Machinekit

Current Status of Machinetalk

Alexander Rössler

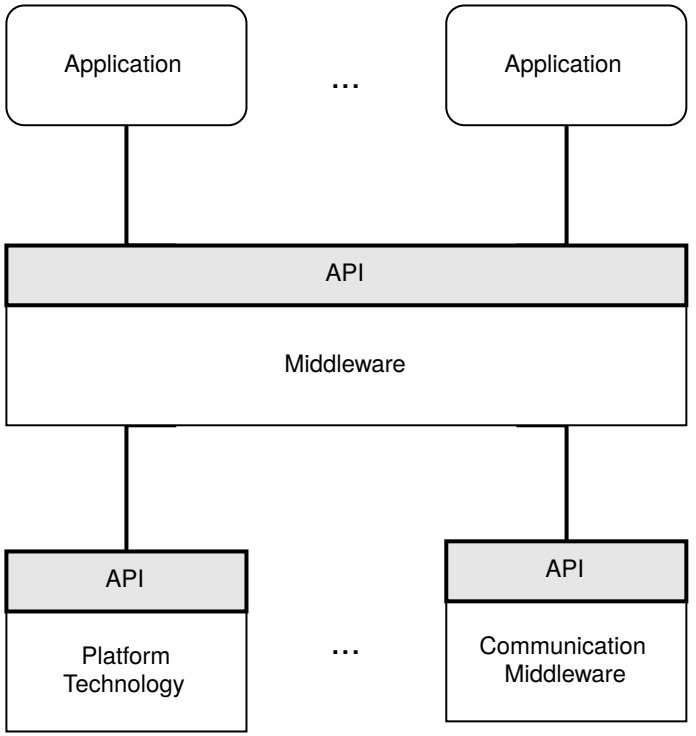
Overview

- Machinetalk Quick Intro
- Use in Machinekit
- Machinetalk GSL
- Machinetalk Generic
- Pymachinetalk

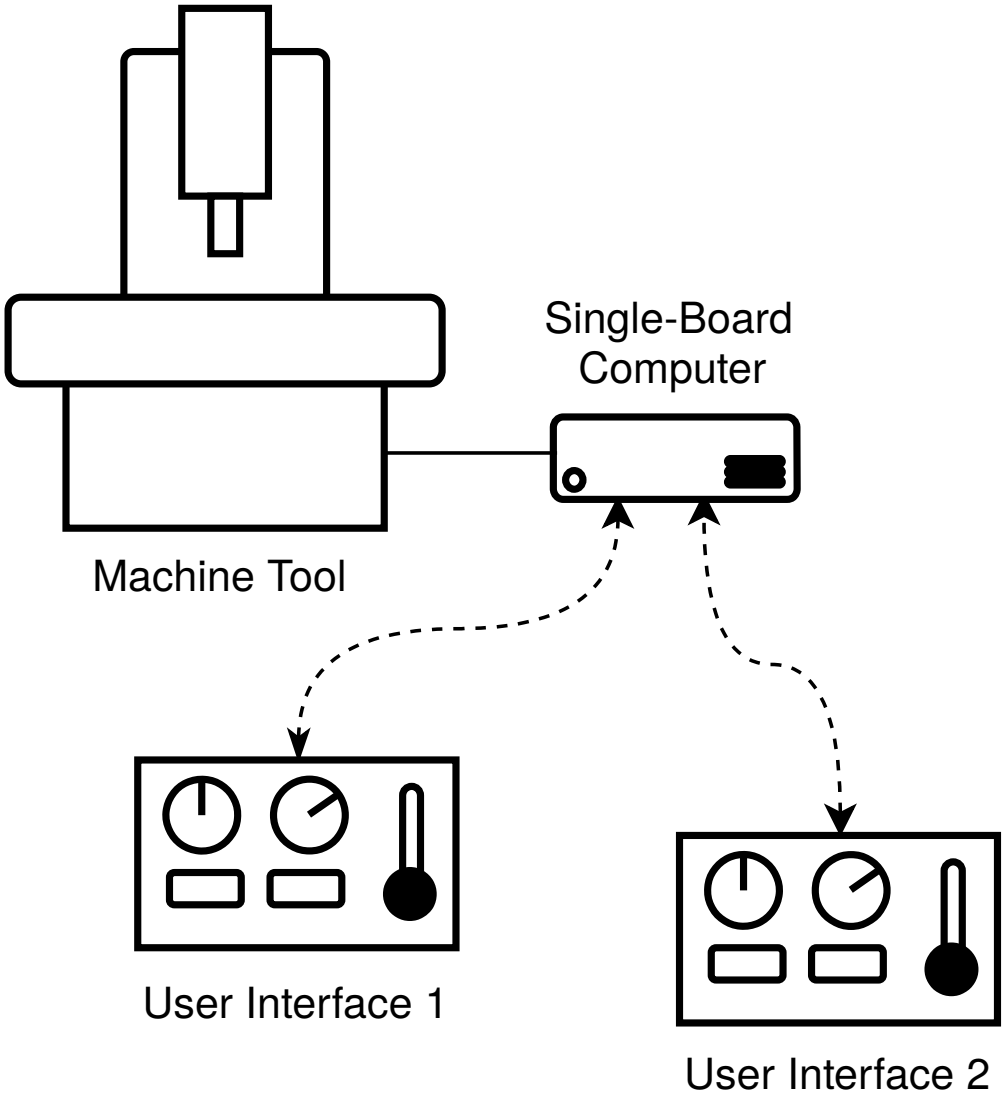
What is Machinetalk?

