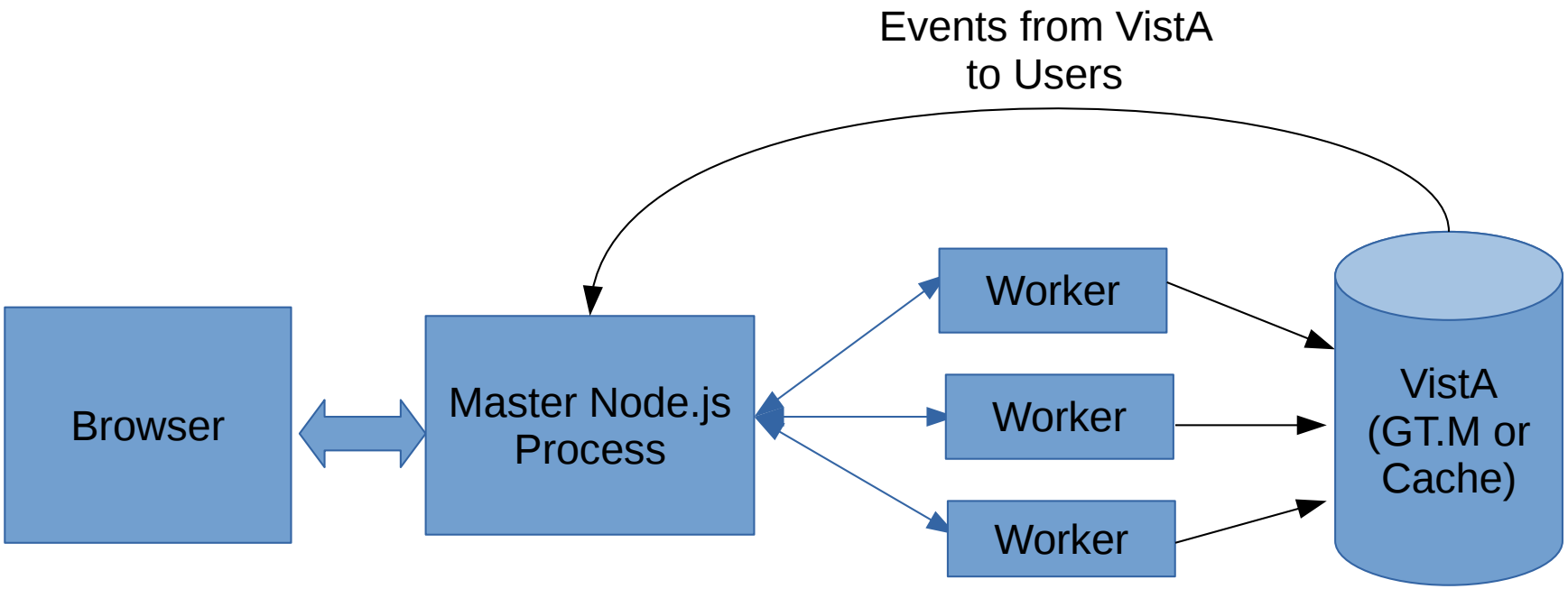
The Panorama Framework

By Sam Habiel, Pharm.D.
Technical Fellow
OSEHRA
Arlington, VA
&
Alexis Carlson
Seattle, WA

