Software Release Plan

Last Modified: 28/12/2020

myDrive:
SOFTWARE RELEASE PLAN

Document identifier:	**myDrive-12-D20.1**
Date:	**28/12/2020**
Author:	**Samir Adrik,** *Technical coordinator* *Full-Stack developer*
Document status:	**APPROVED**
Deliverable identifier:	**D21.2.1**

Abstract: This document describes a proposed project-wide policy for organising the production of successive software releases within the myDrive project.

Document Log

Version	Date	Comment	Author(s)
1.0	12/12/2020	First Draft	Samir Adrik
1.1	19/12/2020	Approved after meeting with the team	Samir Adrik

Delivery Slip

	Name	Role	Date	Signature
From	Samir Adrik	Technical coordinator, Full-Stack developer	28/12/2020	
Verfied by	Asso Rostampoor, Mohammed Usman	Full-Stack developer(s)	28/12/2020	
Approved by	Umair Iqbal	Project Manager	28/12/2020	

Files

Document identifier	User file
myDrive-12-D20.2	myDrive: Design Documentation

Contents

1 Introduction

1.1 Objectives of this document

This document describes a proposed policy for organising the production and deployment of successive software releases within the myDrive project.

The overall myDrive project schedule includes 2 testbeds and 8 releases over a 6 month period. For each testbed or release, a software release is conducted. Such software releases integrate the latest developments from each of the feature branches from the team developers with the `dev` branch and furthermore an integration is also done with the `master` branch, i.e., the production environment.

The coordination of the software developments in each feature branch and their integration is an important task that requires considerable effort.

On the **1st of February 2021** the myDrive project should have a "working-product" in production and model in which integration happens more frequently with incremental steps to minimise the effort and time involved and permits user feedback to be incorporated more quickly.

This document presents a proposed policy for iterative software releases including milestones, time-scales and supporting techniques and tools to be employed.

The policy outlined in this document is presented as a detailed implementation plan for each software release associated with the myDrive project. Each team member is recommended to read and supply feedback to the contents in the document.

1.2 Application area

This document concerns the technical coordination of the software releases for the myDrive project. It has a consequence on the manner in which the software of the project is produced, integrated and tested by future users and stakeholders.

1.3 Applicable and reference resources

Reference resources

- myDrive Central Github Repository - `https://github.com/seemir/myDrive`;
- Official Slack workplace for the myDrive project - `https://app.slack.com/client/T01A0N5PDAL`;

1.4 Document amendment procedure

Readers are invited to send comments on the myDrive slack project to the author who will consolidate all feedback and produce updated versions of the document when significant changes are made.

1.5 Terminology

Definitions:

- myDrive - Project for the introduction of a Customer-driven logistics platform primary to be used in the transportation of goods and services market in Pakistan;

1.5.1 Glossary

- Github - Version control (also known as revision control, source control, or source code management) systems responsible for managing changes to the myDrive app;
- Slack - Business communication platform used by the myDrive team to communicate changes, news, updates and any questions or feedback;

1.6 Role of the technical coordinator

It is the role of the technical coordinator to organise and oversee the software release planning of the project in consultation with the project management and team members. These tasks include:

- Proposing a software release policy and procedure to drive the development of each software release;
- Defining a software release calendar including release dates and content outlines;
- Organising a forum so that technical issues involving multiple issues can be addressed;
- Working with the project managers to ensure work progress is harmonized with overall project plans, including input from users and stakeholders;
- Establishing a basic software infrastructure and toolset to facilitate software development, continuous integration and testing;
- Tracking the progress of the work to ensure the software release schedule progresses as smoothly as possible.

2 Executive summary

This document presents the policy for software releases proposed for the myDrive project. The initial project plan foresaw software releases corresponding to an initial final milestone on 1st of January 2021. At the 7th of December 2020 the myDrive project came to a general consensus that the remaining work to reach completed "working product" and the adaptation of a model in which integration happens more frequently than defined in the original project plan with incremental steps to minimise the effort and time involved and permit user feedback to be incorporated more quickly was postposed to the **1st of February 2021**.

The proposed software release schedule includes incremental releases in January (0.1), January (0.2), January (0.3) and January 2021 (0.4) leading up to testbed 1 (1.0). A similar release schedule is planned for the preceding periods when more experience has been gained with iterative releases.

The proposed procedure for the planning and deployment of successive software releases is intended to ensure development remains focused on the highest priority issues and to improve the manner in which subsequent releases are produced.

