GTS 2.0

High Level Design

By David Van Hoecke, september 2016

*The goal of this document it to describe at a high level the features, requirements and needed time / resources of the envisioned GTS 2.0 solution.*

[**1 Introduction**](#_e2nsleiguhyt) **1**

[1.1 What is GTS 2.0?](#_wdel9n4252q0) 1

[1.2 Scope](#_v9nzgba0razq) 2

[1.3 How will GTS be developed?](#_1r0ikvot3njn) 2

[1.4 Stakeholders](#_977t3ixzn3h5) 3

[**2 Description**](#_1srxfj1e3z5i) **3**

[2.1 Definitions & Terms](#_qasil6iu3z9r) 3

[2.2 Summary](#_yq88vy7ef0fl) 3

[2.3 Used technologies](#_42l1rcmrrglt) 5

[2.4 General design guidelines](#_z4qnx24oqb13) 5

[2.5 Security and data protection](#_v2t6jp96i5m7) 6

[**3 Opportunities and risks**](#_f71s66syk9ge) **6**

[3.1 Opportunities](#_fzyv1mrj9aos) 6

[3.2 Risks](#_scdn2nsupo6w) 7

[**4 Resource and time budget**](#_csi2z4m0wkfj) **7**

[4.1 Resources](#_x5c5fg7szsv2) 7

[4.2 Time budget](#_3wfucu2rdj7f) 7

# 1 Introduction

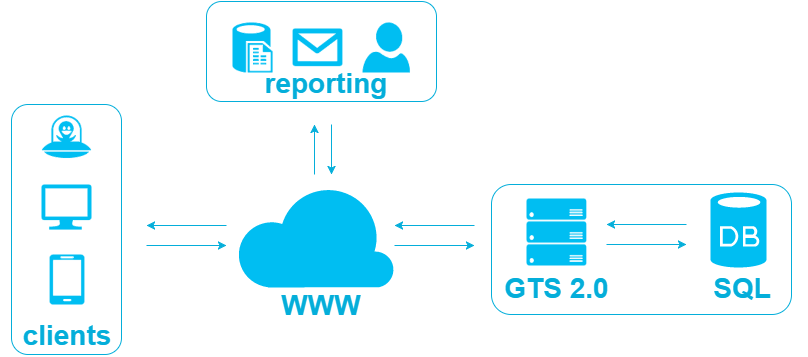
## 1.1 What is GTS 2.0?

GTS 2.0 stands for the software-as-a-service implementation of a Gatewing / Delair-Tech Track&Trace solution. The goal of this service is to be able to track the production process from start to finish as well as provide means to track and analyze user details and any problems and solutions that might occur during the production process. The provided service will focus on tracking and tracing, ease-of-use and accessibility and will provide automatic reporting and process analysis / statistics.

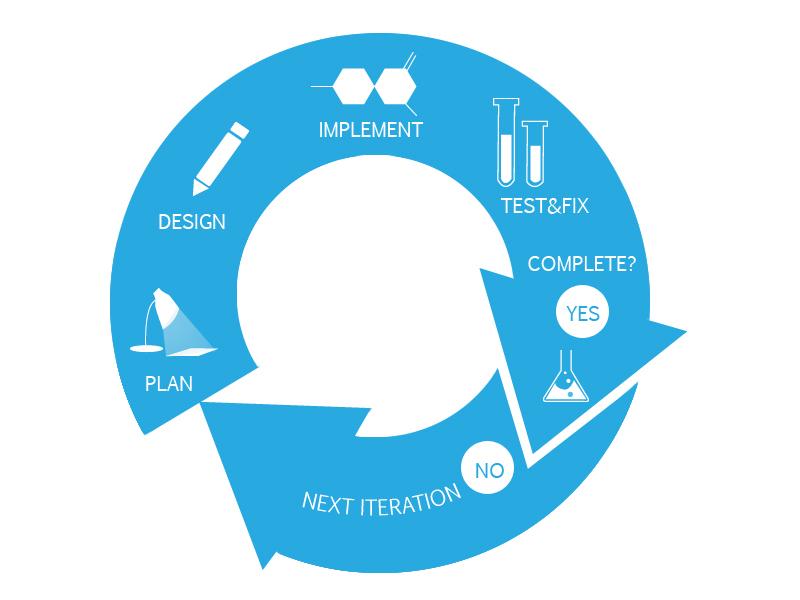
At this time we do not see the need for an interface between SAGE and GTS 2.0 although we will prepare a framework for this for when the need arises.

## 1.2 Scope

The software will be offered as a client-server solution. This enables the client side to use existing features and technologies (internet access and an internet browser, e.g.) and requires no actual rollout. It also ensures the service can be used on a wide range of devices. The server side will consist of a web environment and a datastore such as an SQL database. Additional platforms may be used such as pdf docs, spreadsheet and reporting services.



## 1.3 How will GTS be developed?

An agile, “sprint” based project track will be followed where, ideally, every sprint will provide a usable, rolled out portion of the application. Each portion will require a sign-off after acceptance by the relevant stakeholders so its features may be locked down and used as a base for further sprints. A sprint is typically two calendar weeks.