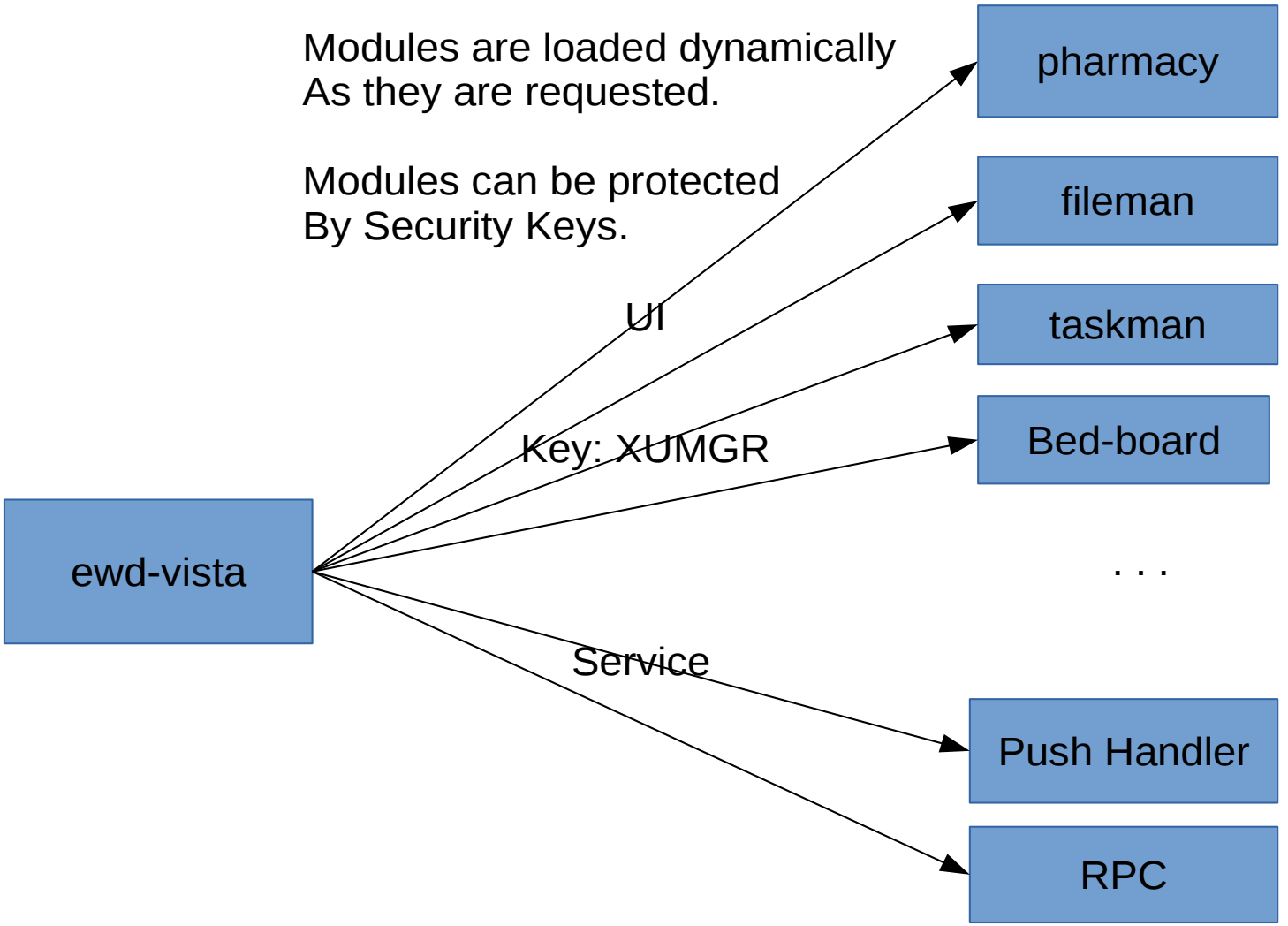
What is it?