MACHINETALK

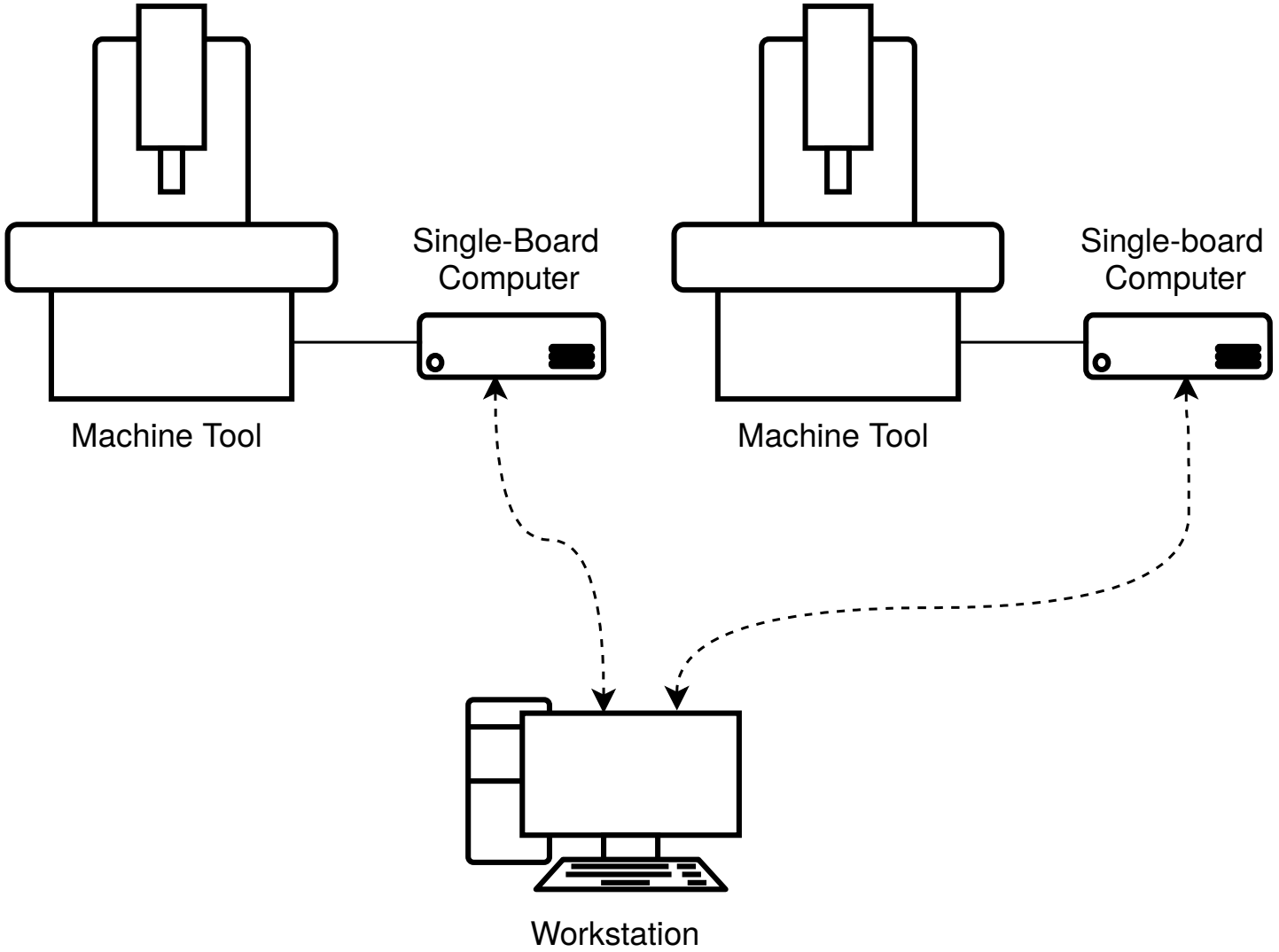
- Middleware
- Component ↔ Component
- RT ↔ Userland
- Local ↔ Remote



