



# Code Review Meeting, June/July 2018

---

## Executive Summary

This report documents the results of a technical review of the software architecture and code base of the openIMIS package. The review was done In June / July 2018 by a group of technical experts from GIZ, SwissTPH and independent consultants to support the strategic and technical roadmaps for openIMIS. Several influencing aspects were analysed for the whole application (crosscutting) or according to functional areas of the application.

The reviewed functional areas, which could form a basis for a future modularisation, are defined as Master Data Management, Insurance Product Definition, Insuree Management, Health Facility Registration, Claiming, Client Feedback, and Analytics. The aspects for the analysis of each area included the adaptability of the application to user defined business processes, data needs and local settings such as the support for local languages. Another analysed aspect is complexity if one functional area was to be isolated as an independent module.

While some functional areas do offer a good amount of flexibility within existing parameters (for example product definitions), the implementation of user defined business processes or additional data needs require the interventions of developers in all functional areas. Two languages are currently supported but further development work is needed to fully support internationalization. While there are isolated aspects in the current architecture (mobile apps, analytics), significant amount of work is needed to reveal the dependencies between different functional areas and modularise the application.

The analysis of cross-cutting aspects related to security, authentication but also to the standards expected from open source applications (technology stack, openness to collaboration, ease of installation, development and code quality). There is no urgent need for an immediate “big-bang” rewrite to move away from the currently used .NET Framework, but there are severe security issues that need to be urgently addressed in the current code base. There are also issues in the public code repository which can impact the perception of openIMIS by new developers, which includes a clean code-base split into well-defined boundaries, regular releases with documented changelogs, improvements to the easiness of setting up a development environment and packaging and distribution of new versions of the application.

In summary the results from the code review sessions support the overall strategy for the further development of openIMIS. The review has helped identify the requirements for modularization of the existing system without changing the entire code base, which in turn will allow for gradually introducing functional modules based on a technology stack more in-line with the OpenHIE community. Additionally there needs to be changes to foster the collaborative development expected by Open Source solutions. The seamless and successive integration of new modules based on a different technology stack and their distribution will require immediate action in the current code base. As a whole these changes will open a path towards a more maintainable and modular system.

# Content

## [1 Introduction](#)

[1.1 Background](#)

[1.2 Proposed Team](#)

[1.3 Suggested Reading](#)

## [2 Evaluation of Cross-Cutting Topics](#)

### [2.1 Security](#)

[2.1.1 Authentication](#)

[2.1.2 Authorisation](#)

[2.1.3 Data Transfer Encryption](#)

[2.1.4 Audit Logs](#)

[2.1.5 Data Protection \(according to GDPR\)](#)

### [2.2 Localisation](#)

[2.2.1 Languages](#)

[2.2.2 Date Time Format / Calendar](#)

[2.2.3 Local Alphabets](#)

[2.2.4 Right-to-Left Writing Alignment](#)

[2.2.5 Conclusion](#)

### [2.3 Tech stack](#)

[2.3.1 NET Framework](#)

[2.3.2 Visual Basic](#)

### [2.4 Platform Independence](#)

### [2.5 Open Source](#)

[2.5.1 Code Availability](#)

[2.5.2 Code Quality](#)

[2.5.3 Open Source](#)

[2.5.4 Documentation](#)

### [2.6 Packaging / Distribution](#)

[2.6.1 Build](#)

[2.6.2 Packaging \(Publishing\)](#)

[2.6.3 Installation](#)

[2.6.4 Upgrades](#)

[2.6.5 Move an Instance to New Server](#)

### [2.7 Testing](#)

### [2.8 Contribution Model](#)

### [2.9 Differences between Implementations](#)

## [3 Evaluation by Functional Area](#)

### [3.1 Master Data Management](#)

[3.1.1 Business Processes](#)

[3.1.2 Data Elements / Attributes](#)

[3.1.3 Localization](#)

[3.1.4 Modularisation](#)

[3.2 Insurance Product Management](#)

[3.2.1 Business Processes](#)

[3.2.2 Data Elements / Attributes](#)

[3.2.3 Localisation](#)

[3.2.4 Modularisation](#)

[3.3 Insuree Management](#)

[3.3.1 Business Processes](#)

[3.3.2 Data Elements / Attributes](#)

[3.3.3 Localisation](#)

[3.3.4 Modularisation](#)

[3.4 Registration with a Facility, Pre-authorization](#)

[3.4.1 Business Processes](#)

[3.4.2 Data Elements / Attributes](#)

[3.4.3 Localization](#)

[3.4.4 Modularisation](#)

[3.5 Claiming](#)

[3.5.1 Code Base](#)

[3.5.2 Business Processes](#)

[3.5.2.1 Data Flow:](#)

[3.5.2.2 Web Service Level](#)

[3.5.2.3 Processing at Server Level:](#)

[3.5.2.4 Submission:](#)

[3.5.2.5 Conclusion:](#)

[3.5.3 Data elements / Attributes](#)

[3.5.4 Localization](#)

[3.5.5 Modularisation](#)

[3.6 Feedback Loop \(Insuree / Insurance Operator\)](#)

[3.6.1 Code Base](#)

[3.6.2 Business Processes](#)

[3.6.3 Data Elements / Attributes](#)

[3.6.4 Localization](#)

[3.6.5 Modularisation](#)

[3.7 Analytics Functions](#)

[3.7.1 Code Base](#)

[3.7.2 Business Processes](#)

[3.7.3 Data Elements / Attributes](#)

[3.7.4 Localization](#)

[3.7.5 Modularisation](#)

[4 Summary](#)

[4.1 Cross Cutting Aspects](#)

[4.2 Functional Areas / Modules](#)

[4.3 Pain points](#)

[5 Appendix](#)

[5.1 TRM Suggestions for a “Way Forward”](#)

[5.2 Out of Scope for analysis - Interdependencies between Modules](#)

# 1 Introduction

## 1.1 Background

openIMIS needs to evolve from MS IMIS to a generic standard product that can be fully customized and scaled to the needs of a growing number of implementing organisations. We need to realize a quick analysis of the source code considering its performance and persistence in terms of developments foreseen in the technical roadmap. The idea is to maintain the MS Visual Basic core while at the same time starting changes towards modularity and to explore options for a successively adding new / replacing old modules on the basis of open source technologies.

Therefore we need to analyze the following aspects:

- How future-proof is the actual core in this regard
- How long / to what extent does the core support existing technical and functional requirements,
- If changes are needed, do these apply to the entire core, or only to parts among others.

The analysis will be very high level with short examples from real code on the basis of the current masterversion in the github repository.

The review should cover all aspects from the suggestions for a [way forward](#) from the technical roadmap, and is divided between [evaluation of cross-cutting aspects](#) and an [evaluation per functional area](#).

