Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○○○○○

afspm Overview

Nick Sullivan

National Research Council Canada (NRC/CNRC)

May 9, 2024

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○○○○○

Outline

1 Justification

2 Introduction to afspm

3 Design Particulars

SPM Basics

In Scanning Probe Microscopy (SPM), an **atomically-sharp tip** is **scanned** above a **surface**, while **measuring** one or more **properties** gleaned from this tip.

This process allows **atomic-level imaging** of properties, spectroscopic analysis, and even manipulation of a sample (toward **atomic-scale manufacturing**).
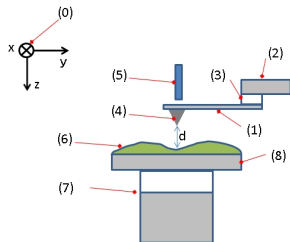

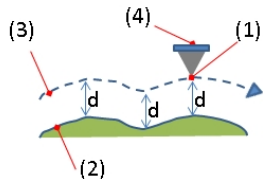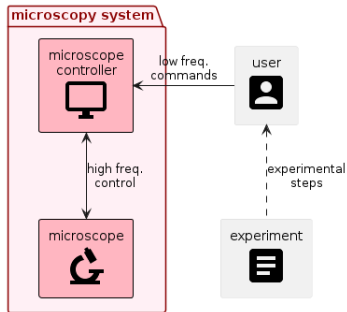Figure: Typical AFM Configuration, Tom Toyosaki, Wikimedia Commons.

Figure: Schematics of AFM topographic image forming, Tom Toyosaki, Wikimedia Commons.
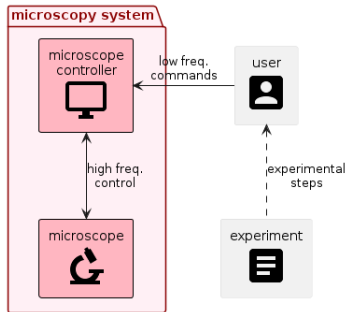
Justification
○●○

Introduction to afspm
○○○○

Design Particulars
○○○○○○

Standard Experiment

In a traditional SPM experiment, a researcher with domain knowledge will:

1. **Prepare the system**: including defining the SPM mode (e.g. FM-AFM).

2. **Run the experiment**: monitoring collected scans, deciding on next scans, and updating any aspects of the experiment.

3. **Finalize the experiment**: by undoing any experiment-specific setup needed to run.

Justification
○●○

Introduction to afspm
○○○○

Design Particulars
○○○○○○

Standard Experiment

In a traditional SPM experiment, a researcher with domain knowledge will:

1. **Prepare the system**: including defining the SPM mode (e.g. FM-AFM).

2. **Run the experiment**: monitoring collected scans, deciding on next scans, and updating any aspects of the experiment.

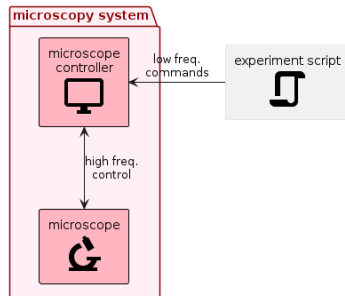3. **Finalize the experiment**: by undoing any experiment-specific setup needed to run.

*Running the experiment is often **long**, and requires **constant** researcher **attention**.*

Justification
ooo

Introduction to afspm
oooo

Design Particulars
oooooo

System Scripting and Code Reuse

Many SPM systems allow custom scripts to run an experiment.

However:

microscopy system

- Scripts written for a **specific** SPM system **cannot be re-used** for other SPM systems: different API/language constraints.
- While **decoupling** of SPM device and experiment logic **is possible**, it is **rarely** a **priority** for researchers.

Justification
○○●

Introduction to afspm
○○○○

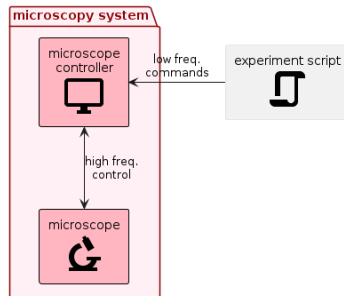Design Particulars
○○○○○○

System Scripting and Code Reuse

Many SPM systems allow custom scripts to run an experiment.

However:

- Scripts written for a **specific** SPM system **cannot be re-used** for other SPM systems: different API/language constraints.
- While **decoupling** of SPM device and experiment logic **is possible**, it is **rarely** a **priority** for researchers.

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

Justification
○○○

Introduction to afspm
●○○○

Design Particulars
○○○○○○

Goals and Scope

Justification
OOO

Introduction to afspm
●OOO

Design Particulars
OOOOOO

Goals and Scope

Goals

- **Clear Decoupling**: of SPM device specifics from the desired experiment; to allow reuse of a given experiment on multiple devices.
- **Multi-Language Support**: for integration with SPM devices; we should not be the limiting factor to integration.
- **Pausable Automation**: to allow a researcher to take over.
- **Separable on Multiple Devices**: composed of concise, separable components.

