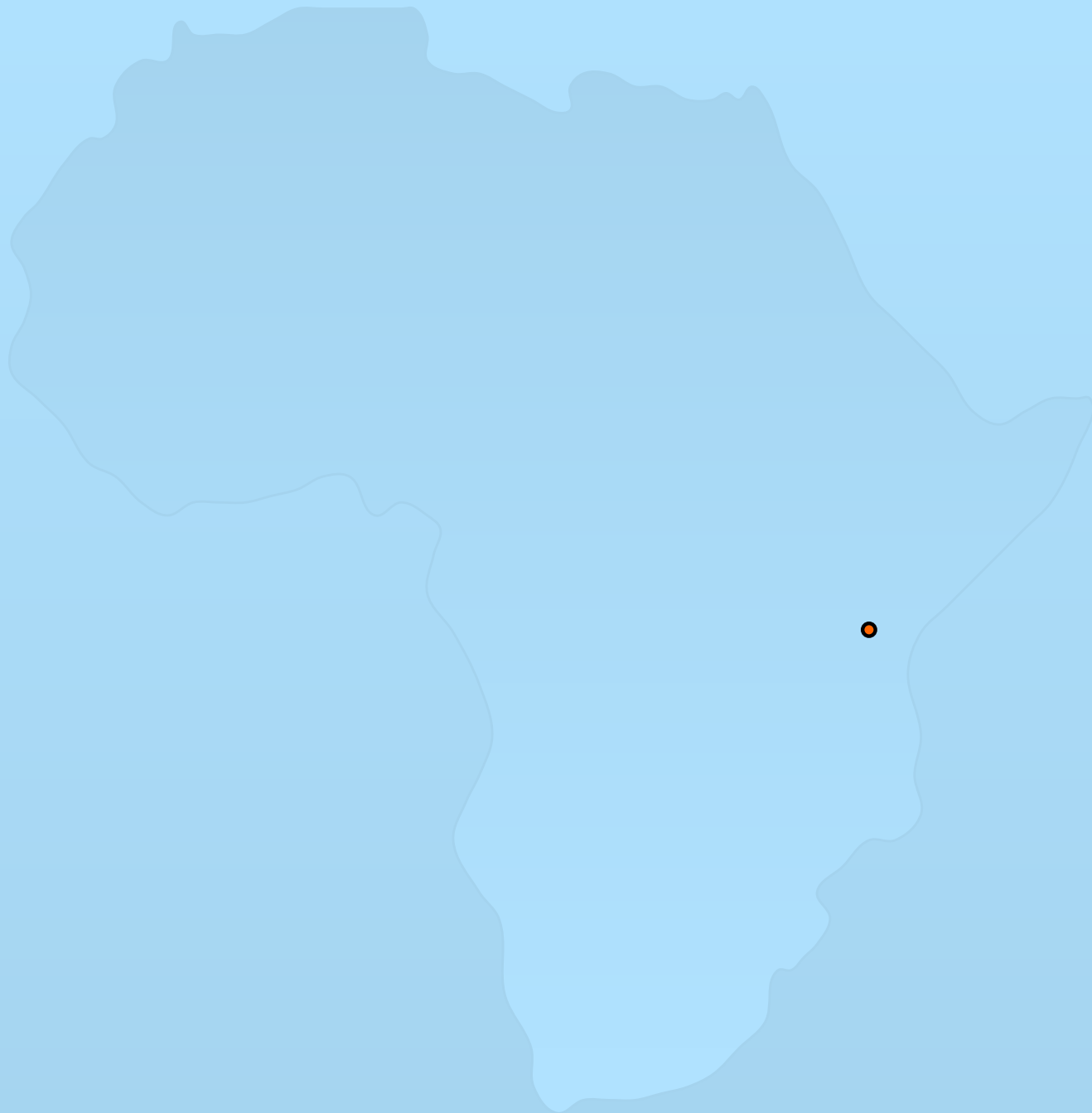


**System configuration as  
institutional memory**  
*an Arusha Project perspective*

Will Partain



# Arusha Project distinctives

- Semi-formal notation that is **universal**  
“Send Christmas card to vendor only if 90% of support calls were resolved within 24 hours.”

