Naming Your Vista Environments & Repositories

On the Road to Automating Vista Distribution & Standardization

Vista Community Meeting 2019-06-04/05
George Mason University
Fairfax, Virginia

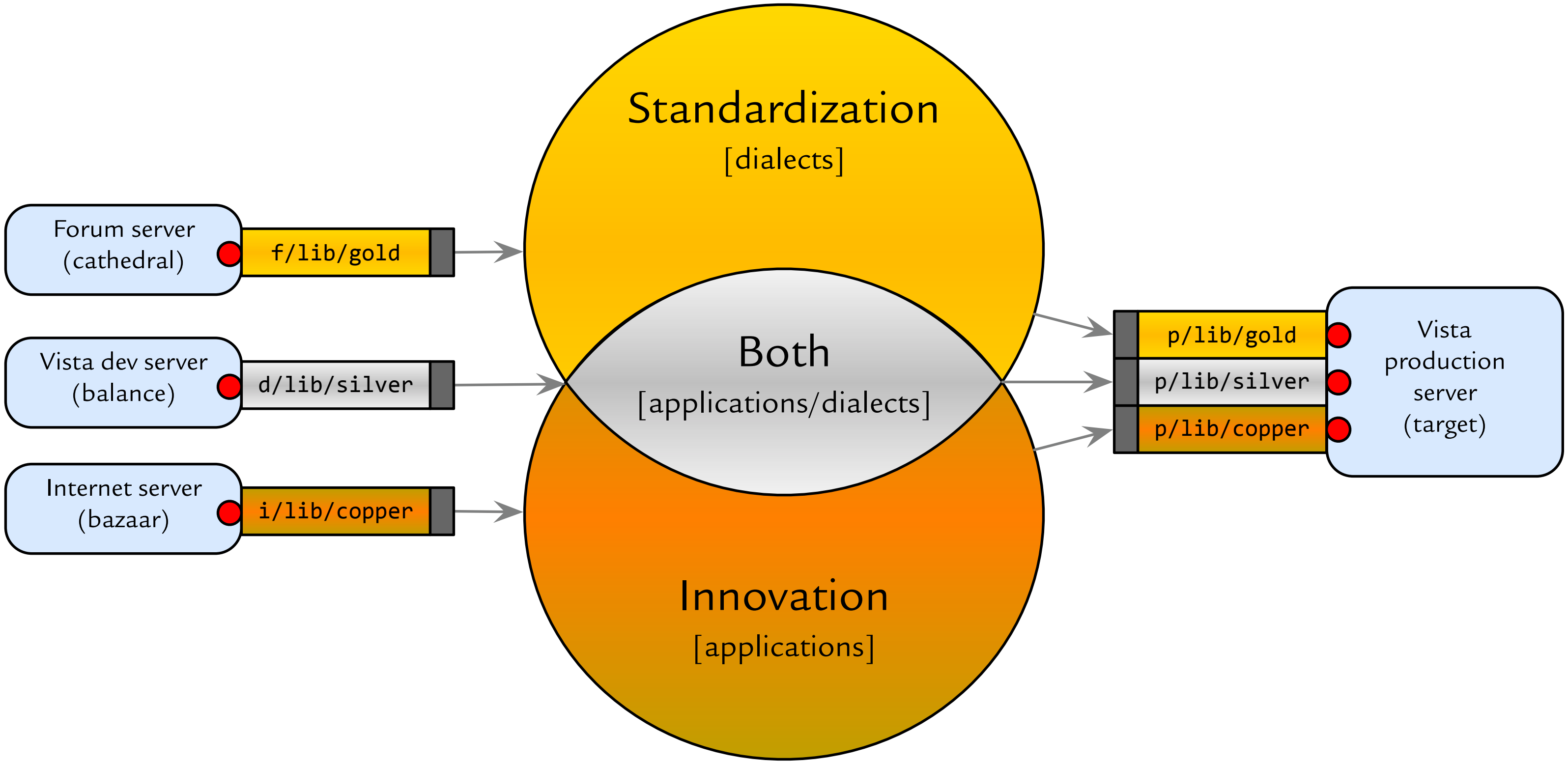
Frederick D. S. Marshall
Vista Expertise Network
rick.marshall@vistaexpertise.net

Repository Classes

(draft, 2019-06-05)

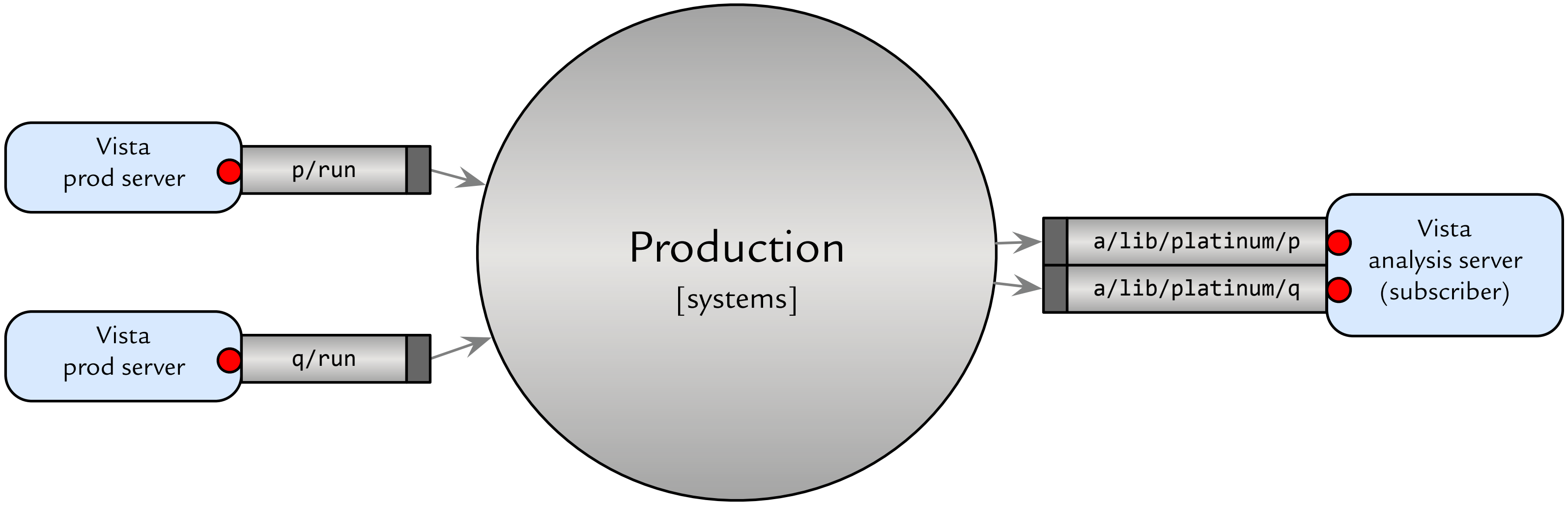
Distribution Repo Classes

Au	Gold repository
Ag	Silver repository
Cu	Copper repository



Production Repo Class

Pt	Platinum repository



Original Mumps VM: Jobs, Directories, UCIs, Volume Sets, & Machines

(draft, 2019-06-05)

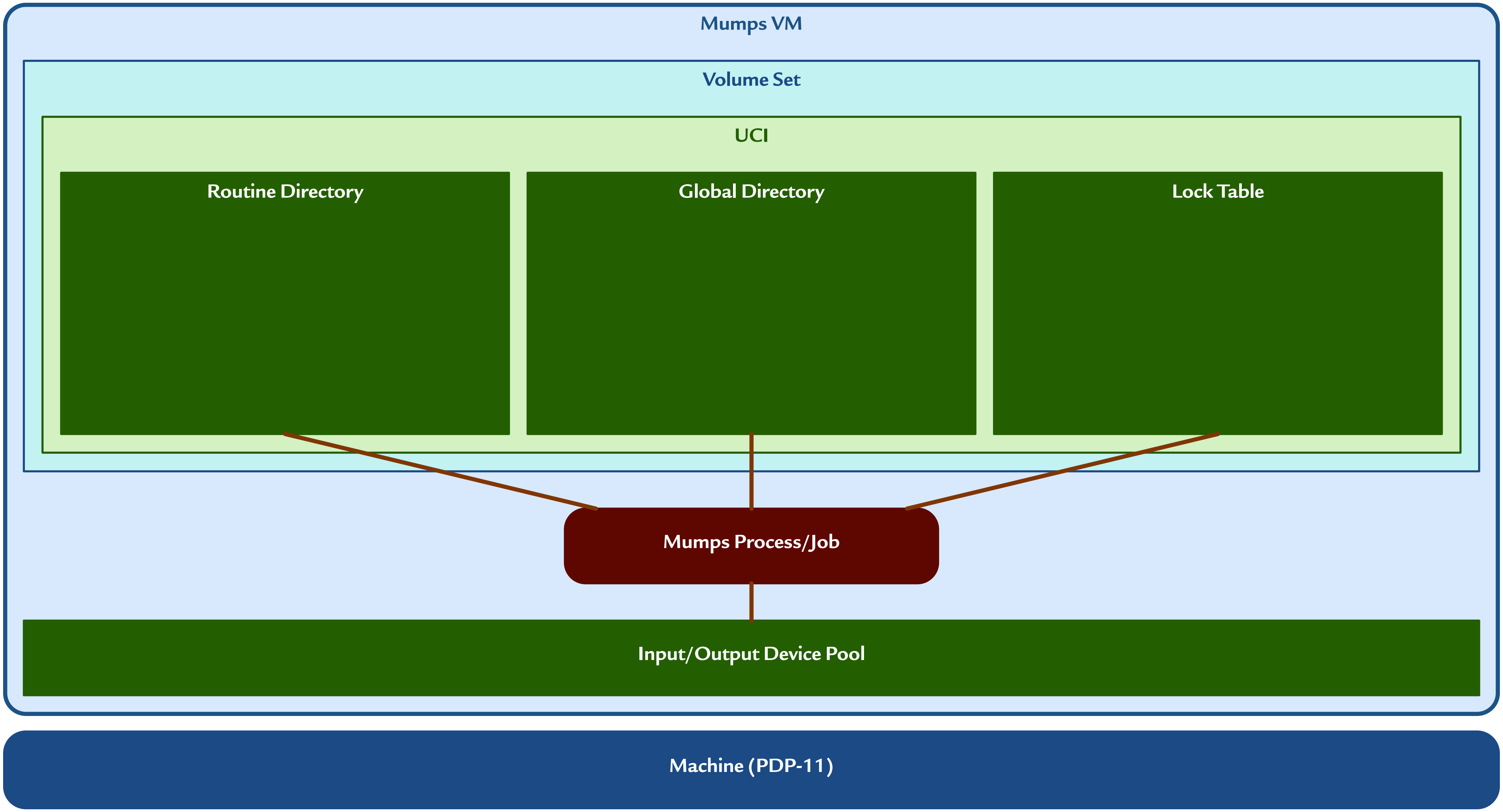
Mumps Directory: a set of names within which each name is unique, mainly globals & routines, sort of lock-names & devices.

User Class Identifier (UCI): a Mumps environment, containing a directory of globals, one of routines, & a lock-names and device pool.

Volume Set: a small set of related UCIs.

Mumps Virtual Machine (VM): Mumps as operating system for computer, hiding nonportable details, appearing as an abstract computer.

Machine: the computer hardware controlled by the Mumps virtual machine.



Original Mumps VM 2: Prototype Configuration

(draft, 2019-06-05)

Prototype DHCP Configuration: Plessey desktop machine with one UCI per major application, sign in by selecting UCI

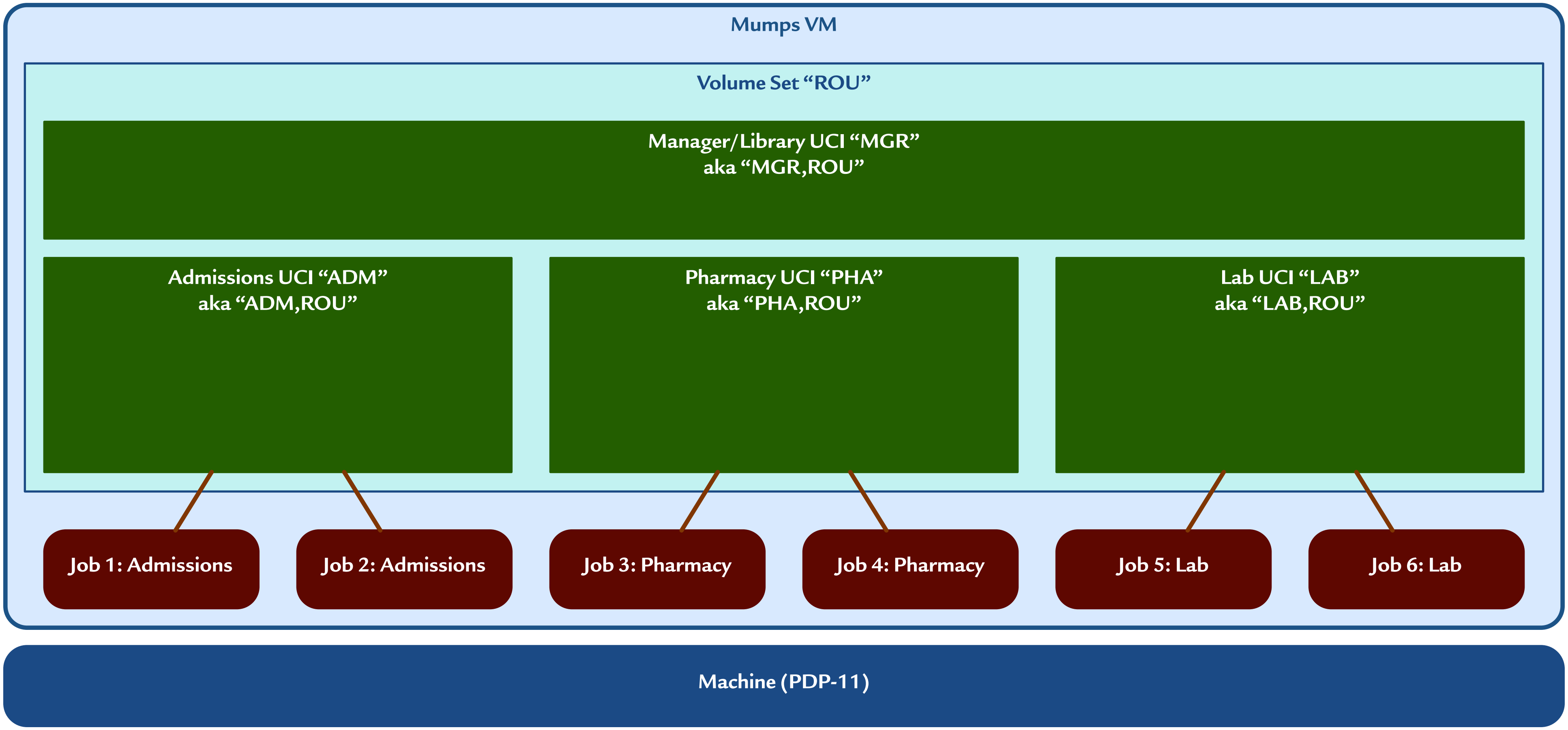
Vista Cookbook: a manual describing how to configure the Mumps VM & machine to support Vista.

