[Cormat\_py] Requirements Specification

Version 1.0

January 15, 2019

Use this Requirements Specification template to document the requirements for your product or service, including priority and approval. Tailor the specification to suit your project, organizing the applicable sections in a way that works best, and use the checklist to record the decisions about what is applicable and what isn't.

The format of the requirements depends on what works best for your project.

This document contains instructions and examples which are for the benefit of the person writing the document and should be removed before the document is finalized.

To regenerate the TOC, select all (CTL-A) and press F9.

Table of Contents

[1. Executive Summary 3](#_Toc191724230)

[1.1 Project Overview 3](#_Toc191724231)

[1.2 Purpose and Scope of this Specification 3](#_Toc191724232)

[2. Product/Service Description 3](#_Toc191724233)

[2.1 Product Context 3](#_Toc191724234)

[2.2 User Characteristics 3](#_Toc191724235)

[2.3 Assumptions 3](#_Toc191724236)

[2.4 Constraints 3](#_Toc191724237)

[2.5 Dependencies 4](#_Toc191724238)

[3. Requirements 4](#_Toc191724239)

[3.1 Functional Requirements 5](#_Toc191724240)

[3.2 User Interface Requirements 5](#_Toc191724241)

[3.3 Usability 5](#_Toc191724242)

[3.4 Performance 6](#_Toc191724243)

[3.4.1 Capacity 6](#_Toc191724244)

[3.4.2 Availability 6](#_Toc191724245)

[3.4.3 Latency 6](#_Toc191724246)

[3.5 Manageability/Maintainability 6](#_Toc191724247)

[3.5.1 Monitoring 6](#_Toc191724248)

[3.5.2 Maintenance 6](#_Toc191724249)

[3.5.3 Operations 6](#_Toc191724250)

[3.6 System Interface/Integration 7](#_Toc191724251)

[3.6.1 Network and Hardware Interfaces 7](#_Toc191724252)

[3.6.2 Systems Interfaces 7](#_Toc191724253)

[3.7 Security 8](#_Toc191724254)

[3.7.1 Protection 8](#_Toc191724255)

[3.7.2 Authorization and Authentication 8](#_Toc191724256)

[3.8 Data Management 8](#_Toc191724257)

[3.9 Standards Compliance 8](#_Toc191724258)

[3.10 Portability 8](#_Toc191724259)

[4. User Scenarios/Use Cases 9](#_Toc191724260)

[5. Deleted or Deferred Requirements 9](#_Toc191724261)

[6. Requirements Confirmation/Stakeholder sign-off 10](#_Toc191724262)

[APPENDIX 11](#_Toc191724263)

[Appendix A. Definitions, Acronyms, and Abbreviations 11](#_Toc191724264)

[Appendix B. References 11](#_Toc191724265)

[Appendix C. Requirements Traceability Matrix 11](#_Toc191724266)

[Appendix D. Organizing the Requirements 13](#_Toc191724267)

# Executive Summary

## Project Overview

The program Cormat\_py, a handy tool, written in Python 3.5, for the manual/automatic correction of frinje Jumps in KG1V data.

## Purpose and Scope of this Specification

This code is intended to be used by KG1V ROs and other people authorized to handle KG1 data. This code.

In scope

This document is intended as being an introduction to the Cormat\_py code, with reference to further documentation on KG1 system and the KG1R\_py code as required. It is also a snapshot of the code in its current state (as January 2019), with the hope that will help with any further development as required. The code can be found on Gitlab [here](https://git.ccfe.ac.uk/bviola/Cormat_py). Documentation has not been released yet.

Out of Scope

It is outside the scope of these specification to describe how KG1 system works

# Product/Service Description

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements (provide the reasons why certain specific requirements are later specified).

## Product Context

How does this product relate to other products? Is it independent and self-contained? Does it interface with a variety of related systems? Describe these relationships or use a diagram to show the major components of the larger system, interconnections, and external interfaces.

