How MRI as a Micro-Service Architecture can Accelerate Research and Commercialization

Stewart Bright[1], Philip J Beatty[2], David Kennedy[2], and Andrea F Vargas[3]

[1]Engineering, Synaptive Medical, Toronto, ON, Canada, [2]Research and Development, Synaptive Medical, Toronto, ON, Canada, [3]System Test, Synaptive Medical, Toronto, ON, Canada

Synopsis

We have developed an MRI system having a modern micro-service style distributed software architecture, with the intent of building a flexible, sustainable platform. To describe the architecture, we provide two demonstrations: 1) fully scripted, perform a phantom experiment, extract the images, and upload them to a cloud server for analysis, and 2) update a protocol, append to the queue of scans to be acquired at the simplified operator console, all remotely from another user interface. Through these demonstrations and discussion of its current uses, we aim to illustrate how it could help accelerate collaborative research and commercialization efforts.

Introduction

We have developed an MRI system having a modern micro-service style distributed software architecture, with the intent of building a flexible, sustainable platform. The architecture is extensible, simplifying integration of new services developed internally, and potentially by independent parties. A micro-service architecture allows system behaviour to be specialized by substituting services, without having to build a new system from the ground up. In addition, micro-services offer flexibility in terms of where (what computer) services run. Advantages specific to MRI have been discussed[1], but we are unaware of another MRI system built with such an architecture.We believe that the full potential of this architecture remains unimagined, and although we have already built a great deal around it, we have only scratched the surface of possibilities. Here, we present the architecture to the community via two working examples that illustrate its utility in a research context. We also discuss how it has helped accelerate our own research and development, and how it might enable others to do the same.

Methods

Both demonstrations use our MRI software system, automatically deployed from a build server to the computers attached to the target scanner.

Demonstration 1

We have an automation framework, called Oto, that provides a JavaScript SDK for writing code against the micro-service APIs. Oto is used primarily for testing, though it has many other potential uses.

We will write an Oto script to perform an experiment where the script iterates through a set of T1 target values. For each T1 target value, a T2 spin echo FLAIR pulse sequence with inversion time targeted to null signal from species with the target T1 value will be run on the scanner.

After the experiment completes, the script will extract the DICOM images directly from the system and upload them to a cloud server for sharing or further analysis.

Demonstration 2

We have decoupled the expert user interface for protocol configuration from the acquisition console, which is a push-button design intended for non-expert operators. The configuration interface is a web page served by each MRI system, that can be made accessible externally. For safety reasons, the console is only accessible from the workstation at the scanner.

To facilitate rapid protocol development, an expert user can make changes to protocols and append individual scans to the queue to be acquired at the operator console, all from the configuration interface.

A practical application of this is a physicist sitting at his desk, whose student asks for help acquiring phantom images at the operator console.

Both demonstrations will be illustrated through figures that show how each goal is accomplished via the architecture.

Results

Demonstration 1

On our working system, the Oto script successfully acquires the phantom study, extracts the DICOM images from the Blob Store service, then uploads them to the cloud.

Figure 1 is the Oto script. Figure 2 shows the services with which the script interacts, simplified by excluding most of the scanning services. Figure 3 is a simplified view of the scanning subsystem to complete the picture of the relevant parts of the architecture. Figure 4 is a screenshot of the phantom and a representative image in the cloud repository's embedded viewer.

Demonstration 2

After saving the protocol and appending to the scan queue, the student can finish acquiring the phantom study, including the scan that uses the remotely modified protocol.

Figure 5 shows how the micro-service design enables the physicist to modify and save a protocol, then by clicking "Add to Current Study" in the configuration interface, append a scan to the active study via the Scan Workflow service. The Scan Workflow service notifies subscribers - the console interface, in this case - and in response to the notification, it updates its visual representation of the scan queue.

Discussion

The choice to build an MRI system with a micro-service software architecture has already paid tremendous dividends. Some examples of what we regularly do with the system are:

- Auto-deploy the system to the cloud, with a few key services replaced, and run continuous integration tests of the majority of the system, without a physical scanner
- Quickly install or replace Scan Applications, Pulse Sequence Generators, and Reconstructors on a running system with no other changes, for troubleshooting purposes
- Retrospectively capture raw data for a previous scan in order to perform offline analysis

All of this contributes to our ability to perform rapid, iterative research and development, while simultaneously providing confidence in our changes and software quality.

Conclusion

We believe others can leverage the system in much the same way we do, accelerating important research and the path to commercialization, which ultimately translates to quicker delivery of much needed advancements in patient care.

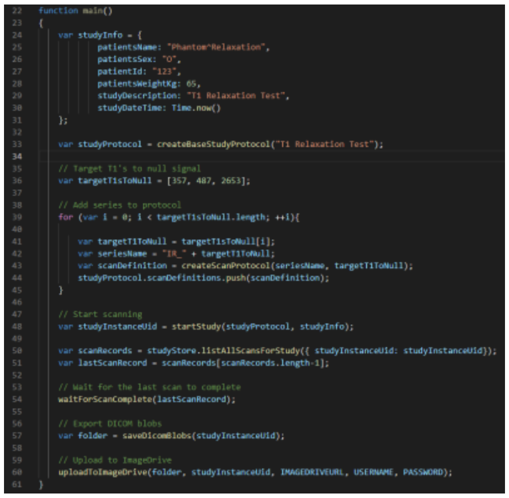
Acknowledgements

Synaptive Medical for allowing this work to be presented.

