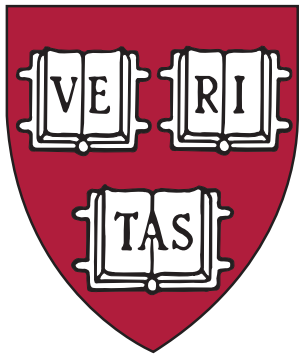
HARVARD
UNIVERSITY



Information Technology

Integer - An Integrated Management Environment
Design Review - v1.0

June 25, 2014

Topics

- Introduction - Jim Waldo - why we are doing the design review.
- What's the problem?
- The Integer Initiative.
- Technical decisions and opportunities.
- Design - how it works.
- Discovery - initial objectives and deployment.
- Information requirements for discovery.
- Processing and sub-system details.
- Plans.
- Development environment and tools.
- Domain knowledge.
- Using Integer.
- Contact us and information.
- Backup topics and information.

Discussion Format

- Ask questions as they come up.
- Stop at the end of each section for questions and comments.
- Feel free to go to integer.harvard.edu to submit comments on the “Join Us” page.
- Send mail to info@integer.harvard.edu

What’s the Problem?

What's the Problem?

- We don't have an integrated view of our systems and services.
- Information gaps cause downtime:
  - Isolated management systems/approaches 'databases'
  - Absent/uncoordinated monitoring across environments
- Patchwork of non-integrated systems from multiple sources is not cost-effective:
  - Multiple groups doing small, sometimes overlapping development.
  - Problems with access to timely information.
  - Duplication of effort.
  - Can increase time to identify and resolve problems.
  - Makes service provisioning and de-provisioning more complex, error-prone and less agile.
- New environments like AWS add complexity and proprietary methods further fragmenting our view; more stove pipes

Service Outage Analysis - Final Report Findings

- No service configuration software - software that understands the relationships between all the elements of a service.
- Installation and upgrade procedures require significant human interaction, increase error opportunities.
- Root Cause/Systems Failure Analyses are not routinely performed - do not have data and systems that facilitate such analyses.
- Unknown Service Dependencies - 74% of a key service's outage time was the result of unknown dependencies. We did not understand how a change (or a fault), on apparently unrelated systems would impact the service.
- Configuration Error - Configuration error is a leading cause of failure in the industry. Harvard still has a largely manual configuration process - many steps can introduce errors.

HUIT Key Systems Study Recommendations

- Develop an integrated service management system that incorporates:
  - A centralized data collection engine.
  - A cross organization service configuration system.
  - Distributed role-based access to the data and functions.
  - Automated deployment/installation and verification software.
- System must span
  - Vendors
  - Technologies
  - Have a service-wide view
- Support service level objectives data collection, analysis and reporting

The Integer Initiative

Integer

- Cross HUIT funding with headcount and resources contributions across organization.
- Initiative to create a unified management environment to operate increasingly complex geographically distributed computing environments.
  - A single screen
  - Selectable perspectives
  - Not all perspectives are available to all classes of users
  - Provides status and control of service delivery development, test and production environments.
- Dictionary definitions of “integer”
  - A complete unit or entity.
  - An individual entity or whole unit.
  - A thing complete in itself.

The Free Dictionary, Oxford Dictionary

Strategic Vision

Integer

An integrated environment that empowers efficient error-free technical operation of our services.

Strategic Objectives	Guiding Principles	Key Performance Indicators
• Create integrated and simplified deployment & operations environment • Increase operational efficiency • Reduce failures • Enable best operational practices • Create actionable information • Increase technical agility	• Integration of information will be a driver in all decisions • Must foster cross intra and inter Harvard collaboration • Will leverage open source technologies and values • Should balance urgent needs with long-term objectives • Will actively balance flexibility, complexity and usability	• Automated inventory management to testing in Q1 and Q2 FY 15 • Functions added semiannually FY 16 - 17 • Lay foundation for Integrated network and server monitoring (existing tools) Q3 FY 15 • Establish verifiable baselines for failures, time to add/change resources and costs related to deployment and operation • Set goals for improvement of each baseline

Alignment with Strategic Objectives

- The initiative supports many strategic objectives emerging in recently formed work groups (workgroups have not yet completed so these are subject to change):
  - Cloud and DevOps.
  - Networking.
- Integer supports these other areas:
  - Cost/efficiency improvement.
  - Increased agility.
  - Reduction of errors (improve reliability/resiliency).
  - A service-oriented approach.

Integer's Scope

- Covers essential areas of management with a service-wide view:
  - Fault - detecting and repairing failures.
  - Configuration:
    - Automated and reliable configuration of systems and services - including deployment of systems & images.
  - Accounting - quantity of work done on behalf of a service.
  - Performance:
    - Issues related to latency and capacity.
    - User level to fine-grained per-system details.
  - Security - Service wide perspective security controls.
- Key elements of the environment:
  - Users.
  - Technical Components:
    - Servers.
    - Network elements like routers, firewalls, load balancers, DNS system and other physical and virtual network elements.
    - Software from the virtualization layer to high-level web services.

Integer as a Point of Integration

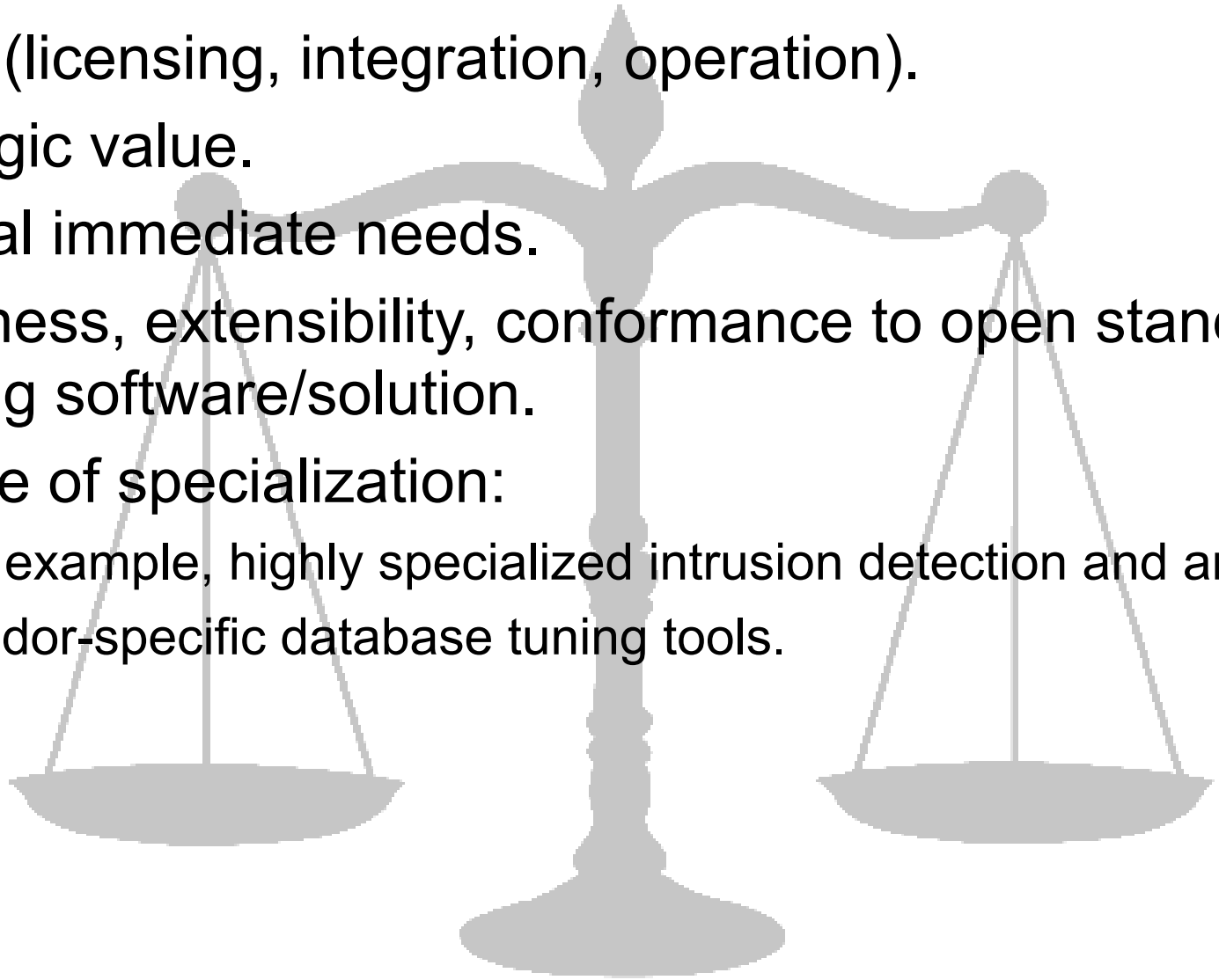
- A point of integration for other specialized systems or systems where there is insufficient cost/benefit for implementation of functions in Integer.
- A platform for the development of a wide range of functions by Harvard and open source community.

Technical Decisions and Opportunities

Integer - No Protocol or Information Model Religions

- Take systems as they come.
- Accept reality of multiple monitoring and control protocols per system.
- Do not assume managed systems adopt CIM, WebM or any other model or current fad.

