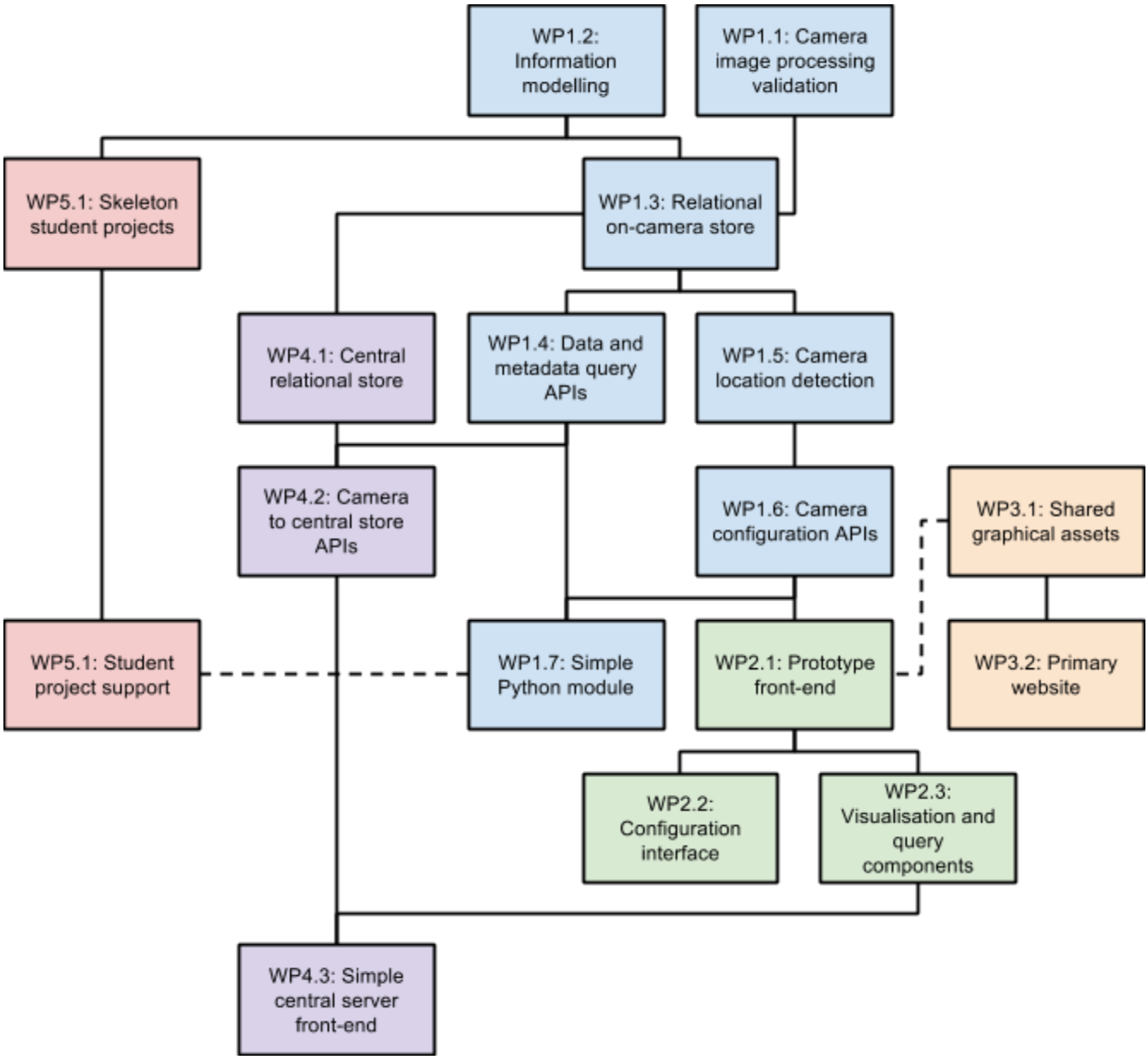
Plan for basic minimal functionality

Tom Oinn, tomoinn@crypticsquid.com, 1/2/2015 - work in progress

This document is split into phases - these are not necessarily chronological, as the diagram below shows:



At the completion of the work described here we will have a basic, functional system. It won't be particularly smart - there's no data analysis on the central server and visualisation of events will be simplistic. The hope is that we get to this point well before the end of June, it

acts as a fallback position such that we have working meteor detection, an API, a website, teaching material etc.

