BBN IMMoRTALS Phase 1 Challenge Problem Announcement

Navigate Space

Expand all Collapse all

Platform Overview

Tactical situation awareness is the domain chosen by IMMoRTALS. The IMMoRTALS platform is a collection of hardware and software that supports domain operation, e.g., a tactical SA missions. The IMMoRTALS platform consists of Android based clients and a Marti server that facilitates interaction among the Android clients. The clien produces a self-location message periodically, and subscribes to position reports sent by other clients. The server's most basic functionality takes position reports and forwards them to all connected clients. Variations on the client and server include sending other data besides position reports, such as images or other sensor data.As such, the development/build view of the baseline (i.e., without any IMMoRTALS technology) platform consists of the client ar server code base, and build and test tools (Fig 1(a)), and the runtime view of the baseline consists of the Android client(s) connecting to the Marti server over a network (Fig 1(b)).

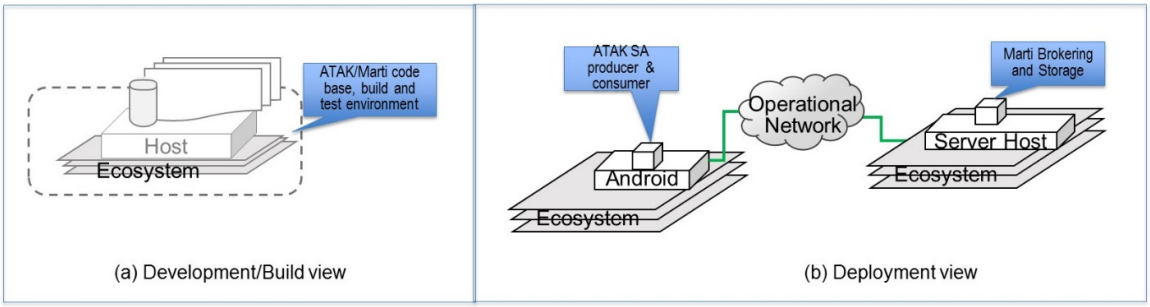


Figure 1

The ecosystem includes "resources" that the SA applications – the Android client app and the Linux server side application- rely on to function, including both SW and HW resources. That is the ecosystem includes systems resources such as CPU/memory/network, software elements such as libraries and packages, as well as other services that must be running.

The IMMoRTALS DAS is the baseline platform + all our technologies to detect and respond to changes + an interface/instrumentation to inject change and observe our response (see Fig 2 below).