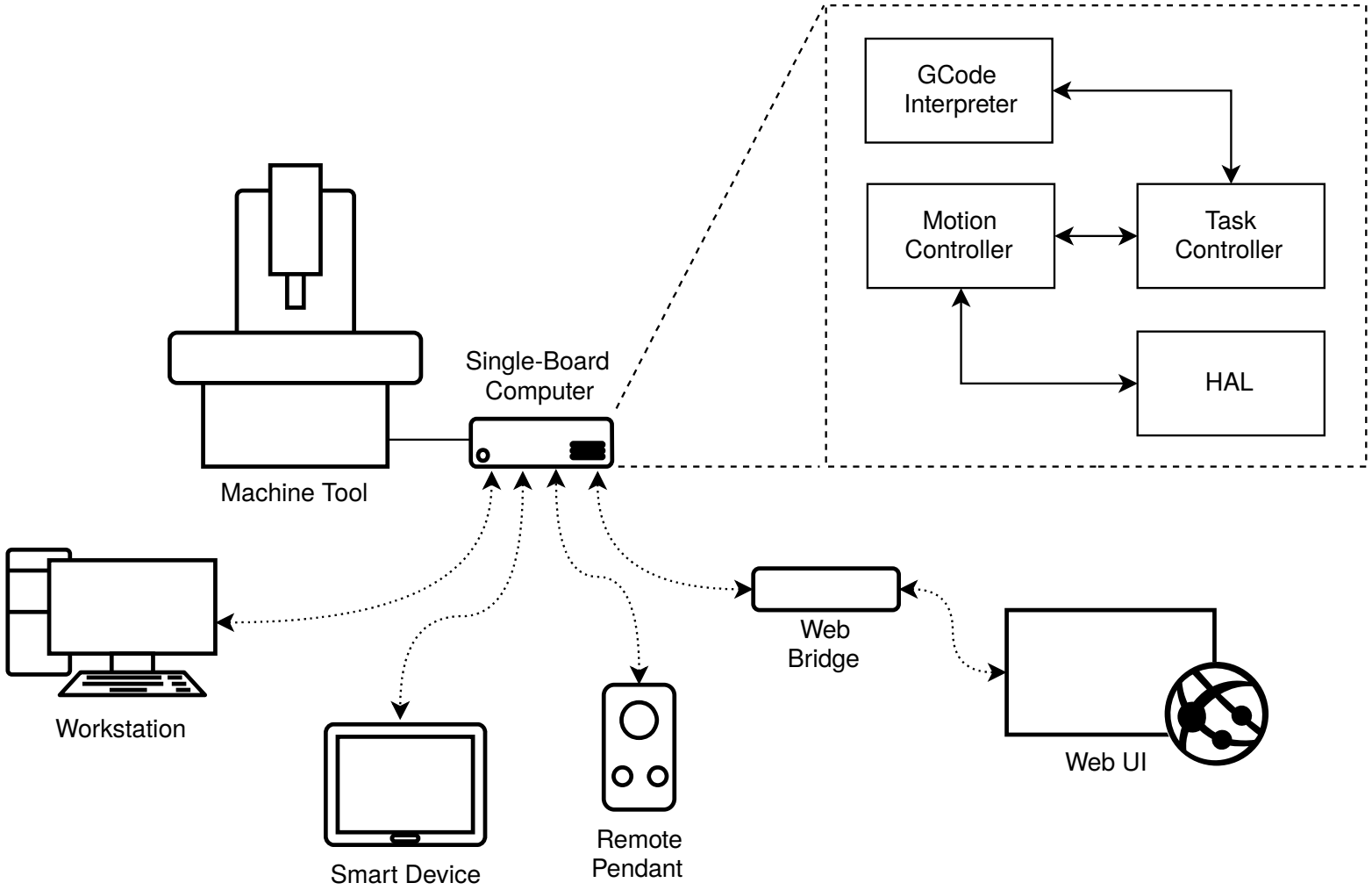
Use Case 1 – Multi UI



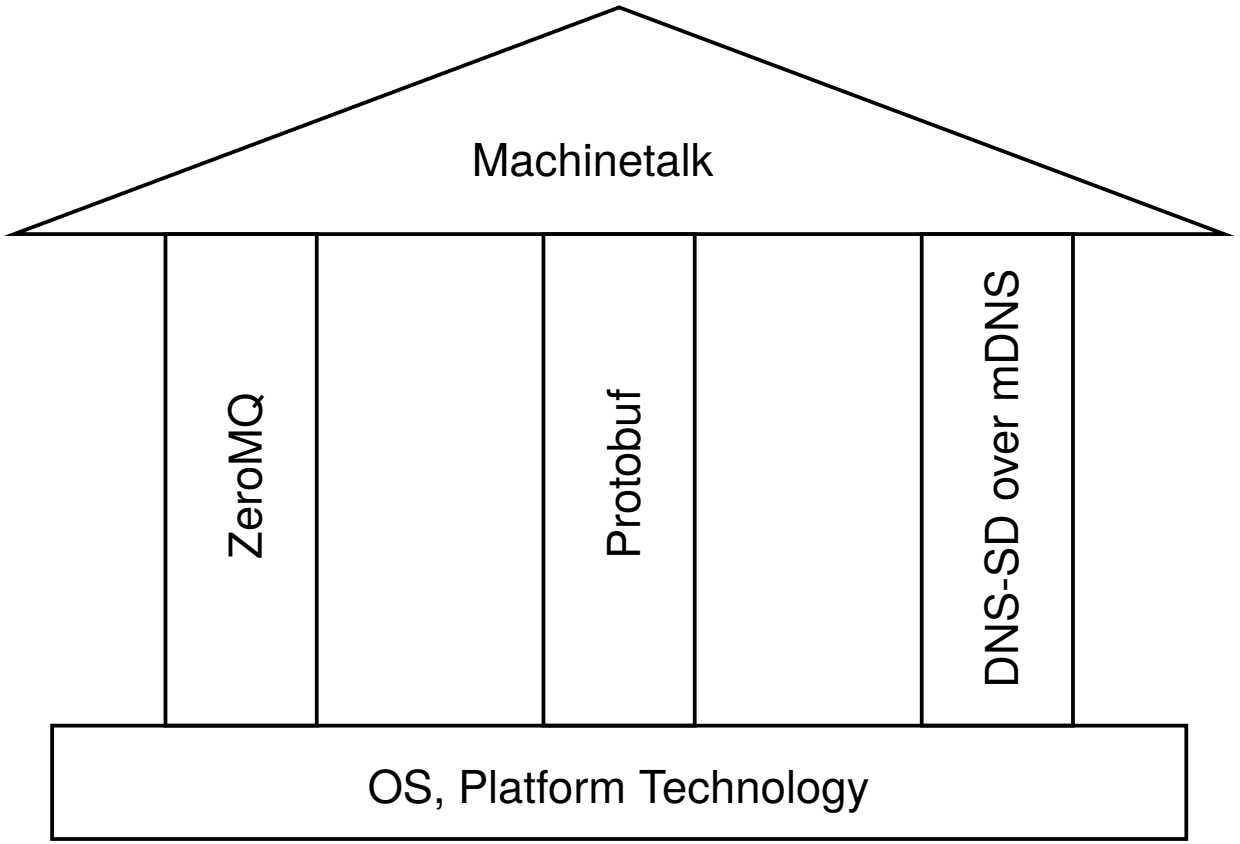
Use Case 2 – Multi Control



Use Case 3 – Internal and External