References

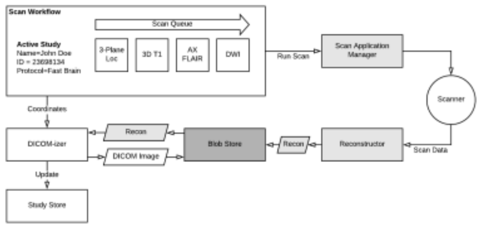
1. Block K, Challenging the Assumptions of MRI: Data Platforms and Architecture. ISMRM 2018 Plenary Sessions. 19 June 2018. https://www.ismrm.org/18/program_files/P02.htm

Figures

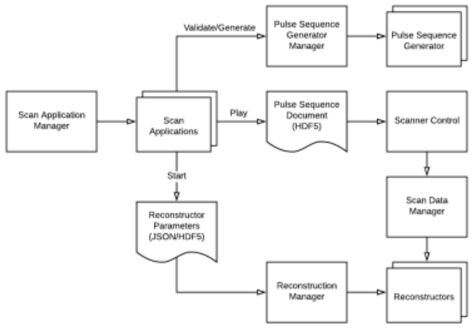


```
function main()
{
    var studyInfo = {
            patientsName: "Phantom^Relaxation",
            patientsSex: "O",
            patientId: "123",
            patientsWeightKg: 65,
            studyDescription: "T1 Relaxation Test",
            studyDateTime: Time.now()
    };

    var studyProtocol = createBaseStudyProtocol("T1 Relaxation Test");

    // Target T1's to null signal
    var targetT1sToNull = [357, 487, 2653];

    // Add series to protocol
    for (var i = 0; i < targetT1sToNull.length; ++i){

        var targetT1ToNull = targetT1sToNull[i];
        var seriesName = "IR_" + targetT1ToNull;
        var scanDefinition = createScanProtocol(seriesName, targetT1ToNull);
        studyProtocol.scanDefinitions.push(scanDefinition);
    }

    // Start scanning
    var studyInstanceUid = startStudy(studyProtocol, studyInfo);

    var scanRecords = studyStore.listAllScansForStudy({ studyInstanceUid: studyInstanceUid});
    var lastScanRecord = scanRecords[scanRecords.length-1];

    // Wait for the last scan to complete
    waitForScanComplete(lastScanRecord);

    // Export DICOM blobs
    var folder = saveDicomBlobs(studyInstanceUid);

    // Upload to ImageDrive
    uploadToImageDrive(folder, studyInstanceUid, IMAGEDRIVEURL, USERNAME, PASSWORD);
}
```

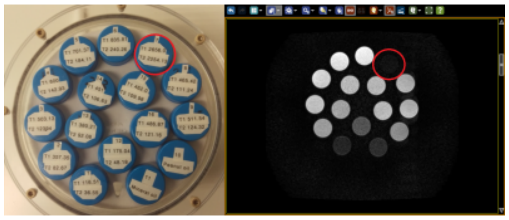
Code written in JavaScript uses the Oto SDK to to perform an experiment by iterating through a set of T1 target values. For each T1 target, a T2 spin echo FLAIR pulse sequence with inversion time targeted to null signal from species with the target T1 value is run on the scanner. After completion, the Blob IDs of the DICOM images are retrieved from the Study Store, the images are downloaded from the (binary) Blob Store, and finally uploaded to a cloud server for sharing, archival, and further analysis. Because of the length of the script, only the main function is shown.



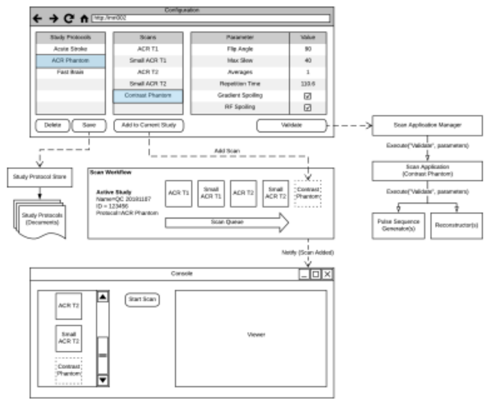
The Scan Workflow service manages execution of a queue of scans that comprise the imaging study being acquired. The Scan Application Manager coordinates pulse sequence generation, scanning and reconstruction. Reconstruction output is stored in the (binary) Blob Store, and is input to DICOM conversion. DICOM conversion of reconstruction output is coordinated by Scan Workflow, but the DICOM-izer service is responsible for performing DICOM conversion and updating series records in the Study Store with Blob IDs.



A Scan Application is an executable program that can combine arbitrary pre-scanning and calibration activities, generate and execute multiple pulse sequences, and run any number of reconstruction algorithms to produce image outputs. Scan Applications, Pulse Sequence Generators and Reconstructors (also executable programs), all written in common languages like Python and C++, can be independently installed or updated on a running system. Scanner Control plays out generated pulse sequences on the scanner. Real-time scanning is not currently available, but will be added in future.



The phantom used in the experiment, next to an image with a target T1 of 2653, which is a screen capture from the embedded DICOM viewer in our cloud repository.



At the request of the non-expert operator sitting at the console (bottom), the expert user modifies the parameters of the "Contrast Phantom" protocol from the configuration interface (top), validates it via the Scan Application Manager, saves it to the Study Protocol Store, and clicks "Add to Current Study". The operator, instead of having to find the saved protocol and add it manually, watches it automatically appear in the console at the end of the scan queue. This is possible because the Scan Workflow service is managing the queue of scans.