Manager UCI: a Mumps environment containing system-management software plus software shared with other UCIs in volume set.

Mumps Implementor: the person or institution providing or selling a standard-compliant version of Mumps.

Job ID: a value that uniquely identifies this process within a pool of jobs determined by the Mumps implementor; typically per machine.

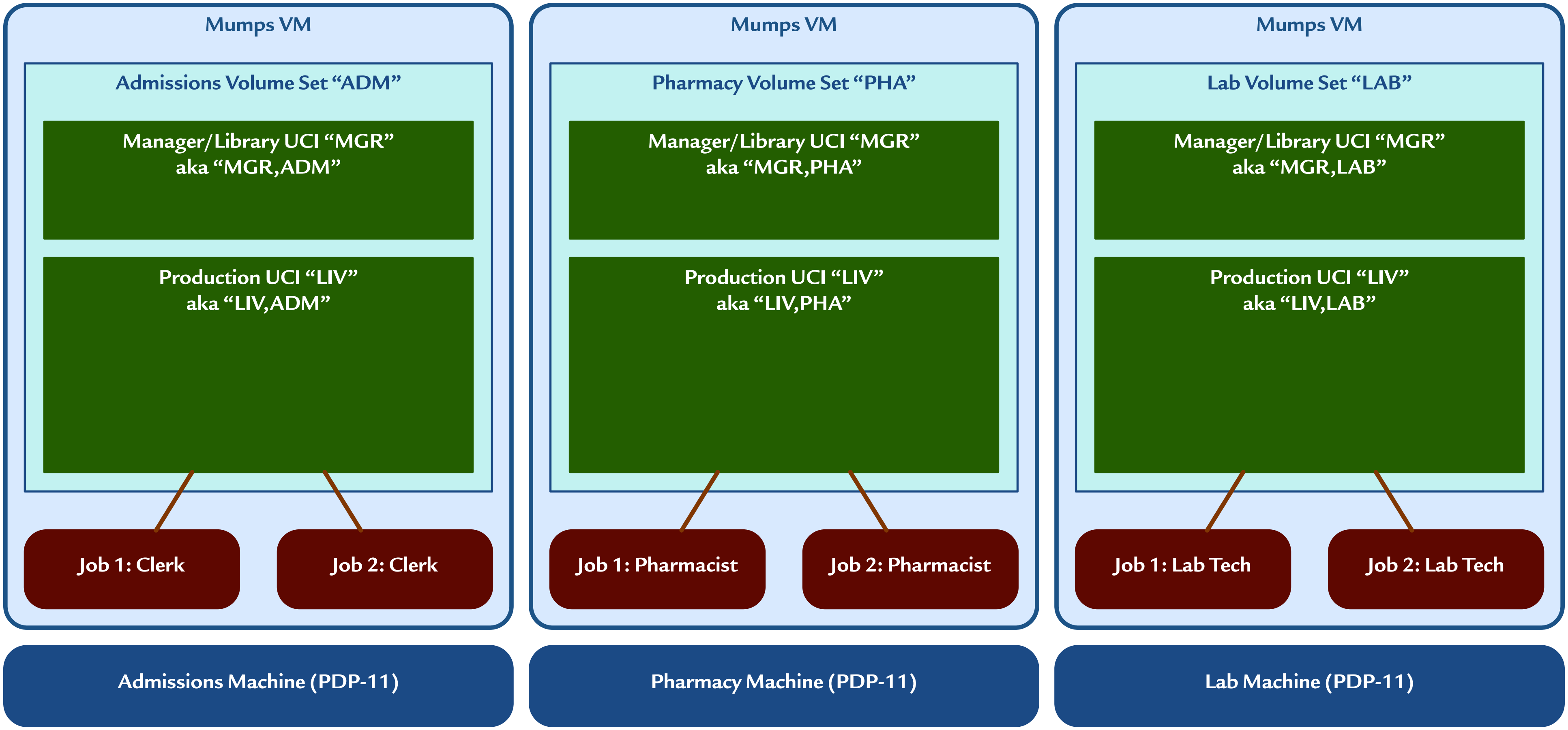
Extended Reference: to access a routine ^RUNME or global ^DATA in another UCI, do ^[“PHA”]RUNME or write ^[“LAB”]DATA.
to access it in another volume set, do ^[“PHA”,”ROU”]RUNME or write ^[“LAB”,”ROU”]DATA.



Original Mumps VM 3: Typical Early Configuration

(draft, 2019-06-05)

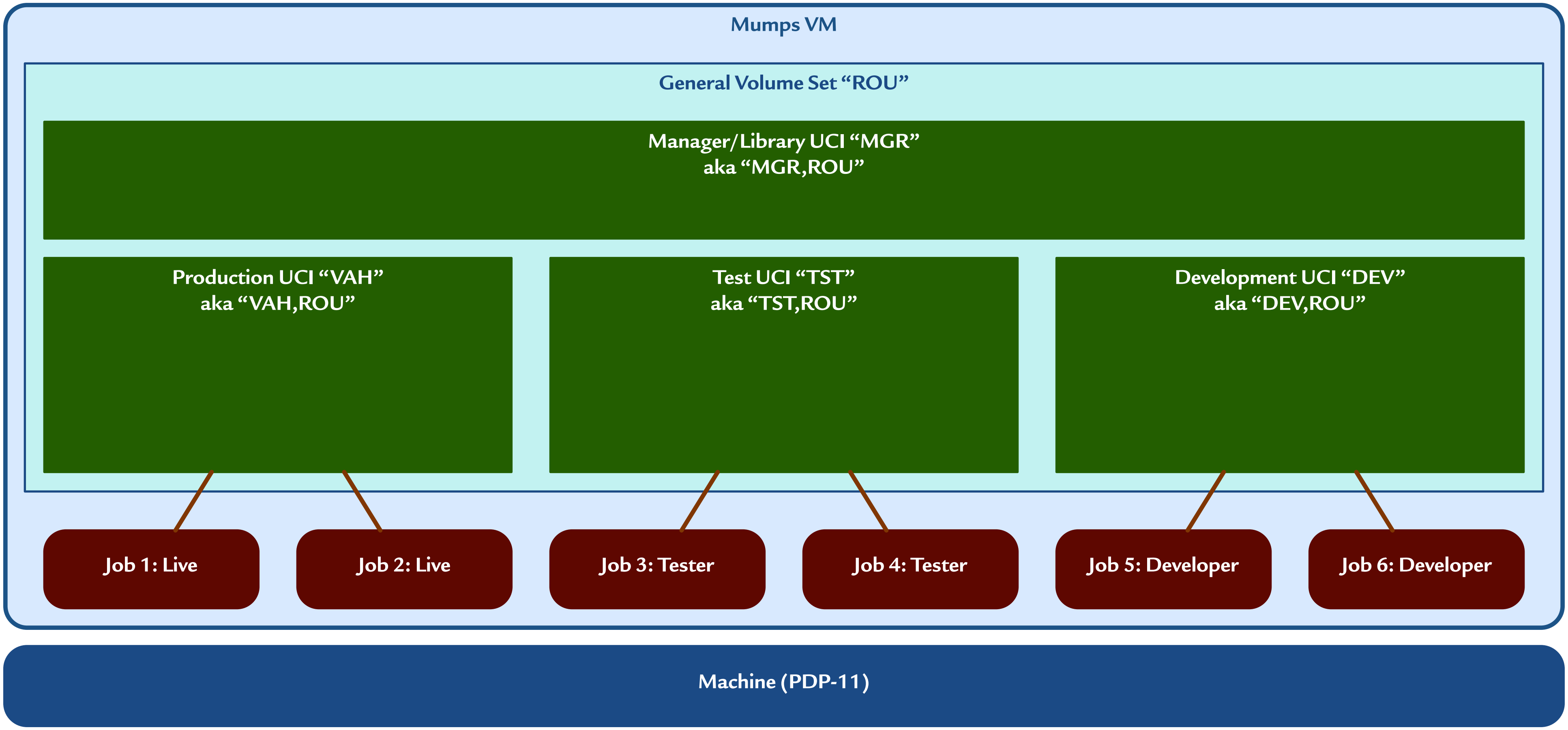
Early DHCP Configuration: each major application on a separate PDP-11, users segregated by application. Environments named according to the application they contain.


Mumps VM
Admissions Volume Set “ADM”
Manager/Library UCI “MGR” aka “MGR,ADM”
Production UCI “LIV” aka “LIV,ADM”
Job 1: Clerk
Job 2: Clerk
Admissions Machine (PDP-11)
Mumps VM
Pharmacy Volume Set “PHA”
Manager/Library UCI “MGR” aka “MGR,PHA”
Production UCI “LIV” aka “LIV,PHA”
Job 1: Pharmacist
Job 2: Pharmacist
Pharmacy Machine (PDP-11)
Mumps VM
Lab Volume Set “LAB”
Manager/Library UCI “MGR” aka “MGR,LAB”
Production UCI “LIV” aka “LIV,LAB”
Job 1: Lab Tech
Job 2: Lab Tech
Lab Machine (PDP-11)

Original Mumps VM 4: Prototype Lifecycle Configuration

(draft, 2019-06-05)

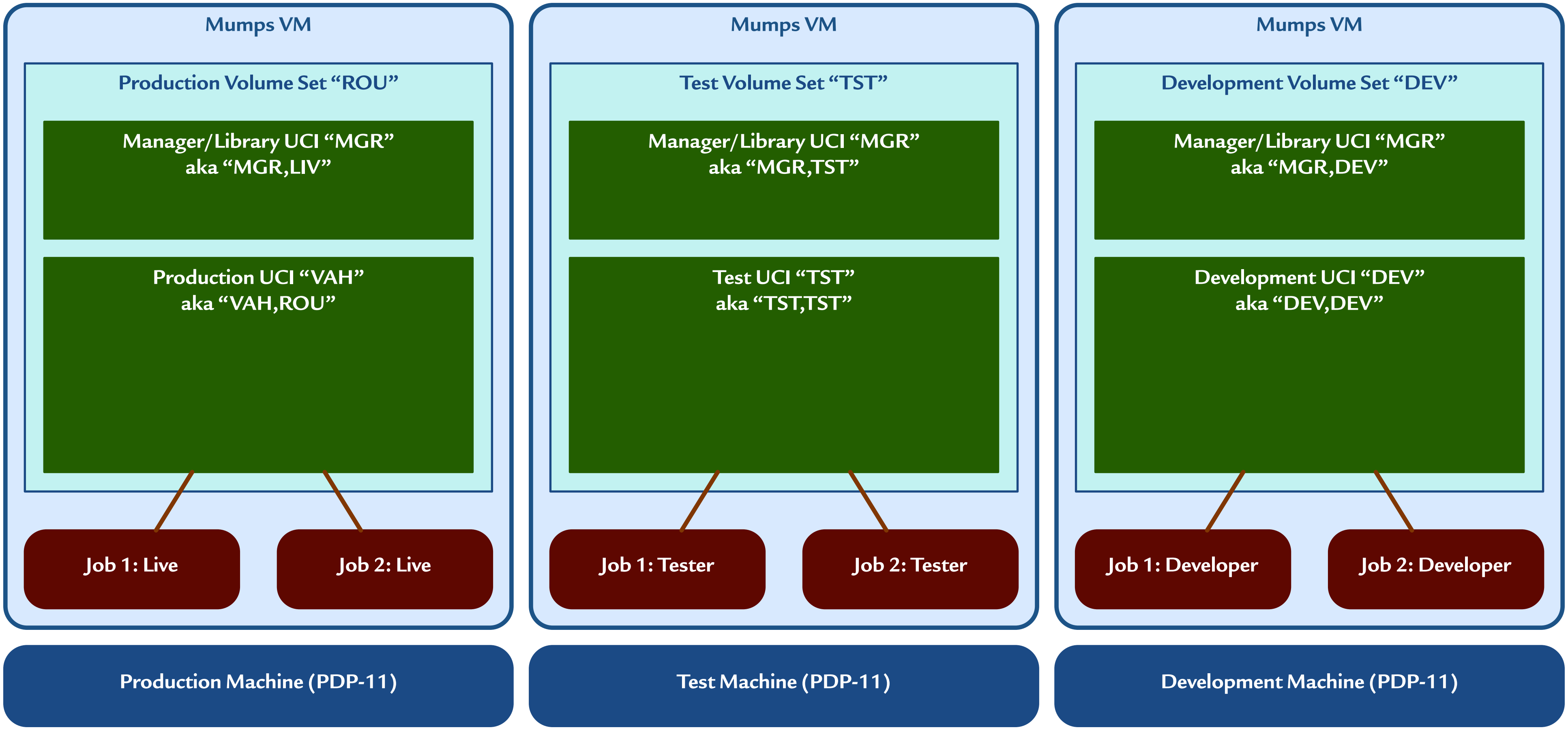
Prototype Lifecycle Configuration: to develop one application, separate live production code in VAH from code being tested in TST from code being developed in DEV. Unfortunately, shared MGR UCI means development of MGR routines can disrupt production. Environment names differ to help clue everyone in about how carefully to tread in the current environment.