Parts

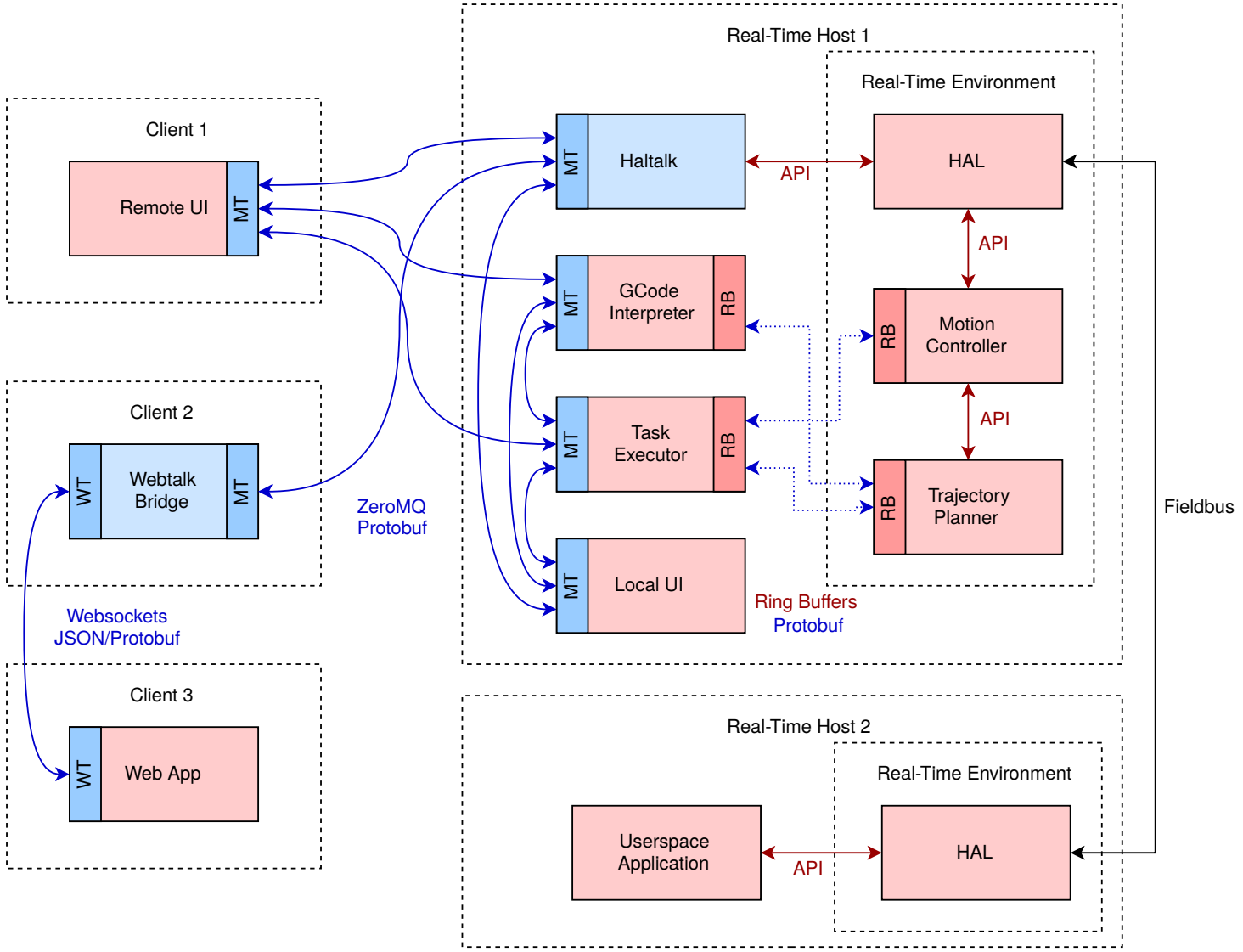


Protocol
Components
Services

Transport / Messaging
Serialization
Service Discovery

How is Machinetalk used in machinekit?

