

afspm: A Manufacturer-Agnostic Automation Framework for SPM

Nicholas J. Sullivan^{1,2}, Julio J. Valdés³, Kirk H. Bevan^{1,2}, Peter Grutter²

Division of Materials Engineering, Faculty of Engineering, McGill University¹
Department of Physics, McGill University²
National Research Council Canada, Digital Technologies Research Centre³



Outline

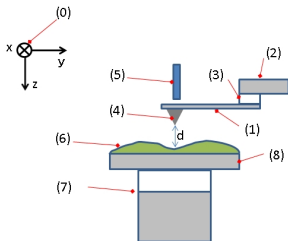
1 Justification

2 Introduction to afspm

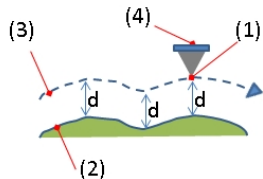
3 Design Particulars

SPM Basics

- An **atomically-sharp tip** is **scanned** above a **surface** measuring properties.
 - Allows spatial imaging, spectroscopic analysis, sample manipulation.
- Usable on various surfaces, only require **relatively flat** sample ($\sim 5\mu m$).



Typical AFM Configuration, Tom Toyosaki, Wikimedia Commons.



Schematics of AFM topographic image forming, Tom Toyosaki, Wikimedia Commons.

Challenges and Automation

Challenges and Automation

Challenges for Wider Adoption

- 1 Preparing, running, and analyzing requires significant **domain knowledge**.
- 2 Running an experiment requires **constant user attention**.
- 3 **Statistical** understanding is **limited** by the researcher's **decisions**.

Challenges and Automation

Challenges for Wider Adoption

- 1 Preparing, running, and analyzing requires significant **domain knowledge**.
- 2 Running an experiment requires **constant user attention**.
- 3 **Statistical** understanding is **limited** by the researcher's **decisions**.

Automation Prior Art

- **Conditioning** of the **tip** for proper surface characterization.
- **Structure classification**, for **where to scan** next.
- **Bayesian/Active Learning** for statistical decision making.

A Remaining Limitation

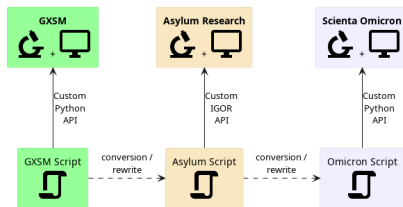
Many SPM systems allow custom scripts to run an experiment. But:

- Scripts written for a **specific** SPM system **cannot be re-used** on others.
- While **decoupling** is possible, it is **rarely** a **priority** for researchers.

A Remaining Limitation

Many SPM systems allow custom scripts to run an experiment. But:

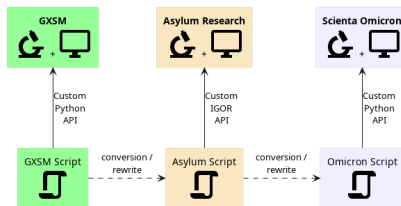
- Scripts written for a **specific** SPM system **cannot be re-used** on others.
- While **decoupling** is possible, it is **rarely** a **priority** for researchers.



A Remaining Limitation

Many SPM systems allow custom scripts to run an experiment. But:

- Scripts written for a **specific** SPM system **cannot be re-used** on others.
- While **decoupling** is possible, it is **rarely** a **priority** for researchers.



*Code **reuse** is **rare**.*

Goals, Scope, and Design Characteristics

Goals, Scope, and Design Characteristics

Goals

To facilitate **code sharing** and **reusability** of developed automation.

Goals, Scope, and Design Characteristics

Goals

To facilitate **code sharing** and **reusability** of developed automation.

Scope

Concerned with the **choices** a researcher would perform **during** an experiment.

Goals, Scope, and Design Characteristics

Goals

To facilitate **code sharing** and **reusability** of developed automation.