Original Mumps VM 5: Safer Lifecycle Configuration

(draft, 2019-06-05)

Improved Lifecycle Configuration: separating development from testing from production insulates production users from development and testing activities. Even a runaway job that consumes or crashes the CPU can't hurt production in this configuration. Environment names are usually different, to help make the configuration system by ensuring programmers are less likely to be confused about which environment they are in, so they don't accidentally do development in production.


Mumps VM
Production Volume Set "ROU"
Manager/Library UCI "MGR" aka "MGR,LIV"
Production UCI "VAH" aka "VAH,ROU"
Job 1: Live
Job 2: Live
Production Machine (PDP-11)
Mumps VM
Test Volume Set "TST"
Manager/Library UCI "MGR" aka "MGR,TST"
Test UCI "TST" aka "TST,TST"
Job 1: Tester
Job 2: Tester
Test Machine (PDP-11)
Mumps VM
Development Volume Set "DEV"
Manager/Library UCI "MGR" aka "MGR,DEV"
Development UCI "DEV" aka "DEV,DEV"
Job 1: Developer
Job 2: Developer
Development Machine (PDP-11)

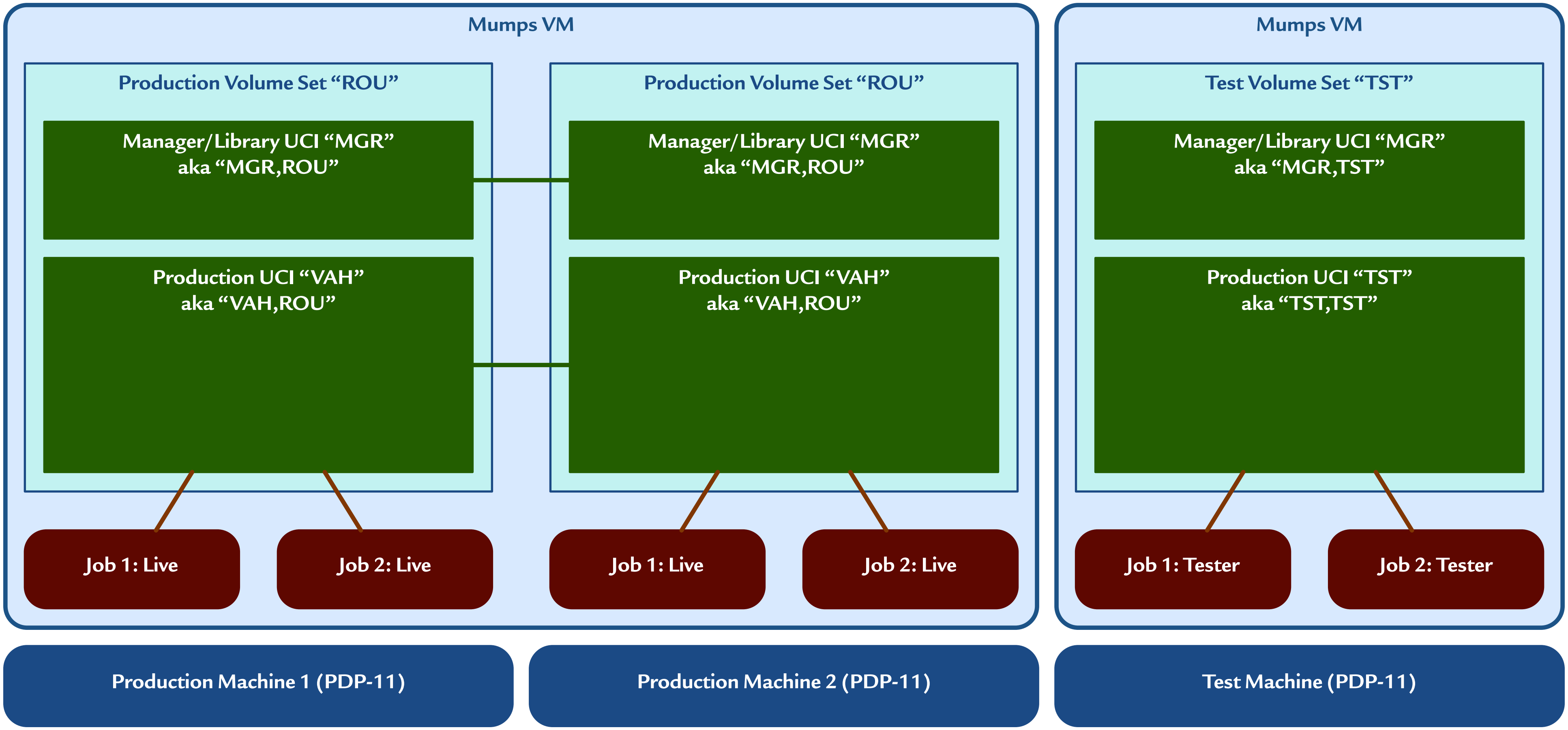
Original Mumps VM 6: Replication/Translation Configuration

(draft, 2019-06-05)

Incremental Server Upgrade Configuration: used to add more horsepower by adding more servers, which share the globals and routines through one of two strategies. Environment names are kept identical, along with the rest of the Mumps database.

Replication: global sets & kills on machine 1 copied to machine 2.

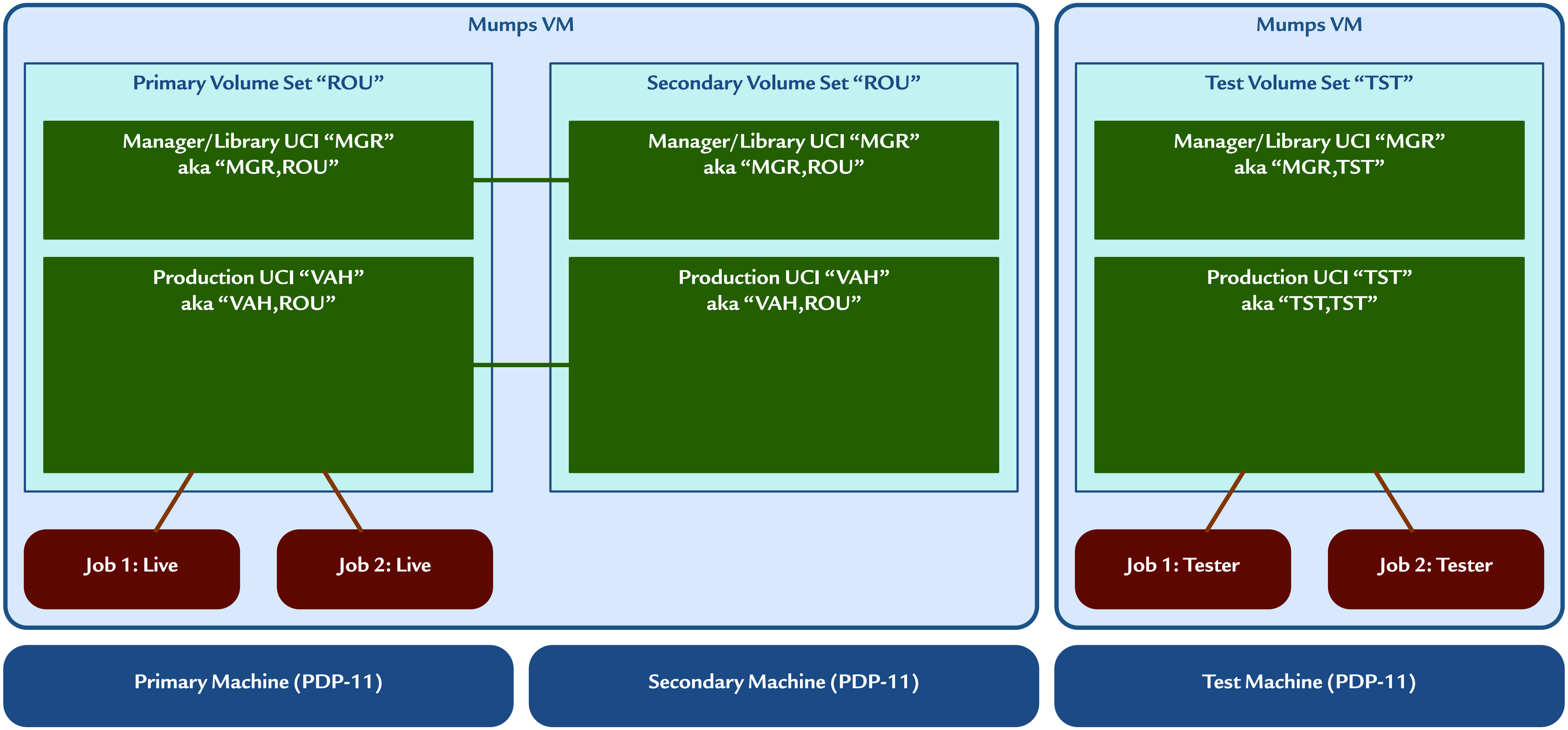
Translation: global sets & kills on machine 2 reach across local network to machine 1.



Original Mumps VM 7: Failover Configuration

(draft, 2019-06-05)

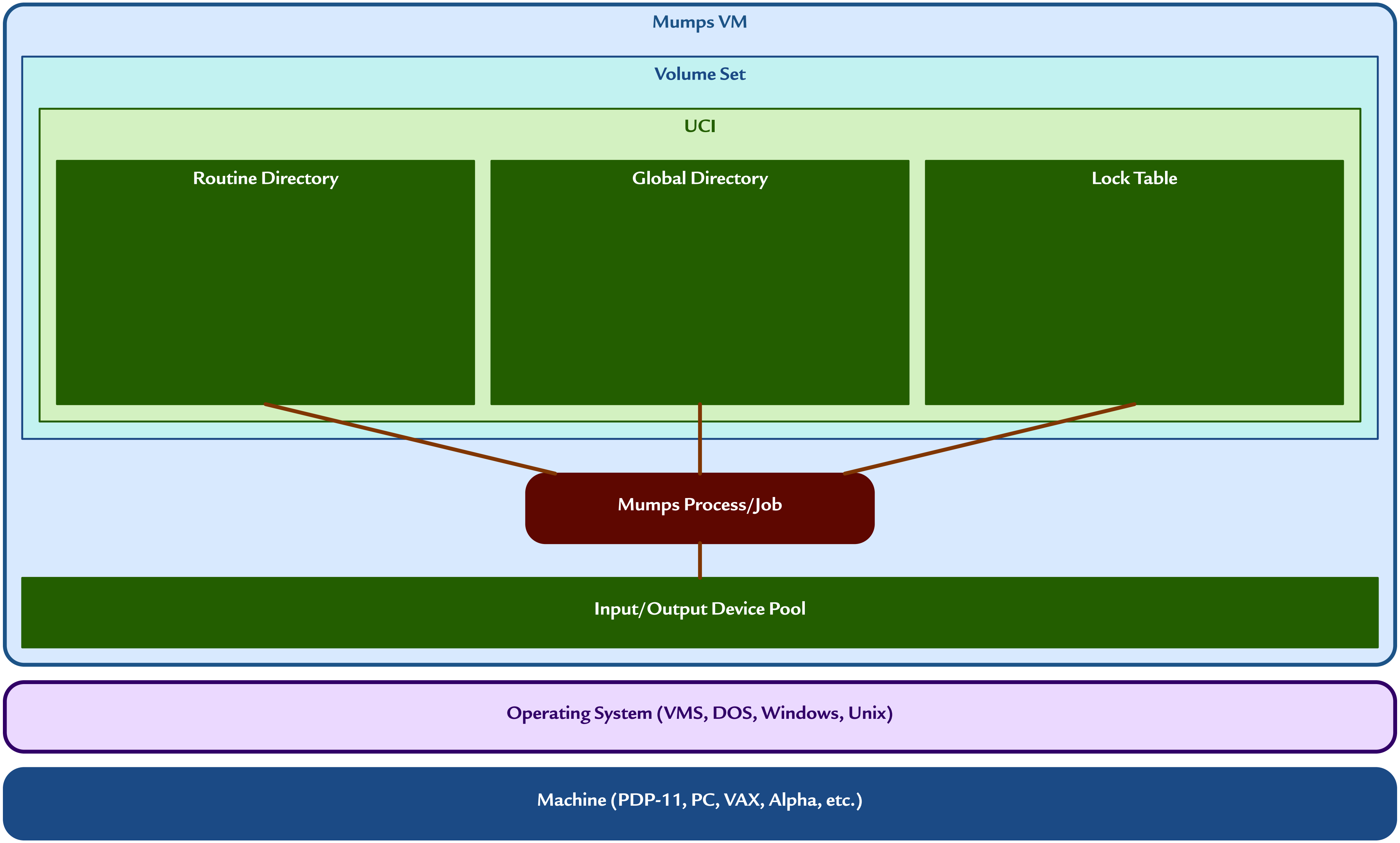
Failover Configuration: for 24/7/365 reliable uptime, we need to plan for failure. We set up a failover machine as a safety net, with duplicated contents from the primary the users are on. If the primary crashes, it takes only seconds to switch users over to the secondary, a process called failover, where they can pick up exactly where they left off on the primary, since it was a copy kept in synch by replication. The primary machine can then be repaired or replaced and then reintroduced as the failover machine. Environment names are intentionally kept identical, so the secondary looks to the users as though it is the same machine.


Mumps VM
Primary Volume Set "ROU"
Manager/Library UCI "MGR" aka "MGR,ROU"
Production UCI "VAH" aka "VAH,ROU"
Secondary Volume Set "ROU"
Manager/Library UCI "MGR" aka "MGR,ROU"
Production UCI "VAH" aka "VAH,ROU"
Job 1: Live
Job 2: Live
Mumps VM
Test Volume Set "TST"
Manager/Library UCI "MGR" aka "MGR,TST"
Production UCI "TST" aka "TST,TST"
Job 1: Tester
Job 2: Tester
Primary Machine (PDP-11)
Secondary Machine (PDP-11)
Test Machine (PDP-11)

Hosted Mumps VM: Original Mumps Virtual Machine + Operating System

(draft, 2019-06-05)

Operating System: at first, a set of features to insulate most Mumps jobs from; increasingly, a set of services to provide portably.