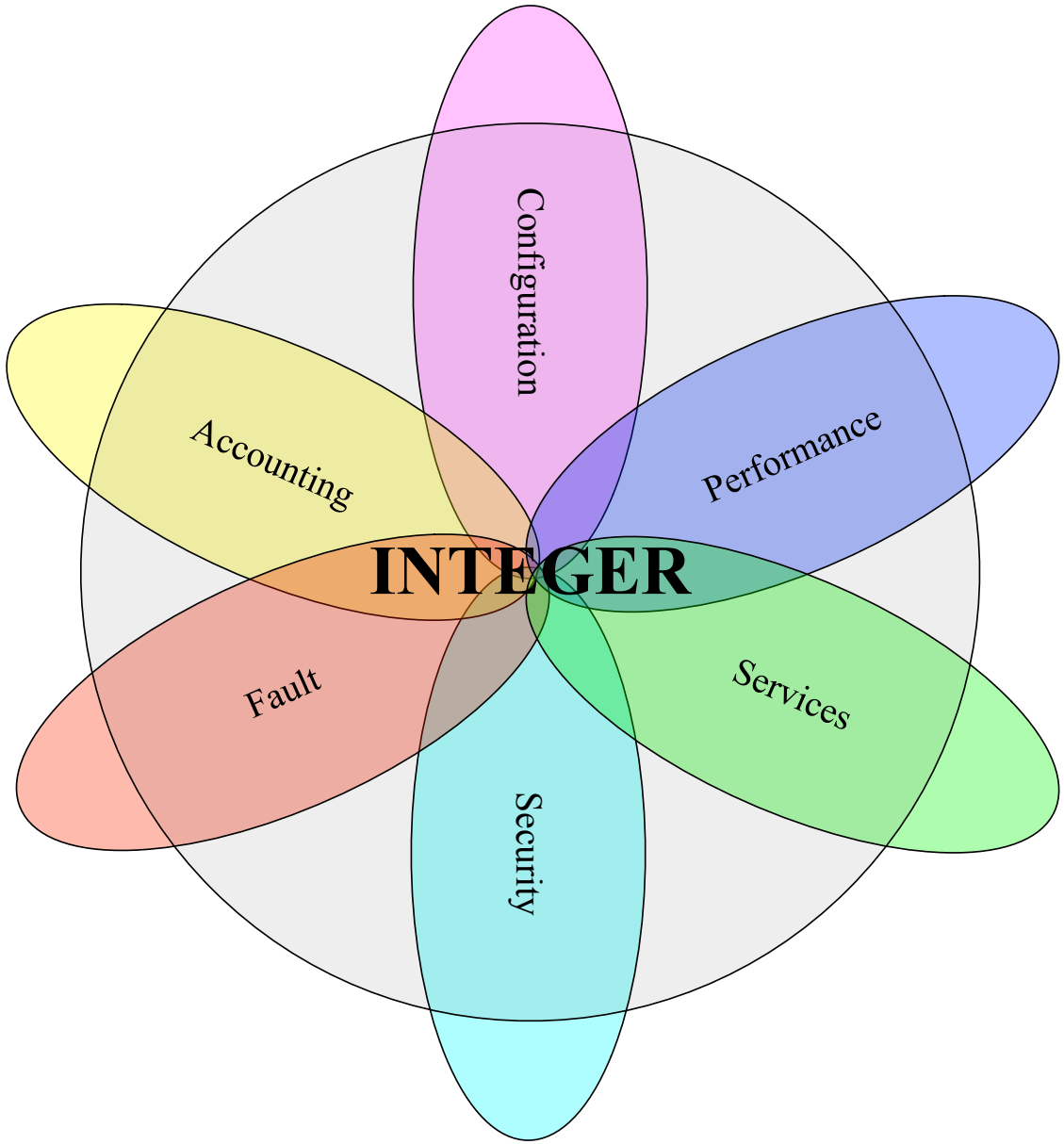
Make vs. Buy vs. Integrate by Integer



- Costs (licensing, integration, operation).
- Strategic value.
- Tactical immediate needs.
- Openness, extensibility, conformance to open standards of existing software/solution.
- Degree of specialization:
  - For example, highly specialized intrusion detection and analysis.
  - Vendor-specific database tuning tools.

Relationship to Existing Management Systems

- Some overlap with existing systems until they are retired.
- Uses/relies and integrates with many other systems:
  - Central logging.
  - Automation tools like Puppet or Chef.



Integrates with key Systems

- Must integrate with development, test and production environments.
- DevOps:
  - Source code control (can be first step toward treating infrastructure as code).
  - Specialized systems such as:
    - Puppet.
    - Logging systems.
  - Cloud technologies:
    - Preference for open APIs and functions.
    - Integration with AWS via Java API.
- High-level service management systems:
  - Two way communication.
  - Published APIs.

Opportunity/Results - Institutional

- Consistency throughout operational environment - the lack of which has been the source of service outages
  - A single system where one instruction (such as permit TCP port 80 for a specific service) is translated to the service element specific commands, e.g., configure:
    - Host Firewalls.
    - Firewalls.
    - ACIs.
    - Middleware.
    - Other elements.
- A single integrated set of data about our environment for better generation of actionable information.
- A single monitoring environment to view the entire technology stack from different perspectives including application/service.
- Reduces need for different groups to write one-off tools that they must maintain.
- Net reduced cost of operation - few management/support systems to develop, license, operate and maintain.
- A common human interface adjusted by role to all functions - a common view of our services and the systems the realize them.
- Highly configurable system empowering domain experts to pro-actively manage environment.

Opportunity/Results - Staff

- Time focused on improving services through analysis of data rather than tools maintenance.
  - Creates opportunity to be more proactive.
  - Opportunities to do more validation of options/what-if analyses.
- Opportunity to learn more about technologies being managed.
- A broader view of systems rather than one or two types.
- Increased opportunity to collaborate with a broader environment of users.

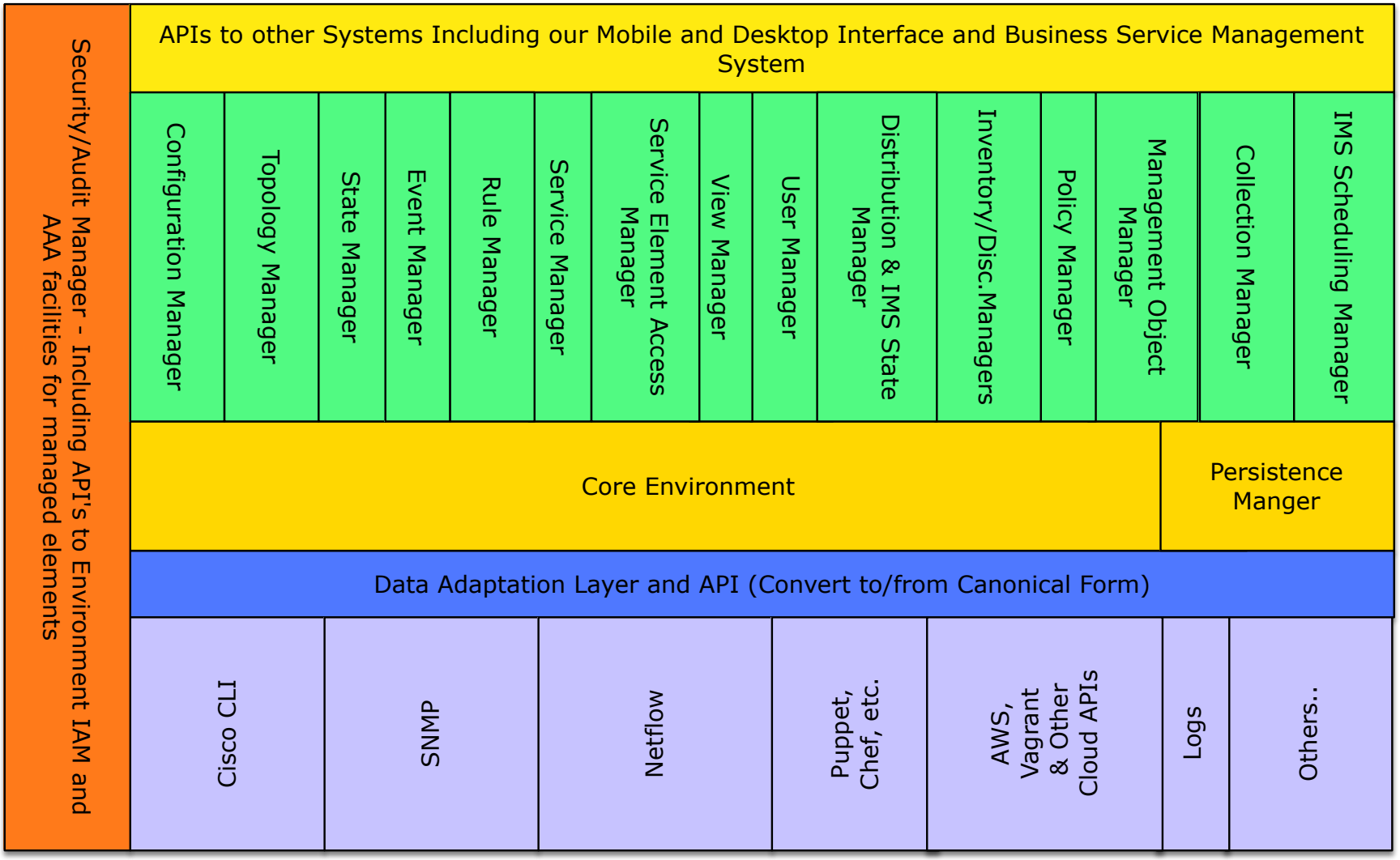
Integer is not magic or an 'AI' system. There is nothing that Integer can do that is not done, or could not be done by operational personnel (though it will eventually do it faster and more reliably). Everything Integer knows, has been configured into it or is learned from the environment.