## User Characteristics

Create general customer profiles for each type of user who will be using the product. Profiles should include:

* Student/faculty/staff/other
* experience
* technical expertise
* other general characteristics that may influence the product

## Assumptions

List any assumptions that affect the requirements, for example, equipment availability, user expertise, etc. For example, a specific operating system is assumed to be available; if the operating system is not available, the Requirements Specification would then have to change accordingly.

## Constraints

Describe any items that will constrain the design options, including

* parallel operation with an old system
* audit functions (audit trail, log files, etc.)
* access, management and security
* criticality of the application
* system resource constraints (e.g., limits on disk space or other hardware limitations)
* other design constraints (e.g., design or other standards, such as programming language or framework)

## Dependencies

List dependencies that affect the requirements. Examples:

* This new product will require a daily download of data from X,
* Module X needs to be completed before this module can be built.

# Requirements

* Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.
* Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.
* Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)
* Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

Priority Definitions

The following definitions are intended as a guideline to prioritize requirements.

* Priority 1 – The requirement is a “must have” as outlined by policy/law
* Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
* Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

* A good requirement is:
* Correct
* Unambiguous (all statements have exactly one interpretation)
* Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
* Consistent
* Ranked for importance and/or stability
* Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)
* Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
* Does not specify any particular design
* Traceable (cross-reference with source documents and spawned documents).

## Functional Requirements

In the example below, the requirement numbering has a scheme - BR\_LR\_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

The following table is an example format for requirements. Choose whatever format works best for your project.

For Example:

| Req# | Requirement | Comments | Priority | Date Rvwd | SME Reviewed / Approved |
| --- | --- | --- | --- | --- | --- |
| BR\_LR\_05 | The system should associate a supervisor indicator with each job class. | Business Process = “Maintenance | 3 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_08 | The system should handle any number of fees (existing and new) associated with unions. | Business Process = “Changing Dues in the System”  An example of a new fee is an initiation fee. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_10 | The system should capture and maintain job class status (i.e., active or inactive) | Business Process = “Maintenance”  Some job classes are old and are no longer used. However, they still need to be maintained for legal, contract and historical purposes. | 2 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_16 | The system should assign the Supervisor Code based on the value in the Job Class table and additional criteria as specified by the clients. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_03. | 2 |  |  |
| BR\_LR\_18 | The system should provide the Labor Relations office with the ability to override the system-derived Bargaining Unit code and the Union Code for to-be-determined employee types, including hourly appointments. | April 2005 – New requirement. It is one of three new requirements from BR\_LR\_04.  5/11/2005 – Priority changed from 2 to 3. | ~~2~~  3 |  |  |

## User Interface Requirements

In addition to functions required, describe the characteristics of each interface between the product and its users (e.g., required screen formats/organization, report layouts, menu structures, error and other messages, or function keys).

## Usability

Include any specific usability requirements, for example,

Learnability

* The user documentation and help should be complete
* The help should be context sensitive and explain how to achieve common tasks
* The system should be easy to learn

(See <http://www.usabilitynet.org/>)

## Performance

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

* Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.
* Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.

All of these requirements should be stated in measurable form. For example, "95% of the transactions shall be processed in less than 1 second" rather than “an operator shall not have to wait for the transaction to complete”.

### Capacity

Include measurable capacity requirements (e.g., the number of simultaneous users to be supported, the maximum simultaneous user load, per-user memory requirements, expected application throughput)

### Availability

Include specific and measurable requirements for:

* Hours of operation
* Level of availability required
* Coverage for geographic areas
* Impact of downtime on users and business operations
* Impact of scheduled and unscheduled maintenance on uptime and maintenance communications procedures
* reliability (e.g., acceptable mean time between failures (MTBF), or the maximum permitted number of failures per hour).

### Latency

Include explicit latency requirements, e.g., the maximum acceptable time (or average time) for a service request.