The following elements were identified as out of scope for the current review

- Link individual code files to functional areas (see [here](#))
- Interdependencies between modules
- C# API

## 1.2 Proposed Team

Name	Organisation	Designation
Uwe Wahser	GIZ HSP Kenya	Coordination
Nirmal Dhakal	GIZ Nepal	Developer, Systems Architect
Saurav Bhattarai	GIZ Nepal	Developer, Systems Architect
Alex Vanobberghen	SwissTPH	Developer, Systems Architect

Hans van Hoppe	Exact	Developer, Systems Architect
Nils Kaiser	Freelance	Open Source Consultant
Dragos Dobre	SwissTPH	Developer, Systems Architect

### 1.3 Suggested Reading

The following documents can serve for further reference and preparation:

- [Technical Documentation](#)
- [Code Repository](#)
- [Health Insurance Board](#) and [Indicators](#) from Nepal
- [SonarQube analysis](#)

## 2 Evaluation of Cross-Cutting Topics

### 2.1 Security

#### 2.1.1 Authentication

- Web Application:
  - username and password authentication with session storage ([code here](#))
  - The password is encrypted using a symmetric key created by executing the SETUP-IMIS stored procedure ([code here](#))
  - Issue: passwords can be retrieved through the database and through the web application!
- Web Services: no authentication
  - Issue: freely accessible by anyone
- Mobile applications:
  - IMIS app: username and password for synchronisation of the data with the web services. (Note: the IMIS app is a compilation of the former individual apps: Enrolment, Enquire, Renewals, Feedback)
  - Issue: Claim & Enquire apps: no authentication required to use the app, all local are accessible

Score: 2

#### 2.1.2 Authorisation

- Web Application:
  - user role based authorisation for executing actions (i.e. add/edit/delete a family)
    - The roles of the user is stored in the DB: binary encoded limiting the number of possible roles ([code here](#) and [code here](#))
    - The possible actions and their associated rights are hardcoded in the code ([code here](#)). In the application logic, checks are done according to the predefined rights ([code here](#))
    - Issue: Only for the user interface and not on the business layer, there might be a chance for malicious clients to bypass the authorisation
  - Location based authorisation
    - User can not access data from other location levels which he/she 's assigned to
- Web Services:
  - Issue: do not have authorisation
- Mobile applications
  - IMIS app has built-in Enrollment Officer ID based authorisation (asks for EO id at startup)
  - Username and password is required to synchronise data (Enrolment, Feedback, Photos) with the central database
  - Claim & Enquire apps: no authorisation

Score: 2

### 2.1.3 Data Transfer Encryption

- Web Application: supports SSL data transfer, managed by IIS
- Web Services and Mobile Applications:
  - Supports SSL for form data transfer (to be confirmed [code here](#))
  - Use FTP for insuree photo transfer from mobile app to the WS ([code here](#), replacement with HTTP transfer is in progress)

As part of the web server configuration an implementer or administrator can ensure completely encrypted data transfers. Only transfer of photos is running through an unencrypted FTP server.

Score: 1

### 2.1.4 Audit Logs

- Logins and logouts are registered in logs
- Every record in the database has a “ValidityFrom” and a “ValidityTo” field which can be used to track changes that were made, when and by whom.
- Existing data records are not modified, only their validity status is updated. Any change generated by a user action results in a new record.
- Issue: the non-functional fields ValidityFrom and ValidityTo are also used for business logic.
- Historization of data is available. But no real technical audit trail.

Score: 2

### 2.1.5 Data Protection (according to GDPR)

- Issue: Clients’ personal data, and medical history are stored in plain texts in the database
- Issue: No measures taken to comply with personal or medical data protection standards
- Issue: Data registered on the mobile devices are stored not encrypted

Score: 3

## 2.2 Localisation

The localisation management is configured on application level, which means that particular components of an application uses the same configuration and the same functionality as the whole app. The applications which integrate localisation functionality are the Web application and all Mobile applications.

### 2.2.1 Languages

- Web application:
  - Supports limited to 2 languages
  - The translation is handled through resource files and database tables (the latter being the restriction to 2 languages: only 2 columns to handle 2 translations)
  - Changing translation resource needs recompiling the Web App and the Mobile Apps
- Web services:



- Languages are handled through the DB
- Result codes are sent as integer
- Mobile application:
  - The same logic applies for [retrieving](#) the dropdown menus' options from the database, as mentioned above for the web application.
  - Needs to be translated separately, i.e. not using the same resource file as the web application
  - Translation is managed through XML files, one for each language. [English](#), Kiswahili, etc. In order to add or change a language, the apps must be recompiled.

When the user logs into the web application, the user's languageID is retrieved from the DB and is stored in a cookie as a value ("en" or "fr", etc) and used for all subsequent requests: code [here](#).

The language resources are stored in separate [files](#) (one per language). These resource files contains all UI labels and messages (error, notice, ...). From this perspective, there is no limitation in the number of languages.

The translation tables in the DB can only handle 2 languages: English (default) and an alternative table. When rendering the web page, the logic is to test the user's language cookie's value and choose the appropriate language accordingly: [code here](#).

Score: 2

### 2.2.2 Date Time Format / Calendar

- The date format expected by openIMIS system is DD/mm/YYYY. The [installation procedure](#) explain how to change the server date format in order to match the required openIMIS format.
- Some functions use string operators on dates that only work on DD/mm/YYYY.
- Issue: Special calendar systems are not customizable (Nepali calendar was only integrated in the Nepali version, only in the web-frontend)
- Issue: Not customisable

Score: 3

### 2.2.3 Local Alphabets

- Unicode support

Score: 1

### 2.2.4 Right-to-Left Writing Alignment

- Only left-to-right user interfaces
- Issue: Right-to-left not supported in the applications (web app and mobile apps)

Score: 3

## 2.2.5 Conclusion

- The translation process is defined (by editing resource files), The language resource files contains error messages and gui-labels. As the Web application is not separated in modules, this translation system is applied on all functionalities of the Web application.
- Dropdown menus' options are stored in the DB and limited to 2 languages.
- Editing these dropdown options requires direct access to the database
- As a result, the entire solution is limited to 2 languages

Total Score: 2

## 2.3 Tech stack

### 2.3.1 NET Framework

OpenIMIS was developed based on .NET Framework 3.5, which was released in 2007<sup>1</sup> and is currently on version 4.7.2, with 10 versions released since that release. IMIS' web interface uses the Web Forms, which is still available in the current .NET version 4.7.2. The application is known to run flawlessly in .NET 4.5. Different installations are using different versions. As .NET is backward-compatible, updating to the newest version is easy.

Note: [ASP.NET Core](#) is a new cross-platform environment of the .NET family, which is a completely different technology and is not compatible with .NET Framework.

