Future Architecture Investigations at Diamond

Keith Ralphs, Joe Handford, Diamond Light Source, Harwell Science and Innovation Campus, Didcot, Oxon OX11 0DE

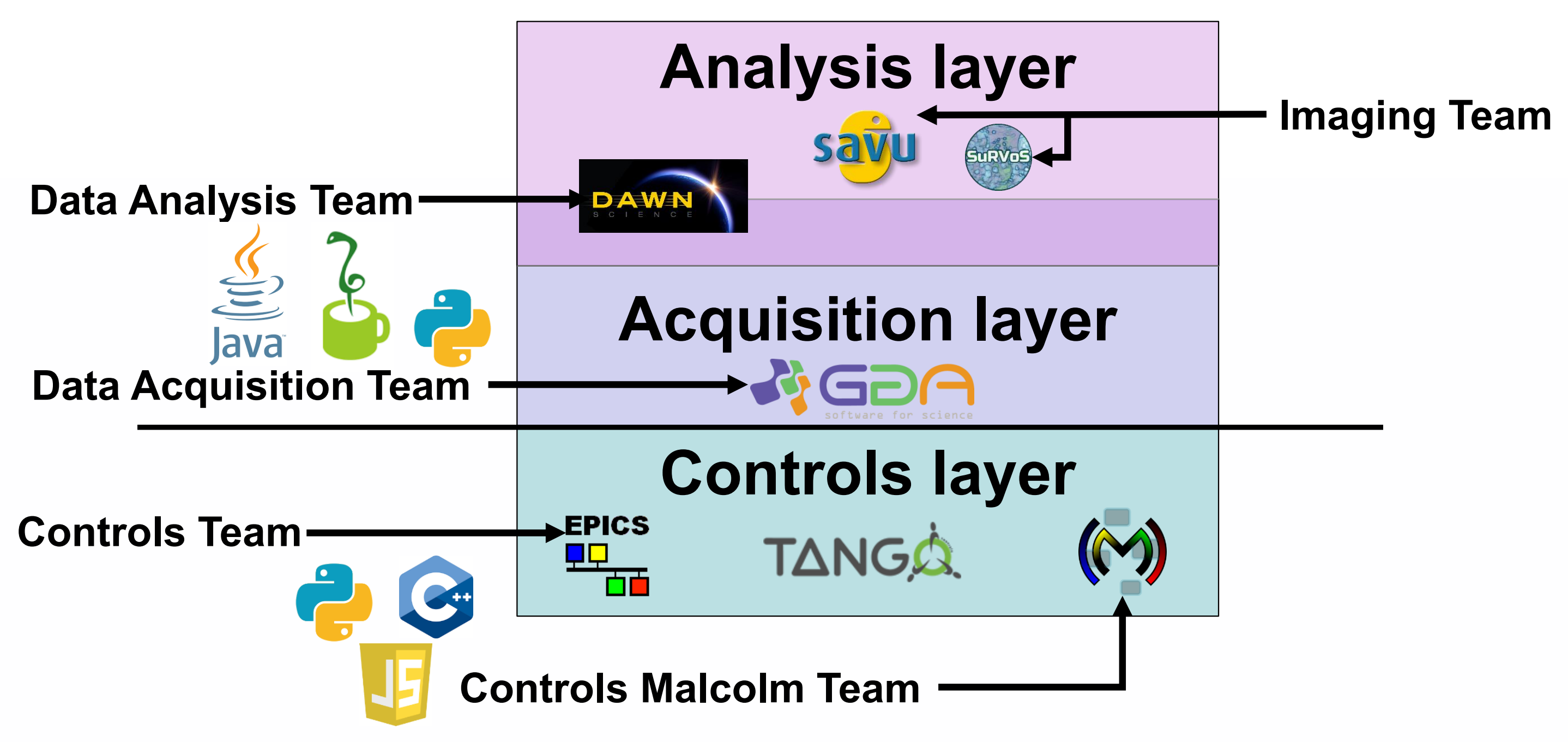
CURRENT SOFTWARE STACK LIMITATIONS

Diamond has a well-established stack of applications developed over many years which provides users with Data Acquisition and Analysis functionality from the Controls Hardware interface right up to the live and offline post processing and visualisation of experimental data. However the stack has the following limitations:

- Controls, Acquisition and Analysis layers tend to have been developed by different teams, in some cases leading to hard boundaries of technological and operational knowledge.
- Generic Data Acquisition (GDA) software has grown up organically over a period of 15 years leading to bad structure and other forms of technical debt making it hard to maintain and difficult to develop.

To support Diamond II, we want to review our software with a view to designing a revised platform architecture to address these and other problems and to take advantage of industry best practises and technologies. Our goal is to:

- Repackage the existing proven functionality in a more flexible structure behind a common platform API.
- Revise some existing implementations, adding new capabilities and features along the way.
- Migrate to a stable consistent platform that is easier to maintain and support.
- Move toward a solution that offers more flexibility to cross the old boundaries to get to the information required by the user.