Three (or four) environments will be maintained for this solution. Development, Acceptance (, Test) and Release. Where Development speaks for itself, Acceptance is the environment where the stakeholders sign off on the development after approving the implementation after which it moves to Release. A sprint can iterate between Development and Acceptance but once it escalates to Release any changes or bugfixes will result in new versions.

## 1.4 Stakeholders

Delair-Tech and Gatewing will both be stakeholders in the long run but initially only Gatewing features this solution. Primary users will be production employees and their management. Administration of the solution will be done by production management.

# 2 Description

## 2.1 Definitions & Terms

* **GTS:** (name to be decided) Gatewing Tracking System
* **TTS:** pre-existing t&t implementation
* **SQL:** Microsoft database server
* **Software-as-a-service (SaaS):** a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software". SaaS is typically accessed by users using a thin client via a web browser.
* **Sprint:** a short development cycle after which a usable, accepted part of the software is released
* **T&T:** tracking and tracing or Track & Trace, concerns a process of determining the current and past locations (and other information) of a unique item or property. This is not limited to physical locations and can be conceptual.
* **Client:** a software resource that installed on a user’s device, in this case we refer to thin clients
* **Thin client:** a lightweight computer that is purpose-built for remote access to a server (typically cloud or desktop virtualization environments). It depends heavily on another computer (its server) to fulfill its computational roles.
* **Server:** a software resource which runs independently of a user interaction and which manages access to a centralized resource or service in a network.
* **Rollout:** the process of releasing software and installing it on a client’s resource.
* **.NET (dot net):** The microsoft .NET platform includes servers; building-block services, such as Web-based data storage; and device software.

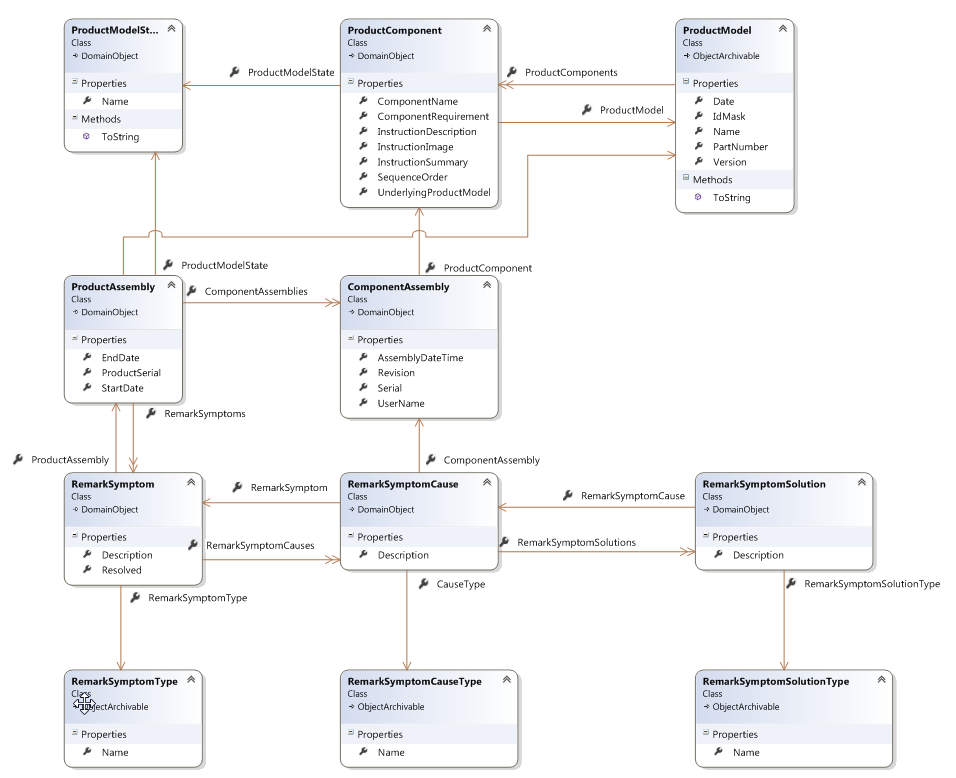
## 2.2 Summary

A product will be represented in the system as a “Product Model” which consists of one or more “Product Components” that each fall under a certain “Model State”. These components have a particular order in which they are executed. A “Product Assembly” is a physical instance of this and consist of one or more “Component Assembly” items. A component assembly can be anything from an installed component to a manually executed action or even a Product Assembly itself.

Per product assembly a current model state is persisted (saved). A product assembly state can move to a next state but not to a previous state. An exception to this is placing a product in the “Remark” state to indicate a problem. For troubleshooting purposes, administrators will be able to return to previous states.

A user identifies his/herself and persists per component assembly some metadata (e.g. a serial of a hardware component, or user initials after executing a manual action). After all components for a given state are finished (have persisted metadata), the product model is moved to the next state. This enables data entry for components of this state and prevents any changes to components of previous states. On each component created or changed, the user and corresponding time will also be persisted to enable in-depth analysis. This functions as a sort of sign-off of every part of production and effectively places the responsibility with the user that executed the action.