Hosted Mumps VM 2: Cluster Configuration

(draft, 2019-06-05)

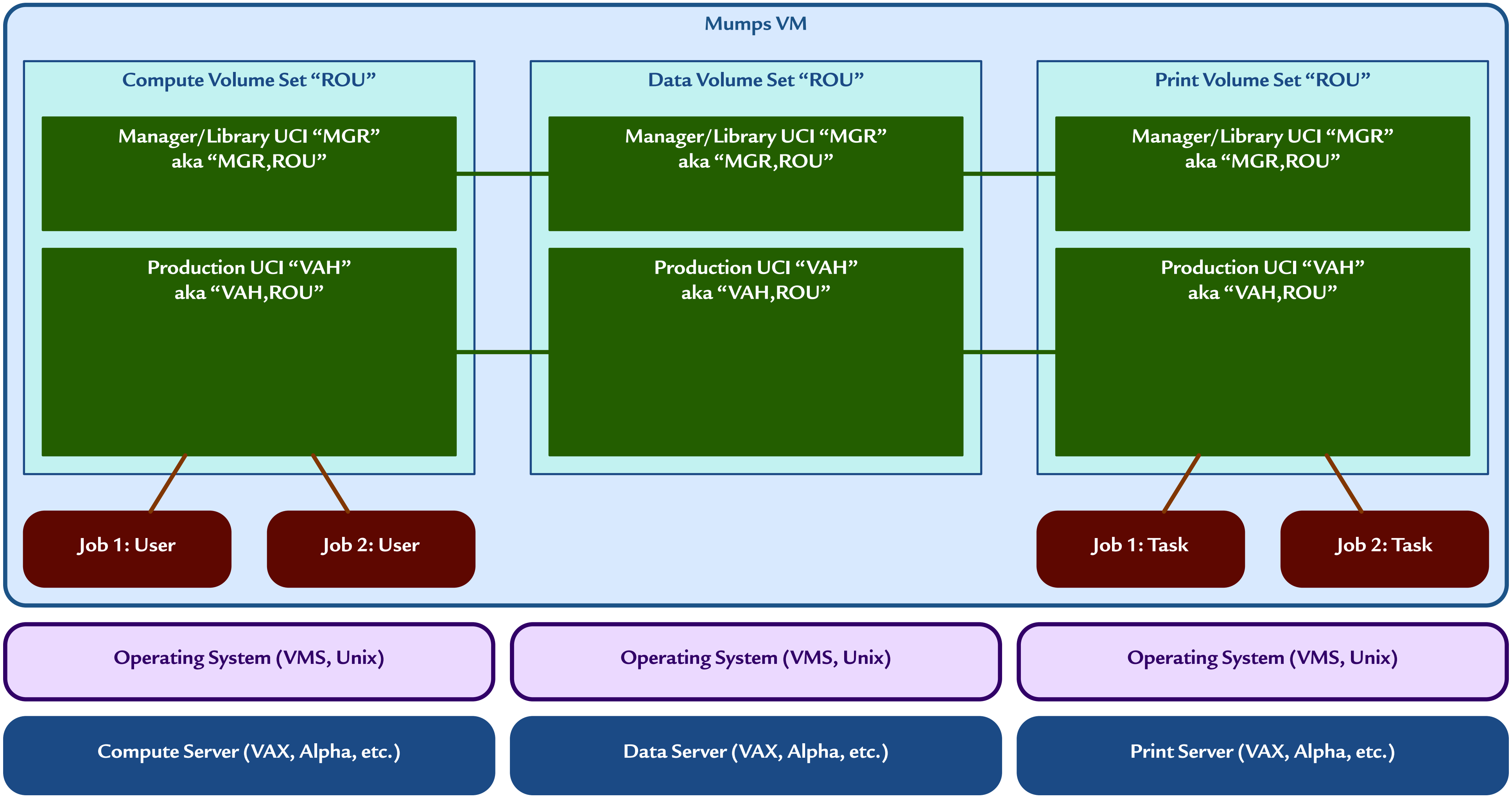
Cluster: a group of nodes (computers, "boxes") that share a database, spreading out the load of the jobs across multiple servers, often specialized.

Data Server: a computer dedicated more or less full time to serving up translated globals to other servers.

Compute Servers: a group of computers dedicated to running Mumps user jobs; almost all global accesses are translated to data server(s).

Print Servers: one or more computers dedicated to running Mumps background jobs (tasks); most globals are translated.

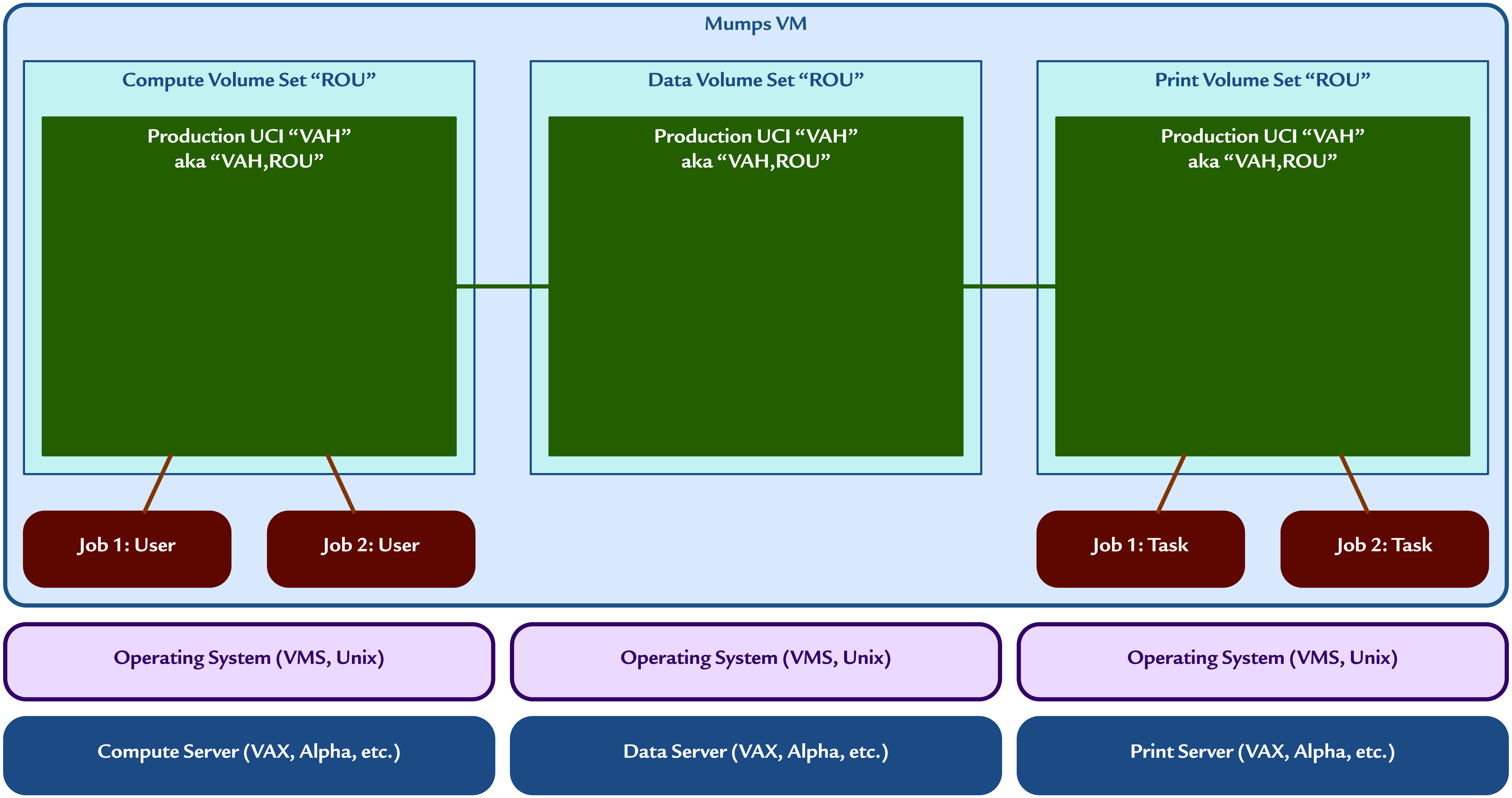
Load Balancing: a system of ensuring no server is underutilized or overutilized.


Mumps VM
Compute Volume Set "ROU"
Data Volume Set "ROU"
Print Volume Set "ROU"
Manager/Library UCI "MGR" aka "MGR,ROU"
Manager/Library UCI "MGR" aka "MGR,ROU"
Manager/Library UCI "MGR" aka "MGR,ROU"
Production UCI "VAH" aka "VAH,ROU"
Production UCI "VAH" aka "VAH,ROU"
Production UCI "VAH" aka "VAH,ROU"
Job 1: User
Job 2: User
Job 1: Task
Job 2: Task
Operating System (VMS, Unix)
Operating System (VMS, Unix)
Operating System (VMS, Unix)
Compute Server (VAX, Alpha, etc.)
Data Server (VAX, Alpha, etc.)
Print Server (VAX, Alpha, etc.)

Hosted Mumps VM 3: Simplified Cluster Configuration

(draft, 2019-06-05)

No More MGR UCI: Eventually, surprisingly late, we noticed there was no longer any reason to have a MGR UCI, because these more powerful computers and Mumps implementations are perfectly capable of storing all applications in the same UCI. Indeed, we had been doing so for some time - we had to - because in the more modern Vista architecture the applications are integrated and communicate closely, which is awkward if they are siloed into separate UCIs. With only one production UCI, there's no point any more to MGR UCIs, nor to %-routines and -globals, so in the late 1990s we finally noticed and eliminated the MGR UCI. %-routines and -globals are slowly being phased out.


Mumps VM
Compute Volume Set "ROU"
Production UCI "VAH" aka "VAH,ROU"
Data Volume Set "ROU"
Production UCI "VAH" aka "VAH,ROU"
Print Volume Set "ROU"
Production UCI "VAH" aka "VAH,ROU"
Job 1: User
Job 2: User
Job 1: Task
Job 2: Task
Operating System (VMS, Unix)
Operating System (VMS, Unix)
Operating System (VMS, Unix)
Compute Server (VAX, Alpha, etc.)
Data Server (VAX, Alpha, etc.)
Print Server (VAX, Alpha, etc.)

Hosted Mumps VM 4: Native Configuration (Datatree & Greystone Implementations)

(draft, 2019-06-05)

The Native Configuration Revolution: what if the Mumps job contained its own virtual machine, instead of the other way around?

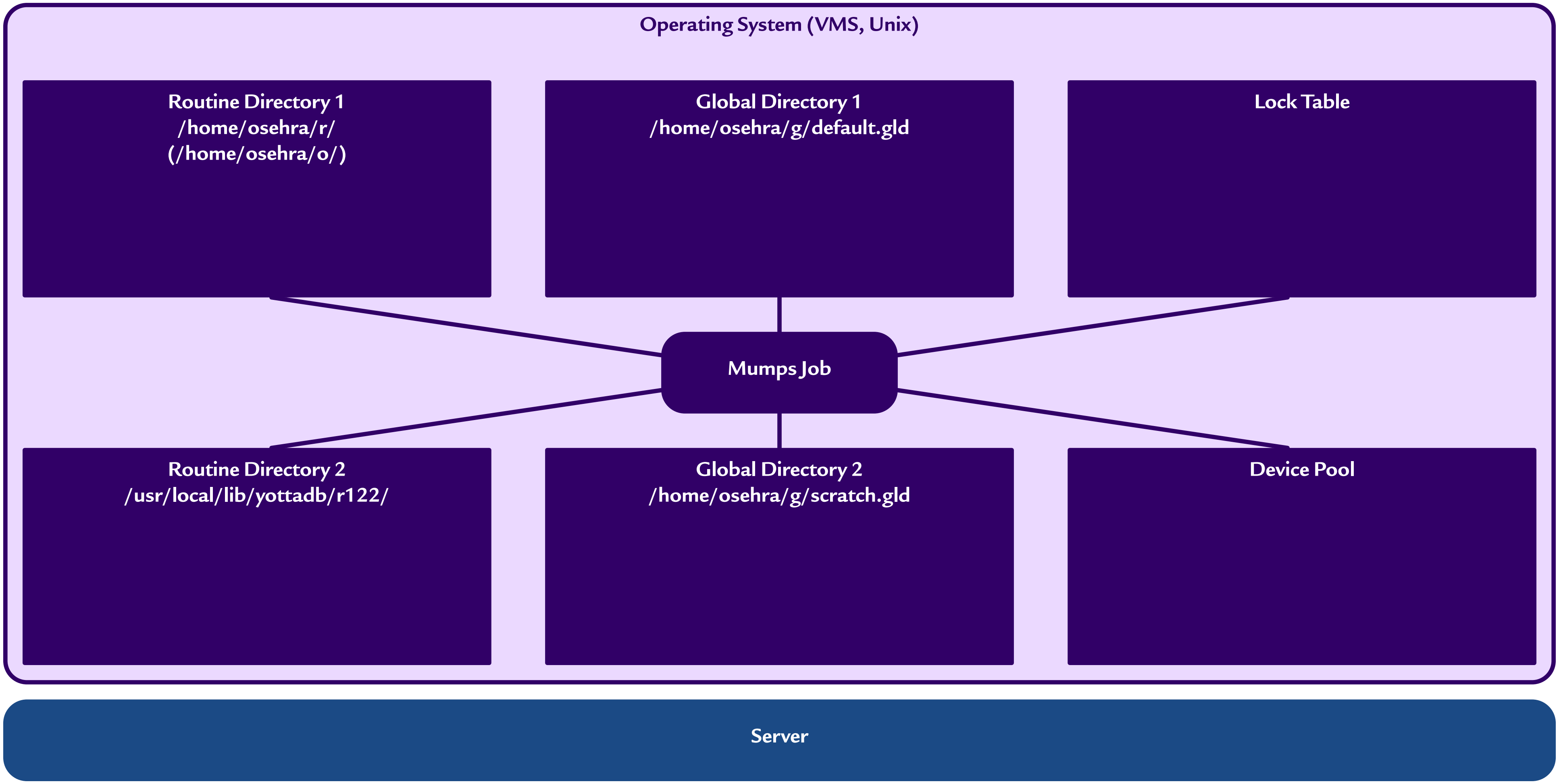
Mumps Job: an operating-system process that is running the Mumps VM software.