# Arusha Project distinctives

- Semi-formal notation that is **universal**  
“Send Christmas card to vendor only if 90% of support calls were resolved within 24 hours.”
- Massive **collaboration**, possibly global  
“Same Apache configuration as Paul’s but without the SSL stuff”

# Institutional learning

**Learning + Refresh  $\Rightarrow$  Memory**



# Coordinated distributed learning

McBreen: “The process of developing software involves taking both explicit and tacit knowledge and *embodying it in software*. Howard Baetjer sees the key challenge of software development as *coordinating distributed learning* and the key limitation as ‘our sheer ability to understand what it is we are trying to do.’” [emphases mine]

# Pair programming

- two heads better than one
- one head stays sharper
- fuzzy head helped by clear one
- new head learns from old head
- (old head learns from upstart head)
- $\Rightarrow$  collective code ownership

# Tests and code, not documentation

“Keep it clean and green”

- Write the unit tests first
- Tests and code are (can be) self-checking
- Small is beautiful
- Refactoring: yes, do the ‘useless’ tidying up
- Bad code smells fog the memory

# Nouvelle software: lessons to learn

- Remember as **little** as possible

# Nouvelle software: lessons to learn

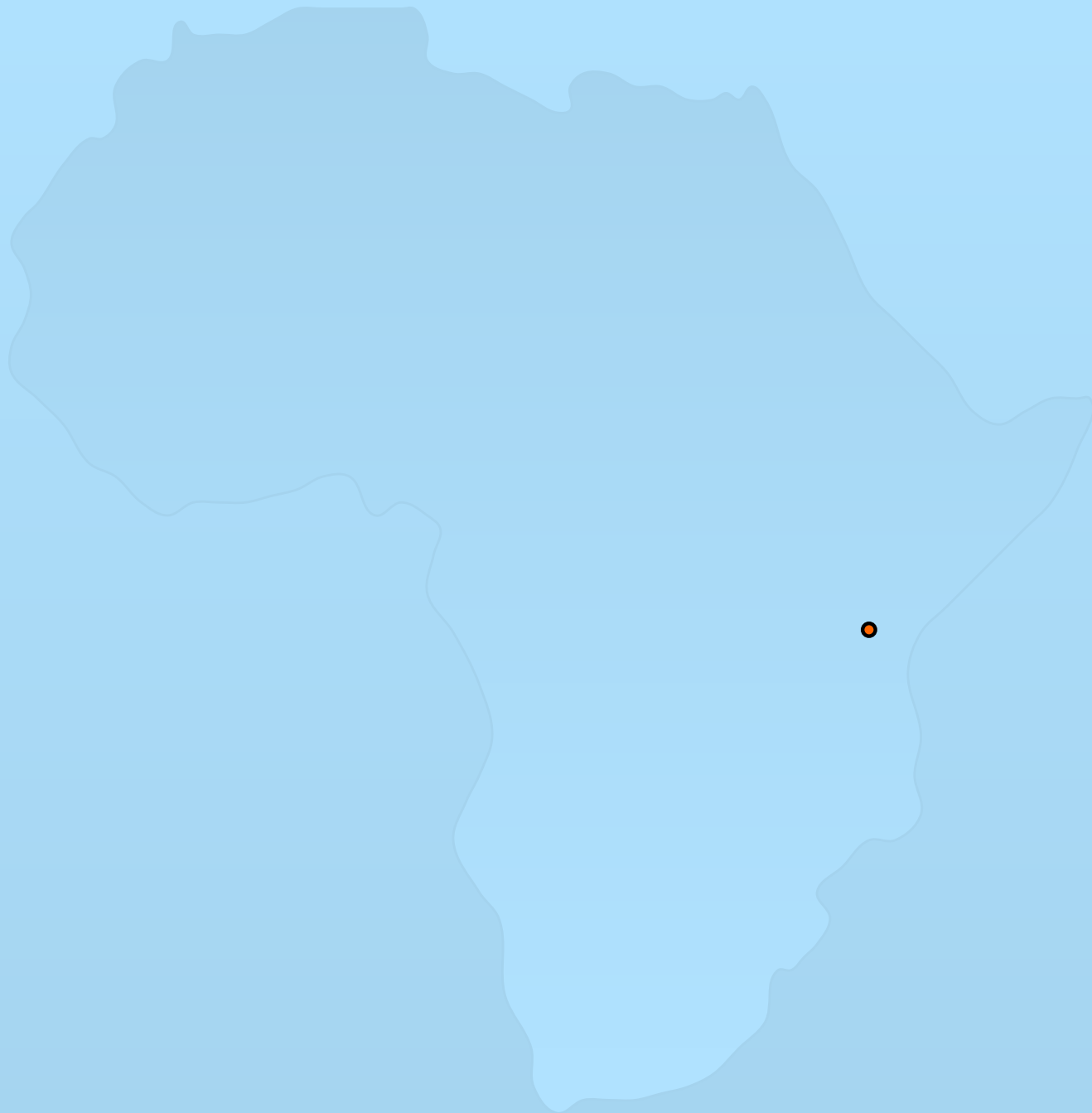
- Remember as **little** as possible
- Say it with **code**