Justification
OOO

Introduction to afspm
●OOO

Design Particulars
OOOOOO

Goals and Scope

Goals

- **Clear Decoupling**: of SPM device specifics from the desired experiment; to allow reuse of a given experiment on multiple devices.
- **Multi-Language Support**: for integration with SPM devices; we should not be the limiting factor to integration.
- **Pausable Automation**: to allow a researcher to take over.
- **Separable on Multiple Devices**: composed of concise, separable components.

Scope

afspm will concern itself **only** with automation of high-level, low-frequency decisions a researcher would perform **during** an experiment.

Justification
○○○

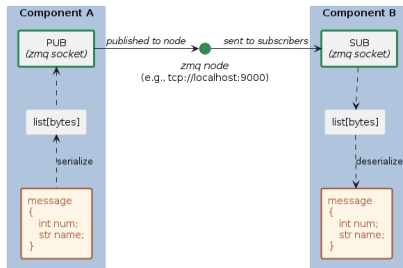Introduction to afspm
○●○○

Design Particulars
○○○○○○

High-Level Design
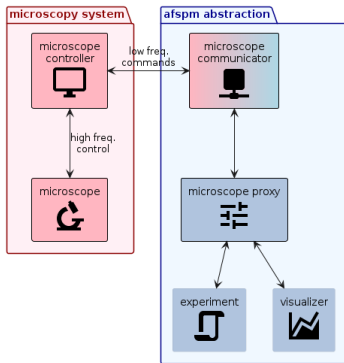
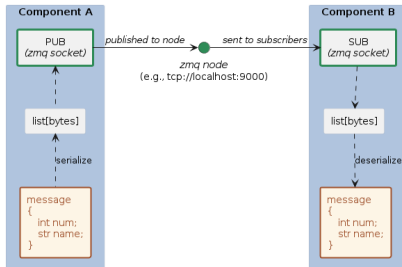afspm is designed around 'computation' components that correspond to nodes in a network. An experiment contains:

- **Microscope Communicator**: communicates with microscope, translating between afspm-generic and microscope-specific language.

- **Microscope Proxy**: mediates control of the microscope (only 1 component in control at a time) and caches data received from microscope.

- **Other Components**: the one or more components the user requires to run their experiment.

Justification
000

Introduction to afspm
○○●○

Design Particulars
000000

Communication Protocol

afspm uses protobuffers and ZeroMQ (both cross-platform / cross-language).

Justification
000

Introduction to afspm
0000

Design Particulars
000000

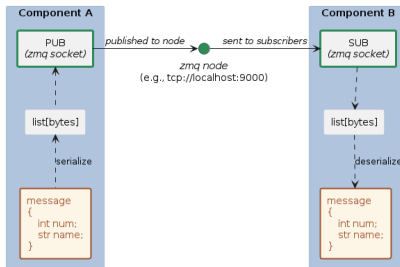## Communication Protocol

afspm uses protobuffers and ZeroMQ (both cross-platform / cross-language).

protobuffers: Serialization / Deserialization Library

Translates data structures into a format that can be stored/communicated).

Justification
ooo

Introduction to afspm
oooo

Design Particulars
oooooo

Communication Protocol

**afspm** uses protobuffers and ZeroMQ (both cross-platform / cross-language).

protobuffers: Serialization / Deserialization Library

Translates data structures into a format that can be stored/communicated).

ZeroMQ: Communication / Concurrency Library

Allows data to be sent between 'sockets' via common 'nodes'. Abstracts away protocols used, allowing easy switching between different protocols (e.g., TCP/IP, interprocess communication, threads).

Justification
000

Introduction to afspm
000●

Design Particulars
000000

Example
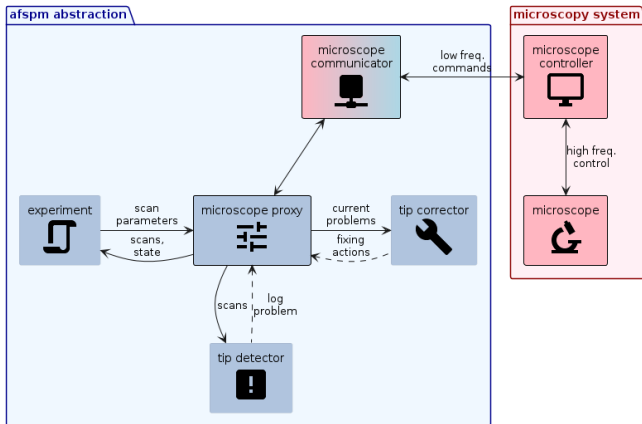
- **Experiment** constantly decides on the next region to scan.
- **Tip Detector** constantly evaluates the state of the tip, logging a problem if deemed poor.
- **Tip Corrector** takes control if 'bad tip' problem has been logged, takes steps to fix it, and releases control once fixed (according to the detector).

