|  |  |
| --- | --- |
|  | |
| **CONFIDENTIAL DOCUMENT**  The reproduction and distribution of this document is forbidden without written authorization from Medial EarlySign LTD | |
|  | |
| **Document type**  **Title** | Quality Management System Work Instructions  ***Development Work Instructions*** |
| **Doc No’**  **Supersedes** | RDI-04-11-01, Rev. E, Effective Date: 29-May-2024  None |

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared/ Revised by:** | | | |
| **Title** | **Name** | **Signature** | **Date** |
| Head of Data Science | Alon Lanyado |  | 29-May-2024 |
| **Approved by:** | | | |
| **Title** | **Name** | **Signature** | **Date** |
| Senior Director of Engineering | Eldan Halberthal |  | 29-May- 2024 |
| VP of Clinical Research and Compliance Officer | Eran Choman |  | 29-May- 2024 |
| CEO | Ori Geva |  | 29-May-2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Log | | | |
| Revision  Status | Change Summary | DCR No. | Date |
| A | First issue | 13-24 | 29-May-2024 |

***Development Work Instructions***

|  |  |
| --- | --- |
| **General/ Scope** | This Work Instruction (WI) prescribes the instructions of Medial’s development of AlgoMarker framework. |
| **Purpose** | The purpose of this procedure is to define the process of development. This procedure covers AlgoMarker’s development activities for Medial Medial EarlySign Ltd. |
| **Applicable Standards and**  **Regulations**  **Reference Documents** | * ISO 13485:2016 * ISO 9001:2015 * ISO 27001:2022 * FDA Quality System Regulations 21 CFR 820 * RDP-04-11 - Software Development Life Cycle Procedure |

# Abbreviations and Definitions

| **#** | **Term/ Acronym** | **Definition** |
| --- | --- | --- |
|  | API | Application Programming Interface |
|  | OTS | Off The Shelf Software |
|  | SDLC | Software Development Life Cycle |
|  | SOUP | Software Of Unknown Providence |
|  | SRS | Software Requirements Document |
|  | PRD | Product Requirement Documents |
|  | SW | Software |

# Development Process Overview

Development of predictive models follows the SDLC path, however in most cases there is a research project/period that precedes the decision to “productize” a predictive capability, that is to develop an AlgoMarker and package it. During the research an initial PRD may be worked, on and software requirements may be created for that, however, after the decision to “productive” the PRD and SRS will be revisited, updated and rewritten to align with the research findings and to maximize the potential product market fit and viability.

We highlight in the process the starting and ending points in terms of configuration management and the process that takes place during the coding, unit test and V&V activities.

The following diagram visualizes the process:



1. Configuration Management Procedures

## Opening Project

When development starts, each developer creates or connects to the git repository for the project as described below:

The source code is divided into several git repositories:

* MR\_LIBS – main library code, contains the MedModel + AlgoMarker. This part is backward compatible with old AlgoMarkers. The AlgoMarker shared library requires this git repository and only this.
* MR\_Tools – Our tools/ETL tools that uses MR\_LIBs for internal usages. Not exposed to client
* MR\_Scripts – some scripts, environment tools we use
* MR\_Resources – some resource we use or want to document like configuration examples
* Specific git repository for each project/AlgoMarker (see table below)

The full structure is described in the internal confluence .(<http://confluence:8090/display/WIK/Moving+to+the+new+git+repositories>). New employee landing page contains more information on how to setup the development environment. There is a convention on where to store those git repositories for each user.

The new repositories will all be under /nas1/UsersData/{user}/MR  
Here's a sketch of the structure to create and work with (each leaf in this drawing is a git repository):

| Folder | Description |
| --- | --- |
| nas1 |  |
| UsersData |  |
| {user} | For example : Alon |
| MR |  |
| Libs | Shared repository |
| Tools | Shared repository |
| Projects |  |
| Resources | Shared repository |
| Scripts | Shared repository |
| Shared | Underneath this folder will have project repositories which may or may not be shared, depending on team structure. |
| { work project } | This is a project or a project group, for example: LungFlag. Each project has its own folder and git repository placed in the project folder or project group.  you can put any depth hierarchy of directories and git them at any height, it is recommended to do so if you need. |

After opening the first project, the user needs just to create the { work project } folder for his project.

* 1. Git methodology:
* The following methodology and principles are used during the development process -

1. Before starting to work on a new feature/fix, please git pull all the git repositories.
2. When our tools get complied, the git commit “hash” is stored in our tools/library. We store the version for each of our main git repositories in each of our tools. It is done automatically.

If needed to jump back to a certain version, the git hash is documented in the model (the code version we used to train the model) and in the AlgoMarker, the library code version that actually runs the model and can be different. No need to use git “tags”.

1. Preferred to work on master branch – no need for other branches and merges. Open a new branch only for major and breaking changes. Breaking means old AlgoMarkers will fail to execute.
2. Avoid “big” pushes with multiple changes. Each push should be a single/several commits for a single feature.
3. Your code must be compiled without any warnings before each commit! Even local commit
4. You must test your code. It is divided into two use cases:
   1. Your code is a new feature with no effect on old AlgoMarkers or minor effect. Minor means have no effect on model scoring/performance. For example, new imputation method that can be used in new models, have no effect on old models that use different imputation method. It’s enough to document the usage in confluence with a test case and push the new feature.  
      In the code documentation and in the confluence, you MUST specify the new feature are “Experimental”/Not fully tested.
   2. If a change can break old models
      1. Documented test of the code change – please put the test code in certain git repository.
      2. Document this change in the confluence.
      3. Code review is required + going over the test in the code review is required.
5. When changing some part in the code in MR\_LIBS or MR\_TOOLS, it is recommended to check who wrote the code using the ”blame” feature in git to consult with before doing the change.

### Data Wherehouse

We will store the Raw data files recieved from clients under drive T in windows or under linux in /nas1/Data. We will open new folder for each data source. directory and the processed repository files under /nas1/Work/CancerData/Repositories

* 1. Pre-Release Configuration Management (on kit construction)

For each AlgoMarker we will store “score compare” files that contains “raw data” inputs, and their “results” for future validation that results are unchanged when provided with same input.

In each AlgoMarker a directory with “score\_compare” will appear inside the folder. There is also “scripts” folder that creates those scores compare files and has reference to the data repository to pull data from and MedSamples.

There will be a README file with a pointer to the directory with the final develop model test results – it will contain the model, the samples used for training, logs, performance analysis and results of Test Kit on the model.

The README file will contain also changes from the previous version.

The library is backward compatible, future release will support old AlgoMarkers. No need to return to exact model/library version to reproduce results. Anyway, exact model/library git commit can be found in the discovery file. The AlgoMarker folder is in standard format that is ready for middleware/AlgoAnalyzer

SonarQube will be executed in this phase to find security vulnerabilities.

Additional vulnerability checks are preformed manually for SOUP/OTS

# Tools Being Used

| **Tool** | **Description** |
| --- | --- |
| Bitbucket | Installed internally, the git repository |
| SonarQube | For SAST |
| Visual Studio | IDE for C++ |
| Standard linux build tools | cmake, gcc |

# Roles and Responsibilities

| **Role** | **Responsibility** |
| --- | --- |
| Head of Data Science | Overall responsibility for this procedure and training the team on how to work under these processes. |
| VP Product and R&D | authorized to approve modifications or revisions to this procedure. |
| Management Representative (MR) | maintaining the relevance of this procedure to Company needs and applicable requirements and for ensuring ongoing compliance with this procedure. |