The software release procedure is composed of the following steps:

- *Coordination meeting* - project participants provide feedback on the previous release and develop a basic work-plan for the next software release;
- *Team follow-up meetings* – Team members elaborate their specific work-plans for the next release;
- *Software Release work-plan coordination* - the technical coordinator consults with the project managers and takes the output of the coordination and teams members follow-up meetings to establish an overall plan for the next release;
- *The technical coordinator provides unit-tested software and associated documentation for the myDrive project as it is produced;*
- *The technical coordinator integrates internal and external software packages and performs integration tests;*
- *An integrated software release is made available for validation and acceptance testing;*
- *The release roll-out* is accompanied by documentation for the end users, stakeholders and software developers. A meeting is hosted by the technical coordinator to present the contents of the new software release and indicate the changes made since the previous release;

- The software release is made available to all end users and stakeholders for general deployment;

In order to support more frequent software releases, a number of tools and techniques need to be put in place by the technical coordinator. Such a toolset will provide a convenient, standardised, project-wide mechanism for building, distributing and documenting the software. It is important that all the team members learn and start using the toolset as early as possible. The details of this toolset are described later.

As a means of providing feedback on integration issues to software developers in each release as early as possible, mechanisms to automatically build, test and document all the software on a daily basis will be put in place. The responsible party for this implementation is the technical-coordinator.

The policy outlined in this document is presented as a detailed implementation plan for each software release.

2.1 Frequency of release

The overall project plan foresees 6 months between testbed 1 (project month 1) and testbed 2 (project month 7). A period of 6 months between testbeds limits the speed of reaction to evolving requirements from the end users and stakeholders and deployed in the overall software planning.

Adding more releases to the project plan increases the total work-load for the team members and users and stakeholders alike but allows more frequent feedback to ensure development concentrates on the most relevant points and reduces the time required for the integration phase of each release. One of the goals of the proposed approach is to maximise the advantage that can be gained from iterative software releases while reducing to a minimum the extra work required.

While every effort will be made to minimise the impact on the users and stakeholders of upgrading from one software release to the next, it will be productive to guarantee backward compatibility between releases in this overall project. However, the use of techniques, such as abstract interfaces and encapsulation to isolate users from changes in the software, should be considered adopted by the team members.

The effects of pending extensions/improvements must be made known to the team members well in advance so that their impact can be estimated. Hence the decision about when and how to introduce major modifications must be made in consultation with the team members at all times.

2.2 Release schedule

Given the size of the project in terms of software packages and groups of people involved, the following iterations of software releases between testbed 1 and testbed 2 are proposed:

- **Release 0.1** (10th of January 2021);
  Usman (#5, #9), Asso (#10), Samir (#3, #6, #7, #8, #12)
- **Release 0.2** (17th of January 2021);
  Usman (#11), Asso (#1), Samir (#2)
- **Release 0.3** (24th of January 2021);
  Usman, Asso (#21), Samir (#28)
- **Release 0.4** (31th of January 2021);
  Usman (#26), Asso, Samir (#27)
- **Testbed 1 - February 2021**;
  Originally scheduled for January 2021 and delivered in February 2021
  First integrated release produced by the project, i.e. a operational and working product should be completed.

- **Release 1.1 - March 2021**;
  Contains bug-fixes, modifications as a result of feedback from use of testbed 1 and first use of release infrastructure by the middle-ware (see Supporting)
- **Release 1.2 - April 2021**;
  Contain bug fixes, modifications as a result of feedback from use of the previous releases by applications, further use of the software infrastructure and extensions/improvements foreseen by team members
- **Release 1.3 - May 2021**;
  Contain bug fixes, modifications as a result of feedback from use of the previous releases by applications, further use of the software infrastructure and extensions/improvements foreseen by team members
- **Release 1.4 - June 2021**;
  Contain bug fixes, modifications as a result of feedback from use of the previous releases by applications, further use of the software infrastructure and extensions/improvements foreseen by team members
- **Testbed 2 - July 2021**;
  A similar release schedule will be planned for the period between testbeds 2, 3 (September 2021) and 4 (December 2021) at a later date when more experience has been gained with iterative releases.

3 Supporting infrastructure

In order to support more frequent software releases, a number of supporting tools and techniques need to be put in place. A sub-set of such tools and techniques are already deployed by technical coordinator to support the testbed 1 of the project. The technical coordinator is responsible for establishing and managing the toolset.