## Manageability/Maintainability

### Monitoring

Include any requirements for product or service health monitoring, failure conditions, error detection, logging, and correction.

### Maintenance

Specify attributes of the system that relate to ease of maintenance. These requirements may relate to modularity, complexity, or interface design. Requirements should not be placed here simply because they are thought to be good design practices.

### Operations

Specify any normal and special operations required by the user, including:

* periods of interactive operations and periods of unattended operations
* data processing support functions
* backup and recovery operations
* safety considerations and requirements
* disaster recovery and business resumption

## System Interface/Integration

Specify the use of other required products (e.g., a database or operating system), and interfaces with other systems (e.g., UWHires package interfaces with PubCookie and ODS, HEPPS system interfaces with Budget system). For each interface, define the interface in terms of message format and content. For well-documented interfaces, simply provide a reference to the documentation.

Outline each interface between the product and the hardware or network components of the system. This includes configuration characteristics (e.g., number of ports, instruction sets), what devices are to be supported, and protocols (e.g., signal handshake protocols).

### Network and Hardware Interfaces

Specify the logical characteristics of each interface between the product and the hardware or network components of the system. This includes configuration characteristics (e.g., number of ports, instruction sets), what devices are to be supported, and protocols (e.g., signal handshake protocols).

### Systems Interfaces

Example systems interface requirements:

1. System1-to-System2 Interface

The <external party> will create and send a fixed length text file as an email attachment to [System2mail@u.washington.edu](mailto:heppsmai@u.washington.edu) to be imported into the System2 system for payroll calculation. This file must be received on EDIT day by 4:00 PM in order to be processed in the EDIT night run. The requirements below document the file specifications, data transfer process, and specific schedule. This file is referred to as "FileName" in this document.

File Structure and Format

* 1. The FileName file is a fixed length text file.
  2. The FileName file is an unformatted ASCII file (text-only).
  3. The FileName file contains a batch totals record and several detail records.

File Description: Batch Totals Record

* 1. The batch totals record can be placed at the beginning, in the middle, or at the end of the file.
  2. The batch totals record contains the following:

Record Type (value: XA)

Process Type (value: A)

Batch Number (3 digit number assigned by Payroll Dept)

Origin Code (AIG)

Total number of detail records

Total deduction amount

File Description: Detail Records

* 1. The FileName file contains a row for each record meeting xxx criteria.
  2. Each row in the FileName file contains the following fields, comma-delimited and encased in double-quotes where the data includes commas or spaces:
* Employee Id
* Record Type
* Process Date (MMDDYY)
* XYG Number
* Element Code
* Amount
* Amount Sign
* Year Flag
* Total Amount
* Total Amt Sign

## Security

### Protection

Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:

* encryption
* activity logging, historical data sets
* restrictions on intermodule communications
* data integrity checks

### Authorization and Authentication

Specify the Authorization and Authentication factors. Consider using standard tools such as PubCookie.

## Data Management

Specify the requirements for any information that is to be placed into a database, including

* types of information used by various functions
* frequency of use
* data access rules
* data entities and relationships
* integrity constraints
* data retention
* valid range, accuracy, and/or tolerance
* units of measure
* data formats
* default or initial values

## Standards Compliance

Specify the requirements derived from existing standards, policies, regulations, or laws (e.g., report format, data naming, accounting procedures, audit tracing). For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

## Portability

If portability is a requirement, specify attributes of the system that relate to the ease of porting the system to other host machines and/or operating systems. For example,

* Percentage of components with host-dependent code;
* Percentage of code that is host dependent;
* Use of a proven portable language;
* Use of a particular compiler or language subset;
* Use of a particular operating system;
* The need for environment-independence - the product must operate the same regardless of operating systems, networks, development or production environments.