### 2.3.2 Visual Basic

Most of the application is written in VB.net. While Visual Basic retains some popularity as a simple language to learn, there is some indications that Microsoft itself rather sees the future in C# an F# rather than VB. For example, while all 3 languages are supported in the .NET environment, VB only supports a subset of functionality<sup>2</sup>. VB is not regarded as a popular language choice among open source developers.

General popularity of Visual Basic vs. other Web languages:

List	VB	Java	C#	JavaScript	PHP	Ruby	Python
<a href="#">Tiobe</a>	5 (.net) and 19 (vb)	1	6	8		12	4
<a href="#">PYPL</a>	14	2	5	3	4	11	1
<a href="#">Github</a>	-	3	8	1	5	4	2

<sup>1</sup> [https://en.wikipedia.org/wiki/.NET\\_Framework#Release\\_history](https://en.wikipedia.org/wiki/.NET_Framework#Release_history)

<sup>2</sup>

<https://www.infoworld.com/article/3051066/application-development/microsoft-c-visual-basic-are-now-set-to-diverge.html>

## 2.4 Platform Independence

Achieving platform independence allows to reduce license costs, increase the reach of the application, lowers the barrier for participation and increases future-readiness of the product.

For reference, examples of license costs required to run OpenIMIS today are:

- Microsoft Server:
  - Licenses are usually included in server purchase
- Microsoft SQL Server:
  - Web application runs on SQL express version (free), however using analytics requires a full MSSQL license
  - Nepal: one-time purchase \$2795 for 4 cores (Standard Edition 2016)
  - DRC: one-time 8698CHF Microsoft SQL Server 2016 Std Core min. 4 cores (was not purchased, only a quote)

Platform dependent frameworks and tools:

Module	Platform	Cross-platform approach	Complexity of cross-platform
Web Application & Web services	developed using the .NET Framework 3.5 which is dependent of Microsoft Windows Server operating systems	there are ways to run .NET code on Linux, however <a href="https://www.quora.com/Can-an-ASP-NET-website-be-hosted-on-a-Linux-server">https://www.quora.com/Can-an-ASP-NET-website-be-hosted-on-a-Linux-server</a>	High
Windows Services	Background jobs are implemented as windows services that only run on windows	Extract to platform-independent services (cron-jobs or similar)	Low-Medium
Database	Uses Microsoft SQL Server. 84 Stored procedures	Stored procedures need to be extracted to services in business layer  Ensure compatibility of DB queries with other SQL servers	Medium-High
		MSSQL can run on linux or docker <a href="https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-linux-2017">https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-linux-2017</a>	Low

Analytics	Uses the Integration and Analytics services from SQL Server. (Needs Standard version at least)	MS SQL Server can run on Linux, but Analytics services are not supported on Linux  <a href="https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-release-notes?view=sql-server-linux-2017">https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-release-notes?view=sql-server-linux-2017</a>	High
-----------	--	--	------

Platform independent frameworks and tools:

- The Android mobile applications are developed using Java

Score: 2-3

## 2.5 Open Source

While Open Source by definition relates to the code being available publicly, a secondary aspect is the collaborative nature of development.<sup>3</sup>

### 2.5.1 Code Availability

**Update - This section is outdated as the repositories have been partially cleaned up and reorganized upon sharing of the results.**

The openIMIS application code is available on [Github](#). However, the repository suffers from multiple issues.

1. The github repo contains many different components with different dependencies and installation procedures. A process to split into different repositories is in progress.
2. The initial github repo contains only ~8% of code, with the remaining 92% taken by build artefacts, generated documentation as well as dependencies. New application based repositories with clean source code are created.
3. Stored procedures are contained in the code repository as part of a Database backup file, but not as code which can reflect changes over time
4. Branches are not intuitive and it is unclear how those related to each other.

These issues make it hard for external developers to understand the code and thus will likely have a negative effect on adoption and participation.

The following table offers an overview of the code repository by application modules extracted from [this spreadsheet](#). Note that the stored procedures are not included in this overview.

module	submodule	# of files	# of lines
--------	-----------	------------	------------

<sup>3</sup> See [https://en.wikipedia.org/wiki/Open-source\\_model](https://en.wikipedia.org/wiki/Open-source_model)

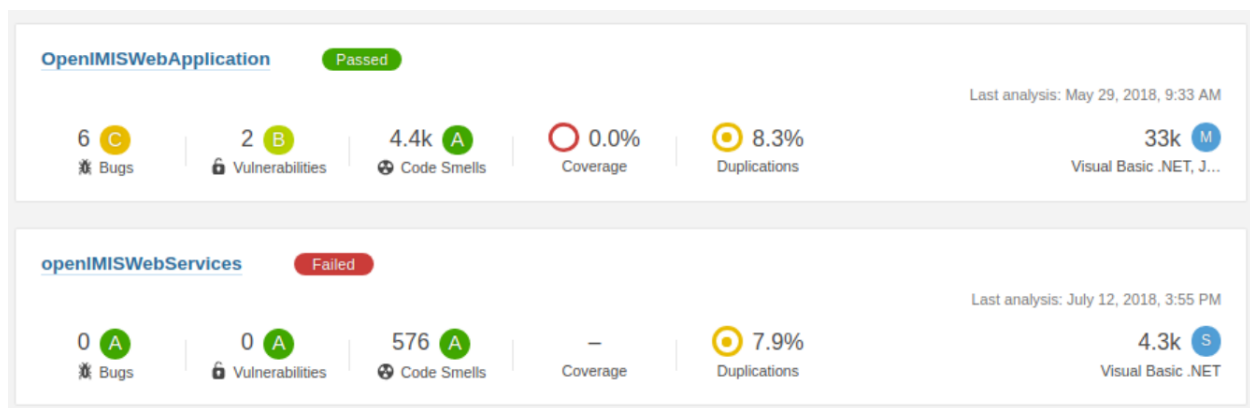
webapp	webapp	192	91598
	webapp-bi	61	4428
	webapp-bl	50	7402
	webapp-dal	58	10316
	webapp-en	22	28306
mobileapp		42	31852
mobileapp-claims		86	12228
mobileapp-enrollment		75	10267
mobileapp-enquire		74	9919
webservice-getcredentials		22	6182
mobileapp-renewal		17	3401
analytics		3	2655
winservice-backup		30	2247
winservice-effectivesms		15	1706
winservice-renewal		14	1693
winservice-assignphoto		14	1484
winservice-feedback		15	1175
		1	33
<b>Grand Total</b>		<b>791</b>	<b>226892</b>

Score: 2

## 2.5.2 Code Quality

SonarCloud (static quality analysis) revealed a number of issues:

<https://sonarcloud.io/organizations/openimis/projects>



Webapp code is organized by application layers, instead of functional areas. This does not allow to understand how different parts of the application relate to each other. However, the business functional behaviours and the database access functions are split by entity related classes.