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
●○○○○○

I/O Paths

Justification
○○○

Introduction to afspm
○○○○

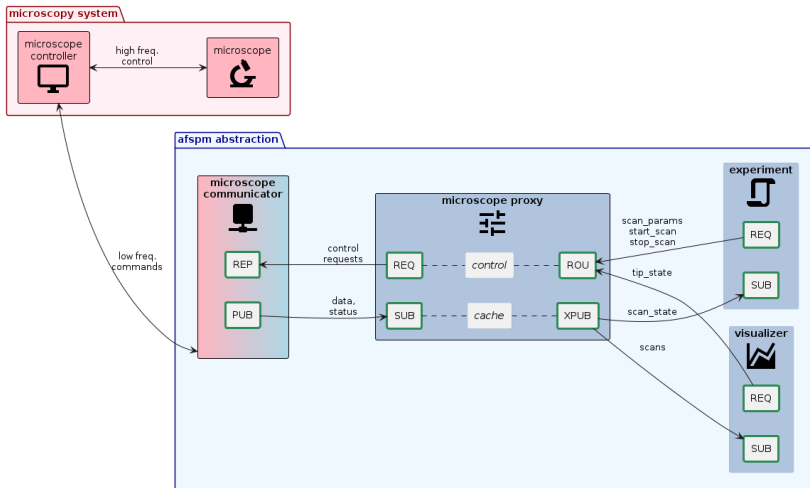Design Particulars
●○○○○○

I/O Paths

Publisher-Subscriber Path

- The MicroscopeCommunicator **publishes** ScanState, ScanParameters, and Scan **changes**.

- These are passed on by the MicroscopeProxy. Data is **stored** in a **cache** and re-sent to new/crashed components.

- Each component choose what aspects to **subscribe** to, and receives data from these.

I/O Paths

Publisher-Subscriber Path

- The MicroscopeCommunicator **publishes** ScanState, ScanParameters, and Scan **changes**.
- These are passed on by the MicroscopeProxy. Data is **stored** in a **cache** and re-sent to new/crashed components.
- Each component choose what aspects to **subscribe** to, and receives data from these.

Control Path

- Each component can send **control requests** over its client.
- The MicroscopeProxy determines which **client** is **in-control**, and **forwards** these to the MicroscopeCommunicator.
- The MicroscopeCommunicator **receives** control requests from one client and **responds**.

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○●○○○○○

afspm: Detailed View

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○●○○○

afspm Controller

afspm Controller

Cache Logic

Data is **stored** into the **cache** according to a **user-defined configuration**.

These map a **protobuf message** to a **cache key** (envelope), and vice-versa.

Justification
OOO

Introduction to afspm
OOOO

Design Particulars
OOO●OOO

afspm Controller

Cache Logic

Data is **stored** into the **cache** according to a **user-defined configuration**.

These map a **protobuf message** to a **cache key** (envelope), and vice-versa.

Experiment Problems

Any component can **report** experiment **problems**, indicating issues that should cause the experiment to **pause** until **resolved**, and can **remove** these problems.

This allows, e.g., detecting a tip crash and attempting to correct it.

Justification
000

Introduction to afspm
0000

Design Particulars
000●000

afspm Controller

Cache Logic

Data is **stored** into the **cache** according to a **user-defined configuration**.

These map a **protobuf message** to a **cache key** (envelope), and vice-versa.

Experiment Problems

Any component can **report** experiment **problems**, indicating issues that should cause the experiment to **pause** until **resolved**, and can **remove** these problems.

This allows, e.g., detecting a tip crash and attempting to correct it.

Control Modes

The MicroscopeProxy defines the **control mode**, which can be:

- **Automated**: default, automation runs.
- **Manual**: pause automation.
- **Problem**: experiment problems are logged, pause automation.

The Config File

afspm uses a **single** TOML **configuration file** per experiment.

Within this file, a user defines:

- The communication protocols used between components.
- Common variables passed between components (e.g. how big the scan size will be).
- The components to spawn.

Top-level definitions can function as **variables**: any **references** deeper in the config are **replaced** by them. This should minimize repeating oneself.

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○○○○●○

Spawning the Experiment

Justification
OOO

Introduction to afspm
OOOO

Design Particulars
OOOOO●O

Spawning the Experiment

Distributed Computing

Components can be **split up** among devices; on startup, the components to spawn can be specified.

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○○○○●○

Spawning the Experiment

Distributed Computing

Components can be **split up** among devices; on startup, the components to spawn can be specified.

Component Monitoring

All spawned components are **monitored**:

- Each sends **heartbeats** at a regular cadence.
- If one **stops** beating, it is **restarted**.

This should minimize a crash breaking experiments.

Justification
○○○

Introduction to afspm
○○○○

Design Particulars
○○○○○○●

The End

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

afspm on github