The myDrive Github repo plays a vital role in the production of software releases. In order to support more frequent software releases the repo must continue to be active between the scheduled testbed integration phases. Its activity in any interim periods will not be so intense, but rapid response to integration issues will be important.

The technical coordinator has already put in place a number of tools and techniques to ease software integration and the production of the software release for testbed 1. These include a code repository, communication platform (Slack) and an incident tracking processing (Github issue management). Further steps can be taken to reduce integration effort by catching potential conflicts earlier in the development cycle. Such measures, as with the software itself, will be extended with each release and subject to more detailed planning.

As a means of providing feedback on integration issues to software developers in each release as early as possible, the aim is to put in place mechanisms to automatically build, test and document all the software on a daily basis. Such "nightly builds" are for the benefit of the team developers and hence will not be systematically distributed to the application groups and users since they will not be the subject of the release testing procedure (see below). However, the contents and status of the nightly builds will be available to everyone in the project via tools such as Travis-CI.

3.1 Centralised code repository

Many of the tools and techniques foreseen within the software infrastructure are linked to the use of a code repository. In the build-up to testbed 1, team members have to produce commits containing their software to either the myDrive central repository or their own repositories that are then mirrored into the `dev` branch before merging to `master` branch.

This central repository will act as the core facility on which the automated building, testing and documentation tools will be based. It is essential that all team members adopt the code repository as an integral part of their daily working practises. Sustained software development without committing modifications to the central repository for extended periods (weeks) will undermine the functioning and value of the nightly build procedure.

To enable the use of several of the proposed tools and to simplify browsing, the source code should be stored in the repository as individual files. The application can then be produced as one of the final parts of the build procedure.

It is expected that detailed documentation including code descriptions, class hierarchies and dependencies can be extracted from comments in the source code and by auto-documentation tools. More high-level documentation will still need to be written by hand. All documentation, both extracted and hand-written, will evolve with each software release. To ensure users get the correct version of the hand-written documentation for a software release it will be necessary to archive the documentation, preferably in the same repository as the code itself. This implies using documentation formats (e.g., separate LaTeX or Dart Docstrings / documentation and style documentation files, e.g. .css, .html or .js) that lends themselves to archiving tools such as Github.

3.2 Development branch

A development branch is required in order to support iterative development, integration and testing without interfering with the validation activities performed by the stakeholders in the `master` branch. In the current planning, it is foreseen that commits for development or integration work will first be committed to the `dev` branch before any merges are done to the `master` branch. The daily development versions of any branches in the software can be tested at any time by any of the team members. It is recommended that all team members get to know the development done on each active branch in the myDrive central repository. The overall progress and status of the development branch will be managed by technical coordinator in conjunction with the team members.

If a developer wants to add features to the myDrive project he needs to branch from the `dev` branch and name the branch in accordance with the following naming convention `feature_name_1`. Ones the feature is ready for deployment to `dev` it can be merged. It needs to be emphasized that any merges to the `dev` branch must be done through a `pull_request_1` (PR) where all developers except the creator of the PR are assigned roles as reviewers. Ones all `pull_requests` are merged to the `dev` branch and the team feels confident in the content of the `dev` branch and the release date has come then the application is ready to be released (`release_1`), i.e. merge `dev` with the `master` branch. The illustration below (Figure **??**) shows the relation between `feature`, `dev` and `master` branches.

All branches must remain available to the developers for the duration of the project. If a branch is to be deleted this must be confirmed and agreed upon by the technical coordinator.