Score: 2

### 2.5.3 Open Source

While the main requirement of Open Source is the public availability of code and a license that allows for modifications, Open Source projects have numerous requirements which will defined their perception by external developers.

We decided to follow a checklist to guide an assessment of different criteria important for Open Source projects. It can be found at: <https://afonsopacifer.github.io/open-source-checklist/>. Click on the individual items for get additional resources.

Item	Result	Notes
<a href="#">Add a README.md file containing all the basic information about your project.</a>	No	Only a pointer to installation instructions. See a template here: <a href="https://gist.github.com/PurpleBooth/109311bb0361f32d87a2">https://gist.github.com/PurpleBooth/109311bb0361f32d87a2</a>
<a href="#">Create a CONTRIBUTING.md file containing all the necessary information so that other developers can help you.</a>	No	No Contributing file. Development file
<a href="#">Add a .gitignore file containing everything you do not want versioning .</a>	No	Git repository is polluted by build artefacts.
<a href="#">Select and add the best license for your project.</a>	Yes	LICENSE.md
<a href="#">Choose a methodology versioning and follow.</a>	No	Legacy versioning methodology is enforced in Tanzania, but not yet reviewed for openIMIS. (2 <a href="#">Releases</a> on github are Tanzania-specific)
<a href="#">Create a roadmap to make clear the course of the project.</a>	No	Current roadmap is a vision document, with high-level goals. However there is no roadmap with estimable, self-contained tasks / features. (There is a <a href="#">placeholder</a> on the Wiki, will be filled by technical advisory group)
<a href="#">Document your project in the best possible way.</a>	Yes	User Manual
<a href="#">Describe your Releases clearly and objectively or create a CHANGELOG.md file and do it there.</a>	No	No changelog file
<a href="#">Use badges to indicate the status of your project.</a>	No	

<a href="#">Configure and add a .editorconfig.</a>	Partial	However, VS project files are included.
<a href="#">When necessary do not hesitate to create a website for your project. Enjoy and publish with github pages.</a>	Yes	
<a href="#">Write automated tests</a>	No	
<a href="#">Integrate your tests a Continuous Integration service.</a>	No	
<a href="#">Provide your project in the appropriate package manager. (EX: bower, npm, etc...)</a>	N/A	No packages provided in project managers nor github, sources and compiled application is committed into github repository

Score: 2

### 2.5.4 Documentation

Different documentation is available

Type	Description	Rating (coverage / quality)
User Manual	Describes each button / feature visible on the web application front-end.	Web application is well covered Mobile application is not covered
Data model & architecture	Functional Design Specification, where the database model is described, along with some business processes within openIMIS	Web application is well covered Mobile application is partly covered Web services are covered in a limited way
Code comments	Code comments, incl. description of classes, interfaces, functions and parameters	Comments are rare both in code and class / method signatures.
Installation Manual	Describes installation and configuration procedures	Web application Web services Windows services Mobile application / compilation

Score: 2

## 2.6 Packaging / Distribution

### 2.6.1 Build

The IMIS Web Application and Web Services are built using Visual Studio tooling. A build with MS build tools is possible, but currently not used. Dependencies are fetched via NuGet.

The application currently does not use continuous integration<sup>4</sup> service which would allow a test / staging instance to be deployed automatically, and help ensure timely integration of code into the main codebase. There are continuous integration services available to build .NET application.

The output of build is in DLL format and depends whether a debug or release build is used as well as the processor architecture.

The mobile applications that are build using [gradle](#). For new instances of openIMIS, the mobile applications have to be recompiled with the new Web Services URL.

A complete rebuild of the application takes approx a minute for the web application, same for the mobile application (20GB RAM, i5).

Score: 2

### 2.6.2 Packaging (Publishing)

The application is distributed via the github release page, i.e. installation files are found in zip-files that a user would need to install. The releases are manually created and uploaded to Github.

- The database structure can be restored from the backup file. It is not possible to extract all SQL scripts from the database because the IMIS-SETUP stored procedure is protected.
- The Web application and the Web services can be published and easily installed on a Windows Server platform
- The Windows Services are packaged with setup files
- The Mobile applications are packaged in APK files, however, they need to be re-compiled for every new instance because the address of the web server needs to be hard coded.

There is currently no documentation of the packaging / publishing process. Nepal is currently not using the github release page as Nepal is not yet using the master-version.

Score: 2

### 2.6.3 Installation

The installation process is manual, no automatic installation via a self-installing package (Setup) is provided at the moment - except for Windows services.

The installation of OpenIMIS takes approx half a day for a new developer. An subsequent upgrade to a new version takes about 1 hour.

Documentation does not include complete instructions to install the analysis module, including importing cubes.

---

<sup>4</sup> [https://en.wikipedia.org/wiki/Continuous\\_integration](https://en.wikipedia.org/wiki/Continuous_integration)



Installing a new instance (excluding the analytics module) can be done by a system administrator using the installation guide without technical assistance. However, note that the application will require some initial configuration (e.g. create users, locations, health facilities).

Score: 2

### 2.6.4 Upgrades

The demo instance of OpenIMIS is also updated manually after every release.

There is no instructions for upgrading an instance. Upgrading an instance requires a downtime.

Database upgrades are made using the provided SQL scripts (from one version to the next).

Score: 2

### 2.6.5 Move an Instance to New Server

Moving an instance from a server to another is rather trivial assuming that the Microsoft server and SQL server are installed and on the same version on both servers. In that case, only the database and application files need to be imported into the new instance.

This is also assuming that the application URL remains the same, otherwise the mobile applications would have to be redeployed.

Score: 2

## 2.7 Testing

- The current version of openIMIS is only tested manually, there is no test cases / scenarios provided to support manual testing.
- No automatic testing, there are no unit tests.

Score: 3

## 2.8 Contribution Model

3 github users have contributed to the codebase (SwissTPH and GIZ Nepal). Tanzanian developers send the sources to a test server where Swiss TPH can retrieve them and upload them on GitHub.

Tanzania and Nepal are currently contributing to different branches, which are to be merged into the master version.

Score: 2

## 2.9 Differences between Implementations

Goal is to highlight customizations that are specific to the different countries.