Design - How it works

Implementation

- Web front end.
- A set of services with APIs that hide DB structures.
- Coded in Java.
- A ‘core’ installation that can be clustered/distributed:
  - Reliability.
  - Performance.
- Multiple distributed server installations.
  - To reduce network impact.
  - Reduce network configuration complexity.

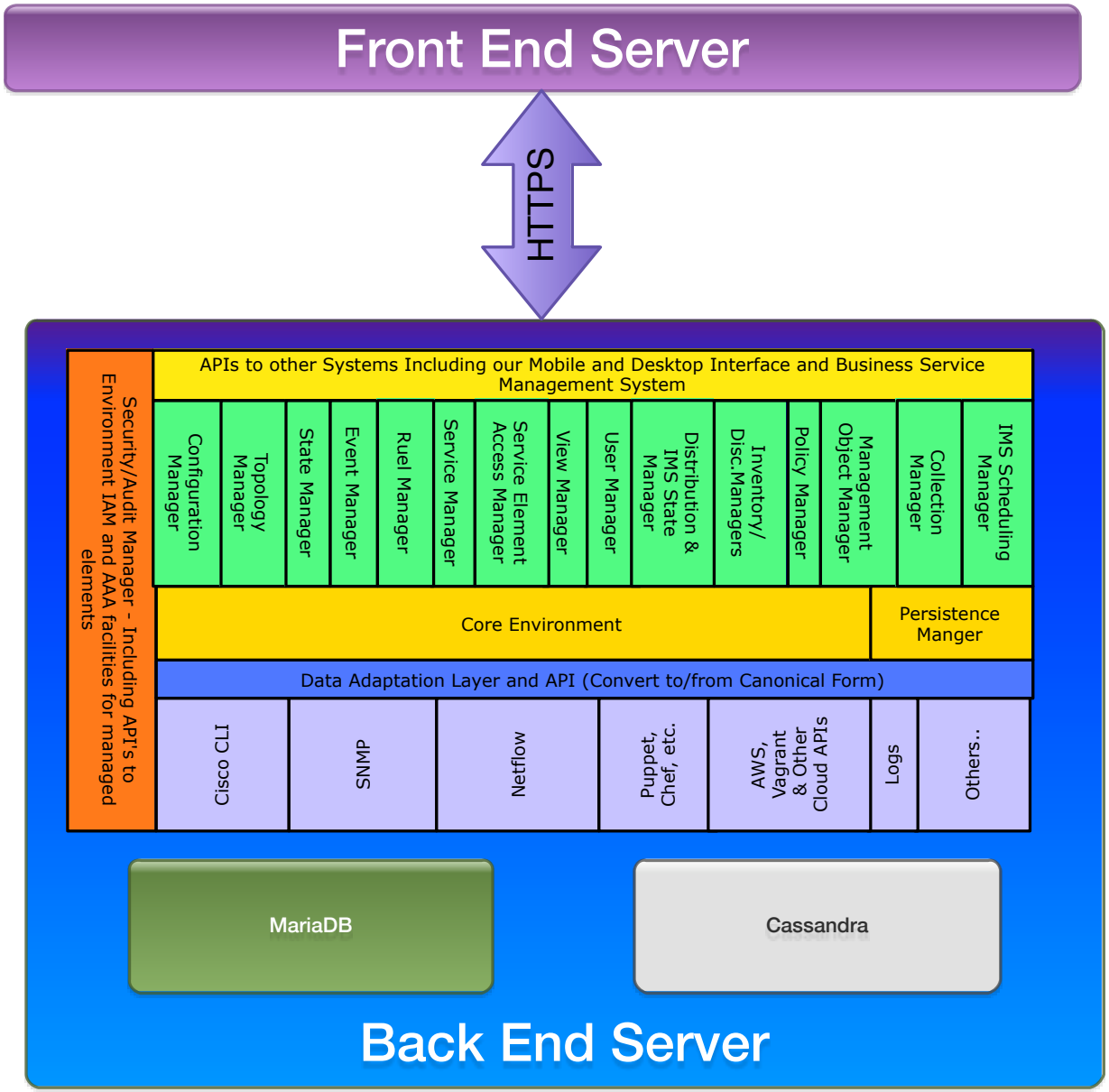
Server Logical Organization



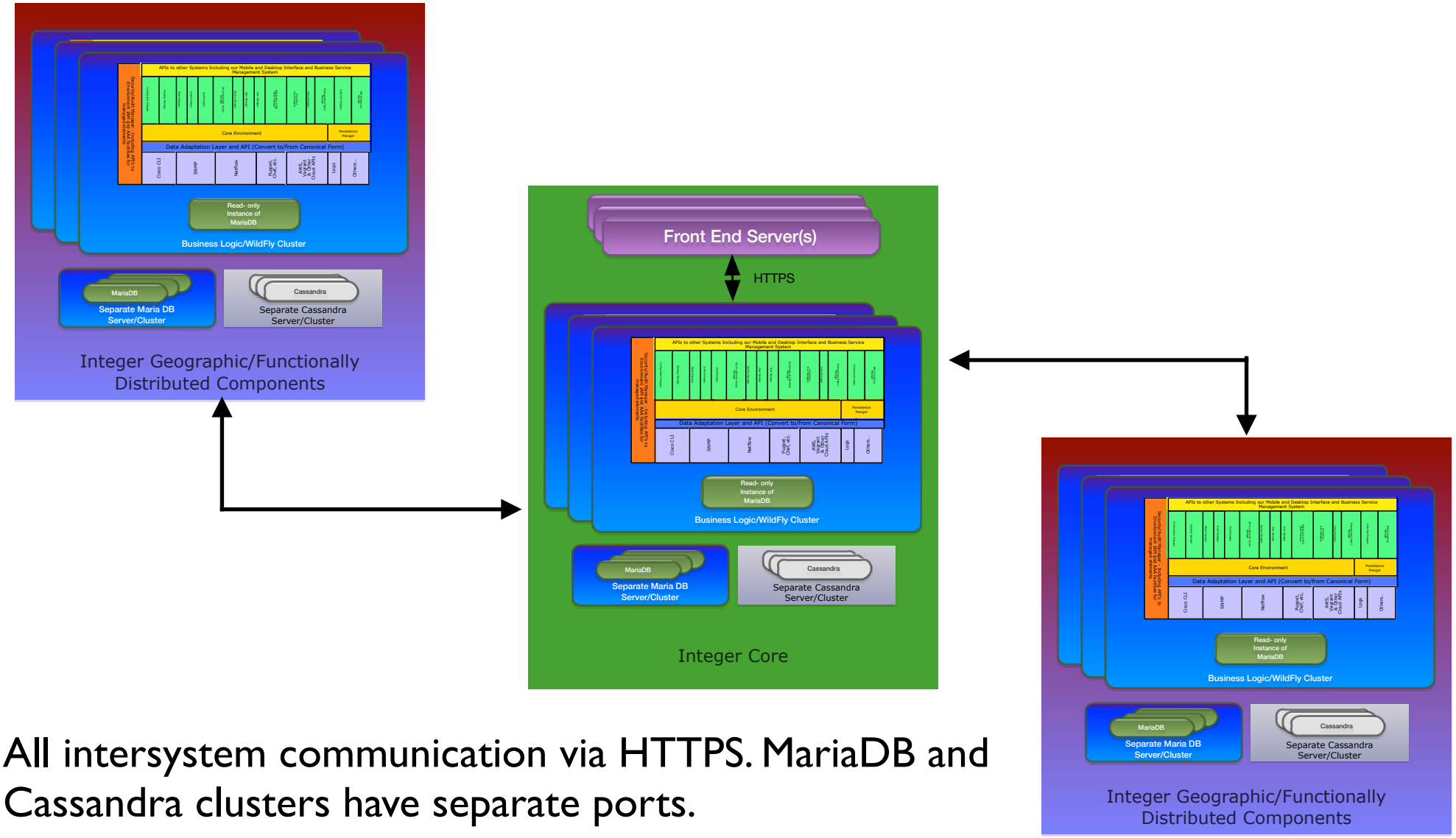
Open Source Technologies/Components

Database	MariaDB
NoSQL	Cassandra - not used in initial discovery release
Middleware	WildFly (a.k.a. JBOSS)
UI	GWT
Scheduling/Time	Quartz
SNMP	SNMP4j
MIB Compiler/ Loader	Mibble
Logging	SLF4J - Simple Logging Facade for Java

