with a GOSSONoT-Things-Agent in the form of a RO-ICM-Performer which gather network performance results to different destinations.
- A GOSSONoT-Module through a corresponding GOSSONoT-Things-Adapter (RO-ICM-Invoker) receives the network performance information from that large number of hosts.
- Based on that, the GOSSONoT-Module then can identify failures and work towards Root-Cause-Analysis (RCA) and "Adjust" appropriate nodes by instructing them through the GOSSONoT-Things-Adapter to use different links.

Notes:

Part XII

Next Steps

43 Next Steps – Evolving GOSSONoT's Core

The Core Of GOSSONoT (ICM, RO-ICM, Model Of Things) is being developed and maintained by a small tight team.

If you have any ideas for improvements and enhancements let us know.

Notes:

44 Additional Modules And Additional Things-Interfaces

As you use GOSSONoT and develop new Things-Interfaces and Modules, we can add them to the common GOSSONoT library. Please let us know.

Notes:

References

[1] " Mohsen BANAN ". " a generalized swagger (openapi) centered web services testing and invocations framework ". Permanent Libre Published Content "180057", Autonomously Self-Published, "December" 2018. http://www.by-star.net/PLPC/180057.

[2] " Mohsen BANAN ". " extending son to clouds and things gossonot: A generalized open-source self organizing network of things platform ". Permanent Libre Published Content "180052", Autonomously Self-Published, "December" 2018. http://www.by-star.net/PLPC/180052.

[3] " Mohsen BANAN ". " remote operations interactive command modules (ro-icm) best current (2018) practices for web services development ". Permanent Libre Published Content "180056", Autonomously Self-Published, "September" 2018. http://www.by-star.net/PLPC/180056.

[4] " Neda Communications Inc". " interactive command modules (icm) and players a framework for cohesive generalized scripting a model for gui-line user experience ". Permanent Libre Published Content "180050", Autonomously Self-Published, "July" 2017. http://www.by-star.net/PLPC/180050.