CRISTAL-iSE Project

Deliverable 3.1: Initial Design Document

Deliverable Data:
Due date: Month 24, April 2014
Actual submission date: Month 30, 31 October 2014
Reason for delay: Alignment with the open source launch, which was a later addition to the project.
Organisation name of lead contractor for this deliverable: UWE
Dissemination level: Internal

Authors:
Mr Andrew Branson
Dr Jetendr Shamdasani

Table of Contents:

1. Introduction

This document describes the implemented and planned design changes to the CRISTAL Kernel to accommodate the requirements placed upon it by the new versions of Agilium and in turn CIMAG-RA. It describes the state of the CRISTAL API for the first open-source release of the CRISTAL Kernel - version 3.0, that had been made possible by this project. This open source version is released under the LGPL Version 3.0 license. It is also been renamed now to the CRISTAL-iSE Kernel and is useable by the wider community at large. The reason for the renaming is to show that the Kernel is now different from the original CRISTAL Kernel and that it breaks compatibility. It also shows that it is now an outcome of the CRISTAL-iSE project. The LGLP version 3.0 license was chosen since we decided that most projects will migrate to this at a later date. We also chose the LGPL since it allows the various aspects of CRISTAL-iSE to be used in a closed source commercial setting.

2. Open source release

The following modules were released as open source :

- **The CRISTAL-iSE Kernel** - This is the “core” of CRISTAL-iSE. This is the version 3.0 release and defines the base API on which the rest of the 3.x release line will maintain compatibility with. It contains new interfaces for users to interact with the Kernel, with many new features and meets several of the requirements laid out in D2.2. A full list of which follows in section 3 of this document.

- **The CRISTAL-iSE Swing GUI** - This is an updated version of the Java desktop UI which users can use to interact with the Kernel.

These are all available on GitHub for the public and they are all available to download freely by anyone. The url is : http://cristal-ise.github.io/ all the documentation is on the website.

Also released at this time was the CRISTAL-iSE Dev Application. This is an example CRISTAL-iSE application that can be downloaded in .zip form from the CRISTAL-iSE website. It contains a set of descriptions that can be used to create Item descriptions, and so serves as a tutorial application on CRISTAL-iSE concepts and development. It is accompanied by a written tutorial describing the development of a clinical patient tracking application using this development environment, which can be found on GitHub.

3. Kernel Enhancement Design

This section describes the design and implementation changes in the CRISTAL-iSE Kernel 3.0, and which requirements in the previous deliverable D2.2 have already been met, which will be met in the future, and those which have been rejected.
References in orange correspond to requirements in D2.2, by section.

3.1. API Changes in Version 3.0

The following changes have been implemented in CRISTAL-iSE version 3.0, and can be found on GitHub at https://github.com/cristal-ise/Kernel.

3.1.1 New interfaces and refactoring

Several new interfaces have been added, refactoring dependencies out of the Kernel into separate modules, and enabling more subsystems to be replaced by plugins. The key refactoring in this process is the removal of the dependency on the OpenLDAP directory server, the Novell JLDAP library, and all references to LDAP and LDAP DNs from the Kernel, which allows for alternative directory implementations to be used. The original LDAP implementation has been spun off into its own module, which implements the new interface. This 'cristal-ldap' module has not yet been published to GitHub. At a later date we will decide whether to supply this as a default directory implementation, or use a newer one we have yet to develop.

3.1.1.1 Authenticator (4.2.5)

The Authenticator interface replaces the previous LDAP authentication. Implementations should verify a username and password when supplied, or read them from configuration, returning a boolean indicating whether or not the authentication was successful. They may optionally store an authentication object i.e. an opened connection that may be used by Lookup or ClusterStorage implementations. A disconnect method is defined to close such connections when the process is shut down.

Authenticators are no longer stored in the Gateway itself, clearing up confusion in multi-user environments such as web servers.

3.1.1.2 Lookup (4.2.5)

The new Lookup interface contains a simplified set of methods previously defined by the LDAPLookup object. All LDAP specific dependencies are now removed. The Path objects (Path, ItemPath, DomainPath, RolePath and AgentPath) are now directory independent. They do not store nor manipulate LDAP DNs.