Distribution - Simple Deployment



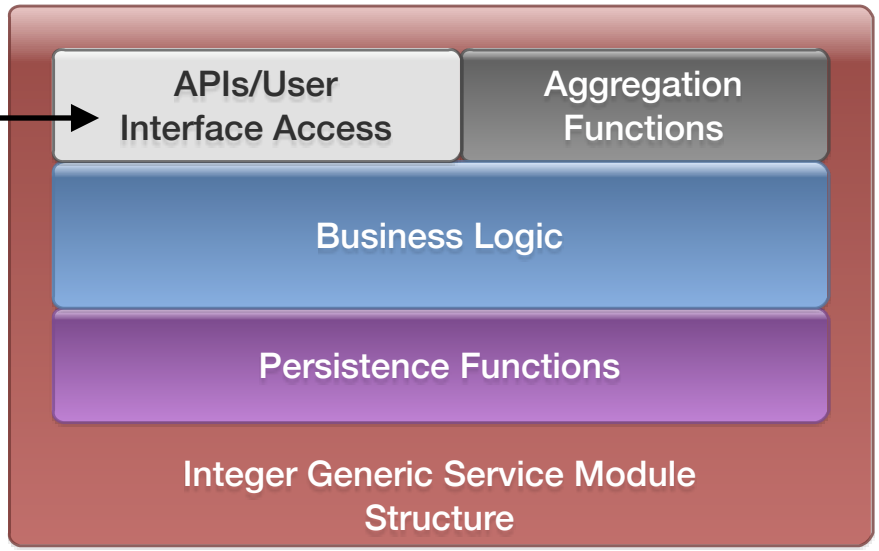
Distribution - Large/Complex Case



All intersystem communication via HTTPS. MariaDB and Cassandra clusters have separate ports.

Aggregation of Distributed Data

Only available on core installation.



Object Hierarchies

- Main conceptual elements
  - Business services and technologies
    - Management objects and capabilities.
  - Providers and organizations.
  - Users, groups, roles.
  - Service elements and multi-layer/technology topology.
  - State.
  - Configuration.
- Others used for various functions and services:
  - Discovery.
  - User selections.
  - Reporting.

Discovery - Initial Objectives and Deployment

Discovery Release Objectives

- Deploy initial components of architecture for validation/correction.
- Covers SNMProwl +.
- Complete inventory of the environment.
  - Network elements - including layer 2 and message stream modification systems like:
    - Firewalls.
    - NAT.
    - Load balancers.
  - Servers.
  - Interconnections.
  - Details of the ‘contents’ of each system - e.g., cards, ports, etc.
  - Some high-level software.
- Automated system for detecting, tracking and reporting changes to systems from the hardware up (expands over time).
- Feed to systems like Nagios that require ‘inventory’ information.

Discovery Deployment

- Can be deployed as a single system or in a distributed fashion
  - Each system can be configured in terms of the scope of subnets included in the discovery.
  - Each of the distributed elements rolls up to the main installation.
  - Each can have a separate set of ‘rules’ that control behavior.
- Systems can run in virtualized, physical or cloud-based environments or any combination.
- Each ‘rule’ may have a different schedule if desired.
- Each system may have multiple tasks discovering devices and topology.

Discovery Mechanism

- First release is exclusively SNMP-based.
- Try to retrieve 6 objects from System Group, identify
  - SNMP version.
  - Community or v3 Administrative information.
  - Port.
- Identify vendor and other information (vendor/system specific).
- Use information to identify device containment hierarchy (e.g, proprietary, entity MIB, Host Resources, SysAppl, etc.).
- Identify each of the children.
- Retrieve interface information via ifMib and extensions.
- Assigns a ServiceElementType based on unique signature.
- Create systems and associated network information for found interfaces.

Distribution Characteristics

- WildFly cluster maintains session state across servers so in the event one fails, users state is preserved.
- Applications written to Integer APIs will include a WildFly library that exchanges information about core cluster nodes so in the event of a failure they seamlessly know which other system to access.
- WildFly to WildFly communications over secure (HTTPS) connection reducing need for access control configuration for multiple ports across the network.
- SNMP and other similar protocols can be deployed locally to further reduce need for complex network configuration.

Information Integer Requires for Discovery

Basic Configuration Information

- In order for Integer to:
  - Identify which are in the systems/networks of interest.
  - What functions they have (e.g., routers, servers, etc.).
  - How they are interconnected.
  - What they are comprised of (e.g, cards, ports, etc.).
- In requires two types of information:
  - User input like what networks to discover (more on this later).
  - General configuration information supplied with Integer, Domain Knowledge:
    - Loaded management objects, for first release MIB Objects only.
      - Standard MIB Modules
      - Vendor-specific MIB Modules
    - Which management objects (and values they return) uniquely identify a type of service element like a router or the cards in the router or a Unix host.
    - What types of functions a particular service element can perform - e.g., routing, load balancing, etc.
    - How specific management objects map to our technology tree (the way we organize our technologies).

Global Discovery Control and Configuration

- Some domain knowledge via YAML. Some may require programatic or other augmentation.
- Whether phases (service element and topology) can overlap on a subnet - to help control load on devices.
- Global SNMP credentials.
- Global SNMP version trial order.
- Per-subnet SNMP credentials to override global credentials.
- Alternate ports to try.
- Protocol time out and retry values.
- Additional attributes to collect for each service element type.
- Calendar policies.

Discovery Control and Configuration - Per Rule Options

- For each discovery rule:
  - Networks to exclude (even if they are within ‘radius’).
  - Devices (gateways) to exclude.
  - Starting subnet(s) for the discovery.
  - Radius.
  - Local Overrides:
    - SNMP Credentials and ports and version trial order.
    - Timeout and retry values (can be separate ones for topology and general device discovery).
  - Technologies to include/exclude - e.g., save only routers or specific types of web servers.
  - Starting router.
  - Remote discovery system.
  - Service element type specific SNMP overrides (e.g,. don’t send too many messages to certain routers).

Processing and Sub-system details

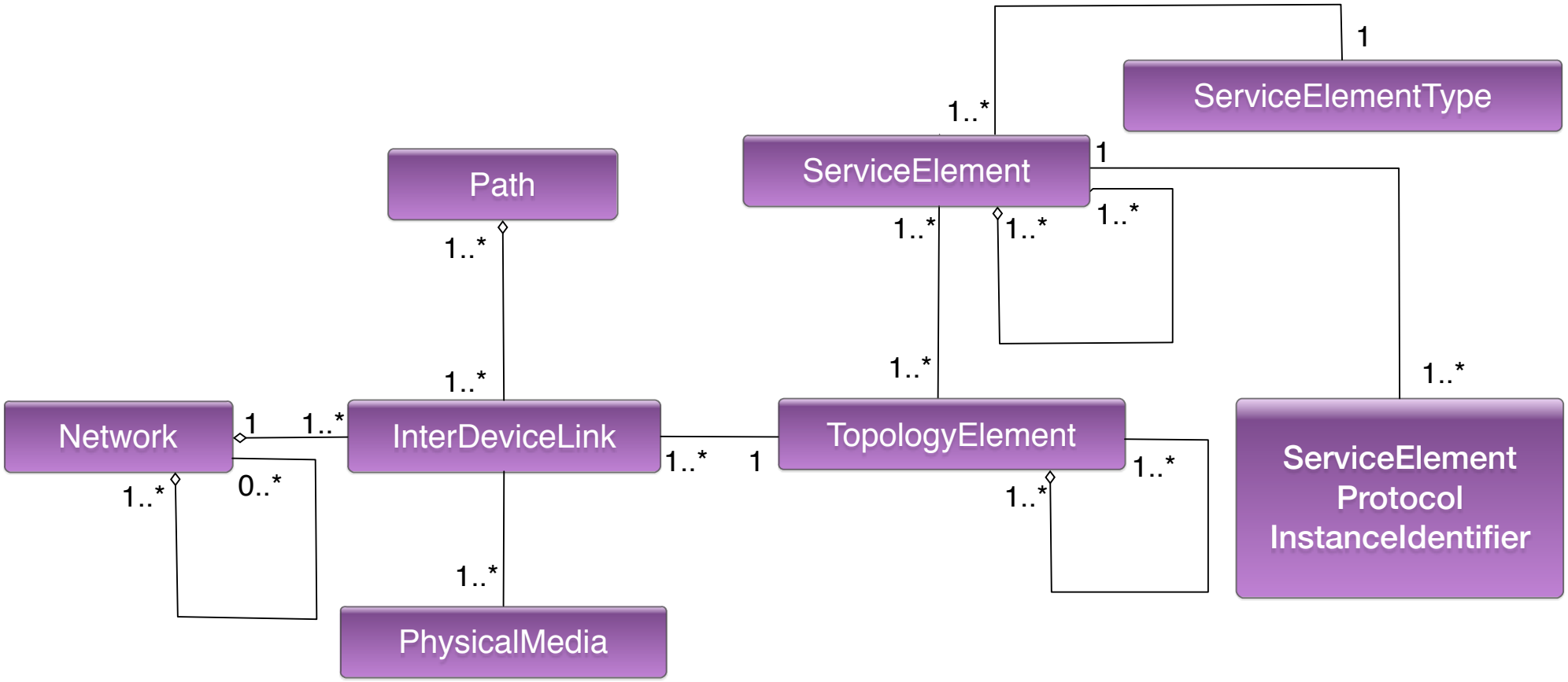
Discovery - System Level - Phase 1

- Determine range of hosts to check on each subnet that has been assigned - based on subnet/mask or CIDR address.
- For each system found in range we attempt to retrieve 6 elements from the System Group we use to:
  - Establish vendor and top level information such as software version.
  - Determine device's containment structure (e.g., cards, ports, memory, storage, etc.
  - Learn about each sub-element (types, whether it contains sub-elements, etc.).
  - Learn about interfaces and network connections and addresses.
- Based on discovery configuration, learn about other technologies from routing to load balancing currently enabled on the service element.
- Save to database.
- Repeat for each network specified in configuration (or learned from topology discovery - next slide).