Implementation	Summary (# of users, version used, environment)	Specific features or customizations
<b>Nepal</b>	Using legacy IMIS version(since April 2016). Windows Server 2012 R2 .NET Framework 4.0 MsSQL Server 2016 Standard Edition Users: -3256 Enrolment Assistants through Android phones -228 Health Facilities submitting claims through web interface -115 HIB District Staff through web and mobile phones -5 Claim reviewers in HIB, Kathmandu -5 IT Administrators as backstoppers - 2 developers	Using Nepal-specific branch and testing master-version.  Configuration: -Cyclic enrollment -Mandatory first service point  Code changes: -Custom reports -Some changes in user interface to make it more user friendly -Inclusion of Nepali Calendar on user interface  - Analytics Dashboard <a href="http://www.shs.gov.np/dashboard/">http://www.shs.gov.np/dashboard/</a> is used instead of MS SQL analytics (due to license costs).
<b>Tanzania</b>	Using the master version (18.0.0) Windows Server 2012 R2 .NET Framework 4.0 MsSQL Server 2016 Standard Edition Implemented in 3 regions (23 districts) and currently being rolled out nationally to all 23 regions of Tanzania mainland Users: -3000+ Enrolment Assistants through Android phones (and offline) - 900+ Health Facilities submitting claims through web interface or offline - 4 IT Administrators as backstoppers - 6 developers.	Using master-version.  Configuration: - <a href="#">validate insurance number</a>  Code change - None

There are [additional](#) small-scale deployments in Cameroon, DRC and Chad.

## 3 Evaluation by Functional Area

The functional areas were taken from the roadmap document. For each functional area and each review aspect, a score for the complexity of the necessary changes is given (compare [6.8 Summary](#))

### 3.1 Master Data Management

#### 3.1.1 Business Processes

- Build configuration
  - Languages
  - Mobile Apps WS URL
- Installation configuration
  - IIS configuration
  - Database configuration
  - FTP setups
  - DB connection
  - Windows Services config
  - Web and mobile applications defaults in DB
- System setup (could be updated over time)
  - Locations
  - Facilities, payment methods, services and goods, price lists, insurance products
  - Users, Enrolment Officers, Claim Administrators, Payers
  - Mobile application master data

Master data are only being updated / synchronised via Web interface, flat files, api, xml-files. There are no customizable workflows for the introduction of new medical services, medical items, health facilities (accreditation process) etc. Products can be configured via the interface related to supported services and medicine.

Score: 4

#### 3.1.2 Data Elements / Attributes

- Build configuration
  - Languages to be used within the system → English and local language (see [Localisation section](#))
  - Configuration: the URL of the Web Services is hardcoded in the code and any new releases or instances must replace the URL and recompile the applications ([code here](#))
- Installation configuration
  - Connection string for DB access ([doc here](#))
  - Location for backup, SMS Gateway configuration on Windows Services ([doc here](#))
  - Defaults configuration (see [page 149 section 6.1.39 in TechDoc](#))
    - Stored in DB tables tblIMISDefaults and tblIMISDetailsPhone
    - Used on system runtime
  - Attribute customization
    - Stored in the DB tables for family relations, professions, ...

- Must be related to Localisation
- System setup
  - Locations management
    - One DB table (tblLocations)
    - Uses the Composite Pattern: Region → District → Municipality → Village
    - Uses SQL Server Views to simulate DB tables for the 4 levels (i.e. [method to get districts](#))
    - Hardcoded in the application → no remove or add a location level ([code here](#))
  - Facilities, payment methods, services and goods, price lists
    - Each data type is stored in separate DB tables
    - Hardcoded in the application and difficult to add fields
  - User management
    - Each account type is stored in a different table (tblUsers, tblOfficer, tblClaimAdmin, tblPayer)
    - Only Users from tblUsers can login to the Web Application (see [Security section](#))
    - No link between the account types (an Enrolment Officer must be registered in Users and in Officers)
    - Users defined roles → duplication of role definition?
  - Mobile Apps
    - Master data for IMIS app is retrieved on first run from the WS ([code here](#))
    - Contains the setup and DB language translation data

All attributes for master data are hardcoded into the program code. There are additional, unused attributes exist in database tables that can be used for custom defined attributes, but the need to be coded into the user interfaces.

Score: 4

### 3.1.3 Localization

- Names etc. of master data can only be defined in one language

Localisation is handled according to [4.2 Localization](#).

Score: 3

### 3.1.4 Modularisation

- The Master data is used by other parts of the system and do not depends on other subsystems. The code for manipulating master data is concentrated in several code files, there is no manipulation of master data done in other modules (e.g. Claiming, etc). Master data are accessible via shared database access.
- Access from other system (synchronisation etc) via api is not yet possible.

Score: 3

## 3.2 Insurance Product Management

### 3.2.1 Business Processes

- Business processes for Insurance products are hardcoded, but product instances can be created, deleted and heavily parameterized via the web interface (covered services or medical products).
- Master data can be configured:
  - maximum number of beneficiaries per family,
  - Contribution Amount, Insurance Period, Waiting Periods, Discounts, Cyclic enrolment options (number of cycles, cycle start date etc)

There is no workflow as such for defining new product generations with different parameters than the current defined set.

Score: 3

### 3.2.2 Data Elements / Attributes

- Data elements are not customizable from openIMIS Interface
- New elements/ attributes can be added through code/ database changes only

All attributes for beneficiaries are hardcoded in the database tables and the programm code.

Score: 4

### 3.2.3 Localisation

- Product names etc. can only be defined in one language

Localisation is handled according to [4.2 Localization](#).

Score: 3

### 3.2.4 Modularisation

- The product data is used by other parts of the system and do not depends on other subsystems. The code for manipulating master data is concentrated in product management code files, there is no manipulation of master data done in other modules (e.g. Claiming, Enrolment, etc). Product data are accessible via shared database access.
- Access from other system (synchronisation etc) via api is not yet possible.

Score: 3

## 3.3 Insuree Management

### 3.3.1 Business Processes

- Enrolment
  - A member of family is defined as head of the family, all other members are assigned to that family

- Enrollment Officer's unique code is predefined and entered in IMIS
- Officer's username and password is required in enrollment app to synchronize mobile stored data with IMIS server
- Insuree's Photo and data is taken through the IMIS App and sent to Server (current status as version 17.5.15)
  - Previous Enrolment App took photo and sent it to the Server ; during data entry respective photo is linked with the Insuree ID (using CHFID)
- Paper based enrollment form (Nepal)/register (Tanzania) is filled by Enrollment Officer
- Data entry Clerk collects the forms from Officers (as per old apps) and enter the data in IMIS (in the new apps data can be sent directly from the phone by enrolment officer) (<http://132.148.151.32/Family.aspx>)
- ID card with QR Code is provided to Insuree.
  - QR Code pre-printed on card - graphical representation of InsureeID
- Master data like Confirmation type, group type, Relationship, profession, Education, Identification Type are not configurable through web interface or API
- Renewal
  - The systems send the renewal information to the Enrolment Officers
  - The time period is defined (within which HHs are expiring) for which the system sends the list to the enrolment officer such that the enrolment officer has a sort of "check list" of HHs that they need to renew from their area
  - Renewals are done through Mobile Apps by Enrolment officers and through Web app by Data Entry Clerk

Enrolment currently includes registration of a beneficiary as a person/family and registering that person for a specific insurance scheme. The basic business process is hard coded, values of single attributes can be changed manually. Eligibility for an insurance product is validated automatically via hardcoded business rules dependent on the insurance product. There is no workflow as such.