# Nouvelle software: lessons to learn

- Remember as **little** as possible
- Say it with **code**
- Let **people** hold the 'memory'

# Nouvelle software: lessons to learn

- Remember as **little** as possible
- Say it with **code**
- Let **people** hold the 'memory'
- Keep the memory refreshed with **sharing and interaction**



```
<package name="perl--5.6.1">
```

```
<descr>
```

```
Our current blessed version of Perl.
```

```
</descr>
```

```
<install><doc>
```

```
Get the source and build the GNU way.
```

```
</doc></install>
```

```
</package>
```

```
<package name="perl--5.6.1">
```

```
<install><code>
```

```
cd /system/ark-builds/perl--5.6.1
```

```
./configure && make all install
```

```
</code></install>
```

```
</package>
```

```
<host name="sun3a">  
  
<ip-addresses><table>  
  <entry name="hme0">  
    130.209.242.55  
  </entry>  
  <entry name="hme1">  
    130.209.242.56  
  </entry>  
</table></ip-addresses>  
  
</host>
```

```
<host name="baobab">
<blinken-lights>Yes!</blinken-lights>

<list-net-interfaces>
  <code lang="python"><![CDATA[
ips  = thing.ip_addresses()

if thing.blinken_lights() != 'Yes!':
    raise 'No lights, no joy'

for if in ips.keys():
    print '%s: %s' % (if, ips[if])
]]></code></list-net-interfaces>
</host>
```

```
<support-contract name="hp" >
<expires>2002.03.31</expires>
<terms>next day</terms>
<email>none</email>
<phone><list>
  <item> +1 800 555 4321</item>
  <item>+44 800 600 700</item>
</list></phone>

</support-contract>
```

```
<website name="ok-jazz">
<customer>1493</customer>

<list-unpaid-bills>
  <code lang="perl"><![CDATA[
my $OK = 1;
cd /system/website/1493/billing
while (<*.txt>) {
  print if /ARREARS/;
}
]]></code></list-unpaid-bills>
</website>
```