Discovery - Topology - Phase 2

- Use seed information from Discovery Rules.
- Information from device discovery (if run previously).
- Discovery of different technologies relevant to topology and connectivity:
  - CDP.
  - LLDP.
  - Routing information by protocol if desired:
    - Static.
    - BGP.
    - OSPF.
    - RIP.
    - Etc.
  - Other types of interconnection such as router or other types of redundancy.
  - Layer 2 (VLANS).
  - Layer 2.5 like MPLS.
- Discover additional networks based on rules.
- Save information to database.

How Integer Represents Systems (Service Elements) And Links Between Them- Simplified



Discovery - Inventory Manager

- Like discovery manager's discovery rule, the inventory manager has an inventory rule. Includes controls for:
  - What service elements are covered by the rule:
    - Location.
    - Criticality.
    - Technologies (e.g, routers, name servers, etc.).
    - Changes since last run.
    - New systems.
    - Missing systems.
  - Definition of:
    - What it means to be 'missing' - number of runs absent.
    - Number of runs a new system must exist before being added.
    - Notification actions.
    - Systems to exempt (e.g, those with ifAdmin status down).
    - What a change is for a specific type of service element (based on unique identifiers for service element types).

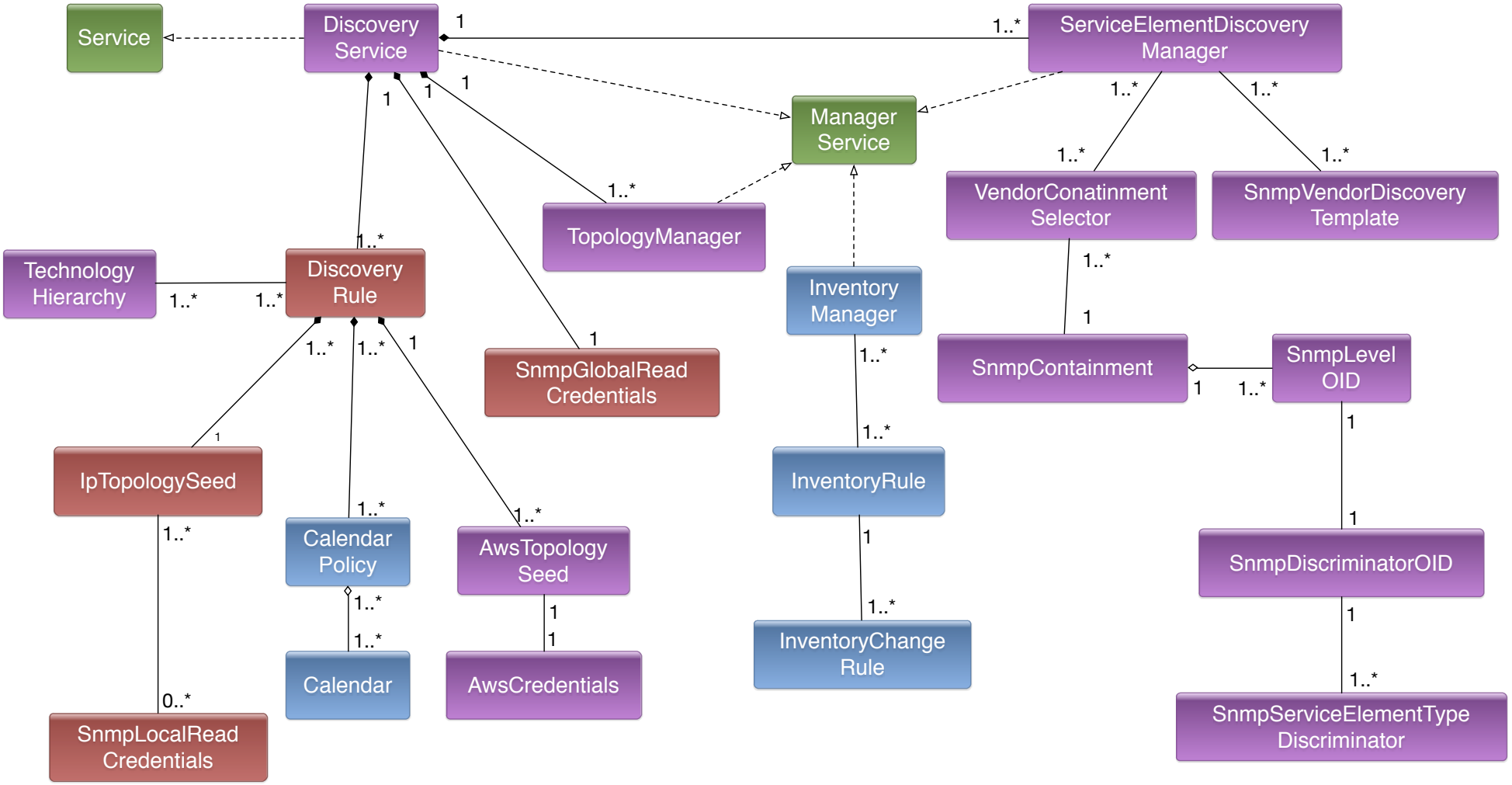
Discovery - Reporting

- Same selection method as other parts of the system.
- For this release:
  - Entire inventory.
  - Changes or other selections.
- Basic format/text report.
- JSON output.
- Email.
- Event/alarm window.

Discovery - Events/Alarms/Logs

- Implements basic framework for future releases.
- Same selection methods as for rest of the system.
- First release limited to basic system function errors, messages and logs.
- Multiple concurrent selections supported e.g.,:
  - A tab for logs.
  - A tab for certain discovery events.

Discovery - Basic Elements



Discovery - Next Steps

- User interface for domain knowledge.
- Manual insertion/placement of layer 1 elements.
- Extend to service dependencies.
- Extend to storage.
- Extend to AWS.
- Extend to virtualization.

Plans

Current Plan - Overview

- Discovery Clean up.
- Configuration, phased:
  - Portions of network elements.
  - All network elements,
  - Servers.
  - Cloud technologies.
  - Middleware.
  - Application tools.
- State/Status.
- Capacity planning and performance.

Releases and focus

Release/Description	Simplification/Integration
Release 1 - Discovery - Delivery of overall architecture with an integrated function, layer 2/3 discovery and service element discovery with inventory reporting. Support for layer 1 elements.	SNMProwl
Release 2 - Discovery and system enhancements - user ability to modify management object definitions, integration of additional data, addition of storage technology and virtualization environments at Harvard. First AWS discovery support.	cust.db, separate supporting spreadsheets for support agreements and other data
Release 3 - Network Element Access Control Configuration - coordinated configuration of access across network elements/infrastructure.	Currently done by hand
Release 4 - Full Network Element Configuration - addition of full network device configuration control.	NetMRI, RANCID
Release 5 - Server configuration/integration with automation technologies. This includes automation of cloud and local server infrastructure configuration.	Many manual scripts, programs, poss. Maestro.
Release 6 - Selected Application Configuration Control	Numerous manual scripts
Release 7 - Refinement of Policy Controls, AWS enhancements and clean up	Manual scripts
Release 8 - Integrated Fault and Performance Monitoring	Statseeker, Nagious, SNMPoll, MRTG
Release 9 - Enhancements for Wireless support	Scripts

Development Environment & Tools

Tools

Security validation	Veracode
IDE	Eclipse - lots of plug ins.
Source Control	Git (GitHub for source and public information) Git (private on our build system for sensitive information)
Build	Gradle
Java	1.7
Continuous Integration	Jenkins - lots of plug ins
Feature/Defect Tracking	Jira with Jira Agile and plug-ins
Collaboration Site	Confluence
IM/Group, Build, Check-in	Slack

Engineering Environment

- Laptops
  - Mavericks.
  - Virtual box.
- AWS
  - Facilities
    - EC2.
    - Cloud Formation.
    - VPC.
    - Route 53.
    - Security Groups.
  - Integration.
  - Integration testing and post build testing
    - Each build creates and deploys test instances in new VPC.
    - Daily security validation with Veracode.