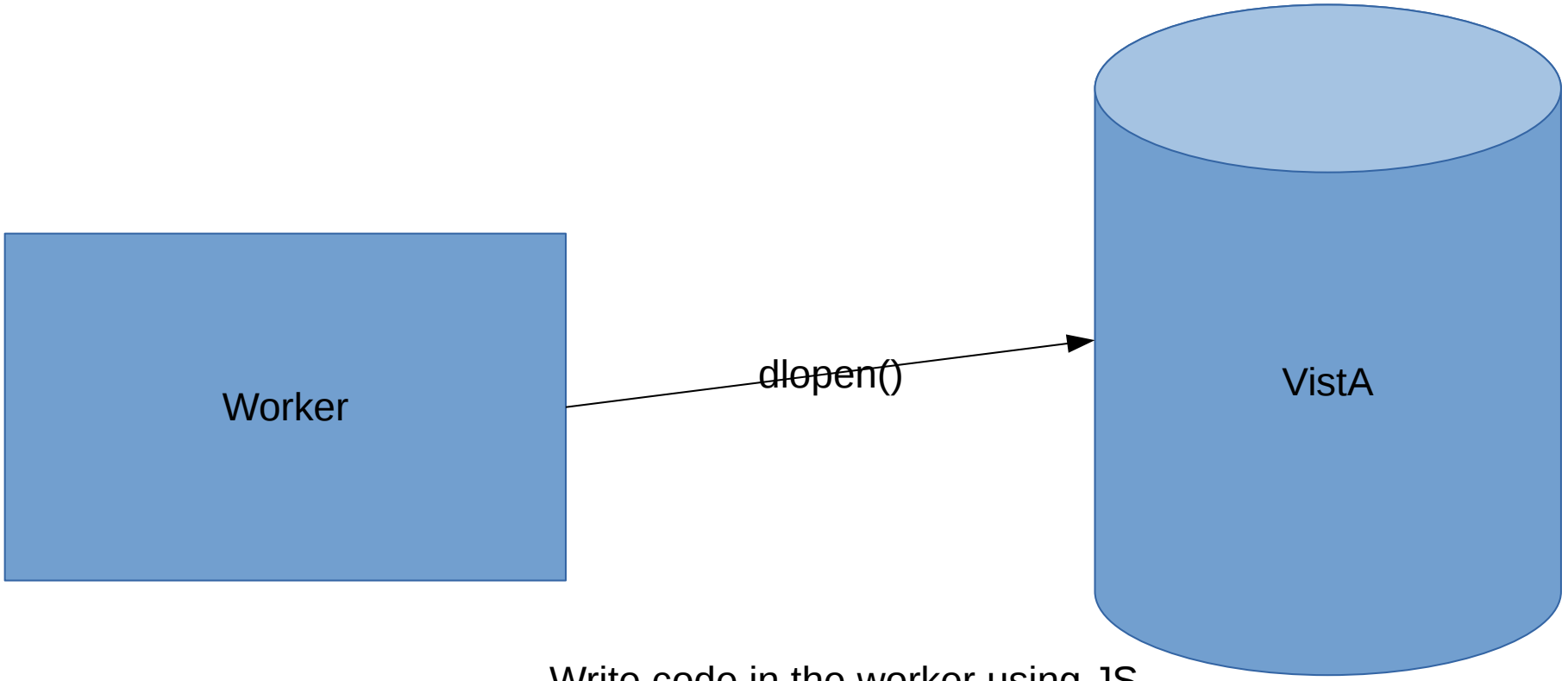
- The next generation of how VistA ought to be developed
- Expressly written to provide a final answer to how VistA should evolve
- Use and Development mirror classic VistA/DHCP in many ways
- Javascript is the primary development language, replacing M for all simple operations.
- Relies on Rob Tweed's QEWD