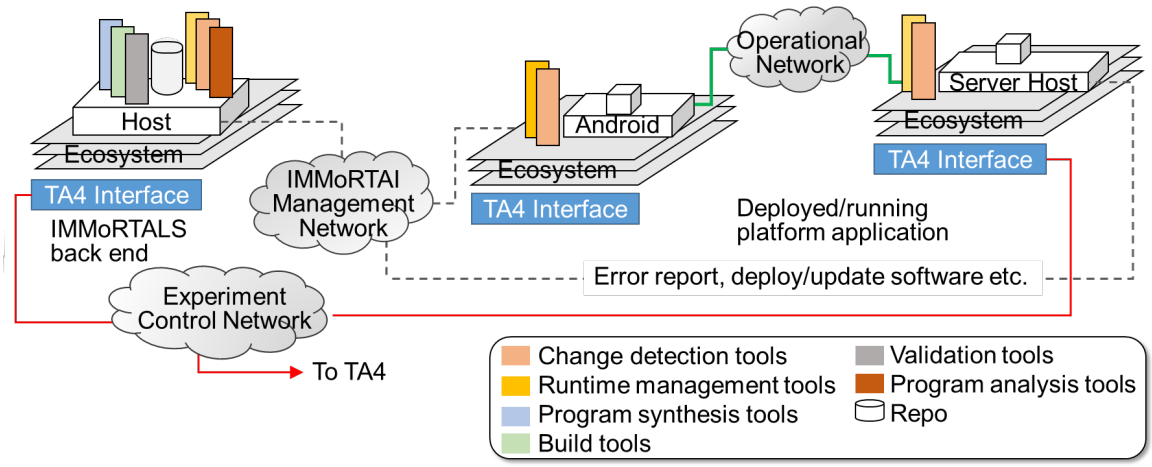


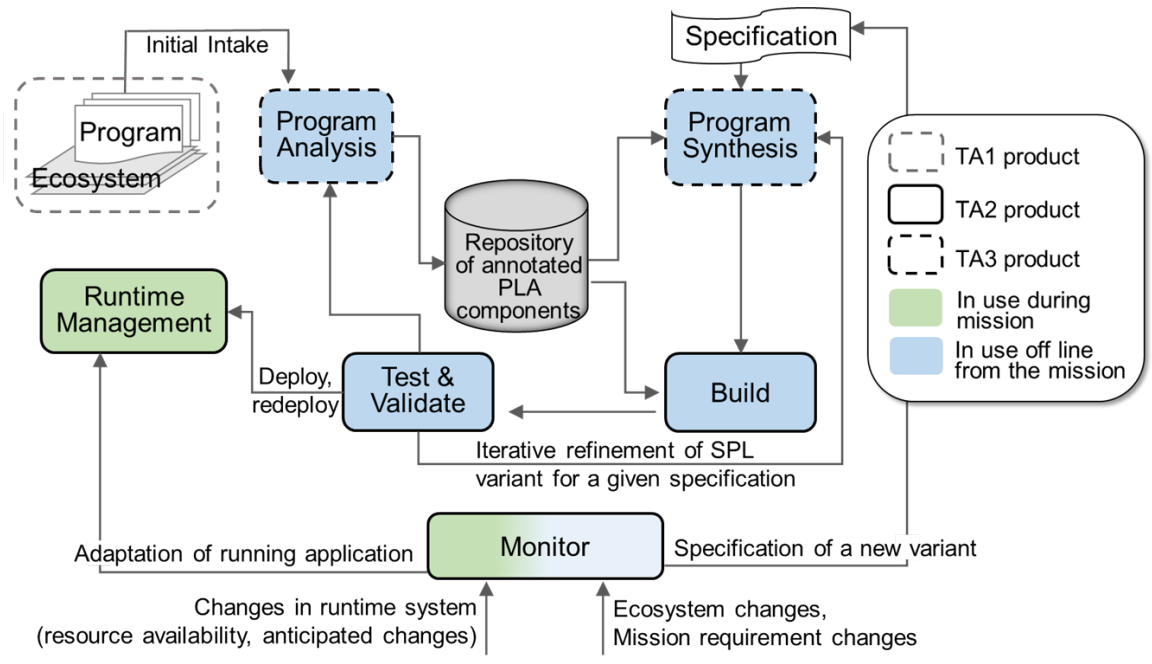
Figure 2

As the legends in the figure above shows, IMMoRTALS technology, in the form of software tools and services, will be used in both the development platform and the deployment platform to support effective and efficient resource-aware program adaptation.

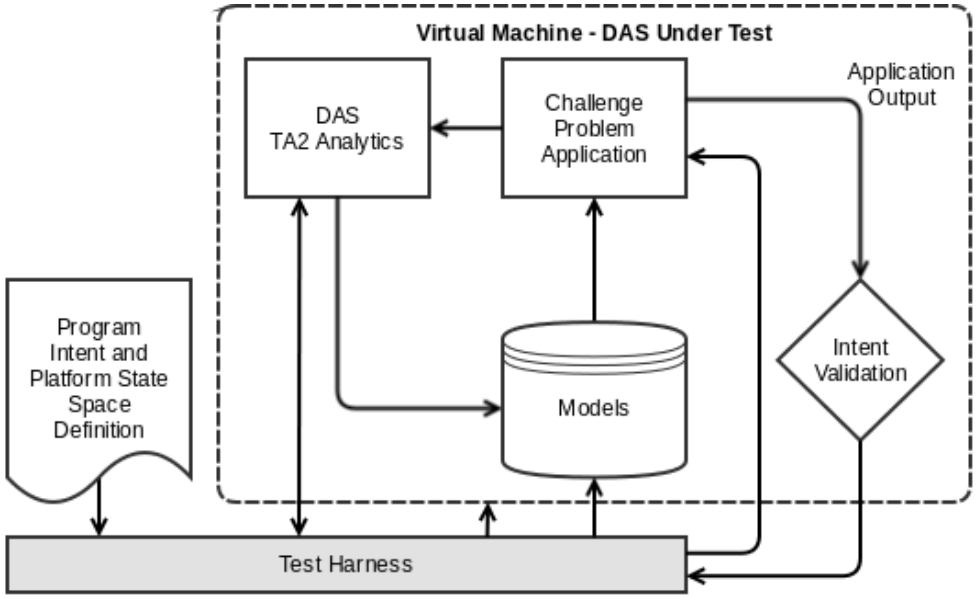
The Fig 3 below shows the end to end operation of the IMMoRTALS DAS covering both the development platform and the deployment platform. "Offline" changes—e.g., changes in mission requirement, a new library update are mostly handled in the development side. Changes in systems systems resources can be both "online"/runtime as well as "offlne". For example, CPU, memory or network resources may diminish in the deployment environment du to failures, load, attacks, contention with other applications etc. At the same time, the build platform can be used to adapt an existing version of the application for a new deployment environment with different systems resource profile.

The "repository" element is a logical entity, but our current implementation plan is that a SPARQL server will be at the core.