Servers in AWS - Interesting Lessons

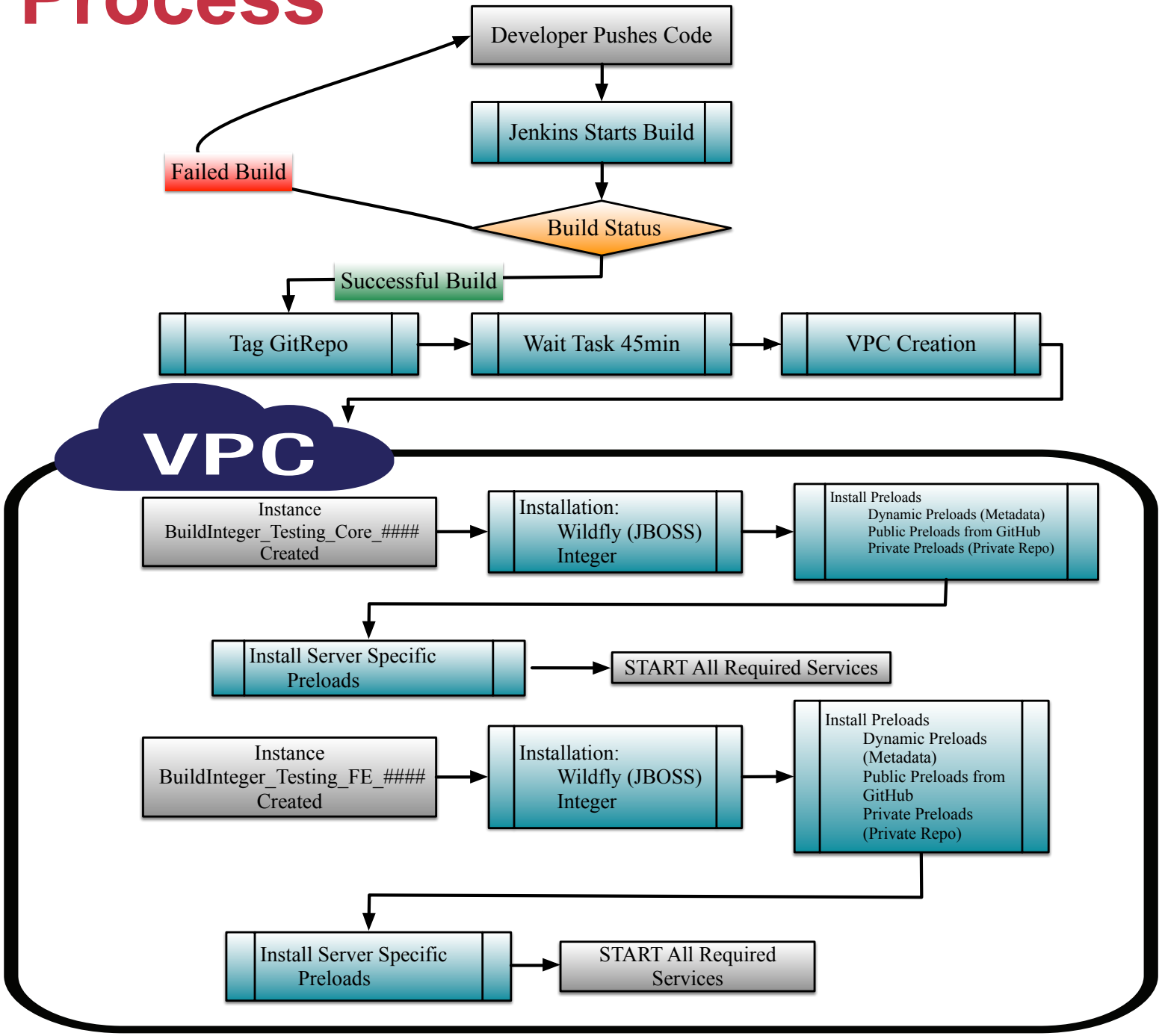
- Main server
  - Build.
  - Jenkins.
  - Confluence.
  - Jira.
- Mail server
  - Mail.
  - integer.harvard.edu
  - Mail list manager.
- RDS is very expensive.

Sometimes a couple big servers are cheaper (and faster) than a bunch of little ones.

Test Systems

- 60 Oxford Street
  - Deployed in two server configuration:
    - Front end.
    - Back end for all other components.
- AWS
  - Per build.
  - ‘Stable’ for regression.
- Other sites in progress.

Build Process



Integer’s Unique Difference - Domain Knowledge

Integer Domain Knowledge

- How Integer ‘learns’ about and normalizes to a canonical form, different types of management objects:
  - SNMP - standard and vendor-specific.
  - A variety of CLIs.
  - RESTful and other interfaces and data formats used on them
  - Different APIs/interfaces for different environments like AWS.
  - DevOps and automation systems like Puppet, Chef, etc.
- How Integer ‘learns’ about what makes a particular device, or part it contains unique in terms of what kinds of things it can do.
- How Integer connects the large number of combinations into knobs that a human can control and understand and use to train Integer further.

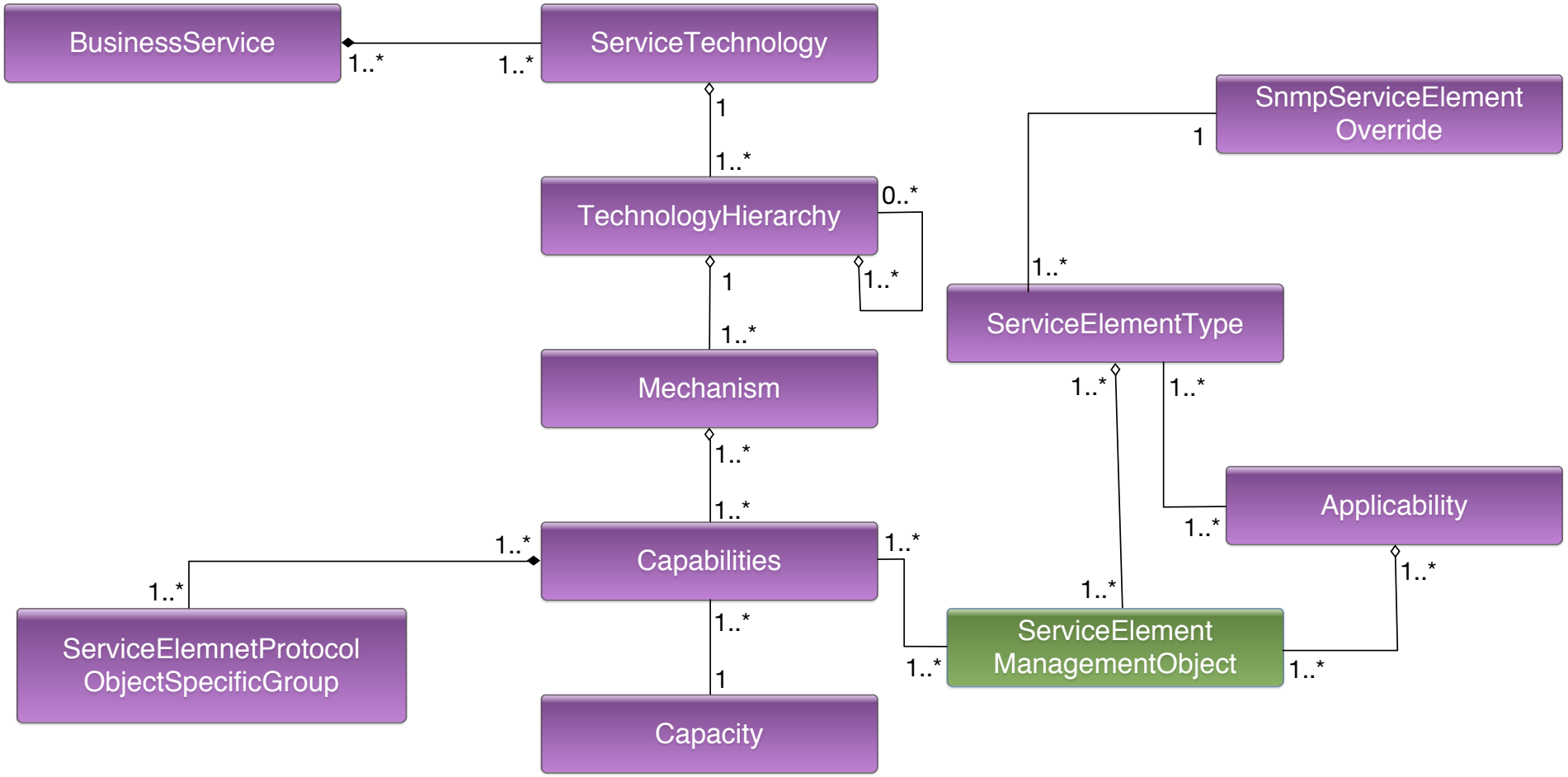
Domain Knowledge

- Some standards such as:
  - Vendors - from IANA.
  - Standard MIB Modules.
- Vendor-specific MIB Modules.
- Best method to understand device structure.
- Association of devices and the elements they contain with service element types.
- Association of each service element type with a set of management objects (for now SNMP, will be adding CLI and others).
- Technology trees that get more and more specific until there are individual capabilities.
- Association of abstract capabilities with service element management objects.