Architecture: Overall



Architecture: Modularity





Write code in the worker using JS
as if you are writing M

```
let name = this.db.function({function: 'GET1^DIQ', arguments: ['2', dfn, '.01']}).result;
let dob  = this.db.function({function: 'GET1^DIQ', arguments: ['2', dfn, 'DOB']}).result;
let routeName = this.db.function({function: 'GET1^DIQ', arguments: ['52.41', ien, 'PICKUP ROUTING']}).result;
let priorityName = this.db.function({function: 'GET1^DIQ', arguments: ['52.41', ien, 'PRIORITY']}).result;
let flagged = this.db.function({function: 'GET1^DIQ', arguments: ['52.41', ien, 'FLAG', 'I']}).result;

// I +$P(OR0,"^",9) S PDEA=$P($G(^PSDRUG($P(OR0,"^",9),0)),"^",3),PDEA=$S(PDEA[2:1,PDEA[3!(PDEA[4)!(PDEA[5):2,1:0)
// E  S PDEA=$$OIDEA^PSSUTLA1($P(OR0,"^",8),"O")
let drugIEN    = this.db.function({function: 'GET1^DIQ', arguments: ['52.41', ien, 'DRUG', 'I']}).result;
let oiIEN      = this.db.function({function: 'GET1^DIQ', arguments: ['52.41', ien, 'PHARMACY ORDERABLE ITEM', 'I']}).result;
let isC2,isC345 = false;
if (drugIEN) {
  let pdea = this.db.function({function: 'GET1^DIQ', arguments: ['50', drugIEN, 'DEA, SPECIAL HDLG', 'I']}).result;
  if (['2'].some(n => pdea.includes(n))) isC2 = true;
  if (['3','4','5'].some(n => pdea.includes(n))) isC345 = true;
}
else {
  let isCS = this.db.function({function: 'OIDEA^PSSUTLA1', arguments: [oiIEN, 'O']}).result;
  if (isCS === 1) isC2 = true;
  if (isCS === 2) isC345 = true;
}
// Count for each patient: by priority; by flag; by route; by CS (2 and 345)
if (patients[dfn] === undefined) {
  patients[dfn] = {};
  patients[dfn].dob = dob;
  patients[dfn].dfn = dfn;
  patients[dfn].bid = bid;
```

Write VistA Code in M or JS

- QEWD provides session and symbol table management
- Lightweight workers
  - On R Pi 3, 5,800 messages/second w/ 3 workers.
- Call VistA functions or procedures directly from Javascript using db.function or db.procedure*.
- Get/Set symbol table variables using db.symbolTable.get or db.symbolTable.set

Write VistA Code in M or JS

- Loop through globals using forEach
  - Javascript syntax is much nicer for looping than the FOR command in M.
- Check existence of nodes using node.exists (returns 0, 1, 10, 11 just like M)