# User Scenarios/Use Cases

Provide a summary of the major functions that the product will perform. Organize the functions to be understandable to the customer or a first time reader. Include use cases and business scenarios, or provide a link to a separate document (or documents). A business scenario:

* Describes a significant business need
* Identifies, documents, and ranks the problem that is driving the scenario
* Describes the business and technical environment that will resolve the problem
* States the desired objectives
* Shows the “Actors” and where they fit in the business model
* Is specific, and measurable, and uses clear metrics for success

# Deleted or Deferred Requirements

Identify any requirements that have been deleted after approval or that may be delayed until future versions of the system. For example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Req# | Business Requirement | Status | Comments | Pri | Date Rvwd | SME Reviewed /Approved |
| BR\_LR\_01 | The system should validate the relationship between Bargaining Unit/Location and Job Class. | April 2005: Deleted.  This requirement has been replaced by BR\_LR\_036 and BR\_CC\_33. | Business Process = “Assigning a Bargaining Unit to an Appointment” | 1 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_02 | The system should validate that the supervisor indicator is correct according to job class.  Deferred to Phase 2B: 3/29/2005 | April 2005: Deferred to Phase 2B. | Business Process = “Assigning a Bargaining Unit to an Appointment” | 3 | 7/13/04 | Bob Dylan, Mick Jagger |
| BR\_LR\_03 | The system should derive the bargaining unit code, union code, and supervisor indicator from the job class code and location. | April 2005: Deleted  Replaced by BR\_LR\_16 and BR\_LR\_17. | Business Process = “Assigning a Bargaining Unit to an Appointment”; This will eliminate the need, typically, for the user to enter the bargaining unit code, union code and supervisor indicator. | 1 | 7/13/04 | Bob Dylan, Mick Jagger |

# Requirements Confirmation/Stakeholder sign-off

Include documentation of the approval or confirmation of the requirements here. For example:

|  |  |  |
| --- | --- | --- |
| Meeting Date | Attendees (name and role) | Comments |
| 7/13/07 | Bob Dylan, Labor Relations SME  Mick Jagger, Labor Relations SME  Ringo Starr, Technical Project Manager  Debbie Harry, Technical Analyst  Janis Joplin, Technical Analyst  Fred Meyer, Project Manager | Confirmed BR\_LR\_01 – BR\_LR\_15 |
| 04/15/05 | Bob Dylan, Labor Relations SME  Mick Jagger, Labor Relations SME  Ringo Starr, Technical Project Manager | Deferred / Deleted: BR\_LR\_01 - BR\_LR\_04, BR\_LR\_07, BR\_LR\_12, BR\_LR\_14, BR\_LR\_15, BR\_LR\_06, BR\_LR\_17 |

APPENDIX

The appendixes are not always considered part of the actual Requirements Specification and are not always necessary. They may include

* Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;
* Supporting or background information that can help the readers of the Requirements Specification;
* A description of the problems to be solved by the system;
* Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

When appendixes are included, the Requirements Specification should explicitly state whether or not the appendixes are to be considered part of the requirements.

1. Definitions, Acronyms, and Abbreviations

Define all terms, acronyms, and abbreviations used in this document.

1. References

List all the documents and other materials referenced in this document.

1. Requirements Traceability Matrix

The following trace matrix examples show one possible use of naming standards for deliverables (FunctionalArea-DocType-NN). The number has no other meaning than to keep the documents unique. For example, the Bargaining Unit Assignment Process Flow would be BUA-PF-01.

For example (1):

| **Business Requirement** | **Area** | **Deliverables** | **Status** |
| --- | --- | --- | --- |
| BR\_LR\_01  The system should validate the relationship between Bargaining Unit/Location and Job Class.---Comments: Business Process = "Assigning a Bargaining Unit to an Appointment" (Priority 1) | BUA | BUA-CD-01  Assign BU Conceptual Design | Accepted |
| BUA-PF-01  Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BUA-PF-01  Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BR\_LR\_09  The system should provide the capability for the Labor Relations Office to maintain the job class/union relationship.---Comments: Business Process = "Maintenance" (Priority 1) | BUA | BUA-CD-01  Assign BU Conceptual Design | Accepted |
| BUA-PF-02  BU Assignment Rules Maint Process Flow Diagram | ReadyForReview |