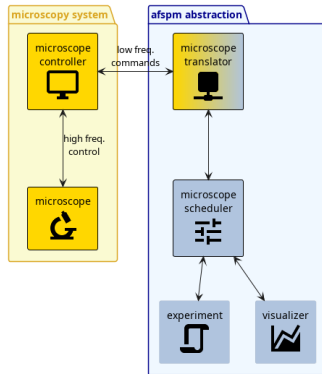
Scope

Concerned with the **choices** a researcher would perform **during** an experiment.

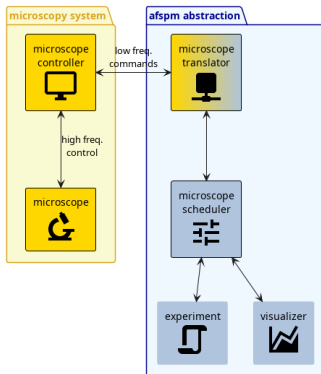
Design Characteristics

- **Standalone components** that each 'do one thing and do it well' (nodes).
- They communicate over '**pipes**' via generic **schemas**.
- It is easy to **split up** components among **different** computing **devices**.

Principal Components



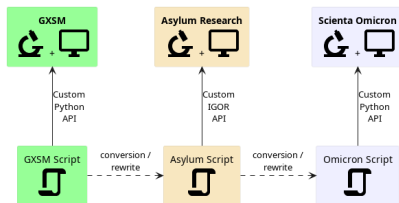
Principal Components



Any experiment must contain:

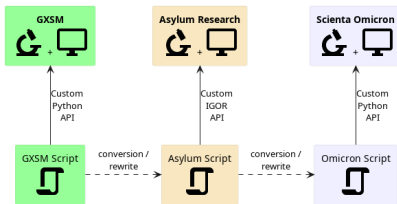
- **Microscope Translator:** translates between afspm-generic and microscope-specific.
- **Microscope Scheduler:** mediates control between components and translator and caches data.

Scripting with / without afspm

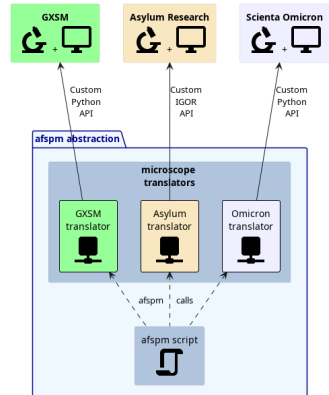


Code Reuse in SPM Scripting

Scripting with / without afspm

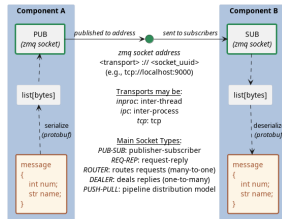


Code Reuse in SPM Scripting



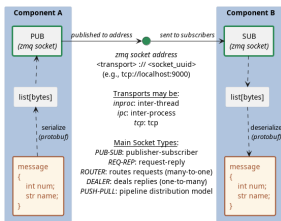
Approach Taken by afspm

Communication Protocol



Generic *schemas* sent between *network sockets* held by components.

Communication Protocol

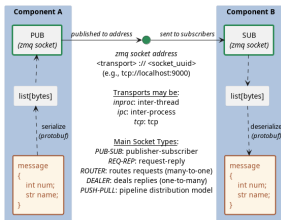


Generic *schemas* sent between *network sockets* held by components.

Serialization / Deserialization

Translates data structures into a format that can be stored/communicated.
Google Protocol Buffers: avoids schema violations, extra error handling.

Communication Protocol



Generic *schemas* sent between *network sockets* held by components.

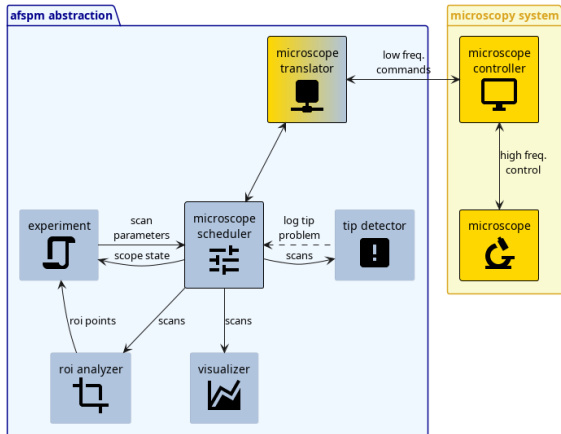
Serialization / Deserialization

Translates data structures into a format that can be stored/communicated.
Google Protocol Buffers: avoids schema violations, extra error handling.