The Lookup only contains directory reading methods, so that read-only implementations may be produced. CRISTAL-iSE client processes use this interface.

3.1.1.3 LookupManager (4.2.5)

The LookupManager extends the Lookup interface to include directory writing methods, so that reading and writing may be separated in implementation. Only servers use the LookupManager methods, and expect the defined Lookup class to be a LookupManager.

3.1.1.4 OutcomeInitiator (4.2.7)

Implementations of this interface are used to create initial forms of Outcomes during Job execution. Activities can specify a property called ‘OutcomeInit’ which contains a reference string. The Job then looks in the configuration for a property called ‘OutcomeInit.<ref>’ which should contain either an instantiated OutcomeInitiator or a class name to instantiate. The OutcomeInitiator is called when `Job.getOutcome()` is called, but the outcome is empty.

This interface will be used to re-implement the Prefill functionality of Agilium in a much more efficient way than before.

3.1.1.5 ShutdownHandler

Another dependency of the Kernel that was factored out was the daemon. The is the usage of the Tanukisoft Java Service Wrapper (JSW) for daemonization of the CRISTAL-iSE server. The JSW is licensed commercially, though there is a GPL2 community edition. This choice restricts use of the Kernel if it were included, so it has been refactored from the Kernel into its own module.

Process shutdown used to detect JSW usage and call the appropriate stop() method if it was being used. Now, a setShutdownHandler method in AbstractMain allows launcher classes to specify how the process should be shut down. The default behaviour when no ShutdownHandler has been specified is System.exit which is a clean exit based on the operating system.

3.1.1.6 ResourceImportHandler

During module import, the five core description resource types can be imported as `<Resource>` entries which synchronize an Item’s outcome with the module contents. Previously, any domain specific resource or extension of the core resources had to be be imported with a verbose `<Item>` entry. The ResourceImportHandler interface allows custom resources to be handled. They produce Outcomes and DomainPaths which the Bootstrap process uses for sync. The generation of the core five processes has been moved into a default ResourceImportHandler, which is used when a custom one is not defined for a resource type.

3.1.2. Item Changes

There have been several fundamental changes to the design of the Item in 3.x. These changes make all existing CRISTAL 2.x data incompatible with 3.x ones without development of migration tools, which is not currently foreseen.

3.1.2.1 SystemKey as UUID

The major change in this release is the redesign of the SystemKey which identifies every Item. The SystemKey is now a Class 4 UUID instead of an integer. The system and admin built-in account are fixed to ‘1’ and ‘2’ respectively, but all other SystemKeys are randomly generated using the Java UUID implementation. They may be supplied in module definitions and in the Kernel resource set.

The Java UUID objects cannot be transmitted over CORBA calls, and the string form is inefficient, so a CORBA structure SystemKey has been created to represent it, consisting of two 64 bit integers containing the most and least significant bits respectively. Previously, an integer sequence was stored in the entity root of the directory. This is now removed and the NextKeyManager lookup object has not been migrated to the LookupManager interface.

3.1.2.2 Agents as Items (4.2.9a)
Agent now extends Item, and so has a lifecycle, and can collect Events and Outcomes. They have their own predefined step container, which in addition to the default steps, contain additional Agent specific steps adding password and role management.

As the concept of ‘Entity’ as an abstract superclass of Agent and Item is no longer necessary, it has been mostly removed from the Kernel. For example, EntityPaths are now ItemPaths.

It was originally stated that Roles would also become Items, and act as AgentDescriptions. As design progressed, it seems that the Role is more of an Agent attribute, and the AgentDescription is more complex and needs further analysis. A CreateAgentFromDescription predefined step has been added to the Agent predefined step container, an AgentDescription is simply an Agent that has the same properties as an ItemDescription for the time being. Initial Roles must be supplied as parameters to CreateAgentFromDescription.

3.1.2.3 Stronger description semantics (5)
Collections and Properties can no longer be directly written to Items through the AddC2KObject predefined step. They must be updated through dedicated steps that enforce constraints given by the Item description.