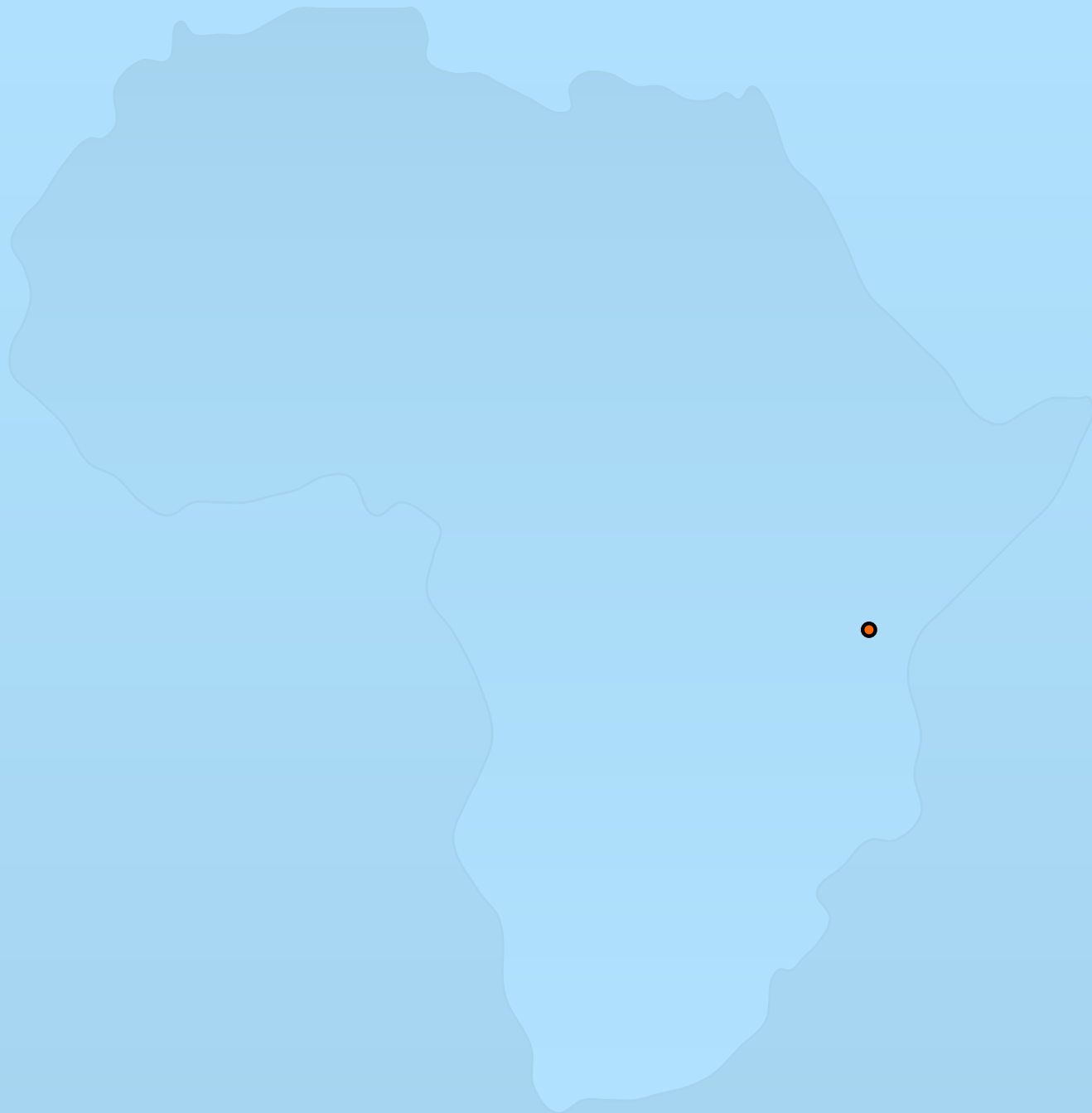
```
<module name="shiftreg14">  
  
<ip-from>motorola.com</ip-from>  
<sr-width>14</sr-width>  
  
<source><table>  
  <entry name="file">  
    /usr/share/sr.v  
  </entry>  
  <entry name="preprocess-with">  
    m4  
  </entry>  
</table></source>  
</module>
```

```
<testbench name="gate-full">
<type>gate-level</type>

<pass-patts><list>
  <item>all tests passed</item>
  <item>very, very happy</item>
</list></pass-patts>

<fail-patts><list>
  <item>incorrect</item>
  <item>segmentation fault</item>
</list></fail-patts>

</testbench>
```



```
<package name="perl--5.6.1">
```

```
<descr>
```

```
Our current blessed version of Perl.
```

```
</descr>
```

```
<install><doc>
```

```
Get the source and build the GNU way.
```

```
</doc></install>
```

```
</package>
```

```
<package name="gnu"    prototype="yes" >
<descr>
Common methods for a GNU-style package.
</descr>

<install>
  <param name="build_dir" default="no" />
  <code>