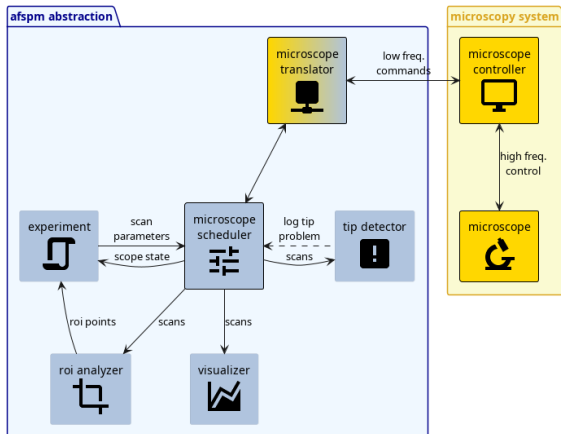
Data Transmission Protocol

Handles sending of data between 'sockets' via common 'nodes'.
ZeroMQ: abstracts away transports used, handles common roadblocks.

Example



Example



- **ROI Analyzer** reviews scans for interesting regions.
- **Experiment** switches between scanning a large region or a suggested ROI.
- **Tip Detector** evaluates the state of the tip, logs problem if deemed poor.
- **Visualizer** visualizes what is being scanned.

Our publisher-subscriber, control-request world

We *subscribe* to events published by the SPM and *request* actions of it.

Our publisher-subscriber, control-request world

We ***subscribe*** to events published by the SPM and ***request*** actions of it.

Publisher-Subscriber Path

- The SPM **publishes** messages types when its 'state' changes.
- Components may **subscribe** to only messages of interest.
- Messages may be **cached** by the Scheduler, to send to new components.

Our publisher-subscriber, control-request world

We **subscribe** to events published by the SPM and **request** actions of it.

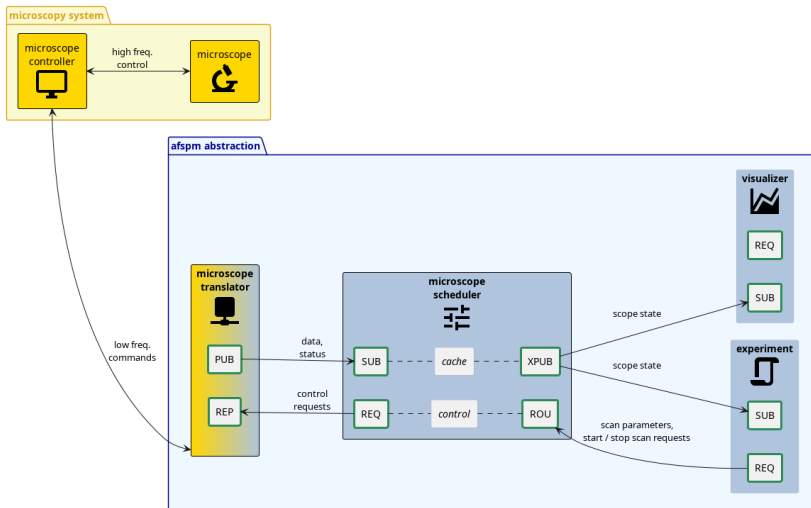
Publisher-Subscriber Path

- The SPM **publishes** messages types when its 'state' changes.
- Components may **subscribe** to only messages of interest.
- Messages may be **cached** by the Scheduler, to send to new components.