Directory: an actual operating-system directory containing routine files or global dataset files.

Routine Source File: the source text for a Mumps routine stored in a text file, e.g., RUNME.m or _wfhform.m.

Routine Object File: the compiled binary file for a Mumps routine, e.g., RUNME.o or _wfhform.o.

Global Dataset File: a binary file containing one or more global variables, often grouped into a database segment, e.g., default.gld.



Standard Mumps VM: Jobs, Environments, Systems, & Machines

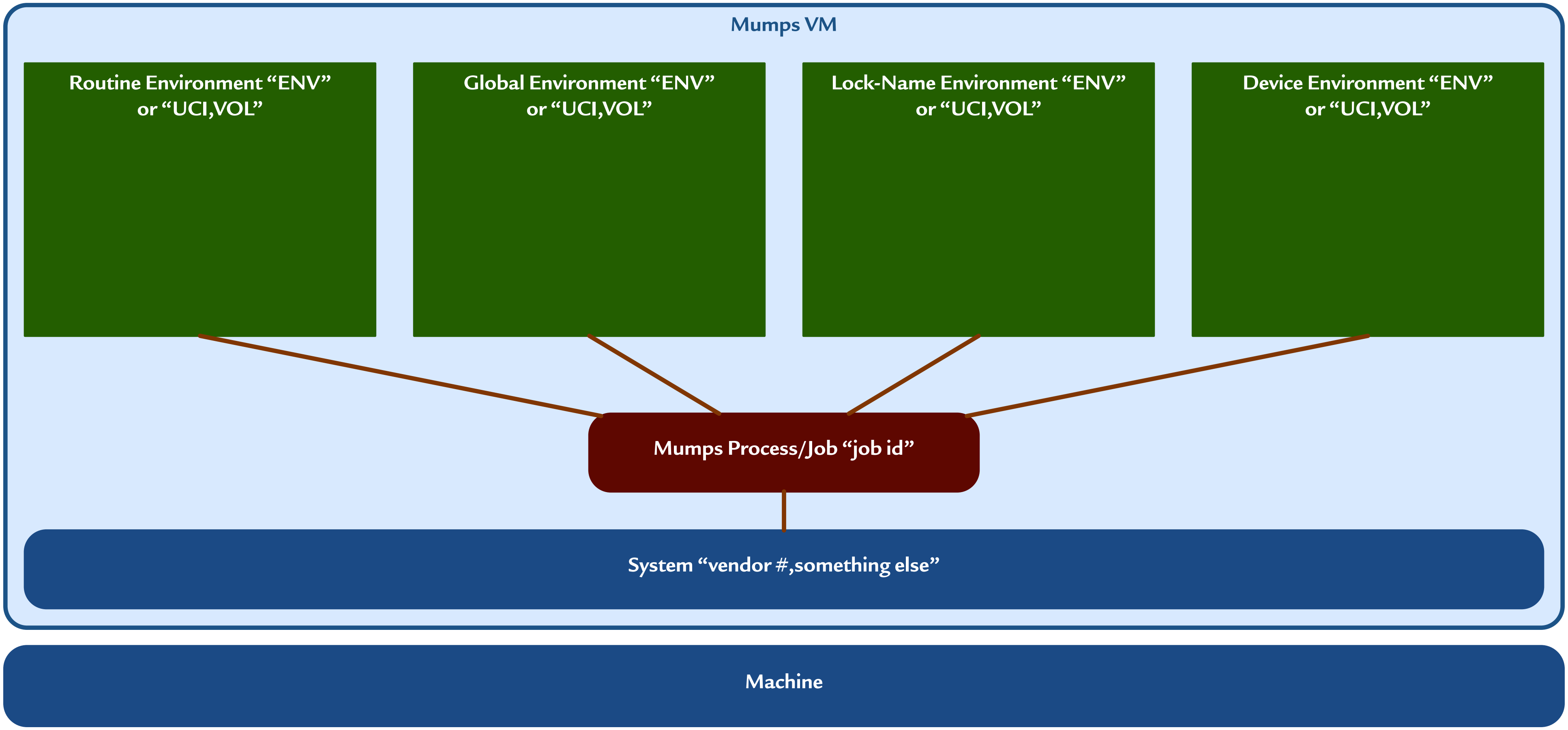
(draft, 2019-06-05)

Mumps 1995 Virtual Machine: the new Mumps standard significantly changed the original Standard Mumps Virtual Machine, with an eye toward connecting Mumps environments with each other and with external software written in other languages. The old conventional implementor terminology & syntax was overlaid with new standards. But although this extended the original Mumps virtual machine, it did not take into account the inside-out transformation of the Native Configuration, nor its re-centering in the operating system. Mumps 1995 focused on portability, with hiding the operating system except through defined interfaces, which left Mumps with two incompatible models.

Mumps Environment: directory of names, set of names within which each name is unique, applies to globals, lock-names, routines, & devices.

Mumps System: a job id environment, a set of jobs within which each job id is unique.

Extended Reference: to access a routine ^RUNME or global ^DATA in another environment, do ^|“LIV”|RUNME or write ^|“LIV”|DATA. there are no UCIs or volumes, but names can look like them, such as do ^|“LIV,VOL”|RUNME or write ^|“LIV,VOL”|DATA.

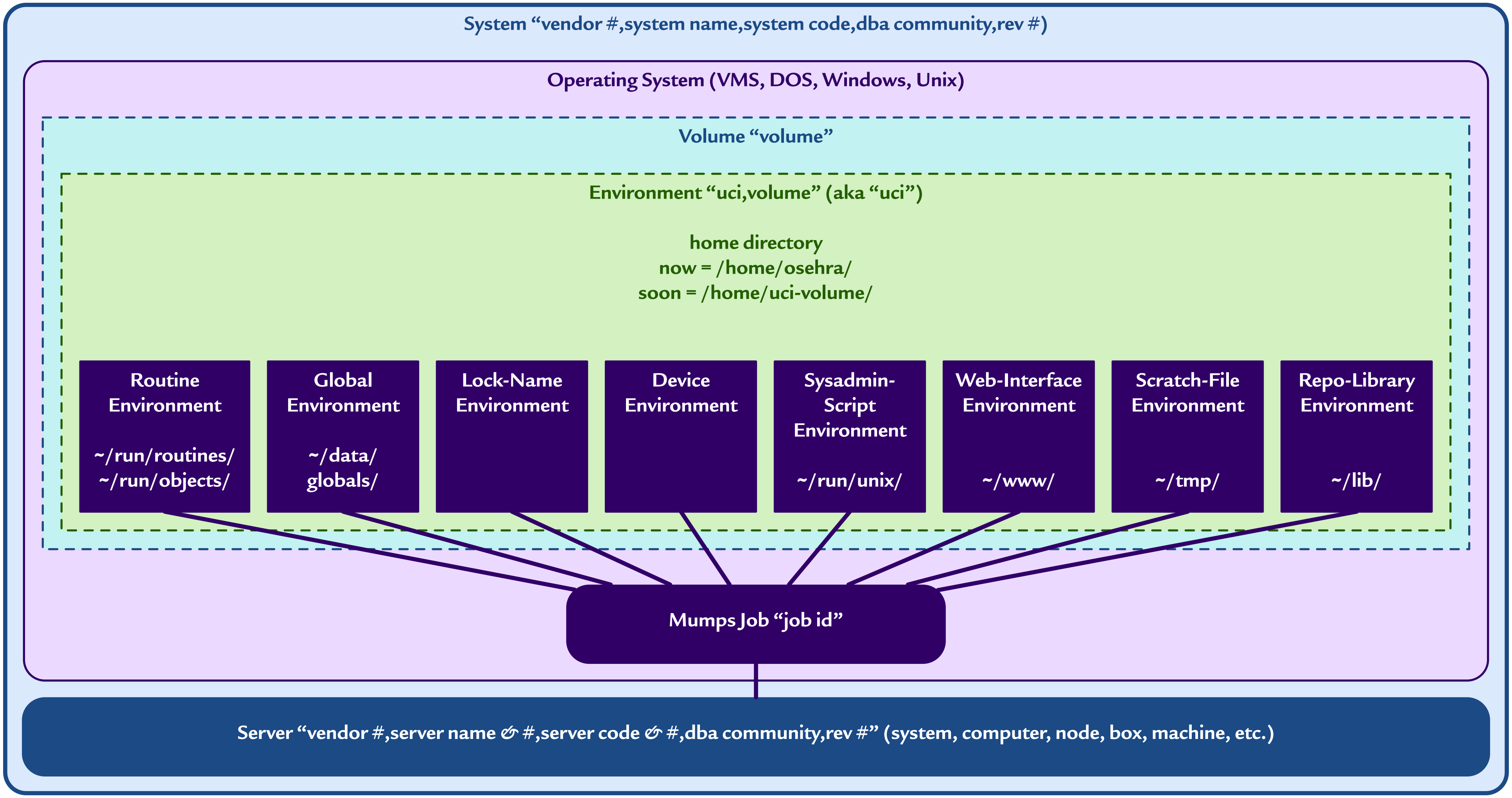

Mumps VM
Routine Environment “ENV” or “UCI,VOL”
Global Environment “ENV” or “UCI,VOL”
Lock-Name Environment “ENV” or “UCI,VOL”
Device Environment “ENV” or “UCI,VOL”
Mumps Process/Job “job id”
System “vendor #,something else”
Machine

Mash VM: Jobs, Environments, Systems, & Operating Systems

(draft, 2019-06-05)

Mumps Advanced Shell Virtual Machine: the new Mumps infrastructure layer needs to marry the Mumps 1995 model with the native-configuration model, to make the operating system visible and central to Mumps computing, to help wrap up and provide access to operating-system services. Mumps needs to fully embrace the larger computing world it is part of, so it needs to standardize how Mumps is configured within that world, to produce predictable, automatable architecture those features can be built upon.

But it also needs to preserve enough of the terminology of the original Mumps Virtual Machine so existing infrastructure software continues to work and make sense to sysadmins. This model maps the original Mumps, Mumps 1995, & Native Configuration models onto a new virtual machine. These are therefore the semantic elements our new Mash standard needs to assign unique names to.


System "vendor #,system name,system code,dba community,rev #)
Operating System (VMS, DOS, Windows, Unix)
Volume "volume"
Environment "uci,volume" (aka "uci")
home directory
now = /home/osehra/
soon = /home/uci-volume/
Routine Environment
~/run/routines/
~/run/objects/
Global Environment
~/data/
globals/
Lock-Name Environment
Device Environment
Sysadmin-Script Environment
~/run/unix/
Web-Interface Environment
~/www/
Scratch-File Environment
~/tmp/
Repo-Library Environment
~/lib/
Mumps Job "job id"
Server "vendor #,server name & #,server code & #,dba community,rev #" (system, computer, node, box, machine, etc.)

Mash Naming 1: Dialect & Developer

(draft, 2019-06-05)

🟦 open & healthy 🟪 incomplete 🟩 proprietary 🟧 troubled 🟥 defunct ⬜ n/a

Dialect: Vista codebase managed as a whole & partly compatible w/other dialects; gold repos transport entire dialects; need codes for names

🟥	vac	CASS Mumps Systems (Veterans Affairs) [FOIA]
🟥	vad	DHCP (Veterans Affairs) [FOIA]
🟦	vav	VA Vista (Veterans Affairs) [none]
🟪	vaf	FOIA Vista (Veterans Affairs) [FOIA]
🟦	vbr	IHS RPMS (Indian Health Service) [none]
🟪	vbf	FOIA RPMS (Indian Health Service) [FOIA]
🟧	vcd	CHCS (Department of Defense/SAIC) [???]
🟧	vho	Hui Open Vista (Pacific Hui) [???]
🟧	vmu	MUSTI (Finnish Universities) [???]
🟩	vmi	MINPHIS (Nigeria) [private]
🟦	vos	Osehra Vista (Open Source EHR Alliance) [Apache 2.0]
🟦	vso	Medsphere Open Vista (Medsphere) [???]
🟪	vst	GFT ("Medsphere") Fileman (gft) [Apache 2.0]
🟦	vvp	VAPALS (Vista Expertise Network) [Apache 2.0]
🟩	vvn	Angel (Vista Expertise Network) [VCL 1.0]
🟥	vvf	FLAP (Vista Expertise Network) [Apache 2.0]
🟥	vwo	Open Vista (WorldVistA) [???]
🟥	vwh	Vista Office EHR (Health & Human Services/WorldVistA) [???]
🟦	vwe	World Vista EHR (WorldVistA) [AGPL 3.0]
🟩	vxd	vx Vista (Document Storage Systems) [private]
🟦	vxo	Open vx Vista (Document Storage Systems) [Apache 2.0]
⬜	vzz	vista dialect undeclared or unknown
🟦	mdc	Mumps Development Committee (Vista Expertise Network) [Apache 2.0?]
🟪	mgy	YottaDB (YottaDB) [AGPL 3.0]
⬜	zzz	not vista-related

Developer/Development Shop: Development organization that manages an entire Vista dialect; used to create dialect names

🟦	va	Veterans Affairs
🟦	vb	Indian Health Service
🟦	vo	OSEHRA
🟦	vs	Medsphere
🟦	vv	Vista Expertise Network
🟦	vw	WorldVistA
🟦	vx	Document Storage Systems

Mash Naming 2: Application, Adopter, Site, & System

(draft, 2019-06-05)

VNNS: The Vista Name- and Number-space System (VNNS), managed by OSEHRA's Database Administration Committee (DBAC), assigns namespaces & numberspaces to each Vista application, as well as to adopters, sites, vendors, and individual hardhats, which can be used to help develop unique names for UCIs, volumes, and systems. For example, SAMI is the namespace for the VAPALS-ELCAP application.