cd $build_dir
./configure && make all install
  </code>
</install>

</package>
```

```
<package name="perl--5.6.1">
  <prototypes>
    <prototype team="ARK" name="gnu" />
  </prototypes>
  <descr>
    Our current blessed version of Perl.
  </descr>
  <install>
    <param name="build_dir">
      /export/builds/perl
    </param>
  </install>
</package>
```

```
<host name="sun3a" >

<ip-addresses><table>
  <entry name="hme0" >
    130.209.242.55
  </entry>
  <entry name="hme1" >
    130.209.242.56
  </entry>
</table></ip-addresses>

</host>
```

```
<host name="baobab">
<blinken-lights>Yes!</blinken-lights>

<list-net-interfaces>
  <code lang="python"><![CDATA[
ips  = thing.ip_addresses()

if thing.blinken_lights() != 'Yes!':
    raise 'No lights, no joy'

for if in ips.keys():
    print '%s: %s' % (if, ips[if])
]]></code></list-net-interfaces>
</host>
```

```
<host name="sun" prototype="yes">
<prototypes>
  <prototype team="ARK" name="ALL" />
</prototypes>
<MAKE>/usr/ccs/bin/make</MAKE>

<reboot>
  <param name="when">now</param>

  <code privileges="root">
    /usr/ucb/shutdown -r $when
  </code></reboot>
</host>
```

```
<host name="ALL" prototype="yes">
<MAKE>/usr/bin/make</MAKE>

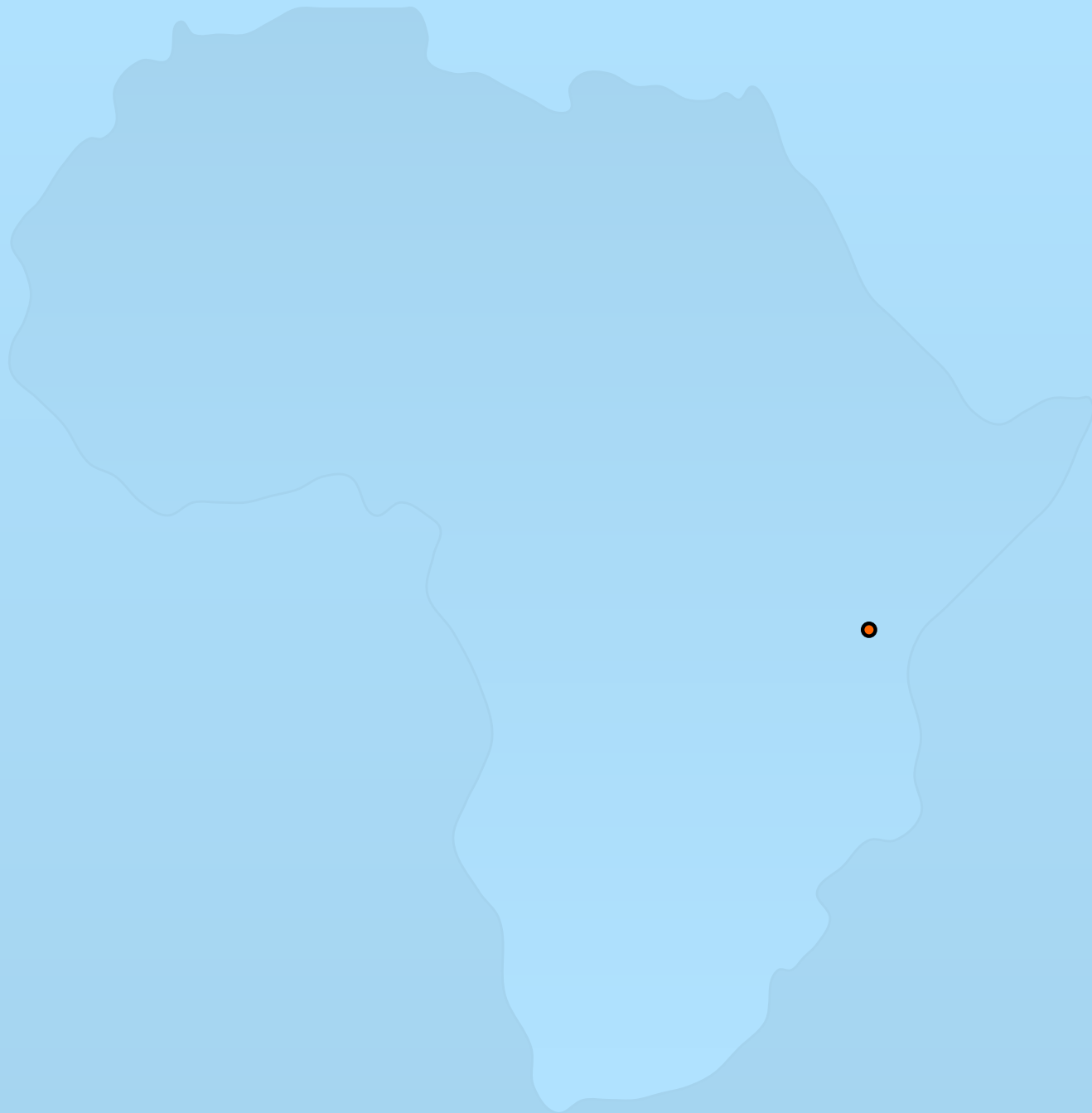
<list-net-interfaces>
  <code lang="python"><![CDATA[
ips  = thing.ip_addresses()
try:
    if thing.blinken_lights() != 'Yes!':
        raise 'No lights, no joy'
except ark.error.ArkUnknown: pass

for if in ips.keys():
    print '%s: %s' % (if, ips[if])
]]></code></list-net-interfaces>
</host>
```

```
<host name="sun3a">
  <prototypes>
    <prototype team="ARK" name="sun" />
    <prototype team="ARK" name="ALL" />
  </prototypes>

  <ip-addresses><table>
    ... as before ...
  </table></ip-addresses>
</host>
```

```
<host name="baobab">  
  <prototypes>  
    <prototype team="ARK" name="sun" />  
    <prototype team="ARK" name="ALL" />  
  </prototypes>  
  
  <blinken-lights>Yes!</blinken-lights>  
  
</host>
```