Control Path

- Components send **requests** to the SPM and receive **responses**.
- The Scheduler **routes** control to the SPM (one component at a time).
- Components not in control may flag **problems** – control is dropped.
 - A user may **manually** fix a problem and unflag the problem.
 - A component that **fixes** a flagged problem can grab control.

afspm: Detailed View



Publisher-Subscriber Path

```
message DataAspects {  
  optional Size2u shape = 1;  
  optional string units = 2;  
}  
  
message SpatialAspects {  
  optional Rect2d roi = 1;  
  optional string units = 2;  
}  
  
message ScanParameters2d {  
  optional SpatialAspects spatial = 1;  
  optional DataAspects data = 2;  
}
```

2D Scan Parameters Schema

```
message Scan2d {  
  optional ScanParameters2d params = 1;  
  optional google.protobuf.Timestamp timestamp = 2;  
  optional string channel = 3;  
  optional string filename = 5;  
  repeated double values = 4;  
}
```

2D Scan Schema

Publisher-Subscriber Path

```
message DataAspects {  
  optional Size2u shape = 1;  
  optional string units = 2;  
}  
  
message SpatialAspects {  
  optional Rect2d roi = 1;  
  optional string units = 2;  
}  
  
message ScanParameters2d {  
  optional SpatialAspects spatial = 1;  
  optional DataAspects data = 2;  
}
```

2D Scan Parameters Schema

```
message Scan2d {  
  optional ScanParameters2d params = 1;  
  optional google.protobuf.Timestamp timestamp = 2;  
  optional string channel = 3;  
  optional string filename = 5;  
  repeated double values = 4;  
}
```

2D Scan Schema

Frame 1

'Scan2d'

Frame 2

Scan2d Data

Envelope

Serialized Data Structure

Message Format

Control Path

TODO: UPDATE!!!

```
enum ControlRequest {  
    REQ_UNDEFINED = 0;  
    REQ_START_SCAN = 1;    // [...]  
    REQ_REQUEST_CTRL = 4;  // [...]  
    REQ_ADD_EXP_PRBLM = 6; // [...]  
    REQ_SET_CONTROL_MODE = 8;  
}
```

Control Request Schema

```
enum ControlResponse {  
    REP_SUCCESS = 0;  
    REP_FAILURE = 1;    // [...]  
    REP_NO_RESPONSE = 3; // [...]  
    REP_NOT_FREE = 7;   // [...]  
    REP_PARAM_NOT_SUPPORTED = 9; // [...]  
}
```

Control Response Schema

Control Path

TODO: UPDATE!!!

```
enum ControlRequest {  
    REQ_UNDEFINED = 0;  
    REQ_START_SCAN = 1;    // [...]  
    REQ_REQUEST_CTRL = 4;  // [...]  
    REQ_ADD_EXP_PRBLM = 6; // [...]  
    REQ_SET_CONTROL_MODE = 8;  
}
```

Control Request Schema

```
enum ControlResponse {  
    REP_SUCCESS = 0;  
    REP_FAILURE = 1;    // [...]  
    REP_NO_RESPONSE = 3; // [...]  
    REP_NOT_FREE = 7;   // [...]  
    REP_PARAM_NOT_SUPPORTED = 9; // [...]  
}
```

Control Response Schema

```
enum ExperimentProblem {  
    EP_NONE = 0;  
    EP_TIP_SHAPE_CHANGED = 1;  
    EP_DEVICE_MALFUNCTION = 2;  
    EP_FEEDBACK_NON_OPTIMAL = 3;  
}
```

Experiment Problem Schema

```
enum ControlMode {  
    CM_UNDEFINED = 0;  
    CM_MANUAL = 1;  
    CM_AUTOMATED = 2;  
}
```

Control Mode Schema

Control Path

TODO: UPDATE!!!

```
enum ControlRequest {
    REQ_UNDEFINED = 0;
    REQ_START_SCAN = 1;      // [...]
    REQ_REQUEST_CTRL = 4;    // [...]
    REQ_ADD_EXP_PRBLM = 6;   // [...]
    REQ_SET_CONTROL_MODE = 8;
}
```

Control Request Schema

```
enum ExperimentProblem {
    EP_NONE = 0;
    EP_TIP_SHAPE_CHANGED = 1;
    EP_DEVICE_MALFUNCTION = 2;
    EP_FEEDBACK_NON_OPTIMAL = 3;
}
```