Our intent is to have the end-to-end "offline" workflow in place for the initial evaluation.



Platform for Evaluation



IMMoRTALS DAS generally fits into the above structure—except that it may not be a single VM.

The runtime components of the IMMoRTALS platform (i.e., the Android client and the Marti server) will be on their own VM. A separate VM (Linux Router?) may be needed to emulate the operational network and offer the TA4 interface to effect changes.

The models, analytics and the discovery machinery will be in the build environment/development platform. Some runtime change detection and runtime managers may be instrumented into the runtime VMs (i.e., the VMs hosting the Android clients and the Marti server).

Evaluation Note:

Given a particular deployment scenario, combined with a set of code artifacts, IMMORTALS will produce one of tw possible answers:

- A solution is possible: given either the baseline application with no changes, or a modified application base on the code available in the repository, it is possible to build an application that will fit in the specified constraints.
- A solution is infeasible: the resource constraints are too tight or we don't have the correct pieces in our code repository to build an application that will fit in the desired constraints/requirements. With this result, there a several possible ways to move forward:
    - Add new code to the repository to fill in the missing gaps, then re-run the analysis.
    - Analyze the resource requirements vs. resource availability to guide development of new code.

Challenge Problems

Challenge Problem 1: Mission Requirements Regarding Location Collection

Overview

Tactical situational awareness spans a large variety of missions. Such missions might have very different requirements for location collection. Some deployments may be satisfied with the GPS built-in to a given Android device. Other deployments might require GPS providers of a very specific nature (e.g. SAASM). Still other deployments might have to assume a *GPS-denied* environment, where the only options are dead-reckoning and potentially inertial sensors.

The accuracy and trustworthiness of the source of a location report come into play in a variety of ways from a mission requirements perspective. For instance, some use-cases by get by with only lat/lon values. Other mechanisms, such as dead-reckoning, can be augmented with a database of elevation data (DTED), at the cost of significant local disk space. Some of the potential mission requirements imply specific hardware being available. Even then, there can be multiple ways to get access to that hardware, for instance wired/USB vs. Bluetooth for accessing an external GPS.

This is primarily an "offline" adaptation scenario in its initial incarnation, although later iterations of this CP could easily be extended to have some “online” aspects. For instance, a mission requirement could be that the applicatio use GPS when available, but be able to dead-reckon if there is no GPS. A further evolution might specify all possib location sources be built-in, and a preference-ordering is supplied as a mission requirement.

This CP can be used to validate the system in a variety of ways. Consider scenarios where there is no code in our repository that can satisfy the requirements versus scenarios where such code does exist:

Let us assume that the *baseline* version of the product is built to use GPS location.

- (offline) A mission requirement to use a different location provider comes in, IMMoRTALS *finds* a location provider that satisfies the mission need in the existing ecosystem
- (offline) A mission requirement to use a different location comes in, but there is no location provider that satisfies the mission need -- may be the semantics do not match, maybe the resource profile of the available provider does not match with what is available in the deployment target
- (online) GPS signals became unavailable during mission, but the deployed program or the deployment environment includes an alternate provider and the system adapts to use the alternate provider
- (online) GPS signal became unavailable during mission, but the deployment set up did not include any alternatives to fall back on, a failure report is provided to IMMoRTALS
    - At this point, this could be treated as the beginning of an "offline" change handling scenario

Note that adapting to offline changes may not be fully automated, IMMoRTALS will likely provide a set of change directives that a human programmer will have to encode. In later phases we will look into automating this step usin program rewriting.

System Perturbations

There are two system perturbations that this challenge problem exposes. First is the mission requirement for a give source of location data. Second is the execution environment, which will dictate availability of specific hardware or external services (e.g. GPS satellites).

Mission requirements

We envision a set of possible mission requirement-based location collection techniques such as

- Dead-reckoning
- Inertial tracking
- Consumer-grade GPS
- Commercial-grade GPS
- SAASM GPS

- DTED (elevation only; assumes another mechanism to find lat/lon)

A given deployment might specify that some options are required. We anticipate this will be input to the DAS as a checkbox or property file (see evaluation API section below).

Execution environment

The set of possible execution environment properties might include:

- GPS satellite availability
- GPS chipset type (e.g. SAASM)
- External GPS connectivity (wired/Bluetooth)
- Availability of sensors appropriate for inertial tracking

These could be set over the TA4 interface the IMMoTALS DAS will provide.

Code Availability

The code that the DAS has available in the repository will have an effect on the results. Submitting a mission requirement for a SAASM GPS might return a working application or declare it infeasible, depending on whether there is code for collecting SAASM GPS data in the repository. So the contents of the DAS code repository might be another system perturbation.