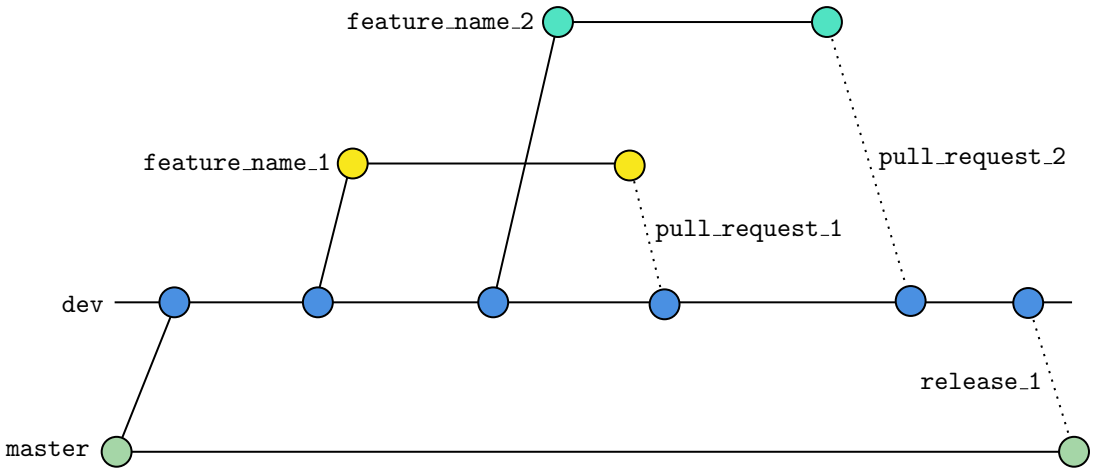

Figure 3.1: Github workflow

3.3 Toolset

Recommendations including guidelines/templates and examples for the use of tools and techniques to support automated building, testing and documentation are currently in preparation. Such a toolset will provide a convenient, standardised, project-wide mechanism for building the software and documentation. It is important that all the team members adopt the toolset as early as possible. The details of this toolset are defined in below, but this list is a non-exhaustive list of these tools and techniques:

- Mark-up techniques (e.g., Dart style comments) for document generation from source code;
- A tool capable of extracting formatted source comments and producing documentation in various formats (e.g., Android Studio build in linters);
- A web interface for browsing the repository (e.g., Google Chrome);
- Github issue management for problem reporting and tracking;
- Test harnesses (e.g., codecov etc.);
- Build tools (e.g., Travic CI etc.) capable of constructing the software on all supported platforms and producing RPMs for distribution;
- A "slack repo notifier" providing a daily digest of commits to the repository from all team members;

- A web page providing access to the results of the most recent automated build, test and documentation cycle;
- Coding style recommendations and tools for checking them (e.g., Dart lint);

The tools at the top of the list are the highest priority and work using them is already started by the team members.

All tools or techniques will be set-up by the technical coordinator using the centralised code repository as a basis and will be deployed in a step-wise manner. It is not intended to develop new tools to support the nightly builds but rather configure existing widely used tools. In this manner, development by team members can be concentrated on exploiting such tools and techniques in their daily work without worrying about the installation and support issues for the tools they use.

4 Testing

This section outlines the software testing procedures foreseen within the project. The basic approach is to address software testing in a bottom-up manner via three distinct phases:

- **Unit tests**;
  Performed on individual software modules to test their basic functionality
- **Integration tests**;
  Performed on integrated software components to verify their correct inter-operation
- **Acceptance tests**;
  Performed using users and test scenarios. The acceptance tests are to be created, organized and managed by the technical coordinator.

4.1 Testplans

To drive the testing procedure a set of testplans are required. The testplans outline the goals and priorities of the testing and list the set of test scenarios foreseen to meet the goals. A description of the tests to be performed for each phase should be documented in the test-plans.

For unit tests, it is the responsibility of the technical coordinator to produce a test plan document based on recommended templates. Each team member should read and understand the test plan. The test plan should include details of testing individual software modules and their integration within the application.

A separate test plan for the project-wide integration tests is also to be produced by the technical coordinator.

Documentation for acceptance tests is foreseen as part of the project deliverables. The three phases of the test procedure are to be performed for each release of the software delivered to users and all stakeholders. The details of each testing phase are described below.

4.2 Unit tests

Unit tests address the basic functionality of a single component of the application. They are to be provided and executed by the team members on their own machines or branches before delivering their software to the dev branch for integration. The individual tests should make use of the test harness software to be provided as part of the toolset and may need to simulate services provided by other team members according to the build dependencies. A subset of the unit tests should be run as part of the nightly build procedure.

Details of the unit tests should be documented as part of the individual test-plans. Sufficient documentation should be provided to allow other people (i.e., not the developers) to execute the tests and interpret the results.

Unit tests should also be established for external software packages and it is necessary to identify individuals or groups responsible for such tests.

4.3 Integration tests

The integration tests verify successful integration of the components developed within the project and underlying external toolkits. Integration tests are executed on the development branch whenever new software for a release is received from team members and built. Their development (though they may be gathered from elsewhere), management and execution are the responsibility of the technical coordinator.

The basic approach is to test all the components via typical user scenarios based on job submission. The integration tests represent a succession of increasingly complex job submissions. The tests cover all aspects of job processing/monitoring that are important to the users and stakeholders. The intention is to add one more component or level of complexity with each job so that by executing the jobs in sequence it should be relatively easy to quickly identify a defective component.