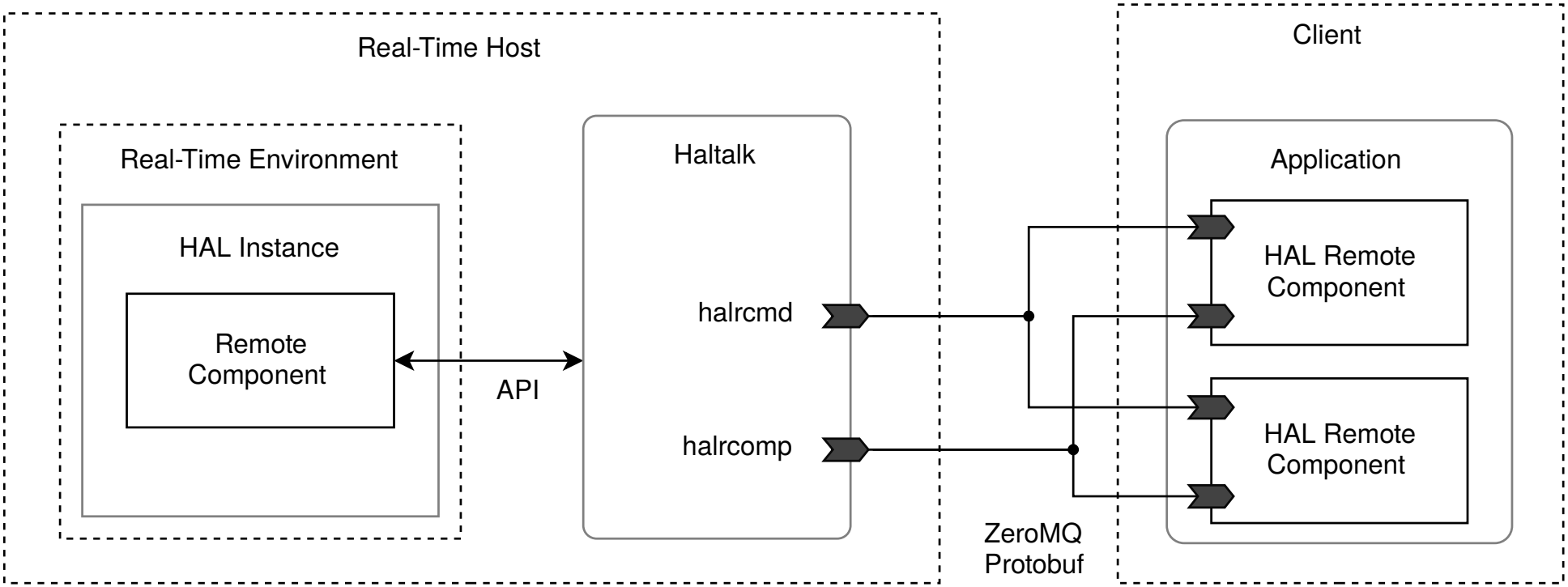
Scope


Real-Time Host 1
Real-Time Environment
Client 1
Remote UI
MT
Haltalk
API
HAL
API
GCode Interpreter
RB
Motion Controller
API
Client 2
Webtalk Bridge
WT
MT
Task Executor
Trajectory Planner
ZeroMQ Protobuf
Fieldbus
Local UI
Ring Buffers Protobuf
Websockets JSON/Protobuf
Client 3
Web App
Real-Time Host 2
Real-Time Environment
Userspace Application
API
HAL

Services & Components

Component	Services
Haltalk	halrcmd, halrcomp, (halgroup)
msgd	log
mklauncher	launcher, launchercmd
configserver	config
mkwrapper	status, command, error, file
previewmodule	preview, previewcmd

Close Up


Real-Time Host
Real-Time Environment
HAL Instance
Remote Component
API
Haltalk
halrcmd
halrcomp
ZeroMQ
Protobuf
Client
Application
HAL Remote Component
HAL Remote Component

What about NML?

- Replace with Machinetalk

  → not completed
- Major refactoring of the CNC Stack

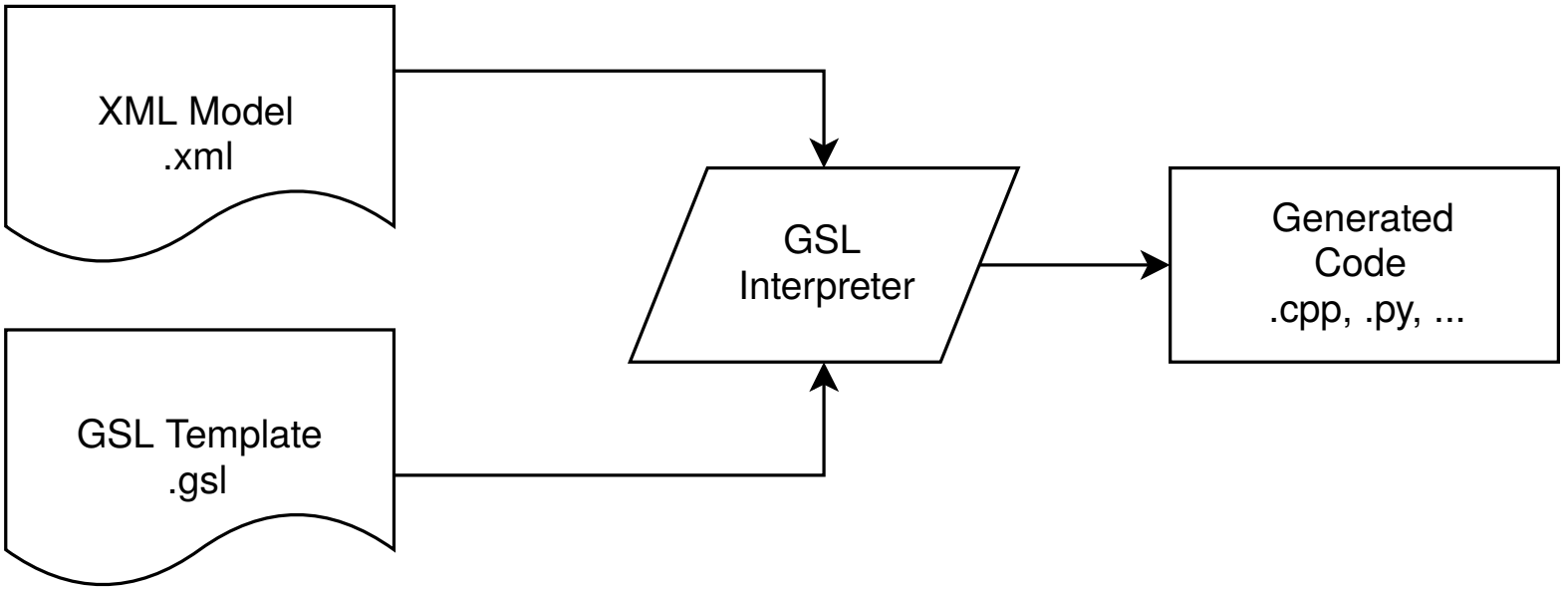
What's the deal with Machinetalk GSL?