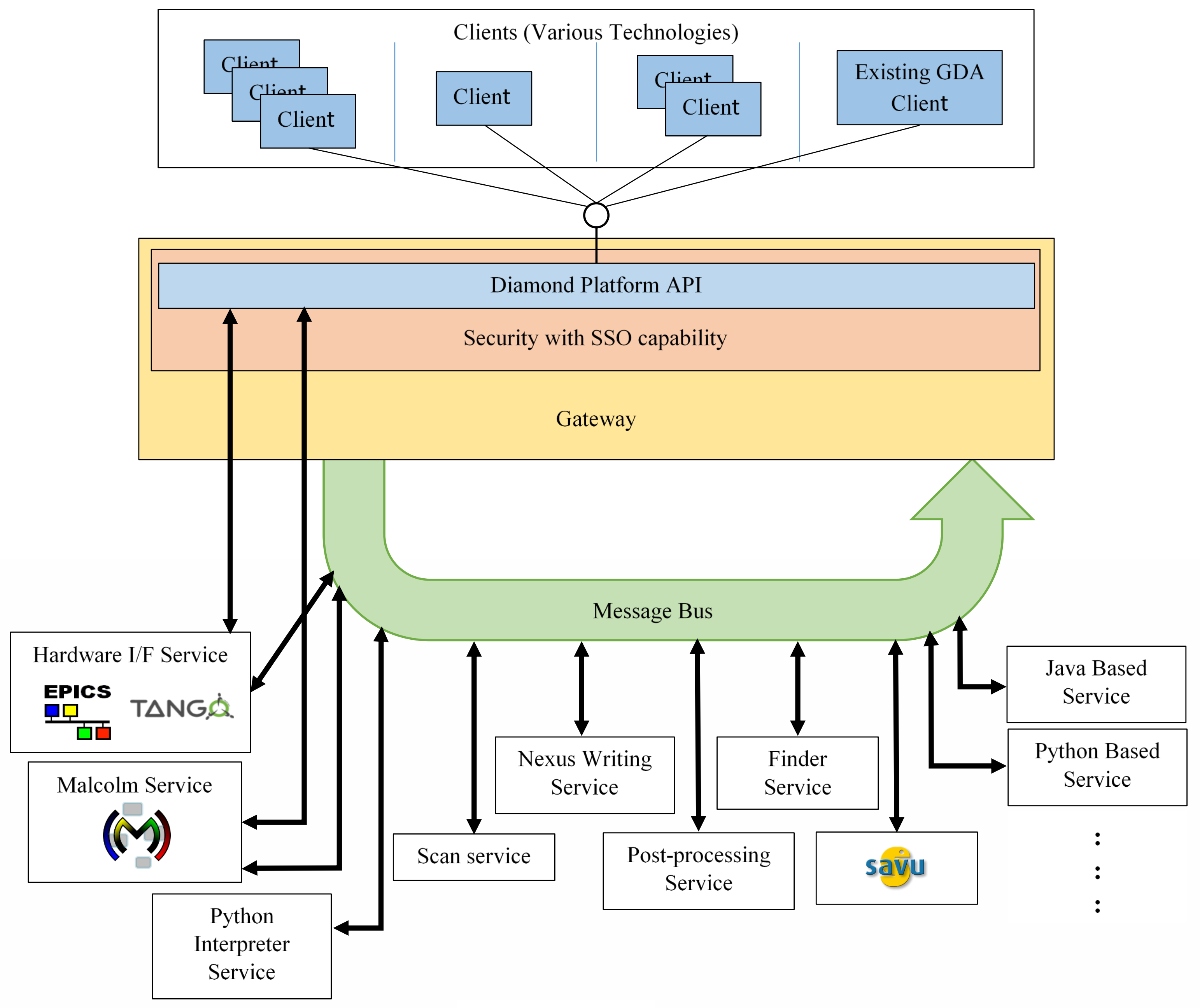


APPROACH UNDER INVESTIGATION

Message based Micro-service Architecture has become ubiquitous in the last decade and is designed to address the sort of problems common in large monolithic applications like GDA. This pattern also provides the capability to implement functionality in the language most appropriate to the task and have it communicate with the rest of the system. Due to its popularity the challenges for its adoption are very well documented and there is a rich tooling ecosystem to integrate it with the API layer already set up for security, containerization etc.

Adoption of a platform wide API using a single technology across the old team functionality boundaries could allow navigation through various layers to facilitate issue diagnosis. To do this the API would need to support:

- Discoverability, i.e. the ability to interrogate the current returned object to determine what lower level calls could be made.
- The ability to communicate using protocols already in use for timing critical direct access to components like PandA and Malcolm.
- Reactivity to allow Observer pattern style feedback to clients at any level within the system minimising polling and simplifying clients.
- Multiple language implementation so that specific functionality could be implemented using the most suitable language without affecting the client's interaction with the API.



The task is to examine the frameworks and technologies that could support this kind of design to select candidates for the various functional blocks. We can then try to produce demonstrator(s) using these technologies along with mock services that behave in similar ways to the real components that would eventually be implemented. This will allow us to discover the issues and test against the key requirements for the architecture as well as examine the hoped for benefits. Once this is done we should be in a position to decide whether to commit to this approach.

CANDIDATE TECHNOLOGIES AND FRAMEWORKS

API

Due to our requirements of discoverability and reactivity ideally with WebSockets compatibility there are relatively few established technologies to choose from. Facebook's GraphQL is a strong candidate though which has obviously been tested and debugged at scale in various languages and is well supported.

Gateway and Security Controller

Requirements for API hosting, Authentication and Authorisation are ubiquitous in Web based applications which we can take advantage of. This is true across several languages and many can provide Single Sign On support with little developer effort. Because most user interaction will take place at the Acquisition and Analysis level, a Java based solution is likely and Spring Cloud is a very strong contender in this area. It meets these requirements and is very well supported and documented. Despite this we will look for other contenders in both Java and Python.

Messaging Backbone

Many options are available here, Active MQ is already in use for similar purposes in Diamond and offers strong routing functionality. Also under consideration is Kafka which gives the ability to replay past messages in the event of a failure and is optimised for streaming which may be important at the Data Analysis level for post processing tasks. Both these and others are again well supported and debugged.

Reactivity

Both Python and Java have ReactiveX implementations that offer the required functionality.

For more information please visit www.diamond.ac.uk or contact Keith Ralphs at keith.ralphs@diamond.ac.uk