Adopter: runs one or more Vista production sites, each with one or more Vista production systems

A*	Veterans Affairs
B*	Indian Health Service
G*/I*	Department of Defense
J*	Nonfederal adopter
K*	Individual Vista hardhat
U*	International adopter
ZZ*	Unknown adopter

Site: Vista production organization or location, with one or more Vista production systems

APG	VA Phoenix
BZX	IHS Phoenix
???	DOD Hospital X
JVV	Vista Expertise Network
KBAF	Frederick D. S. Marshall (toad)
UJO	Kingdom of Jordan
ZZZ	Unknown site

System: Vista production computer or cluster, running one or more Vista environments

APGSAMI	VA Phoenix VAPALS-ELCAP Server
BZXSAMI	IHS Phoenix VAPALS-ELCAP Server
??????	DOD CHCS Server X
JVVSAMI	Avicenna (VAPALS-ELCAP Dev Server)
KBAFHAN	Hanbalur (toad Workstation)
UJOMOH	Jordan Ministry of Health Server
ZZZZZZ	Unknown system

$System: Mumps intrinsic special variable uniquely distinguishing this Mumps system from all others

47,avicenna,jvvsami,vista,1 — $system for Avicenna (VAPALS-ELCAP Dev Server)

System Name: unusual or unique, assigned by site, user friendly — avicenna

vista — **Mumps DBA Code**: uniquely identifying database administration authority/community, registered with MDC

Mumps Implementor #: uniquely assigned by MDC — 47

47,avicenna,jvvsami,vista,1

1 — **System Version #**: increment when upgrade would affect system performance

43 = Micronetics	47 = Greystone
44 = MGlobal	48 = ESI Technology
45 = PFCS Corporation	50 = Mumps v1
46 = Intersystems Corporation	

jvvsami — **System Code**: user-friendly code uniquely identifying system within DBA community

Mash Naming 3: UCI & Volume

(draft, 2019-06-05)

UCI Names: Short UCI names will follow the old conventions. Long UCI names will be uci,vol and serve as environment names.

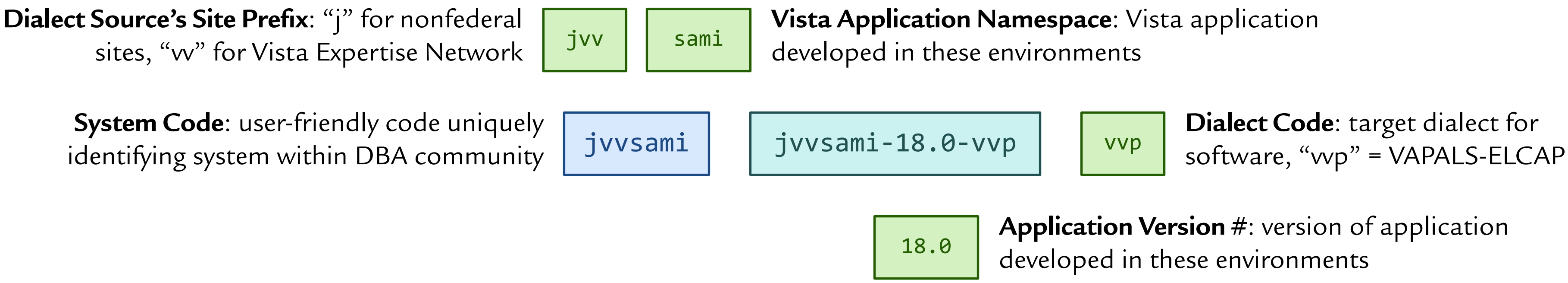
Volume Names: Using dialect names, adopter names, application namespaces, & version #s, we can construct unique volume names.

UCI: old word for an environment, short name of environment, pick from or extend list, describes lifecycle role

`liv`	live/production environment
`tst`	test environment
`dev`	development environment
`fld`	field-test environment
`rel`	release environment
`dem`	demo environment
`ver`	verification environment

Volume: collection of UCIs/environments, aka "volume set"

`apg`	VA Phoenix Vista environments
`apgsami`	VA Phoenix VAPALS-ELCAP environments
`jvva`	VAPALS-ELCAP VA field environments
`jvvref`	VAPALS-ELCAP Vista reference environments
`jvvhop`	VAPALS-ELCAP Vista Forum environments (Hopper system)
`jvvsami-18.0-vvp`	VAPALS-ELCAP v18.0 development environments
`kbafmdc`	Frederick D. S. Marshall Mash development environments
`jvvmdc`	VAPALS-ELCAP Mash development environments
`jvvdi-22.2-vst`	VEN GFT Fileman v22.2 environments
`jvvc9c`	VEN Central Regional Hospital field environments



Dialect Source's Site Prefix: "j" for nonfederal sites, "vv" for Vista Expertise Network

Vista Application Namespace: Vista application developed in these environments

System Code: user-friendly code uniquely identifying system within DBA community

Dialect Code: target dialect for software, "vvp" = VAPALS-ELCAP

Application Version #: version of application developed in these environments

Mash Naming 4: Environment, Home Directory, & Home Path

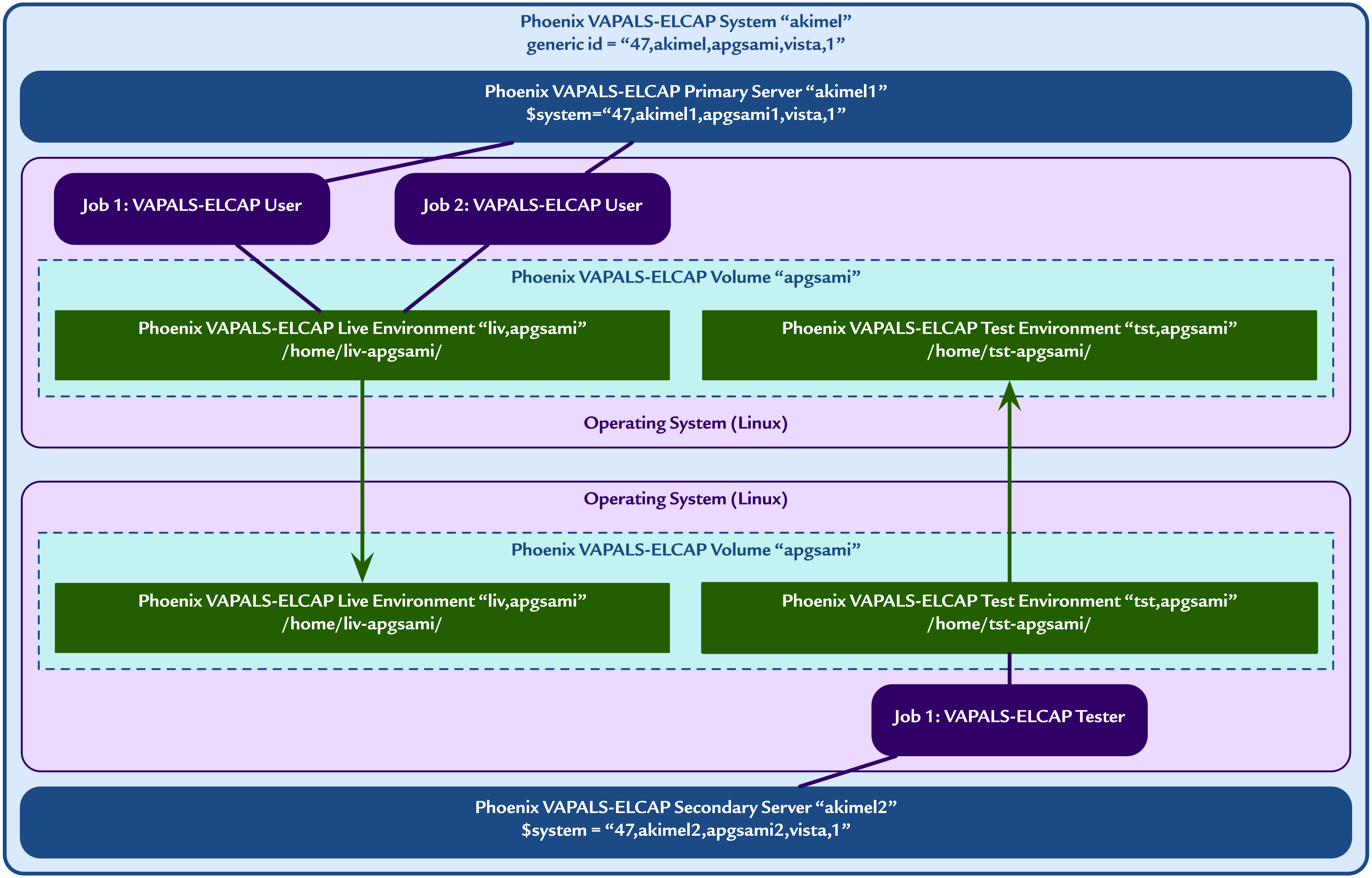
(draft, 2019-06-05)

Under the Mash Virtual Machine model, naming the environment includes standardizing how home directories and paths are named.

Environment Name	Environment Description	Home Directory	Home Path
`liv,apg`	VA Phoenix Vista live environment	`liv-apg`	Would be: `/home/liv-apg/`
`tst,apg`	VA Phoenix Vista test environment	`tst-apg`	
`dev,apg`	VA Phoenix Vista development environment	`dev-apg`	
`liv,apgsami`	VA Phoenix VAPALS-ELCAP live environment	`liv-apgsami`	Now: `/home/osehra/`
`tst,apgsami`	VA Phoenix VAPALS-ELCAP test environment	`tst-apgsami`	Soon: `/home/liv-apgsami/`
`vav,jvva`	VAPALS-ELCAP VA Vista field environment	`vav-jvva`	
`liv,jvva`	VAPALS-ELCAP VA Vista field live environment	`liv-jvva`	Soon: `/home/liv-jvva/`
`tst,jvva`	VAPALS-ELCAP VA Vista field test environment	`tst-jvva`	
`rel,jvva`	VAPALS-ELCAP VA release environment	`rel-jvva`	
`dev,jvvsami-18.0-vvp`	VAPALS-ELCAP v18.0 development environment	`dev-jvvsami-18.0-vvp`	Now: `/home/osehra/`
`tst,jvvsami-18.0-vvp`	VAPALS-ELCAP v18.0 test environment	`tst-jvvsami-18.0-vvp`	Soon: `/home/dev-jvvsami-18.0-vvp/`
`fld,jvvsami-18.0-vvp`	VAPALS-ELCAP v18.0 field environment	`fld-jvvsami-18.0-vvp`	
`dev,jvvsami-19.0-vvp`	VAPALS-ELCAP v19.0 development environment	`dev-jvvsami-19.0-vvp`	
`dev,kbafvvp`	Toad’s VAPALS-ELCAP development environment	`dev-kbafvvp`	Now: `/home/osehra/`
`dev,kbafmdc`	Toad’s Mash development environment	`dev-kbafmdc`	Soon: `/home/kbafvvp/`
`dev,kbafvvn`	Toad’s Angel Taskman development environment	`dev-kbafvvn`	

Mash VM 2: Sample 1, Phoenix VAPALS-ELCAP Server

(draft, 2019-06-05)



Phoenix VAPALS-ELCAP System “akimel”
generic id = “47,akimel,apgsami,vista,1”

Phoenix VAPALS-ELCAP Primary Server “akimel1”
$system=“47,akimel1,apgsami1,vista,1”

Job 1: VAPALS-ELCAP User

Job 2: VAPALS-ELCAP User

Phoenix VAPALS-ELCAP Volume “apgsami”

Phoenix VAPALS-ELCAP Live Environment “liv,apgsami”
/home/liv-apgsami/

Phoenix VAPALS-ELCAP Test Environment “tst,apgsami”
/home/tst-apgsami/

Operating System (Linux)

Operating System (Linux)

Phoenix VAPALS-ELCAP Volume “apgsami”