Reasons

- Not only Protobuf + ZMQ + DNS-SD/mDNS
- Knowledge embedded in code
- Repetitive across languages
- Protocols better formalized
- Protobuf ↔ Messages

Imatix/GSL

- **Model Oriented Programing**
- **Control Model + Generator**

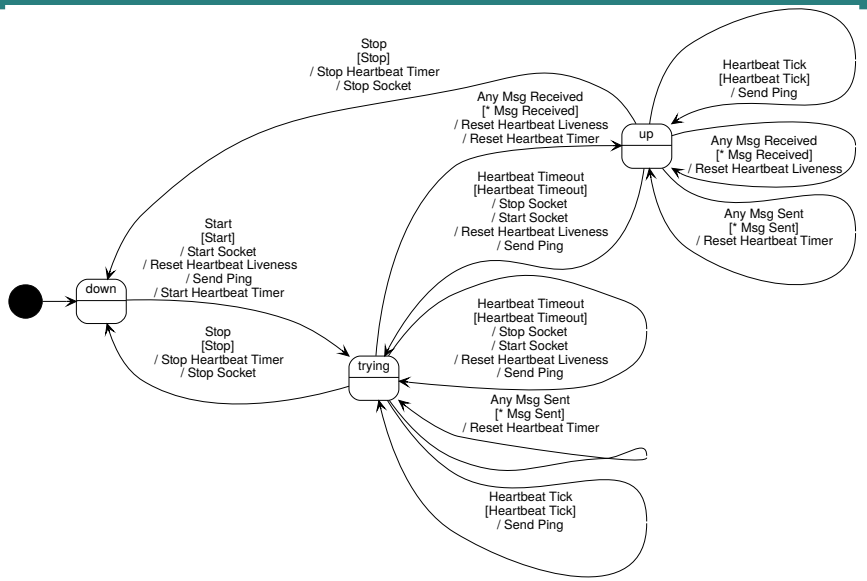


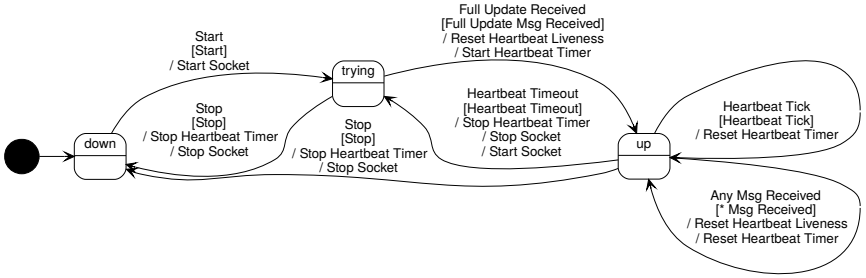
Example

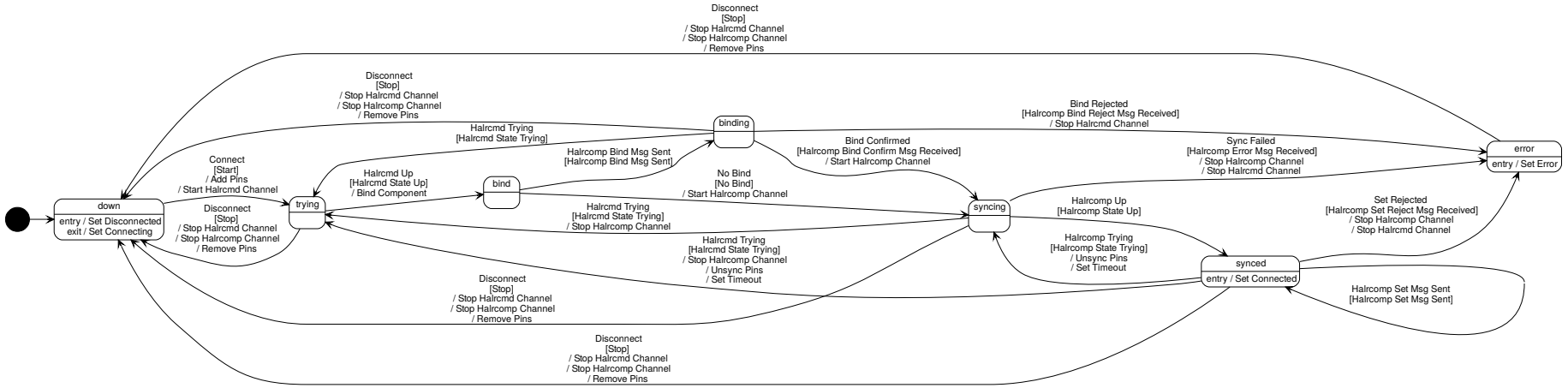
```
.template 1
.output "$(module.name:c).py"
.for class
    class $(class.Name)(object):
        def __init__(self):
.   for property
            self._$(name:c) = None
.   endfor

.   for property
        @property
        def $(name:c)(self):
            print('queried "$(name)"')
            return self._$(name:c)
. endfor
.endfor
.endtemplate
```