For example (2):

| **BizReqID** | **Pri** | **Major Area** | **DevTstItems DelivID** | **Deliv Name** | **Status** |
| --- | --- | --- | --- | --- | --- |
| BR\_LR\_01 | 1 | BUA | BUA-CD-01 | Assign BU Conceptual Design | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-DS-02 | Bargaining Unit Assignment DB Modification Description | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-PF-01 | Derive Bargaining Unit-Process Flow Diagram | Accepted |
| BR\_LR\_01 | 1 | BUA | BUA-UCD-01 | BU Assign LR UseCase Diagram | ReadyForReview |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-001 | BU Assignment by PC UseCase - Add Appointment and Derive UBU | Reviewed |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-002 | BU Assignment by PC UseCase - Add Appointment (UBU Not Found) | Reviewed |
| BR\_LR\_01 | 1 | BUA | BUA-UCT-006 | BU Assignment by PC UseCase - Modify Appointment (Removed UBU) | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-CD-01 | Assign BU Conceptual Design | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-DS-02 | Bargaining Unit Assignment DB Modification Description | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-PF-02 | BU Assignment Rules Maint Process Flow Diagram | Accepted |
| BR\_LR\_09 | 1 | BUA | BUA-UCD-03 | BU Assign Rules Maint UseCase Diagram | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-045 | BU Assignment Rules Maint: Successfully Add New Assignment Rule | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-051 | BU Assignment Rules MaintUseCase: Modify Rule | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-053 | BU Assignment Rules MaintUseCase - Review Assignment Rules | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UCT-057 | BU Assignment Rules MaintUseCase: Inactivate Last Rule for a BU | Reviewed |
| BR\_LR\_09 | 1 | BUA | BUA-UI-02 | BU AssignRules Maint UI Mockups | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-021 | BU Assignment Rules Maint TestCase: Add New Rule (Associated Job Class Does Not Exist) - Success | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-027 | BU Assignment Rules Maint TestCase: Modify Rule - Success | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-035 | BU Assignment Rules Maint TestCase: Add New Rule (Associated Job Class Does Not Exist) - Error Condition | ReadyForReview |
| BR\_LR\_09 | 1 | BUA | BUA-TC-049 | BU Assignment Rules Maint TestCase: Modify Rule - Error Condition | ReadyForReview |

For example (3):

| **BizReqID** | **CD01** | **CD02** | **CD03** | **CD04** | **UI01** | **UI02** | **UCT01** | **UCT02** | **UCT03** | **TC01** | **TC02** | **TC03** | **TC04** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BR\_LR\_01 |  |  | X |  | X |  | X |  |  | X |  | X |  |
| BR\_LR\_09 | X |  |  | X |  | X |  |  | X |  | X |  | X |
| BR\_LR\_10 | X |  |  | X |  |  |  |  | X |  | X |  |  |
| BR\_LR\_11 |  | X |  |  |  |  |  |  |  |  |  |  |  |

1. Organizing the Requirements

This section is for information only as an aid in preparing the requirements document.

Detailed requirements tend to be extensive. Give careful consideration to your organization scheme. Some examples of organization schemes are described below:

By System Mode

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency.

By User Class

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fire fighters.

By Objects

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.

By Feature

A feature is an externally desired service by the system that may require a sequence of inputs to affect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs, and may include validity checks on inputs, exact sequencing of operations, responses to abnormal situations, including error handling and recovery, effects of parameters, relationships of inputs to outputs, including input/output sequences and formulas for input to output.

By Stimulus

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc.

By Response

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc.

By Functional Hierarchy

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.

Additional Comments

Whenever a new Requirements Specification is contemplated, more than one of the organizational techniques given above may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.