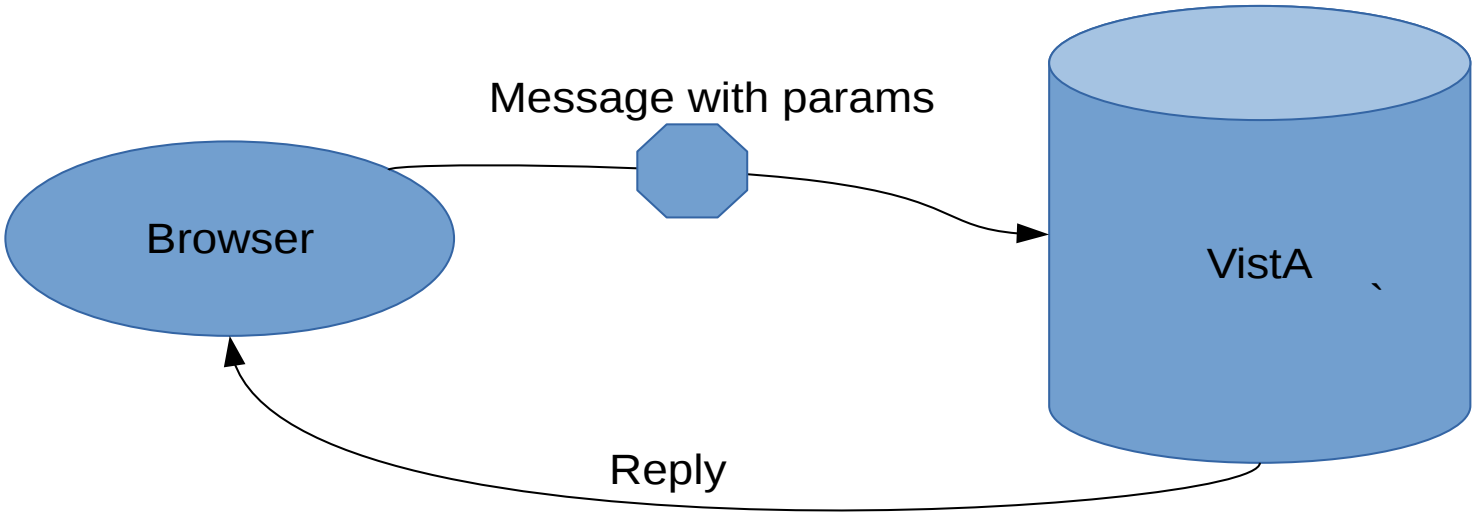
Security Philosophy

- Based on the classic VistA Menu Tree and Key security
- RPC invocations use RPC security
- Access to Data via Fileman uses either Fileman Access Codes or Kernel Part III security, depending on which one is turned on

Security: Log-in implementation

- Classic AC/VC login
- Lock-out implemented
- Change Verify Code implemented
- IP address and client name (IO(“IP”) and IO(“CLNM”)) recorded in the VistA symbol table to assist with lock-out implementation

Architecture: Security Model 1



Is the User Authenticated?

Does User have the correct Key?

Application Level Security:
(e.g. RPC – do you have access to this RPC;
Pharmacy: Display this column only if you are a technician)

Architecture: Security Model 2

- No amount of Architecture can help us avoid the CAPRI security disaster
- You still must write your code to prevent
  - SQL Injections, M style (DDR LIST RPC)
  - Overly broad RPCs with no security checks (DDR RPCs)
- Fileman Widget written to prevent M injection by end users.

Widget Provisions

- Fileman search and select widget
  - Implements LIST^DIC, then FIND^DIC
  - Post selection calls ^DIC to select a record, and if the post selection logic in Fileman fails, deselects the record
- Validation Widget
  - Makes sure that what the user typed is valid for the field

Comparison Tables

How does this Stack up?

Panorama vs Delphi Clients

Panorama	Delphi Clients
Can use RPCs but can also access any part of VistA	Can only use RPCs
Front end and Back end are both JS. Complex stuff will still need to be done in M.	Front End is Delphi. Back End is M.
Security Model is the classic incremental security model based on keys and menu trees	Monolithic security model: If you give access to the application, all its functionality is available.
Extensible	Not extensible (except for VueCentric)
Can receive events from VistA without polling	Must poll for events.

Panorama vs eHMP

Panorama	eHMP
Node.js	Node.js
Fast	Slow
Ugly	Pretty
RPCs supported	RPCs are the only way to access VistA
VistA is the center of your EMR	EHMP is the center of your EMR
Purpose is to provide a new platform for ALL of VistA	Purpose is a replacement of CPRS.
New functionality can be done in JS	New functionality requires a JS developer as well as an M developer
Components are assumed to be in different repositories and are auto-loaded	Components are in a big repository and must be registered
One middleware layer (Web Server)	Up to 3 middleware layers (RDK, vx-sync, and JDS)
Modules are lazy loaded	All modules are loaded all at once—scalability issues
Text still readable	Text too small!

Panorama vs eHMP cont.

Panorama	eHMP
Beyond Bootstrap and Jquery and QEWD, can use JS framework you want. You can use Angular for one module and Ember for another.	Must use Marionette for presentation.
Can respond to events from VistA	Only way to respond to events from VistA is to poll the 3 middleware layers
In its Infancy	Grown up
Will grow up slowly hopefully...	

Panorama: Moving Forward

- This is an ambitious project.
- This is not done yet.
- I think it will be 2 years before it's ready to be used for all sorts of applications
  - Need far more widgets to represent all the different types of inputs Fileman accepts
  - Need auto-save logic
  - Need a layer of Screenman and Listman emulation
  - Need an interop layer to allow users to drop into roll-and-scroll menus if functionality doesn't exist in Panorama.

Demo