Properties now replicate the ‘mutable’ property from the PropertyDescription. The WriteProperty predefined step will only change the property if it already exists and is mutable. Collections have several steps to manipulate them, detailed later.

3.1.2.4 Roles
Roles are now hierarchical, this means that a Role can now inherit from a its parent. Agents are classes as holding all super-roles, but not sub-roles. Another motivation for making the Role an Item and/or Agent to allow it to maintain a Job list for all Jobs assigned to that Role that have not been assigned to an Agent yet. This was felt to be heavy, especially with regards to the number of JobList subscriptions each Agent would have to maintain. It is

expected that domains that require extensive global joblists will accomplish this through XMLDB querying. (4.2.9b)

3.1.2.5 Item initialization

The initialization method of Items is called from the *CreateItemFromDescription* predefined step and parametrized by description data stored within that Item. It has been redesigned to completely initialize the new Item or Agent in a single atomic transaction, ready for its lifecycle to start. The sequence of initialization is as follows:

- Collection descriptions are all instantiated and stored.
- Property descriptions are instantiated, and a new parameter to *CreateItemFromDescription* can supply initial values for any mutable property in the Item.
- The workflow is instantiated, initialized and stored.
- The whole transaction is committed to storage.

All of this is done in a single ClusterStorage transaction, and an Event is now generated too. In the next minor release, this transaction will be committed before the workflow is started, so any automatic activity that runs straight away can expect the Item to be complete.

3.1.3. New Described State Machine (4.2.1)

The Activity state machine of CRISTAL 2.x had hard-coded states and transitions, which were driven by CMS ECAL requirements, with a late Agilium influence. Most of the states were not used by other domains, and were a poor match for predefined steps and composite activities. This has been replaced in CRISTAL-iSE 3.x with a new described state machine.
Each state machine is defined as a description resource, in the same way as activity definitions, scripts and schemas. They are serialized to XML as other description resources, defined by a Kernel resource 'StateMachine' schema. They are stored under the domain path '`/desc/StateMachine/`'.

Each ActivityDef may use the *'StateMachineName'* and *'StateMachineVersion'* properties to use a custom state machine. Each state is defined with a `<State>` element, which contains:

- State Name & integer ID
- Proceeds: a boolean that indicates whether the state is terminal, meaning the workflow can proceed to the next Activity.

Transitions are defined with the `<Transition>` element, specifying:

- Transition Name & integer ID
- Origin state
- Terminal state
- Whether an Outcome required, possible or not allowed. The name and version of the schema is given.
- Whether a Script should be run, and the name and version of the Script.
- A boolean activity property that enables or disables the transition.

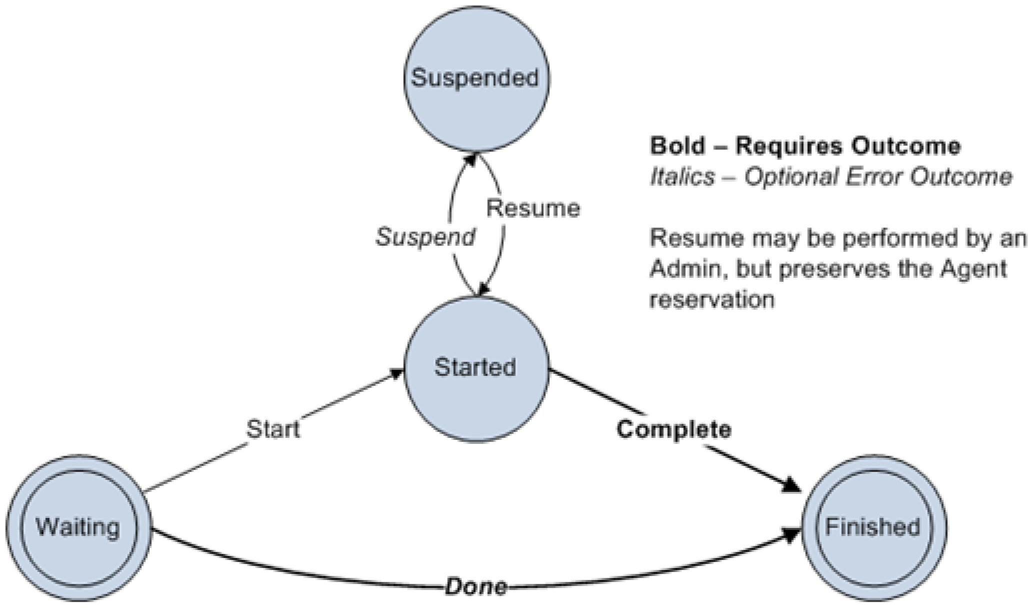
- Role override: A transition may specify its own Role, independent from the defined Activity Role (partially implements 4.2.9c).
- Reservation modification: Allocates Activity to the performing Agent. Each transition may Set, Preserve or Clear the assigned Role. (partially implements 4.2.9c)