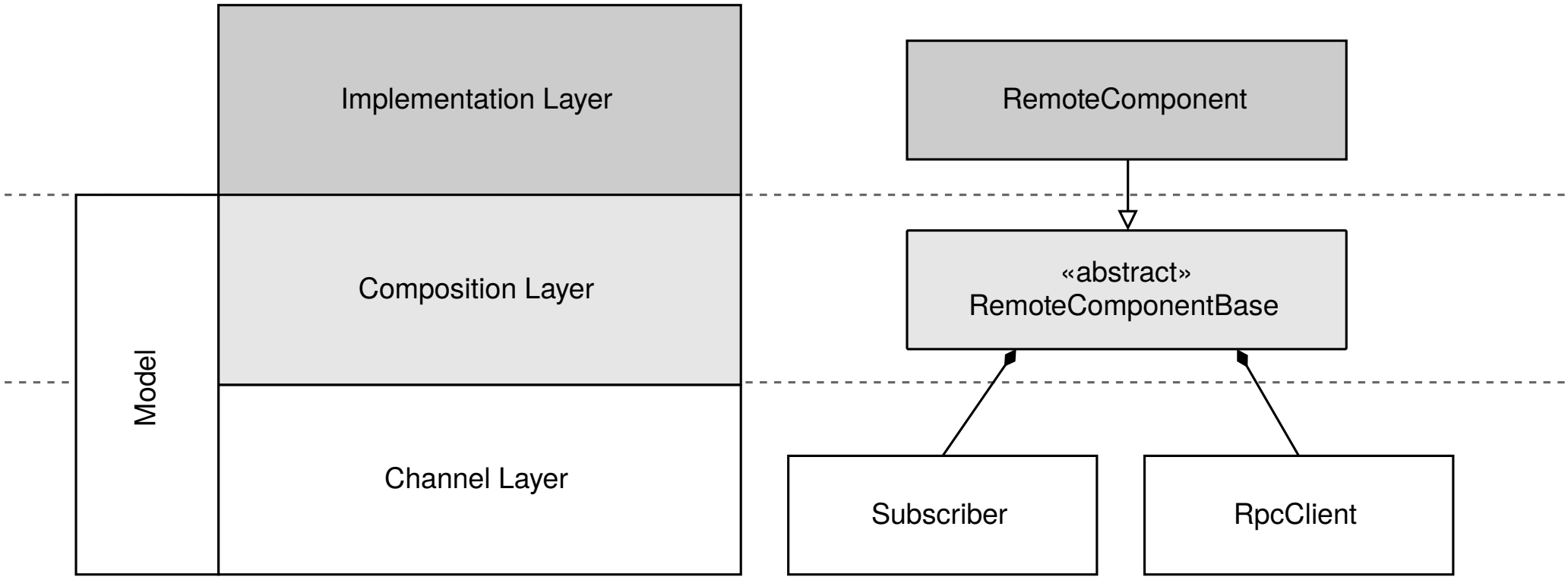
Halremote in FSMs



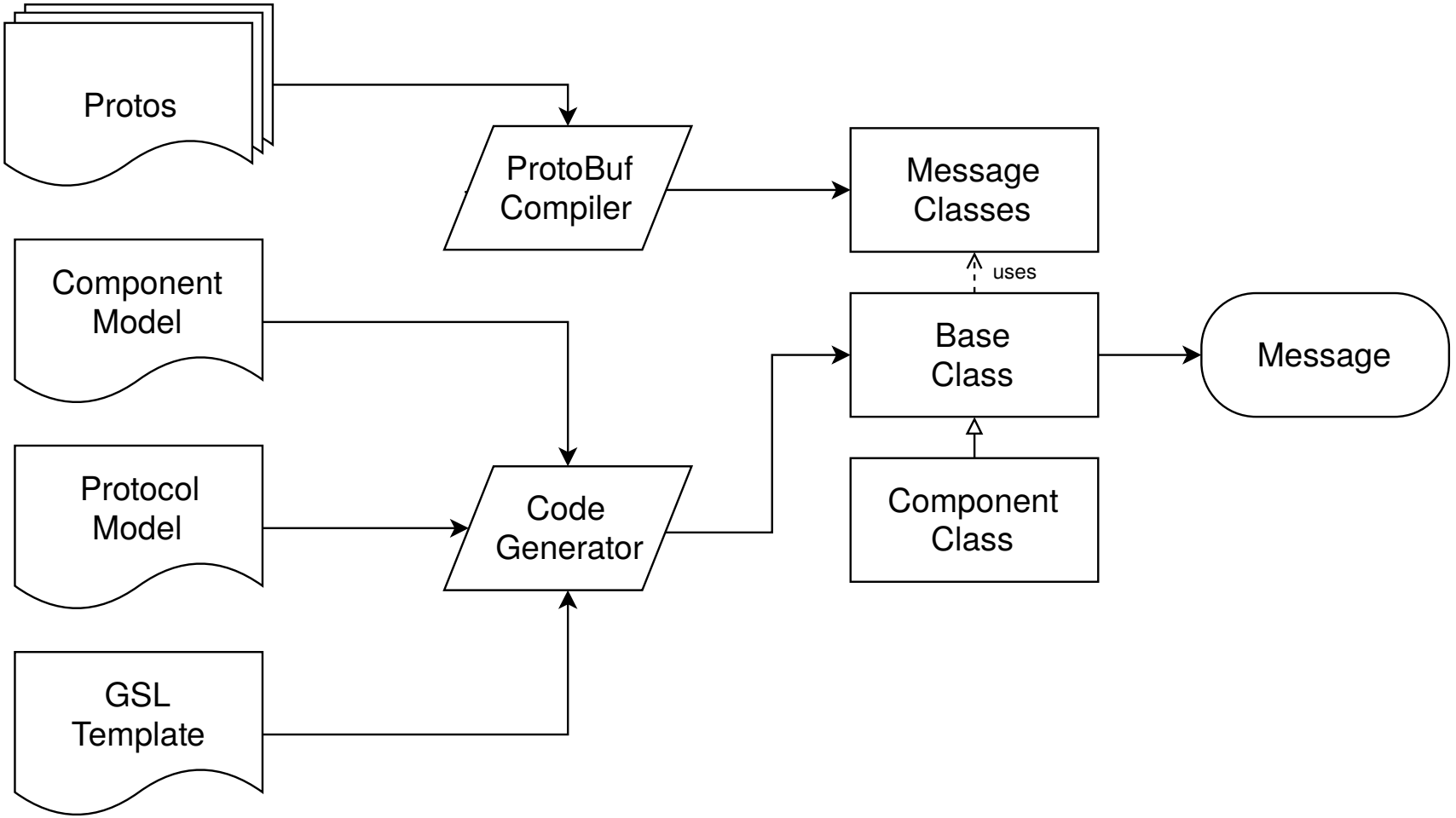




Layering

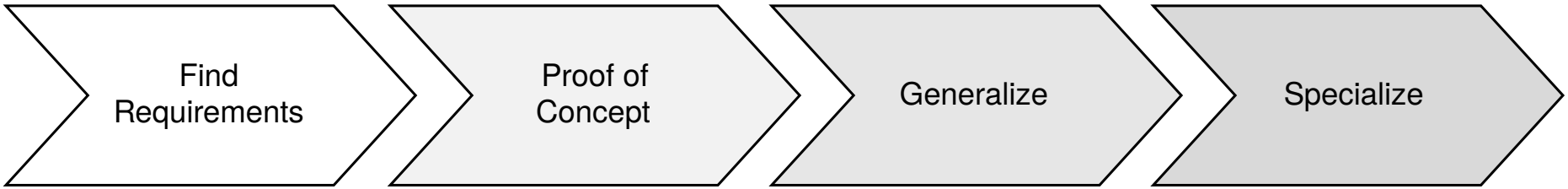

Implementation Layer
Composition Layer
Channel Layer
Model
RemoteComponent
«abstract»
RemoteComponentBase
Subscriber
RpcClient

Code Generation



Adding a new Language Binding

- **Find libs**
- **Write MVP**
- **Build a code generator**
- **Implement details**



GSL Conclusion

- **Perfect for protocols**
- **Less repetitive**
- **Focus on good models**
- **Consistency across languages**

	Python	Node.js	Qt/C++	UPPAAL	Documentation
Code Generator Size	525 LOC	448 LOC	828 LOC	536 LOC	224 LOC
Generated Code Size	3479 LOC	3540 LOC	8183 LOC	22066 LOC	1927 LOC + Dot: 824 LOC
Code Generation Ratio	6.63	7.90	9.89	41.16	12.28
Generated File Types	.py	.js	.h,.cpp	.xml	.md, .dot

Applications

- QtQuickVcp
- pymachinetalk
- WebVCP (webtalk-ng)
- node-machinetalk (not in production)

More

- **Formal Verification - UPPAAL**
- **SCXML support**

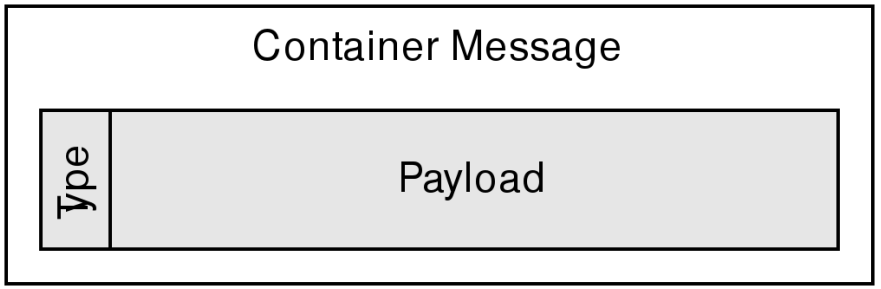
Can we use Machinetalk for other applications?

Machinetalk Generic

- **Generalizes**
  - machinetalk-protobuf