At the moment we don't have resources / time boxes for any of this, I'm hoping to work those out with Dominic when we meet up this Wednesday.

Table of Contents

- Plan for basic minimal functionality
- Table of Contents
  - Phase 1
    - WP1.1 - Camera image processing validation
    - WP1.2 - Information modelling
    - WP1.3 - Relational on-camera store
    - WP1.4 - Data and metadata query APIs
    - WP1.5 - Camera location detection
    - WP1.6 - Camera configuration APIs
    - WP1.7 - Simple Python module
  - Phase 2
    - WP2.1 - Prototype front-end
    - WP2.2 - Configuration interface
    - WP2.3 - Visualisation and query components
  - Phase 3
    - WP3.1 - Shared graphical assets
    - WP3.2 - Primary website
  - Phase 4
    - WP4.1 - Central relational store
    - WP4.2 - Camera to central store API
    - WP4.3 - Simple central server front-end
  - Phase 5
    - WP5.1 - Skeleton student projects
    - WP5.2 - Student project support

Phase 1

This phase of the project is intended to get a bare minimum of end-to-end functionality, using a single camera installation without any central server. Its end point is a working camera unit, capable of locating and orienting itself through GPS and astrometrics, and of logging detected events to an internal data store. It presents a simple REST based API to the outside world allowing for querying and data retrieval, but without any more user friendly web interface. The camera can have metadata properties set through this same API, including manual correction of location information if required. Finally, a simple Python module wraps the HTTP API and allows for programmatic access to the camera - this is expected to be refined in subsequent phases.

WP1.1 - Camera image processing validation

- Assemble enough of the camera hardware to validate that images can be captured from the camera module and processed by the main board, and that events can be detected.

WP1.2 - Information modelling

- Capture the information model for:
    - A single detected event
    - A camera installation
    - A basic report on camera status
- The objective is to have sufficient information to create APIs transporting these types of data.

WP1.3 - Relational on-camera store

- An implementation of the parts of the information model sufficient to capture observations from the camera, probably as a relational database.
    - This will form the basis for the central server's relational store, so should be designed with this in mind.
- A bridge between the the output of WP1 and this relational store allowing for observed events to be stored.

WP1.4 - Data and metadata query APIs

- HTTP REST (Representational State Transfer) APIs sufficient to allow the querying of:
    - Metadata - to which installations does the API provide access, and what are their capabilities, location etc. Use a traversal mechanism to allow a browse pattern for nested capabilities.
    - Data - return observed events and general observations along with any computed information (i.e. correlated event IDs)

        - Enable search by time range and lat / long bounding box for camera location. Allow for more sophisticated searches in the future.
- The API, rather like the RDB schema in WP1.3, should function both for single installations and for a multi-installation central server, i.e. even though we only have one camera on a single site service we still expose a camera discovery metadata API.

WP1.5 - Camera location detection

- Use a combination of GPS (where available) and astrometrics to determine camera location and orientation. Write this information into the on-camera relational store, allowing for time-variant data (i.e. use 'valid from' and 'valid until' timestamps on entries).

WP1.6 - Camera configuration APIs

- The minimal set of REST APIs required to read and write configuration information from and to the camera, including basic properties defined in the information model for the camera installation.
    - Must be secured for write access, at least minimally, with an access key - we can define this from a file on the camera itself.

WP1.7 - Simple Python module

- A Python client module which presents a more user-friendly view over the query and configuration APIs. We expect this module to be progressively refined through subsequent phases in response to the likely user requirements.

Phase 2

This phase is focused on user interaction with a single camera outside of the programmatic API. It therefore consists primarily of a web interface, hosted on the camera itself but running largely as a rich client UI within the user's browser.

While everything at this point is a single camera, this phase forms the basis for the interactive UI which will run on the central server; whether to consider this up-front or to allow for refactoring at a later stage is a decision left to the developer or developers concerned.

At the end of this phase, we should be able to take the camera developed in phase 1 and interact with it through a simple web interface.