Each state machine description value may be explicit, or reference an Activity property using the following syntax: `${Property Name}`, which will be resolved each time the value is needed. The 2.x standard *ScriptName*, *ScriptVersion*, *SchemaType* and *SchemaVersion* properties are now defined in the default state machine, rather than the Activity itself, so are no longer considered hard-coded Activity properties.

Breakpoint
A new 'Breakpoint' boolean property has been added to Activities. If set to true, the workflow will not proceed when that activity completes, which blocks workflow advancement to help with debugging. (4.2.8)

Built-in State Machines
Three state machines are included with the Kernel, and are imported to the domain path '`/desc/StateMachine/system/Kernel`' on boot:

- **Composite**: this is the default state machine for Composite Activities. It can be Waiting, Started or Finished, and doesn't support Outcomes or Scripts.
- **PredefinedStep**: all predefined steps use this null state machine, which only has one state: Available, and one transition: Done.
- **Default**: all other Activities default to this state machines, which is a simplified version of the old 2.x one. It is configured as below:


Suspended
Bold – Requires Outcome
Italics – Optional Error Outcome
Resume
Suspend
Resume may be performed by an Admin, but preserves the Agent reservation
Started
Start
Complete
Waiting
Finished
Done

3.1.4. Predefined Steps

The predefined steps AddC2kObject and RemoveC2KObject have been present since CRISTAL 2.0, and allow any Item local object to be overwritten. Because of this, many Item manipulations were done this way, which didn't leave a very useful audit trail, and didn't do any input validation. In 3.x, these two steps still exist, but are restricted to users holding the 'Admin' role, meaning that it is still possible for administrators to manipulate any object they need to in a traceable way, but regular executions must use a more specific step that logs more meaningful parameters and validated their input. More steps have been created to allow manipulations that were previously only possible through AddC2KObject, such as on collections and viewpoints.

The WriteProperty step has been extended to respect the 'mutable' flag. This flag was previously present in Property descriptions, but not respected in instantiated Item Properties. Now, the mutable flag is replicated in the Property during instantiation, and the WriteProperty step will throw an ObjectCannotBeUpdated exception if it is set to true. WriteProperty will also refuse to create a Property; all Item Properties must now be declared in Item Descriptions.

Other added/modified steps in 3.x:

- **WriteViewpoint** - Changes the referenced Event of an existing Viewpoint. The Event and Viewpoint schema names must agree, though the schema version may change.

- **CreateItemFromDescription** - A new parameter has been added which allows a set of initial Properties to be supplied to override the initial set and avoid having to set them all in a later Script. The Properties must be declared and be mutable in the Item description. These initial properties are stored in an outcome by the initialize method.

- **Collection steps**
    - **AddMemberToCollection** can now can be parametrized with slot properties.
    - **AddNewSlot** can reference a numbered version of its referenced Item description to constrain type membership.
    - **CreateNewCollectionVersion/AddNewCollectionDescription** - part of the new Collection versioning API described below.