Score: 4

### 3.3.2 Data Elements / Attributes

- Data elements are not customizable from openIMIS Interface
  - Fixed number of data elements (tblInsuree)
    - 28 fields
    - Most are already used by the system for logical flow
    - No provision to add new (eg. If some scheme operator requires 'color of eyes' during enrolment, the only option would be to use one of the existing 28 fields)
- New elements/ attributes can be added through code/ database changes only
- The maximum benefit ceiling (value of services in the insurance period) is dependent on a combination of base ceiling + additional ceiling for members beyond threshold.

All attributes for beneficiaries are hardcoded in the database tables and the programm code.

Score: 4

### 3.3.3 Localisation

Localisation is handled according to [4.2 Localization](#).

Score: 3

### 3.3.4 Modularisation

Enrolment -> Insuree Database <- Verification at facilities

Enrolment <-Product (eg. thresholds defined in product would affect enrolment contribution amount)

Enrolment App (Android)-> Enrolment (Web Services) -> Insuree Database <- Enrolment (Web interface)

Enrollment queries a number of master data tables directly from the database ([code here](#) for attaching a product to a family → creating policies). The ceiling for claims is a calculated value from maximum ceiling and threshold from product definition minus sum of claims done.in the family/policy table. After creation of API's for the claims process, insuree management could be quite modular, but affects a number of other modules.

Stored procedures are used when enrolling via the mobile app.

Score: 3

## 3.4 Registration with a Facility, Pre-authorization

### 3.4.1 Business Processes

- Registration
  - Beneficiary goes to Facility → Shows card → Facility uses Enquire app ([source here](#)) to identify via QR code or insuree's number
  - Alternative way is to use IMIS web application
  - Ideally online, but also possible offline
  - The enquiry request provides information on client identity data( Name, Gender and DOB), picture and eligibility (policy status, remaining ceiling, first service point etc. if applicable as per the configured Product/benefit package) configuration)
- Pre-authorization: same as registration with facility, no extra process

Registration with a facility is hardcoded in the server module and inquired by the mobile app. The evaluation is based on the insurance product and current status of the insuree and is hardcoded in a stored database procedure. The workflow in the stored procedure is not customizable.

Offline use of the app needs master data on an SD card allowing for a limited check on validity date of the policy and the picture of the insuree only.

Score: 4

### 3.4.2 Data Elements / Attributes

The data available for the encounter at the facility is hardcoded in the app and on server side and cannot be changed. Different organisation might require additional information to verify an eligible person.

Score: 4

### 3.4.3 Localization

- The mobile app needs to re-compiled for each language before installing the app on a phone according to [4.2 Localization](#).

Score: 3

### 3.4.4 Modularisation

- This functionality access information from Enrolment and Master data subsystems.
- URL to server is hardcoded in the app.
- Stored procedures are used a lot.

Score: 3

## 3.5 Claiming

### 3.5.1 Code Base

#### In the Web application

Front end classes:

Business logic classes in IMIS\_BL project:

- [ClaimAdminBL.vb](#)
- [ClaimDedRemBL.vb](#)
- [ClaimItemsBL.vb](#)
- [ClaimServicesBL.vb](#)
- [ClaimsBL.vb](#)

Data Access Layer classes in IMIS\_DAL project:

- [ClaimsDAL.vb](#)
- [BatchRunDAL.vb](#)
- [ClaimDedRemDAL.vb](#)

SQL Server stored procedures:

- uspSubmitClaims
- uspSubmitSingleClaim
- uspClaimSelection
- uspProcessClaims
- uspProcessSingleClaimStep1
- uspProcessSingleClaimStep2

### 3.5.2 Business Processes

The claiming process allows the healthcare provider to be reimbursed for services provided to a patient by the patient's insurer.

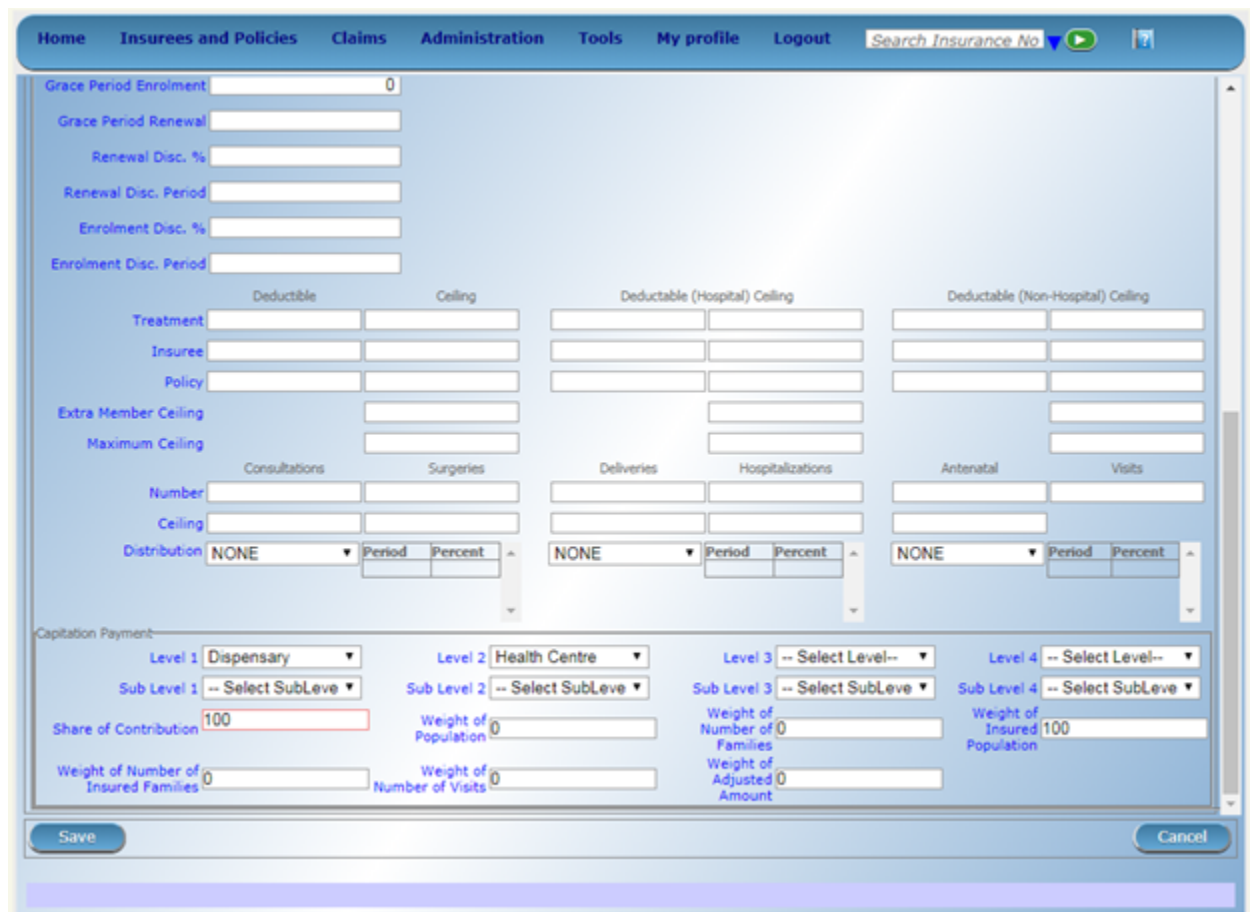
In short, the claim process includes the following steps:

1. Claim Entry
2. Claim Submission
3. Integrity checks
4. Medical review (optional)
5. Valuation of claims (at individual level)
6. Batch run (generates reports for payment at facility level)



From the Functional Design Specification, see section 5.3 (page 64) for the logic applied for the claim submission and section 5.4 for the claim valuation process.

These steps are partly customisable through configuration screens (for steps 1, 2, 3), and through parameters defined at the insurance product level (for steps 4 and 5). See this product page for example: <http://132.148.151.32/Product.aspx?p=4>. The definition of each field can be found here: [http://openimis.readthedocs.io/en/latest/user\\_manual.html#product-page](http://openimis.readthedocs.io/en/latest/user_manual.html#product-page)



It is worth noting that this does not constitute a business rule engine / manager ; any new claiming logic or new date element would require new development.

### 3.5.2.1 Data Flow:

Claims can be sent via the web application or the mobile application. For both options, claims' data are submitted and stored in the database, until the batch process is run at a later stage (end of the reporting period).

Phone data flow: XML file (one per claim) -> Web service -> MSSQL stored procedure

Online web application: Direct call to the stored procedure

Offline web application: XML file (bulk export) -> Web service -> MSSQL stored procedure

### 3.5.2.2 Web Service Level

(<http://132.148.151.32/services/exactservices.asmx>)

At the web services level, the following functions are used:

- [isValidClaim](#)

### 3.5.2.3 Processing at Server Level:

Here the stored procedures handling the claiming processes (to be viewed in SQL Server Management studio)

- uspSubmitClaims
- uspSubmitSingleClaim
- uspClaimSelection
- uspProcessClaims
- uspProcessSingleClaimStep1
- uspProcessSingleClaimStep2

### 3.5.2.4 Submission:

- One claim is checked (integrity) through uspSubmitClaims.
- It is also pre-processed i.e. deductables or ceiling are checked: uspProcessSingleClaimStep1.
- Processing:
- The claim processing takes a batch of claims, and loops through them one by one.

### 3.5.2.5 Conclusion:

- Most of the business processes related to claims are hard coded into the system and no aspects/parameters are modifiable through a GUI.
- Like every other process of the system, the claiming process is part of the core logic, and cannot be used on its own (by a third party system for example).
- There are different data flows to support different operational modes (mobile phones, online, offline). These flows introduce some code redundancy, which could be removed for ease of maintenance.
- The claiming process is rather complex overall, and was developed iteratively based on users' requests. This is reflected in the code, also means that it is not very flexible in case any new data element or logic needs to be added, the whole claiming process needs reviewing.

Score: 4

## 3.5.3 Data elements / Attributes

As described in the above business processes, numerous tables / data fields are being checked at different stages ; see diagram 2, 3, 4 and 5 (from page 158 onwards) from the Functional Design Specification.

No additional attributes can be added to the claim objects.

Score: 4

### 3.5.4 Localization

- The mobile app needs to re-compiled for each language before installing the app on a phone according to [4.2 Localization](#).

Score: 3

### 3.5.5 Modularisation

All parameters that are related to claiming are dealt with by product definition, item, and service definition (master data).

Claim submitting and processing is handled within stored procedures which are isolated and not dependant on other business logic. The results of these stored procedures are stored in tables, which are then referred to by other processes, for example the inquiry of the insuree's status. Changing the claiming module would then also impact other modules.

Score: 3

## 3.6 Feedback Loop (Insuree / Insurance Operator)

### 3.6.1 Code Base

Feedback front end classes:

Feedback stored procedures:

### 3.6.2 Business Processes

Feedback objectives:

- Increasing satisfaction
- Revealing fraud

Current process:

- At the claiming stage, the medical reviewer can flag a number / proportion of claims for feedback
- The feedback requests then become available through the mobile app to the field officer to collect feedback from the insurees.
- The above feedback prompts can received by SMS or through mobile data via the mobile app.

For more information about the Feedback please see section 8.4. Feedback & Renew application (page 190) from the [Technical Documentation](#).

Score: 4

### 3.6.3 Data Elements / Attributes

No additional data elements can be added to the feedback objects.

Score: 4

### 3.6.4 Localization

- The mobile app needs to re-compiled for each language before installing the app on a phone according to [4.2 Localization](#).

Score: 3

### 3.6.5 Modularisation

The feedback functions are part of the mobile app as an isolated class which retrieves a list of feedback requests from the server. The data are isolated in one table in the database. There is also a nightly batch run and a link to the sms system.

The feedback module is optional function and is linked to the claim module: one feedback per selected claim.

Score: 2

## 3.7 Analytics Functions

### 3.7.1 Code Base

Web application report templates:

- [Web application/Sources/IMIS/Reports](#)

### 3.7.2 Business Processes

How can a user create own fixed format (standard) reports?

Tanzania: OLAP reporting in Excel needs BI functionalities that need to be licensed (not free). Are data structures and ETL processes part of the package?

Nepal is using external tools for reporting (Google PHP Libraries) in addition to the fixed reports.

Score: 4

### 3.7.3 Data Elements / Attributes

Creation of user defined reports is very interlinked with the involved business processes.

Score: 4

### 3.7.4 Localization

For the Web Application reports, localisation is handled according to [4.2 Localization](#).  
OLAP reporting in Excel is cannot be localised to languages.