It is expected that the set of test scenarios should be expanded as the project gains more experience with the software and requirements from the users.

Initially the test scenarios will be run by hand but it is hoped that a subset can be automated (e.g., via scripts etc.) once their usefulness has been proven and integrated as part of the nightly build procedure.

Details of the integration tests should be documented as part of the overall test-plan.

4.4 Acceptance tests

Acceptance tests are to be performed in conjunction with the application groups using scenarios provided by the users (typically based on their own application software). Such tests are to be executed before merge is done to the `master` branch in a distributed manner (i.e., typically involving multiple users on multiple mobile phones). The definition of the test scenarios and their execution is the responsibility of the technical coordinator and will be supported by the team members. The acceptance tests correspond to the validation tests that all IT projects are recommended to have.

4.5 Further testing

Scalability, reliability and resilience tests involving the full myDrive software should also be defined. Such tests will require more infrastructure (e.g., more users than are available via the development branch) and planning (e.g., continuous 24 hours tests etc.) and hence will need to be scheduled in advance.

5 Software release procedure

A well-defined procedure related to the planning and deployment of successive software releases will be established.

As part of the software release production procedure, there must be the opportunity for all the parties involved to provide feedback on the software and the release procedure to ensure development remains focused on the highest priority issues and to improve the manner in which subsequent releases are produced.

The planned contents and schedule for subsequent releases should be reviewed by the project manager to ensure the plan remains as accurate as possible.

An overview of the proposed software release procedure is described below as a sequence of steps:

1. **Coordination meeting**

   The purpose of the coordination meeting is to allow the project participants to provide feedback gathering their experiences from working with the previous release and outline a work-plan for the next software release. It should be scheduled after the previous software release has been deployed and used by possible users, but long enough before the next release to enable its findings to be taken into account. The product of the coordination meeting should be an appraisal of the previous release's quality and suitability including how it has been used and a prioritised list of requested modifications/extensions from the next release.

   The coordination meetings will be organised by the technical co-ordination and are open to everyone involved in the project but the participants should include:

   - Project manger;
   - Team developer(s);
   - Any other relevant stakeholders;

   The agenda for the coordination meeting should include presentations by each of the team developers with sufficient time for discussion to establish a list of agreed priorities for the next release. The plan for the next release should include:

   - software components to be included along with their modifications/extensions relative to the previous release;
   - Operating system and compiler versions to be part of the reference platform;
   - Versions of external toolkits and supporting packages to be used by all team members;

   All team members should remember that their input must be collated and hence relative priorities may change in order to establish the most suitable plan for the project.

   Such coordination meetings can also be based on similar format to that used for the *application groups' meeting* done in the industry.

   In preparation for the coordination meeting, it is assumed that each of the team members will have gathered input and clarified their position on relevant topics to be discusses in the coordination meeting.

   Project manager should also hold a preparation meeting where the team members may raise release production issues and make suggestions about how to simplify or improve integration for the next release.

2. **Team follow-up meetings**
   Each team member can hold a meeting to establish its work-plan for the next release. Such meetings should take into account the output of the coordination meeting and produce a work-plan for the next release describing a prioritised list of expected modifications/extensions and including details of responsibilities for performing the work that takes into account an estimation of the effort and resources involved. The team follow-up meetings can be held in parallel but dependencies on other team members should also be respected when scheduling these meetings.

3. **Software release work-plan coordination**
   The technical co-ordination will, in consultation with the project managers, take the output of the coordination and team follow-up meetings to establish an overall plan for the next release. The overall work-plan will be published and made available to all participants of the project.

The rest of the software release procedure should be similar to that of any other release:

4. **Unit tests** should be performed by the individual team members before it is submitted to the `dev` branch. The proposed development can be used for this purpose. The submission of the software should be made via the code repository and clearly tagged for subsequent retrieval.

5. The technical coordinator maintains and integrates dependencies with external software packages. Basic integration tests are performed. The integration step is deemed to have been successfully completed when a stable version of the software passes the integration tests as defined in the test-plan is available in the `dev` branch. During this time the associated documentation for the release (including release notes) is prepared by the technical coordinator.

6. An integrated software release is made available to the users for **acceptance testing**. During this time the team members are available for consultation and support of the new release. The acceptance testing phase is deemed to have been successfully completed when the set of test scenarios defined by the test plan execute correctly.