With the addition of lifecycles to Agents, they now have their own predefined step container containing common steps with Item, and additional Agent specific steps:

- **SetAgentPassword** - changes the Agent's password. Must be executed by the owning Agent, or by an administrator.
- **SetAgentRoles** - resets the Agent's role membership. Administrators only.
- **CreateAgentFromDescription** - creates a new Agent using this one as an Agent description. Takes the same parameters as CreateItemFromDescription, with the addition of a comma-separated lists of Roles.

  - **RemoveAgent** - analogous to *Erase* in the Item, but also removes Role memberships.

The server Item, which was added in CRISTAL 2.2 as an abstraction of the server application, has been enhanced with additional global management steps:

  - **AddDomainContext/RemoveDomainContext** - add and remove Lookup directory contexts.
  - **CreateNewRole/RemoveRole** - Add and remove Roles from the system. Only administrators may remove Roles.

3.1.5. Collections

The Collection API has been simplified and extended, removing the unnecessary superclass *Parent2ChildCollection* and adding collection versioning. Two new predefined steps allow collections to be designed and versioned by domain users.

Collections instantiated from Collection descriptions are now set up in Item.initialize() to set the Item up with its initial collections at the same time as its properties and workflow are initialized. This means the the Item is complete according to its description before its lifecycle is started.
AddNewCollectionDescription is a new predefined step that creates a new, empty Collection description in the current Item, which can be then be instantiated during Item instantiated. Only administrators may create non-description Collections.

Collection versioning allows for numbered snapshots, analogous to the numbered Viewpoint names that snapshot Outcomes for use in descriptions. Because of this addition, the local path of each Collection has changed from `/Collection/<collection name>` to `/Collection/<collection name>/<collection version>`. The old 'current' version of the collection is labelled '*last*', and most applications that use the Collection can hide the new parameter and assume that.

When a Collection is considered to be part of a completed Item description, it can be snapshotted with the new predefined step *CreateNewCollectionVersion*. This makes a copy of the current 'last' version of the Collection, and stores it as the lowest unused real number, just like numbered viewpoints. At this point, a numbered PropertyDescription viewpoint plus any numbered Collection descriptions can be considered a complete Item description version, and can be referenced by Collection descriptions to constrain Collection membership to Items that match the configuration of the description at that time.

3.1.6. Other Item Local Object Changes

- Activity definition properties may be declared as abstract. This means that the functionality of the Activity instance depends on a value being supplied for this property. As before, this is done by overriding the Activity property in the slot of a Composite Activity Definition. Workflow instantiation will fail if any abstract properties in its constituent Activities have not been overridden.

- The Event object contains much more data about any Outcome it may be associated with. Schema name and version, and the Viewpoint name written to are now present, making it easier to fetch an associated Outcome and Viewpoint objects from the Event.

- The Outcome object now makes much better use of DOM. A new DOM may be created and manipulated in a new Outcome object. New methods allow Outcome querying using XPath. The Outcome now caches its DOM, and it remains in memory until the Item is reaped from memory (4.2.10)

3.1.7. Platform Changes

Configuration parameters (*c2kprops*) were previously stored in a Java Properties collection, to configure the CRISTAL instance. This object only supports String values, which was not compatible with plans in Agilium to integrate CRISTAL-iSE into an OSGi environment. In 3.x this has been replaced with `ObjectProperties`, which not only allow for easier access to boolean and numeric properties, but can hold any object including pre-instantiated implementations of plugins.

The CORBA exceptions, which may be thrown from servers to clients, have been extensively reviewed. A `InvalidCollectionModification` exception has been added to the existing set, replacing the Java `MembershipException`. `ClusterStorageException` has been merged and replaced with the existing `PersistencyException`.

The Gateway singleton has been simplified, and it is now much easier to initialize a CRISTAL-iSE process from environments other than the command-line. `AbstractMain` is still present to build the configuration from command-line arguments, but its use is optional.

3.2. Remaining changes to be implemented for CRISTAL-iSE

From the release of CRISTAL-iSE 3.0, the API is considered stable and future versions will be backwards compatible with descriptions and applications written for the current version. It will be extended over the remainder of WP3 to cover more of the requirements for Agilium NG and CIMAG-RA, but all new features will either be optional or migration code will be included to support v3.0+ descriptions.