Phoenix VAPALS-ELCAP Live Environment “liv,apgsami”
/home/liv-apgsami/

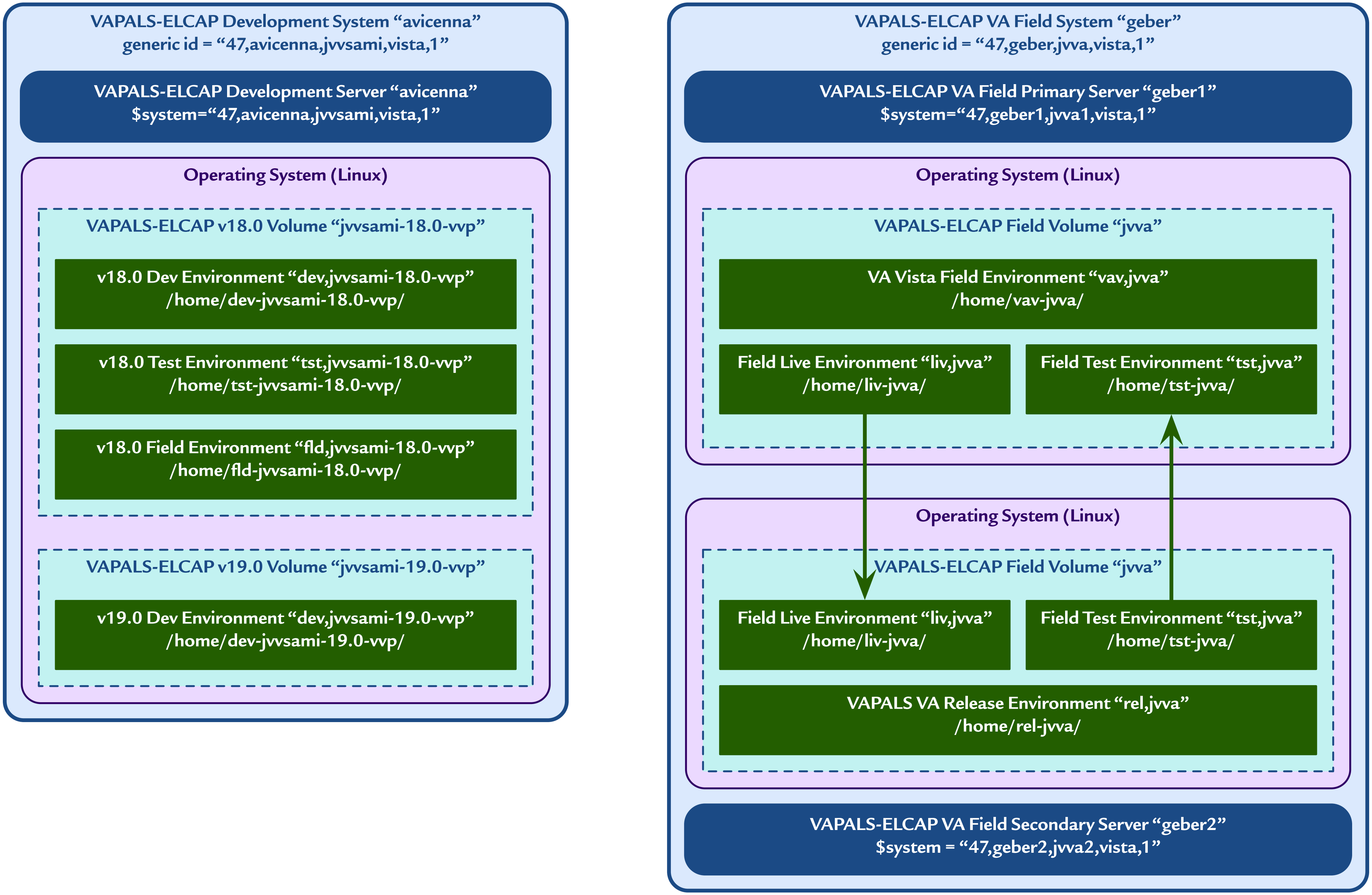
Phoenix VAPALS-ELCAP Test Environment “tst,apgsami”
/home/tst-apgsami/

Job 1: VAPALS-ELCAP Tester

Phoenix VAPALS-ELCAP Secondary Server “akimel2”
$system = “47,akimel2,apgsami2,vista,1”

Mash VM 3: Sample 2, VAPALS-ELCAP Development & Field Servers

(draft, 2019-06-05)



VAPALS-ELCAP Development System “avicenna”
generic id = “47,avicenna,jvvsami,vista,1”

VAPALS-ELCAP Development Server “avicenna”
$system=“47,avicenna,jvvsami,vista,1”

Operating System (Linux)

VAPALS-ELCAP v18.0 Volume “jvvsami-18.0-vvp”

v18.0 Dev Environment “dev,jvvsami-18.0-vvp”
/home/dev-jvvsami-18.0-vvp/

v18.0 Test Environment “tst,jvvsami-18.0-vvp”
/home/tst-jvvsami-18.0-vvp/

v18.0 Field Environment “fld,jvvsami-18.0-vvp”
/home/fld-jvvsami-18.0-vvp/

VAPALS-ELCAP v19.0 Volume “jvvsami-19.0-vvp”

v19.0 Dev Environment “dev,jvvsami-19.0-vvp”
/home/dev-jvvsami-19.0-vvp/

VAPALS-ELCAP VA Field System “geber”
generic id = “47,geber,jvva,vista,1”

VAPALS-ELCAP VA Field Primary Server “geber1”
$system=“47,geber1,jvva1,vista,1”

Operating System (Linux)

VAPALS-ELCAP Field Volume “jvva”

VA Vista Field Environment “vav,jvva”
/home/vav-jvva/

Field Live Environment “liv,jvva”
/home/liv-jvva/

Field Test Environment “tst,jvva”
/home/tst-jvva/

Operating System (Linux)

VAPALS-ELCAP Field Volume “jvva”

Field Live Environment “liv,jvva”
/home/liv-jvva/

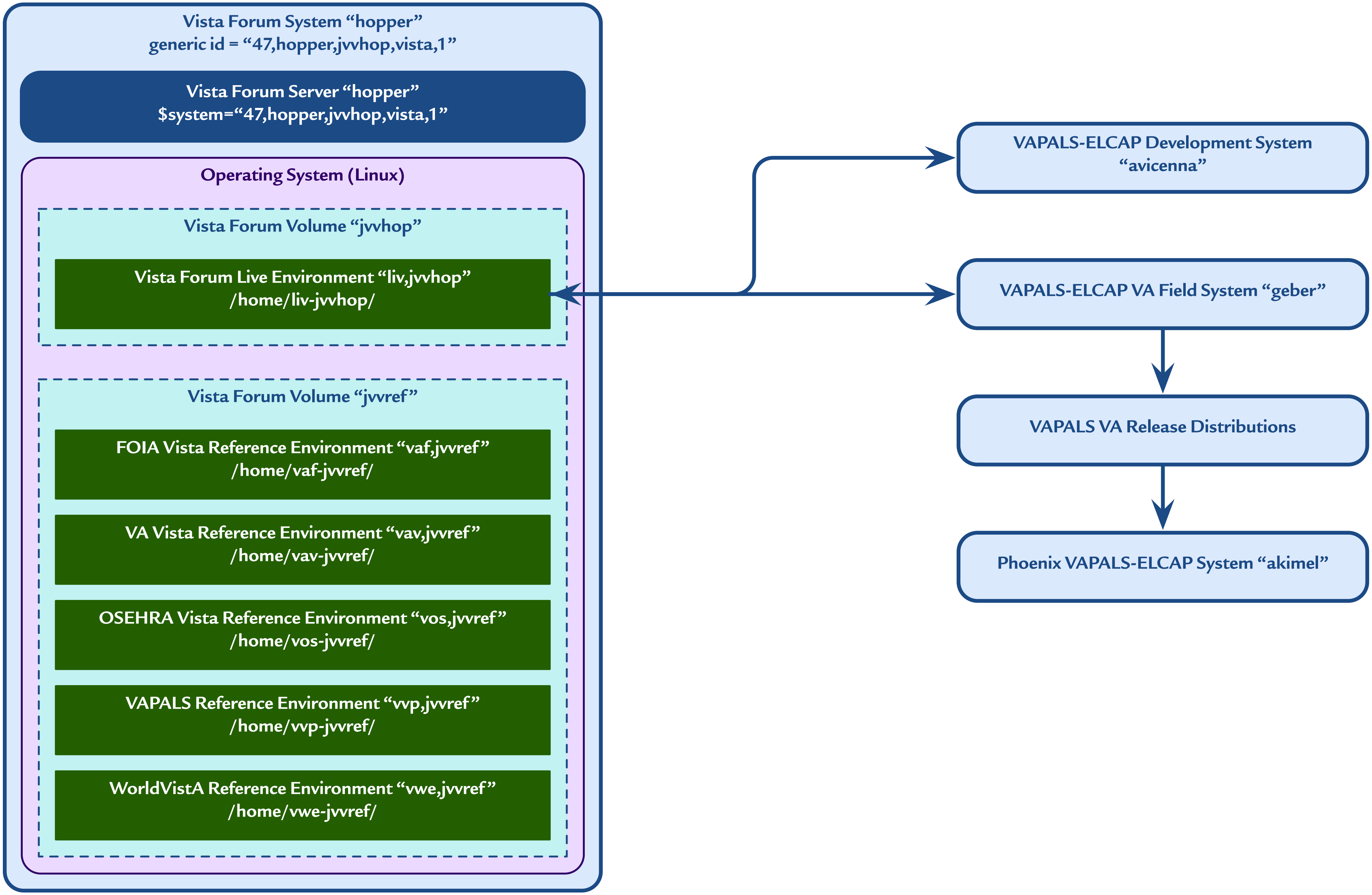
Field Test Environment “tst,jvva”
/home/tst-jvva/

VAPALS VA Release Environment “rel,jvva”
/home/rel-jvva/

VAPALS-ELCAP VA Field Secondary Server “geber2”
$system = “47,geber2,jvva2,vista,1”

Mash VM 4: Sample 3, Vista Forum Server

(draft, 2019-06-05)



Vista Forum System “hopper”
generic id = “47,hopper,jvvhop,vista,1”

Vista Forum Server “hopper”
$system=“47,hopper,jvvhop,vista,1”

Operating System (Linux)

Vista Forum Volume “jvvhop”

Vista Forum Live Environment “liv,jvvhop”
/home/liv-jvvhop/

Vista Forum Volume “jvvref”

FOIA Vista Reference Environment “vaf,jvvref”
/home/vaf-jvvref/

VA Vista Reference Environment “vav,jvvref”
/home/vav-jvvref/

OSEHRA Vista Reference Environment “vos,jvvref”
/home/vos-jvvref/

VAPALS Reference Environment “vvp,jvvref”
/home/vvp-jvvref/

WorldVistA Reference Environment “vwe,jvvref”
/home/vwe-jvvref/

VAPALS-ELCAP Development System
“avicenna”

VAPALS-ELCAP VA Field System “geber”

VAPALS VA Release Distributions

Phoenix VAPALS-ELCAP System “akimel”

Repo Naming: A Complete Lack of Standards

(draft, 2019-06-04)

Repository Name	Repository Description
`FILEMAN`	where should I look to find the application repo?
`Fileman`	
`VA FILEMAN`	
`KERNEL-GTM`	how are these related to official patches?
`Kernel GTM`	
`BROKER`	sort in 3 different places in big directory
`RPC Broker`	
`XWB`	
`M Web Server`	which of these 3 does my environment depend on?
`M-Web Server`	
`m-web-server`	
`M UNIT`	what syntax should repo names have?
`M-Unit`	
`Random Stuff`	what is even in these?
`Random Vista Utilities`	
`Important`	

Imagine a directory containing hundreds or even thousands of repository clone directories, from many dozens of sources, inside and outside the Vista community.

Some are the official software distribution directories our environment depends on. Others are local extensions or development directories. They come from countless developers from many different repo hosting sites.

How can we tell which is which? Where do we look for the software we need?

This is a teeny tiny sampling of such a directory to illustrate a few of the issues with the status quo.

Mash Naming 5: Gold, Silver, & Copper Repo & Clone Names

(draft, 2019-06-05)

Home Path to Repos

`/home/dev-jvvsami-18.0-vvp/lib/`

`gold` standardized, automated distribution of entire vista dialects for production installation

`silver` standardized, manual distribution of dialects & applications for production installation

`copper` non=standardized, manual distribution of software for any purpose

Gold Repo & Clone Names

Dialect Code: FOIA Vista (Veterans Affairs) [FOIA] `vaf`

`vaf-foia-vista`

`foia-vista` **Dialect Name:** FOIA Vista (Veterans Affairs) [FOIA]

Silver Repo Clone Names

`sami-vapals-elcap` **Remote Repo Name:** software namespace-name

Software Namespace: computable code describing repo contents, namespace `sami`

`vapals-elcap` **Software Name:** human-readable name of repo contents, lowercase, hyphens, short

Action Code: r = read-only, a = active development (r/w) `a`

`a-sami-vapals-elcap--vo-osehra-github`

`github` **Software Source System:** name of repo host system.

Software Source Code: hardhat namespace or vista dialect source code or etc., vo = osehra, kban= sam habiel, etc. `vo`

`osehra` **Software Source Username:** username of user at repo hosting system.

Copper Repo Clone Names

`m-unit` **Remote Repo Name:** because the remote repo is not named _ut-m-unit, the actual name is included

`r-_ut-m-unit--m-unit--kbbp-joelivey-github`

VAPALS Repo Clone Names

(draft, 2019-06-05)

Home Path to Repos	Repo Type	Repo Clone Directories
`/home/dev-jvvsami-18.0-vvp/lib/`	`gold/`	none at present, coming in a little over a year for vvp-vapals dialect repo
	`silver/`	`a-mdc-mash--vv-git-mumps`
		`a-sami-vapals-elcap--vo-osehra-github`
		`a-vvp-vapals--vv-git-mumps`
	`copper/`	`r-vos-dist-osehra-vista--vista--vo-osehra-github`
		`r-_web-m-web-server—kban-shabiel-github`
		`r-xugtm-kernel-gtm--kban-shabiel-github`
		`r-xv-vpe--kban-shabiel-github`

The goal in a nutshell is a schema that can:

1. manage thousands of closely related repos
2. without any confusion about what's in each one or how we use it
3. support visually scanning a massive directory full of clone names
4. with names that lend themselves to automation
5. to support software knowing which clones contain the software updates they need, without human intervention
6. while ensuring we can get to tab-completion in the fewest possible characters
7. to keep the names practical for use by sysadmins
8. while also ensuring we can unerringly locate the remote repo
9. and that we know what person or institution is responsible for the repo’s contents.

10. It must also store its contents in a standardized structure, to support automation (see next slide).

Mash Naming 6: Repository Structure

(draft, 2019-06-05)

Status Quo: Vista repos follow no detectable standard in terms of where package elements are stored nor in the syntax used for directory and file names. Processing of such free-for-all repos is therefore strictly manual, labor- and expertise-intensive. To replace this state of affairs with automation therefore requires standardizing repo structures as well as names.