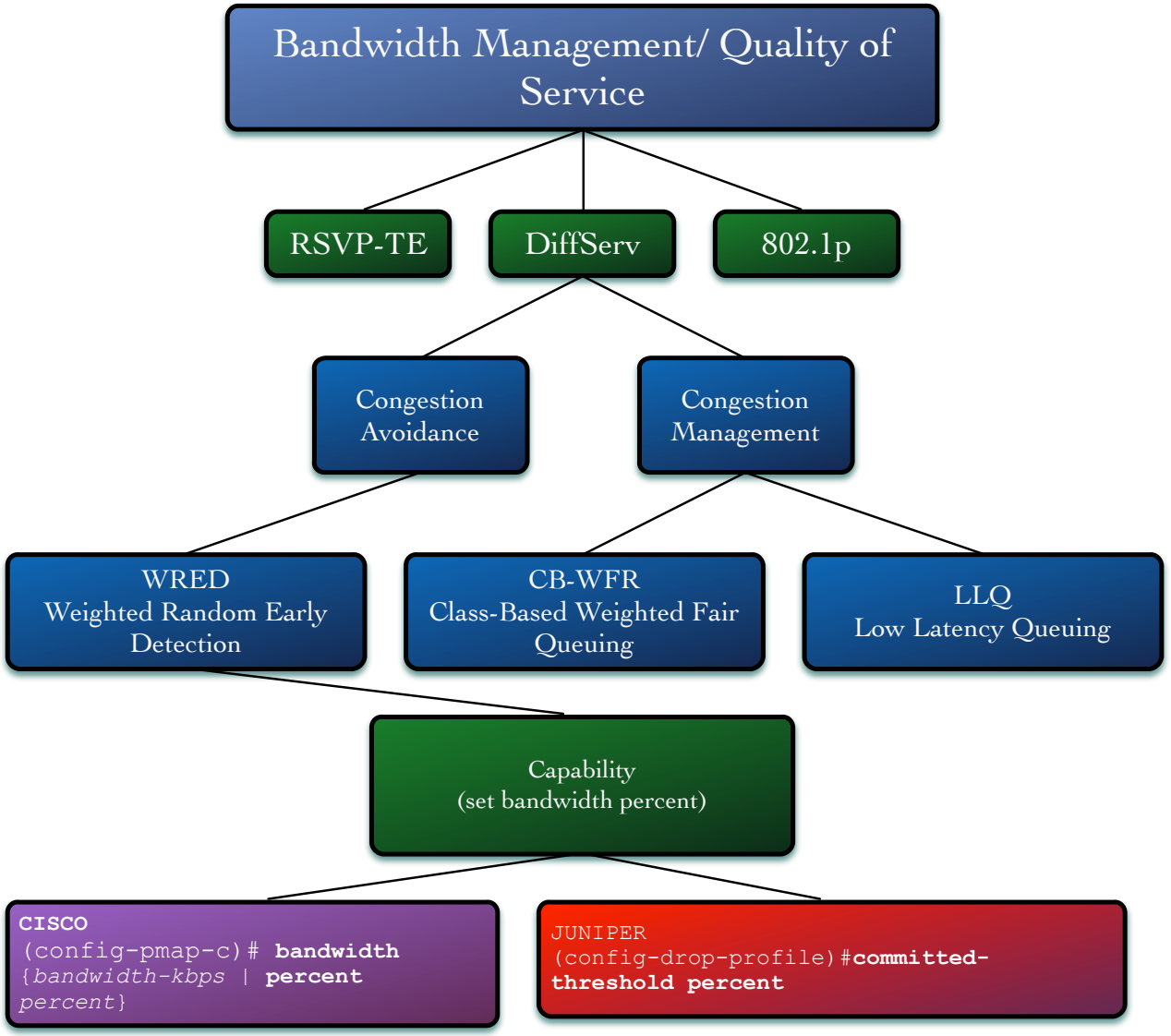
Integer's Challenge

- The critical challenge for a system like Integer:
  - Is it possible to create abstractions for the systems and technologies involved in delivery of modern IT Services?
  - Once the abstractions are created are they sufficiently flexible to reliably deal with all the different equipment, technology, software and hardware vendor differences?
  - Is it possible to connect these abstractions to the realities of each variation of:
    - Vendor.
    - Model.
    - Hardware version (including all the sub elements like disks and interface cards).
    - Software revisions.
    - Feature sets.
    - Differences due to configuration options selected.
  - Can this be done and still have a system that is scaleable and not overly-complex?

Domain Knowledge Enables Service Management



Portion of Bandwidth Management Technology Tree



Configuration of Domain Knowledge

- A lot of Domain Knowledge will come with the system.
- The environment is dynamic, provision must be made to:
  - Add/change/delete types of systems and components they contain.
  - Add/change/delete management objects.
  - Add/change/delete technologies and business services.
  - Add/change/delete relationships between management objects, technologies, service elements, etc.
- Provide a YAML-based format for users to perform these modifications. This may require programatic augmentation to achieve effective normalization across different data representations.
- User interface added later.
- Will develop a user contributions site for this information.

Using Integer

Selections

- Same approach for interacting with Integer for all functions, e.g,:
  - What you want to see in a report.
  - What you want to see on the screen.
  - What to discover.
- Elements of a selection
  - The systems/services.
  - Should topology information be included.
  - The information about the selected systems/services of interest
    - State and (configuration, discovery, fault, etc.).
    - Utilization and other historic information.
    - Capacity.
    - Calculated values input by users.
  - How to present the information, e.g,
    - As a topology map.
    - In a tabular report.
    - Output to CSV or JSON.

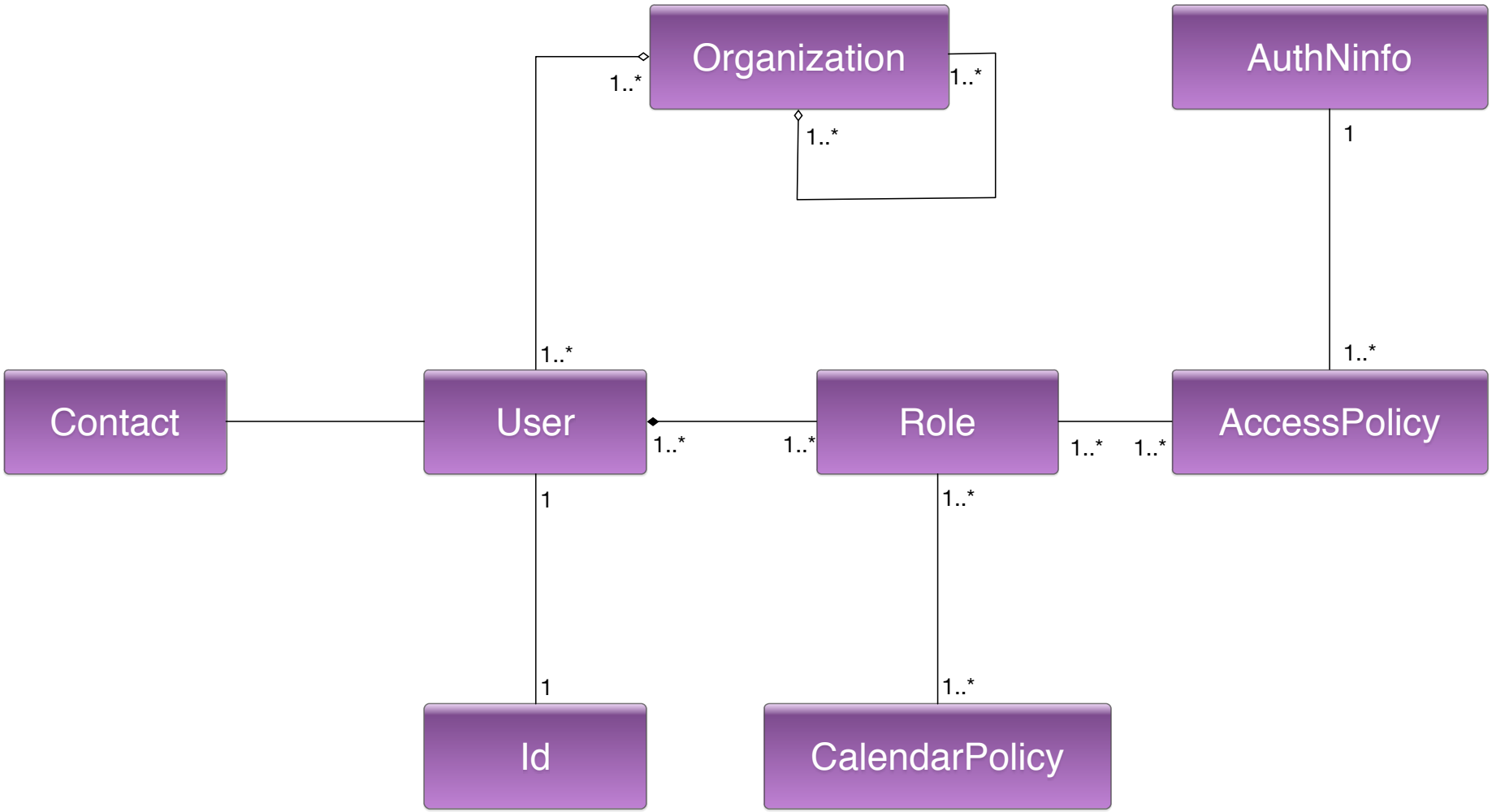
APIs and Documentation

- YAML for domain knowledge and other areas will be developed over time.
- Javadoc for all API's we use which is how data is accessed - not via DB.
- Javadoc generated as part of builds, updates available on public collaboration site.
- System design extensively documented in UML - model and free viewing tool on public web site.