Software Requirements

Android SDK and build environment

Java JDK 1.8

Evaluation API

The mechanisms for perturbing the system for this problem consist of constructing a new set of mission requirements or execution environment descriptions. We will provide a set of properties and their possible values in each of those categories. We will also provide a pre-populated repository of code fragments.

Having actual SAASM hardware as part of the evaluation is infeasible, as well as strictly controlling GPS availability So we will provide drivers that simulate some of the notional code fragments described above. An evaluator will be able to tell both by looking at the code for the final application (e.g. the application instantiates SaasmGpsLocationProvider), and by some markers we will put in the output data, which location provider was chosen.

Challenge Problem 2: Handling the impact of changes in deployment scale and other factors that impact resource consumption

Overview

Tactical SA can vary in scope from small-squad (<10 participants) missions to major National Security Events (>50 participants). Along with the number of participants, other variables might refer to expectations of the mission, such as report rate or additional data beyond basic Position Location Information (PLI) (e.g. pictures). All these factors have an impact on the resource usage of every node in the system, but in most cases the limiting factor is bandwidth. In many cases the cause for limited bandwidth is purely technological, but in other cases bandwidth may have outsized cost (e.g. SATCOM) such that users want to voluntarily limit usage even though the technology would support more bandwidth.

The goal of this challenge problem is to see whether IMMoRTALS can evolve the client and server applications in response to such variations. Like CP1, this CP is also a family of challenge problems. We outline a number of possibilities, and indicate what we propose to achieve in Phase 1.

- The change request comes in, the code to use exist in the current ecosystem or the current system is amenable to changing/tuning the contributing parameter
    - Obvious boundary cases—the deployment environment can only accommodate a certain scale. For example, for a given bandwidth limit, it is easy to set number of client, rate and size in such a way tha no solution can exist—IMMoRTALS should indicate these impossibilities
- Same as above except the code to use is not available, and there is no way to tune the offending parameter
    - The goal here is to eventually generate new code that fits and adapt existing code to allow parameter tuning. Initially, IMMoRTALS will just prescribe the changes. We will try automated handling of this case in later program phases
- Online: rate or number of clients changed during mission and the deployed system is able to accommodate the changes
    - Same caveat about boundary cases—it will be easy to set parameters that the deployment environment cannot sustain

System Perturbations

There are several system perturbations that this challenge problem exposes. The resource constraint that is most prominent is bandwidth. As mentioned in the overview, a bandwidth limit might be hard or soft.

Mission requirements

We envision a set of possible mission requirement variables

- Number of clients
- PLI Report Rate
- Image Report Rate
- Image property requirements (e.g. resolution, color vs. grayscale)

Some deployments may not have a requirement for images. The PLI report rate and number of clients will be in every scenario description

Execution environment

The most critical aspect of the execution environment for this CP is bandwidth availability. For the first iteration, we will be considering a simplified network model where we only consider the "most constrained link". This is sufficient to describe most deployments, especially where the actual network topology may be opaque (e.g. commercial cellular). Later iteration may expand on the network definition to allow more complex topology descriptions.

Code Availability

As in the first CP, availability of code in our repository to satisfy certain functional requirements is something that could be adjusted to get different results from our system.

Software Requirements

Android SDK and build environment

Java JDK 1.8

Evaluation API

The mechanisms for perturbing the system for this problem consist of constructing a new set of mission requirements and execution environment descriptions. The *number of clients*, *PLI report rate,* and *Image Report Rate* are integer values that can be set to any positive value. Likewise the bandwidth availability will be given as ar integer value (likely kbps). We will provide a set of image properties and their possible values. We will also provide pre-populated repository of code fragments.

The evaluation system must be able to limit bandwidth for at least one link (typically will be the link to/from the server). Multiple android "clients" will need to be deployed according to the mission plan.

Platform Integration & Test

Software Requirements

- *What is the anticipated platform processing architecture (i.e. multi-core x86, GPU, ARM, etc.)?*
  - *We anticipate a multi-core x86_64 machine for the server*
  - *We have developed the Android clients to work on ARM, but we will investigate the possibility of running on Android x86 if needed.*
- *What OS will the DAS and CP utilize (please specify both type and version (i.e. Ubuntu 14.04, Windows 95, etc.)?*
  - *We assume a recent linux distribution (no specifics as of yet) for the server*
  - *Android version 5 or greater*
- *Any specific packages or code that the platform will rely on (this doesn't have to include common packages like gcc or Python but should call out any specific (i.e. Robot Operating System version 2.x, Android Emulator, etc.) or supporting packages that your team is planning on developing and how Lincoln can obtain these)?*
  - *TBD*
- *Any specific interfaces or communication protocols that your system will rely on (i.e. multicast network capability, etc.), particularly for interaction with the TA4 test platform*
  - *IP networking*
    - *We have avoided multicast in this iteration because there were known issues with the evaluation platform.*