7. The release roll-out is accompanied by **documentation** for the end users, stakeholders and software developers. A meeting should be hosted by the technical coordinator to present the contents of the new software release and indicate the changes made since the previous release. Issues concerning compatibility with the previous release should be presented together with recipes for how to migrate applications and data. The format of this roll-out meeting should be similar to the coordination meeting.

8. The software release is made available to all users and stakeholders for general deployment on the `master` branch.

5.1 Themed technical meetings

A number of technical issues confront the project which affect multiple parts of the application. The intention is to address such issues via special themed meetings involving representatives from the part and the technical coordinator. Candidate subjects for such themed meetings include any team member. It is expected that the priority for topics for the themed meetings will come from the list of priorities established at the release coordination meetings. Urgent technical clashes that arise on a day-to-day basis at the implementation level are perhaps best solved with direct communication between the team members involved which may take the form of bi-lateral meetings. Only if they affect the overall strategy, external interfaces or if a satisfactory solution cannot be found should it be necessary to present them at the themed technical meetings.

The proposed organisation of the themed meetings is to distribute a description of the issues involved and a suggested approach/solution before the meeting. The proposal is intended to focus the discussion and ensure the meetings result in a clear conclusion.

6 Next steps

In order for the plan described in this document to be put into action, a number of steps are required:

6.1 Establish release contents

An initial set of extensions/changes were presented by each team member for testbed 1 at the 7th of December 2020 meeting. These lists are a starting point for planning the contents of the scheduled releases. Team members are asked to complete the information about each item on their proposed list by providing the following details:

- **Priority:** Prioritise the list of items in consultation with the technical coordinator and project manager;
- **Effort:** Estimate the effort required to develop/modify the software and identify the individuals/groups involved;
- **Dependencies:** Identify the dependencies on other software packages (internal or external software products);

Note: An item that needs to be added to every team member list is the use of the infrastructure tools required to support the release procedure. The training in the use of the infrastructure tools is the responsibility of the technical coordinator.

Once such information is available from all the team member contributing to the software releases, it will be possible to map them on to the foreseen release schedule. This mapping will have to take into account dependencies between team members. Such a plan will be useful to help organise the testing and validation of the application. The input of the team members is important for the release planning procedure. It is expected that, the work of the validation will be affected by making interventions more often but with reduced intensity.

It is hoped that the overall release schedule including each team members extensions/changes can be compiled during December 2020. This release schedule will be updated after each release using the output of the software release work-plan coordination.

Note: Planning is never perfect – team members may under estimate the effort required for a particular item or a new release of an external software package may not be delivered on time. In such cases priority will be given to respecting the release schedule. In other words, the scheduled date for the myDrive software releases will not change but it may contain less changes/extensions than originally planned.

6.2 Develop testplans

An important point to clarify the status of the software and simplify integration, is the development of testplans for all major components of the software. The technical coordinator should develop testplans for the software based on a standard project template. The technical coordinator should also develop an overall testplan corresponding to the integration tests for the application and external packages.

6.3 Deploy infrastructure

In parallel to establishing the contents of each software release, the technical coordinator can start deploying the infrastructure necessary to support the release procedure. This will require the cooperation of the team members developing software that shall be expected to start making use of the various tools and facilities as they become available. In particular, packages that identifying the unit tests that can be performed before delivering the software to the `dev` branch for integration.

6.4 Gather resources for a development deployment

In parallel to the above steps, the technical coordinator can contact the team members to establish a distributed development plan. With the experience already gained from the

releases, the technical coordinator is in a position to give details of the minimum number of pull requests required at each branch to represent a useful development deployment.

7 Checklists

This section is intended to provide convenient check-lists for members of the project concerning software release planning activities.

7.1 Technical coordinator

- Produce basic work-plans identifying, for each proposed software release, prioritised list of extensions/modifications with names of responsible individuals and estimates of effort involved;
- Organise migration to proposed toolset for code repository, build tools, and document production;
- Prepare test-plan document outlining what aspects of the application can be tested in isolation and those that have external dependencies;
- Build/develop features in accordance with priorities and development plans;
- Develop/collect test case scenarios and associated code/data suitable for integration tests;

7.2 Developers / team members

- Propose work-plan for software release integration activity identifying the order in which individual external toolkits, packages and software component should be integrated and the roles and responsibilities for individuals (including release document preparation);
- Build/develop features in accordance with priorities and development plans;
- Propose a testplan document for the integration tests;
- Continue preparation/extension of toolset for project-wide software infrastructure;
- Develop/collect test case scenarios and associated code/data suitable for integration tests;
- Propose procedure for deployment for successive software releases;