WP2.1 - Prototype front-end

- Modify the existing server (used for the API) on the camera to serve a simple home web page. This web page should make the appropriate REST calls back to its own server to display basic (non configurable) information about the camera, such as the ID, and versions of software components. In addition it should:

    - Provide a (brief) explanation of the project and link to the main project website.
    - Allow users to view the current image from the camera - this doesn't have to be a live video, a static image with manual refresh is fine.
- We do not expect cameras to have public IP addresses, access is for those on the same local network (typically within a school or other institution).

WP2.2 - Configuration interface

- A user with the configuration key should be able to access the same properties as those exposed through the simple python module in WP1.7. Users without the configuration key should be able to read these properties - where concepts such as longitude and latitude are referenced the interface should attempt to explain what these are!

WP2.3 - Visualisation and query components

- Provide at least one mechanism to search for, and display, events recorded by this camera. This will make use of the existing REST API to perform querying, and must render the returned information (the exact format of which will be defined as part of the API specification and which will manifest the per-event portion of the information model). At a bare minimum this component should:
    - Provide a form interface for users to specify any or all of:
        - Earliest observation date
        - Latest observation date
        - As we are only managing a single camera there's no requirement at this point to either enumerate and allow filtering by camera installation, or to allow search by installation location.
    - Display the results as a table, showing a summary of the information for each event along with a link to the video or image captured at the time - this link will be provided as part of the results (there's no need to create a hosted video player).

Phase 3

Phase 3 (which, confusingly, doesn't really depend at all on phases 1 and 2) is about the public face of the project. This includes the website, social media presence, blog and graphical assets used throughout.

WP3.1 - Shared graphical assets

- We want to establish a common look and feel across the project, this work package should result in a set of reusable icons, logos, banner images and similar, along with any other user interface guidance (font choices, colour palettes, accessibility advice and similar).

WP3.2 - Primary website

- A main website for the project as a whole. Note that this is not where users will interact with the data produced by the cameras, but is where they will visit to see details of the project aims and progress.
- The main website should be implemented on top of wordpress, this will enable us to have individual accounts to post to the news section, and to retain the ability to have static content through wordpress's 'page' mechanism. The more customisation and identity we can put into this the better.
    - We need a presence of some kind on whatever social networks our target communities use. Wordpress can be set up easily to cross-post to Twitter, Facebook and others, and basic things like adding sharing buttons to the post display should help us here.

Phase 4

Phase 4 introduces the central server, and extends various previous components to make use of it. Cameras push data to it, and the APIs are extended (or their existing extensions used) to allow for query and retrieval by camera ID and location. The corresponding changes are also made to the Python libraries to allow users to code against groups of cameras rather than single ones. At this point no analysis is performed, so we're not doing correlation between events or attempting to calculate trajectories, this is all about making the same information available but for multiple cameras.

WP4.1 - Central relational store

- An extension to the on-camera relational store defined in WP1.3, this may (and hopefully will) be exactly the same database structure, the single camera being a degenerate case of a central store with a single camera ID present.

WP4.2 - Camera to central store API

- To be useful, the camera installations must be able to push data on events, sky state and similar (as defined by the event and status parts of the information model) to the central server. The cameras will do this through an HTTP REST API on the central server defined by this work package.
    - Define and implement the appropriate APIs on the central server itself.
    - Use these APIs to push events from the camera's on-camera store to the central server.

WP4.3 - Simple central server front-end

- Take the existing visualisation, query and status components from WP2.1 and WP2.3, targeting the query portion against the central server's query API, and in addition allowing for:

    - Search for cameras, by location, name etc.
    - Extend event search to include location and camera ID properties
    - Include camera ID in search results - at this point we're still not expecting a sophisticated rendering of the results, we want to see that we're getting data from multiple cameras.

Phase 5

Phase 5 is about the use of the cameras and their data by students (or citizen scientists, for want of a better term). This phase has subtle interdependencies with the various APIs and query specifications. On one hand, the set of possible activities is bounded by the capabilities of the query API, and on the other the query API itself can be informed to an extent by the student activities defined here (once we go beyond very simple query by time and location operations). Because of this interdependency we anticipate an ongoing progressive refinement of the projects and the corresponding query and programmatic APIs. Fundamentally, however, the possible activities depend on the kind of information we can capture.

WP5.1 - Skeleton student projects

- A set of outline projects, including in principle how they will use the information we produce and how they will integrate with the current school curricula.

WP5.2 - Student project support

- For programming projects based on the Python API, we need to provide example code and other assistance to teachers using this project in their classes.