Experiment Problem Schema

```
enum ControlResponse {
    REP_SUCCESS = 0;
    REP_FAILURE = 1;      // [...]
    REP_NO_RESPONSE = 3;  // [...]
    REP_NOT_FREE = 7;     // [...]
    REP_PARAM_NOT_SUPPORTED = 9; // [...]
}
```

Control Response Schema

```
enum ControlMode {
    CM_UNDEFINED = 0;
    CM_MANUAL = 1;
    CM_AUTOMATED = 2;
}
```

Control Mode Schema

Frame 1

REQ_REQUEST_CTRL

Frame 2

EP_TIP_SHAPE_CHANGED

*Command Request Enum**Serialized Data Structure(s)*

Message Format

Spawning, Monitoring, and the Config File

*An experiment defined via a **config** file, with components **spawned** as needed.*

Spawning, Monitoring, and the Config File

*An experiment defined via a **config** file, with components **spawned** as needed.*

Spawning

- Each component is instantiated as a child process (independent memory).
 - One crashed component does not crash all.
- We can spawn different components on different computers.
 - The 'spawn' command allows indication of which components to spawn.

Spawning, Monitoring, and the Config File

*An experiment defined via a **config** file, with components **spawned** as needed.*

Spawning

- Each component is instantiated as a child process (independent memory).
 - One crashed component does not crash all.
- We can spawn different components on different computers.
 - The 'spawn' command allows indication of which components to spawn.

Monitoring

- The parent process monitors spawned components for heartbeats.
- If a process has not 'beat' in a timeframe, it is killed and respawned.
- Then the caching logic chosen is important – must be able to run.

Sample Config File

```
# --- URLs --- #
pub_url = "tcp://127.0.0.1:9000"
psc_url = "tcp://127.0.0.1:9001"

server_url = "tcp://127.0.0.1:6666"
router_url = "tcp://127.0.0.1:6667"
# [...]
exp_scan_res = [256, 256]
# [...]
```

General Variables

```
# ----- PubSub ----- #
[translator_pub]
class = 'afspm.io.pubsub.publisher.Publisher'
url = 'pub_url'
# [...]
[experiment_sub]
class = 'afspm.io.pubsub.subscriber.Subscriber'
sub_url = 'psc_url'
# [...]
```

Intermediary Classes

```
[translator]
component = true
class = 'afspm.components.microscope.translators.gxsm.translator.GxsmTranslator'
publisher = 'translator_pub'
control_server = 'translator_server'
# [...]
```

Config Components

Expandability

*We defined a ‘basis’ set of schemas we believe are standard.
But this is quite arbitrary! Here are ways you can expand upon them.*

Expandability

*We defined a ‘basis’ set of schemas we believe are standard.
But this is quite arbitrary! Here are ways you can expand upon them.*

Parameters

- Can add generic **parameter** IDs and/or create ‘composite’ parameters (e.g. ScanParameters2d).

Expandability

*We defined a **'basis'** set of schemas we believe are standard.
But this is quite arbitrary! Here are ways you can expand upon them.*

Parameters

- Can add generic **parameter** IDs and/or create 'composite' parameters (e.g. ScanParameters2d).

Actions

- Can send these by providing generic **action** ID linked to method.

Expandability

*We defined a **‘basis’** set of schemas we believe are standard.
But this is quite arbitrary! Here are ways you can expand upon them.*

Parameters

- Can add generic **parameter** IDs and/or create ‘composite’ parameters (e.g. ScanParameters2d).

Actions

- Can send these by providing generic **action** ID linked to method.

Custom Messages

- Any user may create custom messages or expand existing ones.

Expandability

*We defined a ‘basis’ set of schemas we believe are standard.
But this is quite arbitrary! Here are ways you can expand upon them.*

Parameters

- Can add generic **parameter** IDs and/or create ‘composite’ parameters (e.g. ScanParameters2d).

Actions

- Can send these by providing generic **action** ID linked to method.

Custom Messages

- Any user may create custom messages or expand existing ones.

Operating Modes

- If switching between operating modes with presets, use actions.
- If needing to explicit parameters, add new ‘composite’ parameter request.

The End

Let us know what you think and help us make it better.

[afspm on github](#)