Copper: no standard structure, bazaar, freedom, manual

Silver: as described below, balanced, automatable

Gold: each subdirectory named & structured like a silver repo, cathedral, automated

Platinum: like src directory below, balanced, automated

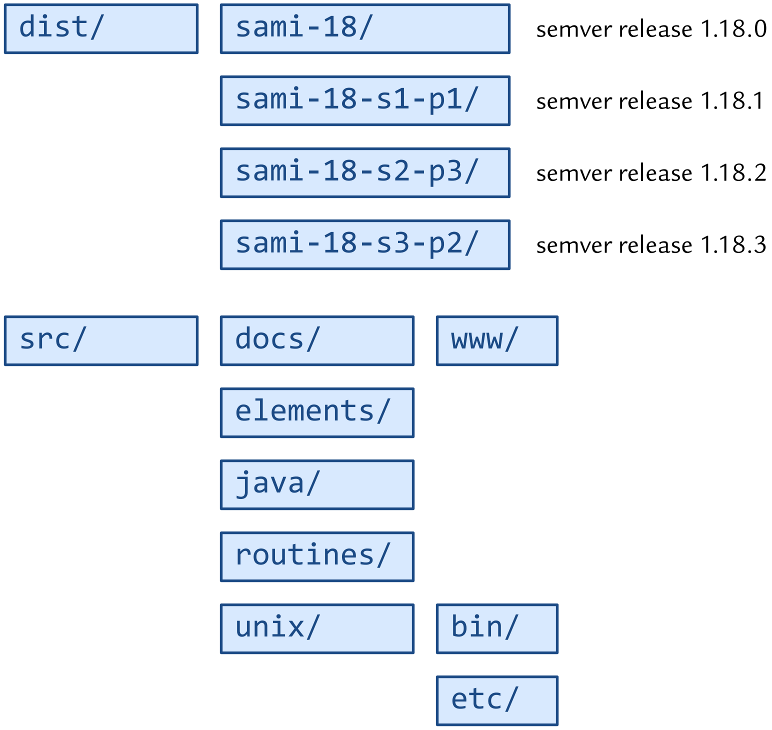
Repo Path

```
/home/<env>/lib/<category>/<repo-name>/
```

Intermediate Standard

`admin/`	unix scripts
`kids/`	kids distribution files
`docs/`	documentation, notes, diagrams
`elements/`	non-routine software elements
`globals/`	global distribution files
`java/`	java source files
`routines/`	mumps source files

Emerging Standard



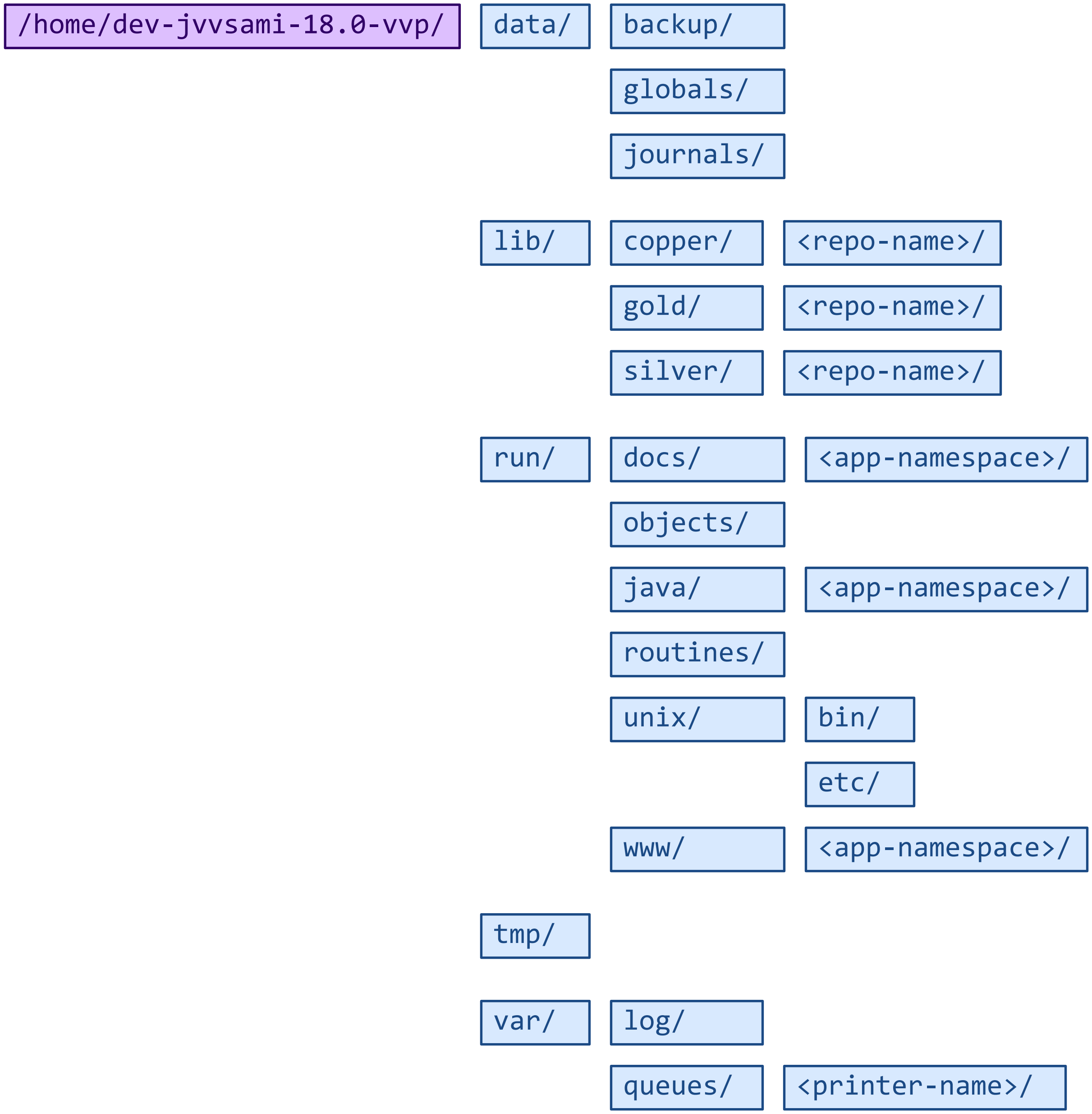
```
dist/   sami-18/        semver release 1.18.0
        sami-18-s1-p1/  semver release 1.18.1
        sami-18-s2-p3/  semver release 1.18.2
        sami-18-s3-p2/  semver release 1.18.3

src/    docs/      www/
        elements/
        java/
        routines/
        unix/      bin/
                   etc/
```

Mash Naming 7: Home Directory Structure

(draft, 2019-06-04)

Status Quo: Automating Vista distribution also requires standardizing the home directory structure of Vista environments.



- /home/dev-jvvsami-18.0-vvp/
  - data/
    - backup/
    - globals/
    - journals/
  - lib/
    - copper/
      - <repo-name>/
    - gold/
      - <repo-name>/
    - silver/
      - <repo-name>/
  - run/
    - docs/
      - <app-namespace>/
    - objects/
    - java/
      - <app-namespace>/
    - routines/
    - unix/
      - bin/
      - etc/
    - www/
      - <app-namespace>/
  - tmp/
  - var/
    - log/
    - queues/
      - <printer-name>/

data directory holds all the environment’s live running data, including the backups and journals used as the safety net for that data, though those two are often mount points for other drives

lib directory holds all incoming software; if this is a development site, then some of these repos are for outgoing software, while others are bidirectional; standardizes vista distribution, so it can be automated

run directory holds the running software, which is then pushed out to the environment’s platinum repository (excluding the object folder) to support environment-to-environment, as well as doing version control on what is and has been actually running in this environment; standardizes vista standardization, so it can be automated

tmp directory is for scratch data

var directory is for highly variable data files, such as logs, print queues, and so on

Coming Soon 1: Environment & Repo Standardization & Automation

(draft, 2019-06-05)

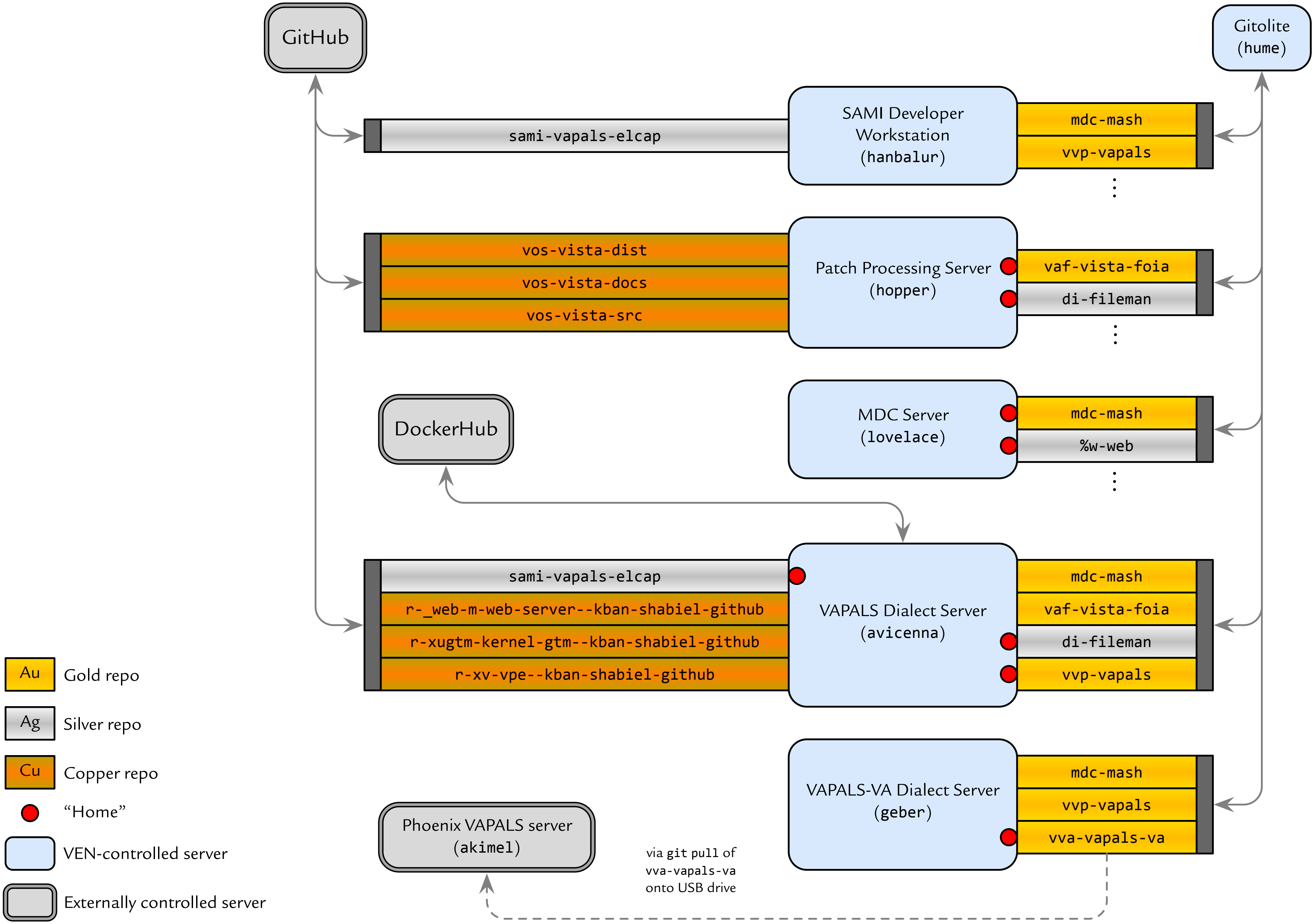
1. **VAPALS Increased Compliance**: In the past couple weeks we significantly improved the compliance of the sami-vapals-elcap repo with this emerging Mash standard. During June 2019, we will bring it and the home directory fully into compliance, before distribution to our first VA system at Phoenix.

2. **Automation for Renaming Environments**: We are partway through writing a new routine, ZTMRENAME, to run while an environment is shut down to change the value of every field in Vista that holds a UCI, volume, or system name, to bring them into compliance with the environment's proper name. We will then write a script to move the home directory, change the startup scripts, and so on to bring its unix elements into compliance with the proper system and environment names. We will then develop files and routines to ask the sysadmin a few questions and use those answers plus a database of namespaces to calculate the proper name for the UCI, volume, environment, and system. Together, these pieces of automation will make it practical for the first time to rename an environment without causing innumerable negative side effects - and without having to be an expert in Mash's naming standards. This software will be distributed in the VAPALS Dialect Repo (vvp-vapals) and MDC Mash Repo (mdc-mash) and VAPALS-VA Dialect Repo (vva-vapals-va) in time for the release of patch SAMI*18*1, before distribution to Phoenix.

3. **Automation for Upgrading Environments**: Over the course of the next year and a half, to be released incrementally, we are iteratively developing new versions of KIDS and the Patch Module to support keeping our VAPALS servers up to date with Vista patches and new application versions without requiring a Vista expert to do the patching. In support of this software, we will be creating and operating what as far as we know is the first fully automated Vista patch stream outside VA, for our VAPALS dialect, which will distribute upgrades to our SAMI software and upgrades to other Vista applications (including preserving our dialect-specific Vista modifications) for installation on our VAPALS servers. We will simultaneously create and operate an MDC-specific patch stream for distribution of upgrades to the Mumps Advanced Shell, for installation on our VAPALS servers.

Coming Soon 2: Patch Stream Architecture

(draft, 2019-06-05)



GitHub

Gitolite (hume)

SAMI Developer Workstation (hanbalur)
- sami-vapals-elcap
- mdc-mash
- vvp-vapals
- ⋮

Patch Processing Server (hopper)
- vos-vista-dist
- vos-vista-docs
- vos-vista-src
- ● vaf-vista-foia
- ● di-fileman
- ⋮

DockerHub

MDC Server (lovelace)
- ● mdc-mash
- ● %w-web
- ⋮

VAPALS Dialect Server (avicenna)
- ● sami-vapals-elcap
- r-_web-m-web-server--kban-shabiel-github
- r-xugtm-kernel-gtm--kban-shabiel-github
- r-xv-vpe--kban-shabiel-github
- mdc-mash
- vaf-vista-foia
- ● di-fileman
- ● vvp-vapals

VAPALS-VA Dialect Server (geber)
- mdc-mash
- vvp-vapals
- ● vva-vapals-va

Phoenix VAPALS server (akimel)

via git pull of vva-vapals-va onto USB drive

Au	Gold repo
Ag	Silver repo
Cu	Copper repo
●	“Home”
▢	VEN-controlled server
▢	Externally controlled server

vista™

Expertise Network

Thank You!

Vista Community Meeting 2019-06-04/05
George Mason University
Fairfax, Virginia

Frederick D. S. Marshall
Vista Expertise Network
rick.marshall@vistaexpertise.net