Score: 3

### 3.7.5 Modularisation

Score: 2

## 4 Summary

### 4.1 Cross Cutting Aspects

Scoring:

- 1: Implemented
- 2: Partially implemented
- 3: Not implemented

Cross Cutting Aspect	Score
<a href="#">Security</a>	2
<a href="#">Authentication</a>	2
<a href="#">Authorisation</a>	2
<a href="#">Data Transfer Encryption</a>	1
<a href="#">Audit Logs</a>	2
<a href="#">Data protection</a>	3
<a href="#">Localisation</a>	2
<a href="#">Languages</a>	2
<a href="#">Dates</a>	3
<a href="#">Local Alphabets</a>	1
<a href="#">Right-To-Left Alignment</a>	3
<a href="#">Tech stack</a>	2
<a href="#">Open Source</a>	2
<a href="#">Code availability</a>	2
<a href="#">Code Quality</a>	2
<a href="#">Open Source Checklist</a>	2
<a href="#">Documentation</a>	2
<a href="#">Packaging / Distribution</a>	2
<a href="#">Build</a>	2
<a href="#">Packaging (publishing)</a>	2

<a href="#">Installation</a>	2
<a href="#">Upgrades</a>	2
<a href="#">Move to New Server</a>	2
<a href="#">Testing</a>	3
<a href="#">Contribution Model</a>	2
<a href="#">Differences between Implementations</a>	n/a

## 4.2 Functional Areas / Modules

### Scoring:

- 1: Nothing to be done
- 2: Impact only on one application (e.g. user registration of web app)
- 3: Impact on multiple application(s) of the system (e.g. claim management on web app and web service)
- 4: Complete module needs to be redesigned

Functional Area	Bus. Proc.	Data Elem.	Loc.	Mod.
<a href="#">Master Data</a>	4	4	3	3
<a href="#">Insurance Product</a>	3	4	3	3
<a href="#">Insurees</a>	4	4	3	3
<a href="#">Registration</a>	4	4	3	3
<a href="#">Claiming</a>	4	4	3	3
<a href="#">Feedback</a>	4	4	3	2
<a href="#">Analytics</a>	4	4	3	2

### 4.3 Pain points

The following pain points have been collected during the code review.

Pain point	Possible mitigation
Versioned code does not reflect all local changes	Ensure that every country runs on a checked in version, even if those are different branches
Different country implementations maintained in branches means that code is diverging	Ensure that a basic customization mechanism is pushed to the master version (via configuration) and that country implementations are submitting pull-requests to master
Implementation is bound to Microsoft SQL (See <a href="#">2.4 Platform Independence</a> )	<p><u>Non-core modules:</u> Ensure that most accesses to data layer are done via API calls rather than database.</p> <p><u>Core modules:</u> Evaluate cost of porting data access layer to a database-independent</p> <p><u>Stored procedures:</u> Redevelop stored procedures as a service with an API.</p>
Code is developed using outdated technology	Incrementally replace non-core modules with new technologies, while development teams are gaining experience with those technologies
Deployment of a new OpenIMIS server instance is complex	Develop a windows-based Docker containerized version of OpenIMIS
Key workflows are not customizable	The application should provide clear extension points so that workflows (for ex. Claimes or registration) can be customized. This customization could happen in Code or via an interface (more complexity)
Security relevant issues (see issues in <a href="#">2.1.1 Authentication</a> , <a href="#">2.1.2 Authorisation</a> and <a href="#">2.1.3 Data Transfer Encryption</a> )	Fixes needed
Audit logs are couple with business logic (see <a href="#">2.1.4 Audit Logs</a> )	Decouple audit logs from business logic.
Data registered on mobile devices is not encrypted	Ensure that personal data in database on device is encrypted.
Data privacy issues - see <a href="#">2.1.5 Data Protection (according to GDPR)</a>	Extensive data audit needed to suggest feasible solutions for encryption of personal data.
Special calendar systems are not customizable	Support custom calendars throughout the application (Nepal support is only in front-end)



Right-to-left not supported in the applications (web app and mobile apps)	Add support for right-to-left interface in both web app and mobile app.
Code repository issues (partially resolved), see <a href="#">2.5.1 Code Availability</a>	Develop code repository guidelines (no build artefacts in repo, ensure separate modules are in separate repositories)
No automated tests or continuous integration can impact software quality and development speed.	Develop tests for new modules, set up continuous integration service.
Documentation is incomplete	Add documentation for mobile application
Manual packaging process	Packaging process should be automatic (connected to continuous integration above) and should not require manual intervention
Product names can only be in one language	Allow for product name and description in multiple languages

## 5 Appendix

### 5.1 TRM Suggestions for a “Way Forward”

From Technical Roadmap, Version 5.2, based on the OpenIMIS workshop from February 2018

Summarized suggestion by SwissTPH	Comment OpenIMIS Initiative
1. OpenIMIS should be developed based on the current IMIS, using MS VBA.	Agreed in the sense that for the time being all existing core functions should remain in Microsoft Technology. For new functions it should be analysed case-by-case whether new elements can be used (micro services, open source, existing libraries), thereby stepwise modularizing the core.
2. IMIS will be modified in such a way that it will support besides MS SQL Server <b>database</b> also other databases (e.g. open source or license free database). This will also allow running on Linux OS and gives partners technology choices. Migration tools will be prepared.	Agreed, based on specific customer demands. <i>(This has a lower priority)</i>
3. The data warehouse of IMIS will be integrated with alternate front-end besides the currently used front-end MS Excel. Priority would be DHIS2, but it should also be possible for other reporting/DWH/BI products.	Agreed. Pro-actively pursuing DHIS2 integration corresponds to requests of different partners. Still, vendor-independent approach is important.
4. The current <b>RESTful API</b> layer will be completed (and made fully compatible with the relevant standards) in order to facilitate full replacement of web form functionality in case it is needed for ensuring online communication with adjacent software systems.	Agreed. It has been a new information for us, that IMIS uses RESTful API.
5. A <b>data format</b> in conformance with HL7 FHIR will be provided for data exchange on claims as an alternative to the current native XML based format.	Agreed. In a later step, it can be analyzed how micro-service based FHIR libraries could also contribute to long-term core modularization.
6. Data formats based on CSV and/or Excel will be offered for initial loading of selected <b>registers</b> (it is already available but only for some registers and it is not published yet).	Agreed. This will speed up implementation and help transfer maintenance responsibility to countries. Also continuous data exchange with specialized registers, such as facility lists, can be explored in line with openHIE approaches.
7. The existing mechanism of <b>exit points and escape procedures</b> will be enriched to allow country specific modification of more aspects of the business logic in case an explicit configuration on the user level is not advantageous for a specific case.	Agreed. This seems like a promising approach to allow countries to inject further functionality. Since this can deeply affect core stability, the procedures should be well structured and documented. <i>(This has a lower priority)</i>

## 5.2 Out of Scope for analysis - Interdependencies between Modules

This exercise would require to assign the code files of the core (webapp) by functional area.

This analysis could be done using Code Maps

<https://docs.microsoft.com/en-us/visualstudio/modeling/create-layer-diagrams-from-your-code>

Ideally, the result would be a matrix of the interdependencies between functional areas

	Master Data	Insurance Product	Enrolment	Registration	Claiming	Feedback	Analytics
Master Data	-						
Insurance Product		-					
Enrolment			-				
Registration				-			
Claiming					-		
Feedback						-	
Analytics							-