The remaining features to be implemented in WP3 have mostly been designed, and are outlined below:

3.2.1. Improved model semantics using collections (5)

In order to more clearly document and encapsulate descriptions, more collections will be used to link associated Items together.

- Each Item will hold a collection pointing to the Item description that created it.
- The links from Activity definitions to their associated Schemas, Scripts and State Machines will be replicated as Collections. The activity properties will be deprecated, but synchronized with the new collections.
- Composite Activity Definitions will hold the links to their child Activity definitions.
- Modules will be developed and managed as Items, holding Item descriptions and resources in Collections. They will be directly exported to jar files from an activity in their workflow.

3.2.2. Enhanced security

CRISTAL at CERN was designed to track data from trusted users, and does not use any security features beyond initial authentication. To be used over the internet and for sensitive information, CRISTAL-iSE must make snooping of client/server connections impossible, and verify that each call came from the correct authenticated Agent. SSL support will be added to CORBA communications to make them more secure. The Authenticator will be extended to securely negotiate a token from the server during login, which can be used to sign any subsequent call.

3.2.3. Workflow changes

The largest remaining chunk of work on the Kernel in this project relates to workflow enhancement. The design of these enhancements has been the hardest to pin down, as the requirements depend on finer points of the Agilium NG design. At this point the following features are earmarked for inclusion, but it is still possible that the requirements will diverge enough to warrant refactoring of the workflow engine completely:

- Activity Bags: Composite Activities without layout whose child Activities are perpetually active and may be executed many times until the composite is closed.
- Implementation of a supervisor relationship between Agents. Supervisors may distribute jobs, and carry a special status when interacting with the state machines of Activities allocated to Agents or Roles they are responsible for. *(remainder for 4.2.9c)*
- BPMN2.0 features: An extensive analysis of the gap between the Agilium workflow and the BPMN 2.0 standard is underway, and changes to the workflow engine may be requested to better support it.
- The Loop vertex in the workflow engine causes many problems because it allows branching to any point - which amounts to a GOTO like functionality that can produce inconsistency. A Do/Loop workflow construct that forbids branching outside of its own block is desired. *(4.2.2)*
- The most ambitious change is a complete transformation of the way Composite Activity Definitions stored in their Items. Up until now, each version is stored complete as anOutcome, but if this were moved to a sequence of XML patches it would become easier to identify changes between versions and would make deployment management easier and make it possible to merge production changes back into

development servers and release new versions the other way *(4.2.3)*. This is the latest thinking to satisfy the ‘customer layer’ problem *(4.2.13).*

3.2.4. Miscellaneous changes

The following features remain on the todo list for the Kernel. This section, combined with the next which lists features which are rejected for Kernel inclusion, cover all requirements from the earlier D2.2 deliverable.

- Redesigned support for translated user text, to better comply with standards that have emerged since CRISTAL was originally developed. *(4.2.6)*
- A REST Client interface which will enable other languages than Java to interact with CRISTAL-iSE. *(4.2.14)*
- A testing framework is in development to automatically perform integration tests on the server and client code during build. *(4.2.11)*
- The Logging API will be mapped onto a standard framework *(4.2.8)*

3.3. Requirements that will not be implemented in the CRISTAL-iSE Kernel

The following features were listed in D2.2 as future Kernel features, but will instead be implemented outside of the Kernel.

- **3.3.1. Asynchronous write-only cluster storage (4.2.4)**
  This comes from a requirement to be able to mirror data asynchronously to another place without blocking activity execution. This will be implemented as a two-stage cluster storage that writes to a cache synchronously, and then asynchronously copies the data to its final destination.

4. Next Steps

Workpackage 3 is scheduled to continue improving the Kernel until June 2014, though the delay in starting the chunk of secondments to UWE means that it may extend into the autumn. After this point, the Kernel will enter a maintenance phase as far as this project is concerned as the focus shifts onto Agilium NG and CIMAG-RA development in Phase 3 of the project. It is hoped that after this open source release, other interested parties will become interested in using and contributing to CRISTAL-iSE, which will affect this roadmap in the long term.