Issues may arise during production which block, alter or influence product or component assemblies. In such a case a product assembly will be placed in the remark state. A product in remark MAY NEVER BE SHIPPED to a customer and must receive additional actions. Only after resolving the problem the product can be removed from remark and placed in a production state again, but only by noting the cause and solution. While noting cause and solution, product component and component assemblies may be referenced. E.g. an issue causes a component to be replaced: both the replaced and the new item’s serial will be mentioned in the remark cause / solution information.



## 2.3 Used technologies

* This application will be written in C#.net, takes the form of an MVC web application hosted on an IIS server and will generate web pages to display to users.
* Data for this backend will be stored in an SQL database, namely MSSQL
* The same database will generate easily accessible reports, either by the application or by inbuilt tools found in MSSQL
* All potential client devices will already have web browsing software installed and require no roll out
* Some potential client devices will have peripherals to ensure ease of use and the application will consider this in its UI design (e.g. use of hand scanners)

## 2.4 General design guidelines

The solution will put a heavy focus on ease-of-use. An effort will be made so that users have no other tasks to perform than what is necessary from a business intelligence point of view. For instance: when a user needs to enter a piece of information that can be scanned, the only action needed would be to pick up the scanner and scan the relevant bar code. No further manipulation of the application will be required.

As much as possible the company styling and colours will be used but overall we will aim to provide a lightweight UI that is responsive, intuitive and communicates clearly.

Different application tiers will designed to implement a “Loose Coupling” strategy to be able to easily change and replace parts of the application.

## 2.5 Security and data protection

Regular database backups must be made. These backups must not overwrite previous backups and must be stored in a different physical location as the database server itself. Ideally the same backup strategy which is already in place for Gatewing will be used.

The built in security features of the ASP.NET platform are sufficient for user management and website security and all critical user details (initially only the user password) will be stored in an encrypted format. Administrators will never know and ask for user passwords but will be able to reset them if needed.

User rights will be assigned to groups and users will be placed in those groups. Two groups are planned at the start namely “Users” and “Administrators”

# 3 Opportunities and risks

## 3.1 Opportunities

A Track & Trace implementation has many benefits from a management, legislative and BI point of view. This solution will enable us to measure production efforts, provide T&T features to be used in support and after sales service and provide an in depth, statistically supported view into our production.

We have the chance to enable users to spend less time and effort on their administration and have more time to actually produce. We will achieve this by designing the UI properly and act on lessons learned from previous similar solutions (such as TTS)

As we will develop this ourselves we can benefit from years of production experience and have a clear idea of where we need to go and what we need to do. We can further base ourselves on the previous TTS solution as a starting point.



## 3.2 Risks

Development time must be limited and hopefully restricted as much as possible to the down time period at the end of 2016.

A potential high risk scenario would be to create a coupling between SAGE and GTS. In all software projects the link to an existing system in a completely different technology is one of the greatest risk factors and great care must be taken to implement this.

Care must be taken to test and debug the application so no corrupted or (unintentionally) falsified data is persisted.

# 4 Resource and time budget

## 4.1 Resources

* Visual Studio IDE license: (**free**, for current team size)
* MSSQL database license: (**free**, for current team size)
* A VisualSVN license (**99€** but we may be able to reuse an existing license for this)
* A windows web server running IIS 7.0 and a database server running an SQL database (this can be one machine for both). We have three options here:
  + We can provide the hosting ourselves, this means we ourselves set up the webserver and database on an (existing) windows machine. For this we need a windows server license (for the webserver) and an SQL license (for the database). It also means we are responsible for administration, maintenance and backup management. We can limit this to just a database license and not particularly a windows server license, since every windows installation can be configured to host a website (IIS, the windows webserver, is a native windows feature)
  + We can have Behind The Buttons provide the hosting, in which case they will make an offer which would presumably include licensing, maintenance and backup. We would still be able to perform administration, together with BtB. At this time however, BtB only provides windows 2003 hosting. This would force us to rewrite the application in a previous version of the programming language and we’d have to redesign the web site portion of it.
  + The third option is to use existing infrastructure. We do not have the infrastructure here in the Buchtenstraat but Delair Tech may have this. For this to work, an existing windows (server) machine and a database server has to be available. In this case we would perform no administration but simply roll out versions of GTS 2.0 to this machine.

## 4.2 Time budget

* Sprint **01**: vertical development (setting up DB, IIS, Website and client side code) DONE
* Sprint **02**: Product model admin pages, remark and cause & solution pages
* Sprint **03**: product page
* Sprint **04**: users and roles management pages
* Sprint **05**: reporting and statistics, backup implementation

# 5 Comments/Future/Other

## 5.1 Comments

* Further define the needs for GTS (indeed we have experience with TTS and know what to do but we should write down the requirements we learned from TTS (=”lastenboek”))

## 5.2 Future

* Couple GTS to workinstructions; be able to show the workinstruction for the current assembly step
* Uploading of test data (for example from test flights) through GTS. This would probably allow for much more structured data (instead of the google sheets/excel file we have today)

## 5.3 Other