# Arusha Project distinctive #1

- Semi-formal notation that is **universal**  
“Send Christmas card to vendor only if 90% of support calls were resolved within 24 hours.”

```
<vendor name="acme">
<send-christmas-card><code lang="python">
import ark.calls
mgr = ark.calls.ArkSupportCallsMgr()
total, quick = 0, 0
for sc in mgr.getAll():
    total = total + 1
    if sc.fixTime() <= 24:
        quick = quick + 1

if total > 2 and quick/total > 0.9:
    print "yes, send 'em a card"
</code></send-christmas-card>
</vendor>
```

# Arusha Project distinctive #2

- Massive **collaboration**, possibly global  
“Same Apache configuration as Paul’s but without the SSL stuff”

```
<package name="apache">
<prototypes>
  <prototype team="paul" name="apache" />
</prototypes>

<configure>
  <param name="args">
    --without-ssl
  </param>
</configure>
</package>
```

# System configuration: lessons to learn?

- Remember as **little** as possible

# System configuration: lessons to learn?

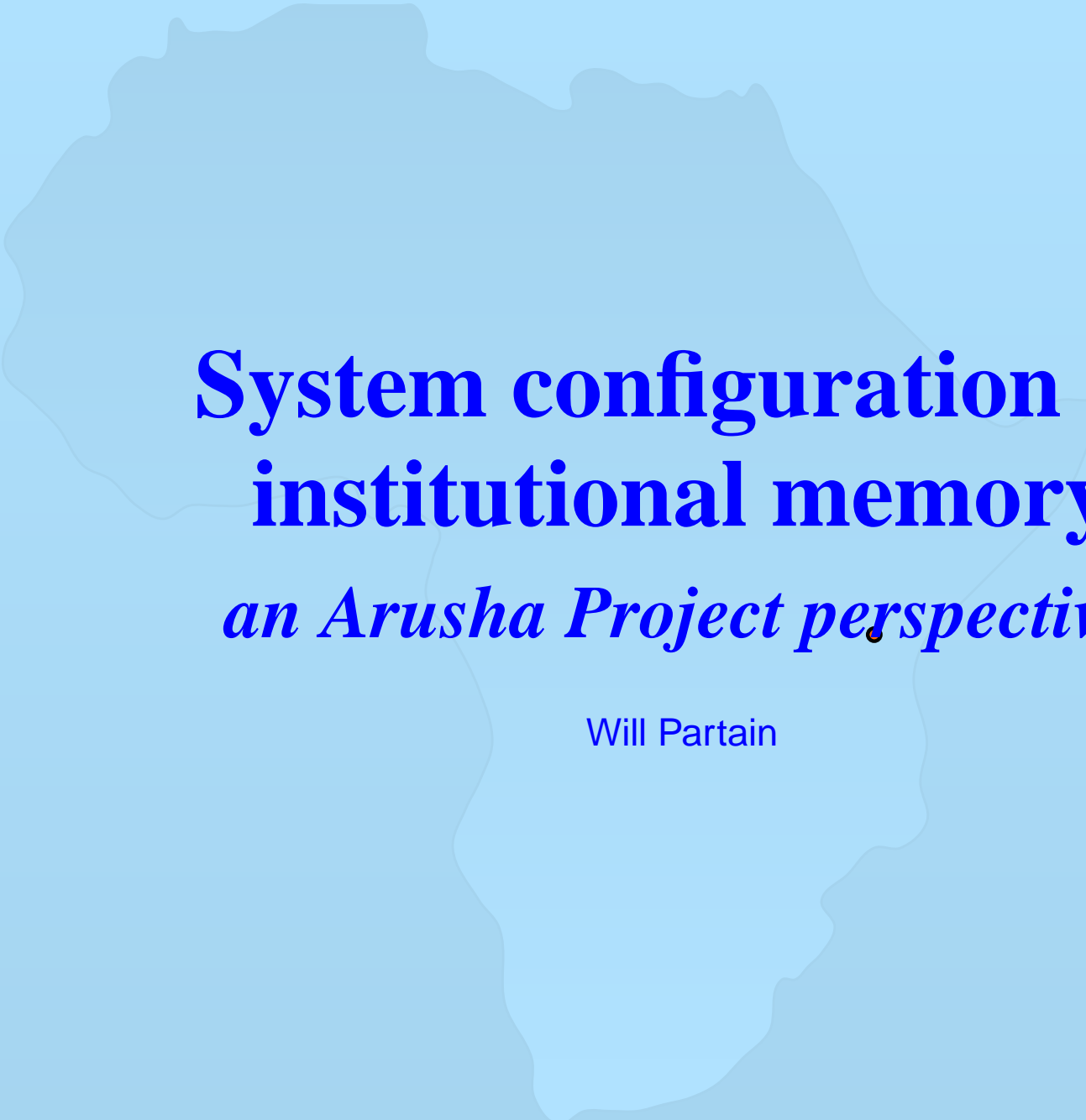
- Remember as **little** as possible
- Say it with **code**

# System configuration: lessons to learn?

- Remember as **little** as possible
- Say it with **code**
- Let **people** hold the 'memory'

# System configuration: lessons to learn?

- Remember as **little** as possible
- Say it with **code**
- Let **people** hold the 'memory'
- Keep the memory refreshed with **sharing and interaction**



**System configuration as  
institutional memory**  
*an Arusha Project perspective*

Will Partain