  - machinetalk-gsl
- **multiple “Machinetalks” can live side-by-side**
- **Adds project namespace/package**
- **Generate client and server base**

How to use

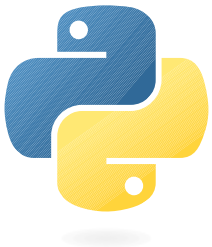
- Fork machinetalk-protobuf
- Fork machinetalk-gsl
- Change/add project name alias
- Implement your protocols
- Define your message structure
- Generate
- Implement



What is the current status of pymachinetalk?

Update

- Continued work
- Implemented mt-gsl
- Replaced avahi with zeroconf
- Easier to use serivce discovery
- Available via PyPI



python™

Example

```
import time
from pymachinetalk.dns_sd import ServiceDiscovery
import pymachinetalk.halremote as halremote

sd = ServiceDiscovery()

rcomp = halremote.RemoteComponent('anddemo', debug=False)
rcomp.newpin('button0', halremote.HAL_BIT, halremote.HAL_OUT)
rcomp.newpin('button1', halremote.HAL_BIT, halremote.HAL_OUT)
led_pin = rcomp.newpin('led', halremote.HAL_BIT, halremote.HAL_IN)
sd.register(rcomp)

sd.start()

try:
    while True:
        if rcomp.connected:
            print('LED status %s' %s str(led_pin.value)
        time.sleep(0.5)
except KeyboardInterrupt:
    pass

sd.stop()
```

Questions?