Authentication & Authorization General

- First release supports CAS
  - Will use central CAS service.
  - Will experiment with two factor authentication for access to more sensitive functions.
- Supports direct log-in as an emergency for a few people in case elements of authorization or authentication infrastructure has failed.
- Other authentication technologies such as Shibboleth are possible.
- Uses WildFly facilities - adjust access privileges by accessed URL, e.g,
  - One is general with restricted privileges.
  - One will be configured in the CAS system with two factor authentication with broader configuration and control permissions.

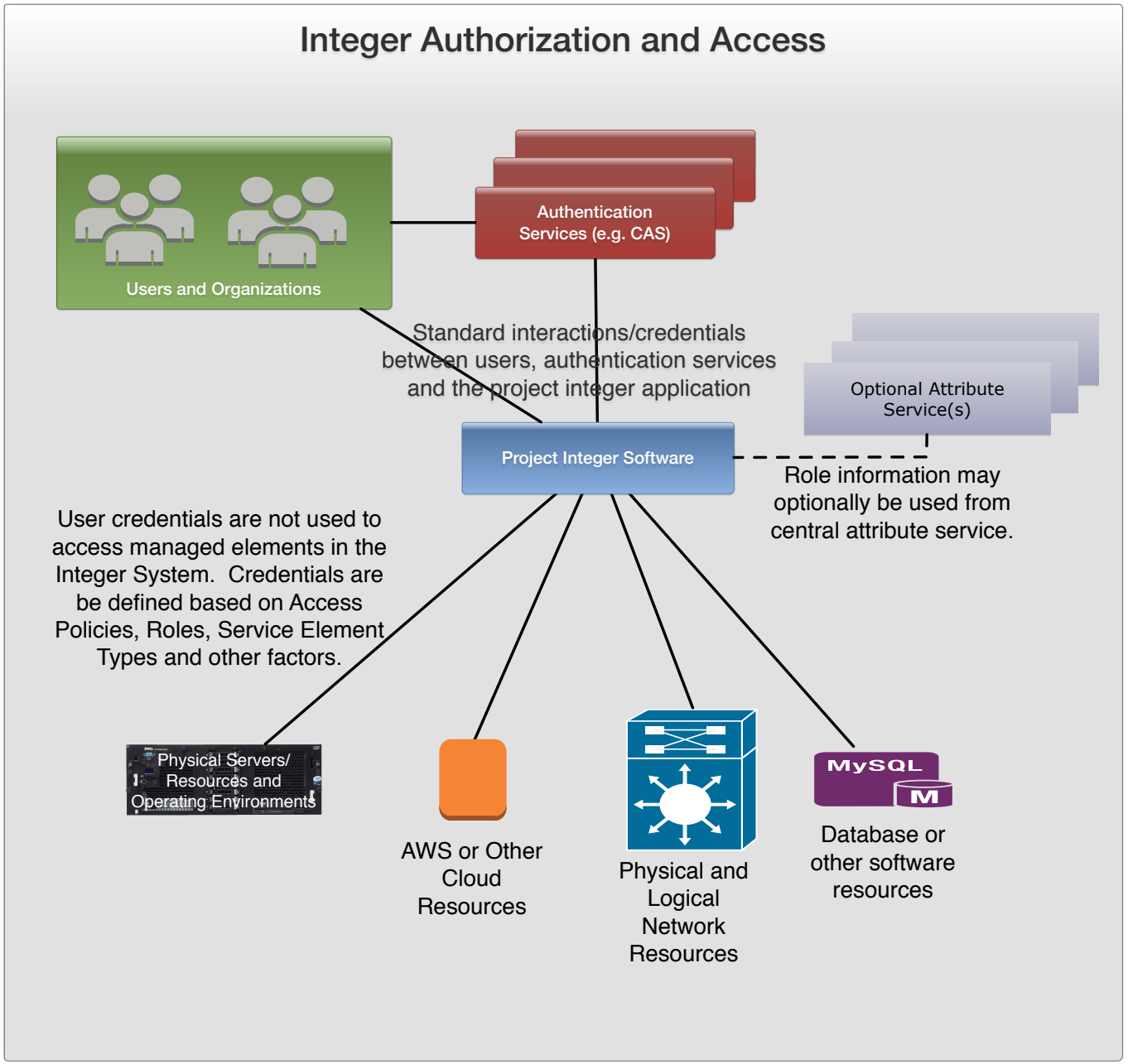
Authorization - Set of Relationships



Controlling Permissions

- Flexible enough to meet requirements of a complex organization.
- Supports role-based access control across several dimensions:
  - Authentication method.
  - Organization (can be hierarchical).
  - Time/calendar.
  - Location of managed systems.
  - Technology/function/type of equipment.
  - Function to be performed
    - Configuration.
    - Control.
    - Reading information.
  - Criticality of the system or system information accessed.
  - Specific device(s).

Access - the Whole Picture



Contact Us and Information

How You can Participate

- info@integer.harvard.edu - specific requests for information or to make comments.
- integerinfo@integer.harvard.edu - mailing list used periodically to send out news and updates on the initiative.
- integerproject@integer.harvard.edu - a subscription mailing list for people to discuss the initiative.
- Become a ‘tester’.
- Contribute other skills and knowledge.

Other Information

- open-integer/integer at GitHub.
- integer.harvard.edu.
  - Information.
  - Bug submissions and feature requests.
  - Information mailing lists.
  - Extensive use of other open source components.
  - Collaboration with other institutions.

integer.harvard.edu

Backup

Approaches and alternatives

- Continue current patchwork.
- More purchases of single purpose components.
- More purchases of large and expensive frameworks that are still substantially limited.
- Try something new.

Sources of Management Software (1)

- Home grown scripts and programs (usually operational staff)
  - Often created in isolation
  - Can be difficult to evolve and maintain them
  - Tend to be single purpose, solving a specific problem
  - Not well integrated with other systems/data.
- Open source solutions
  - Started in1990's but were limited in scope: monitoring basic server or network element statistics.
  - Today do some configuration, especially of servers, but too are limited in scope and do not integrate well with other systems (some trying to add on - just like commercial vendors).

Sources of Management Software (2)

- Commercial management software (e.g., Tivoli, HP, etc.)
  - Driven toward low hanging fruit in terms of function
  - Complex and difficult problems (like configuration) left unsolved.
  - Includes software from various public and private consortia - make assumptions about the ability to drive access methods, management objects or approach.
  - Ignore all outside the range of things over which they have direct control.
- Management software from hardware or software vendors (e.g., network equipment, server, storage, database)
  - Frequently the most complete
  - Targeted at the products produced by that vendor.
  - Costly
  - Poor integration - creates stove pipes. For example, information about servers is separate from network elements in these vendor solutions.

Integrating Stove Pipes

- High-cost
- Information loss
  - N-way translations
- High maintenance
- Fragile
- Subject to changes from any one component

A New Approach

- Not currently available
- A fully integrated environment
- One tool that integrates
  - Across Services
  - Technologies
  - Vendors
  - Management protocols and frameworks

Other Integration Aspects of Integer

- Integrates systems and software into different views:
- Key services like authorization and authentication, ERP applications, and custom applications like learning systems.
- Infrastructure services like routing, DNS, or load balancing
- Views of information based on role, such as high-level service view to details of how a server or router is functioning.

Can Integer Succeed?

Can Integer Succeed When Others Fail?

- Others start too large - Digital had over 100 engineers.
- Small start-ups undercapitalized 6-month focus.
- Operations expertise.
- Software engineering skills.
- Domain experience.
- Ability to collaborate in the open source community.
- Long-term vision and commitment.
- Shameless use of many other open source packages as major building blocks.

Is Integer Reinventing the Wheel?

- If there were a solution (or solutions) they would be widely adopted.
- “Previous attempts were not entirely successful”.
- Claims greatly exceeded reality.
- Integer does replicate existing functions but is unique with respect to:
  - Scope
    - Range of functions.
    - Types of systems.
  - Flexibility.
  - User configurability.

Keeping Integer Honest

- Public design reviews - on a regular basis.
- Open source.
- Close connection to operations.
- Building a community.
- Open APIs.
- Not bound to any protocol or single vendor or consortium view

Risks

- Funding.
- Focus.
- Technology - e.g. mapping between management protocols.
- Adoption:
  - Buy in to approach.
  - Ability to deliver desired features in desired time frame.

Integer is not a Slam Dunk

Security is mostly a superstition. It does not exist in nature, nor do the children of men as a whole experience it. Avoiding danger is no safer in the long run than outright exposure. Life is either a daring adventure, or nothing.

- Helen Keller

Far better is it to dare mighty things, to win glorious triumphs, even though checkered by failure... than to rank with those poor spirits who neither enjoy nor suffer much, because they live in a gray twilight that knows not victory nor defeat.

- Theodore Roosevelt