SCIENTIFIC REPORTS

OPEN

SUBJECT AREAS:
GENETIC DATABASES
SOFTWARE
FUNCTIONAL GENOMICS

Received
14 January 2014

Accepted
18 February 2014

Published
4 March 2014

Correspondence and requests for materials should be addressed to M.L. (mlist@health.sdu.dk)

Efficient Sample Tracking With OpenLabFramework

Markus List[1,2,3], Steffen Schmidt[1,2], Jakub Trojnar[1,2,5], Jochen Thomas[6], Mads Thomassen[1,3], Torben A. Kruse[1,3], Qihua Tan[3,4], Jan Baumbach[7] & Jan Mollenhauer[1,2]

[1]Lundbeckfonden Center of Excellence in Nanomedicine NanoCAN, University of Southern Denmark, Odense, DK, [2]Institute of Molecular Medicin (IMM), University of Southern Denmark, Odense, DK, [3]Clinical Institute (CI), University of Southern Denmark, Odense, DK, [4]Epidemiology, Biostatistics and Biodemography, Institute of Public Health, University of Southern Denmark, Odense, DK, [5]Department of Biochemistry and Molecular Biology (BMB), University of Southern Denmark, Odense, DK, [6]io-consultants GmbH & Co. KG, Heidelberg, DE, [7]Department of Mathematics and Computer Science (IMADA), University of Southern Denmark, Odense, DK.

The advance of new technologies in biomedical research has led to a dramatic growth in experimental throughput. Projects therefore steadily grow in size and involve a larger number of researchers. Spreadsheets traditionally used are thus no longer suitable for keeping track of the vast amounts of samples created and need to be replaced with state-of-the-art laboratory information management systems. Such systems have been developed in large numbers, but they are often limited to specific research domains and types of data. One domain so far neglected is the management of libraries of vector clones and genetically engineered cell lines. OpenLabFramework is a newly developed web-application for sample tracking, particularly laid out to fill this gap, but with an open architecture allowing it to be extended for other biological materials and functional data. Its sample tracking mechanism is fully customizable and aids productivity further through support for mobile devices and barcoded labels.

With the development of high-throughput technologies, laboratory work has seen a paradigm shift from small projects involving single or few researchers towards large-scale projects involving several laboratories and often hundreds or thousands of samples. Sample management is therefore a growing issue, especially since most laboratories still attempt to keep track of their samples using spreadsheet tools. A high turn-over of academic staff coupled with maintenance of individual files that are often locked or outdated, as well as inconsistent nomenclature and labeling, can lead to tedious repetition of previously existing work. The significant amount of time that is often spent on locating samples would be better used for performing experiments. Moreover, expensive storage space is wasted, since samples are often not labeled properly and cannot be identified. Even if a label is given, it usually does not include a standardized minimal amount of information that allows unambiguous identification of the materials or the experiments they were derived from. Numerous commercial and open-source solutions have been developed in an attempt to overcome these problems.

Although solutions are offered by commercial companies like Labvantage®, most academic laboratories find it difficult to afford the license costs, which usually rise with additional users and technical features. The focus of this paper is thus open-source systems.

As Table 1 shows, open-source laboratory information management systems (LIMS) are often customized towards specific types of biomaterials or research data, as for instance genotyping[1–3], protein production[4,5], protein-protein-interaction[6], 2D gel electrophoresis[7], or protein crystallography[8] data. Some generic LIMS target specific laboratory tasks, such as sample management[3,9,10], laboratory work-flows and protocols[2,11–13], documentation, management of lab stocks, or clinical studies[14]. Further solutions exist for molecular genetics and the creation of vector libraries[15]. There is, however, no dedicated LIMS for the management of large vector construct and cell line libraries. At our Lundbeck Foundation Center of Excellence in Nanomedicine (NanoCAN) at the University of Southern Denmark in Odense such large-scale libraries need to be handled efficiently (see[16] for a short overview about our work). This motivated us to develop a novel open-source LIMS platform: OpenLabFramework (OLF).

Results

Any LIMS that involves sample management on a large scale should fulfill a number of requirements listed in the following as R1-15. Existing open-source LIMS fulfill these requirements to varying degrees (Table 2).

Table 1 | Some examples of existing browser-based LIMS solutions. Corresponding project URLs can be found in Supplemental Table 1

Project Name	Ref.	Main purpose	Built with
MMP-LIMS	1	Genome mapping in maize	Java
AGL-LIMS	2	Genotyping work-flow	Java
SMS	3	Gene mutation screening & biobanking	Java
PiMS	4	Sample & experiment tracking for protein production	Java
ProteinTracker	5	Protein production & purification	Java
PARPs Database	6	Protein-protein interaction data and data-mining	Perl/Java
LIPAGE	7	2D gel electrophoresis based proteomics	PHP
LISA	8	Protein crystallography	PHP
EnzymeTracker	9	Data analysis, sample management, spreadsheet functionality	PHP
FreeLIMS		Sample management, reports	Java
YourLabData		Sample tracking and lab notebook	-
Open-LIMS		Experimental work-flow, sample & document management	PHP
OpenFreezer	10	Sample management & tracking	PHP/Python
iLAP	11	Data management, analysis, experimental protocol design	Java
SIGLa	12	Customized experimental work-flows	Java
BIKA		Whole lab work-flow for clinical studies	Python
MicroGen	13	Mircoarray information and work-flow	MS-Access
LabLog		Project Documentation	Java
LabStoRe		Chemical lab stocks	PHP
SIMBioMS	14	Linking experimental, patient, and high-throughput data	Java
MolabIS	15	Molecular genetics data	Perl

Implementation. A LIMS for an academic environment needs to be open-source (R1), in order to save costs and to allow for adaptation to the specific requirements of a given scientific field and laboratory. Since adaptation can be a difficult and time-consuming task, a LIMS that is modular and extensible by design (R2) would be most appropriate. Although difficult to assess for existing projects, a LIMS should be reliable and its implementation simple. Existing frameworks and software packages that are maintained and tested by a large community are often more reliable than individual solutions and should thus be incorporated.

Data handling. Dealing with a large number of samples in a library or biobank requires efficient mechanisms for sample management (R3) and physical sample tracking over several hierarchical levels (R4).

Table 2 | Feature Comparison of requirements across browser-based LIMS solutions for sample management using the following abbreviations: EnzymeTracker (ET), Free-LIMS (FL), SLIMS (SL), YourLabData (YL), Open-LIMS (OL), ProteinTracker (PT), AGL-LIMS (AL), SMS (SM), MolabIS (MI), SIMBioMS (SI), OpenFreezer (OF), and PiMS (PS). o depicts limited fulfilment. Sample Tracking refers to the physical location of samples. Local Deployment refers to a local installation not requiring a database installation. Cloud Deployment refers to documented cases

Requirements		ET	FL	SL	YL	OL	PT	AL	SM	MI	SI	OF	PS	OLF
Open-source	R1	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Modularity	R2	✗	✗	✗	✗	✓	✗	✓	✗	✓	✓	✓	✗	✓
Sample Management	R3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sample Tracking	R4	✗	✗	o	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓
File Management	R5	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✓
Reports	R6	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✗	✓
Multiple DBMS	R7	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓
Local Deployment	R8	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓
Cloud Deployment	R9	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Documentation	R10	✓	✗	✓	o	✓	✓	✗	✓	✓	✓	✓	✓	✓
Barcodes	R11	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Labels	R12	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Mobile Devices	R13	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Data Analysis	R14	✓	✗	✗	✗	o	✗	✗	✗	✗	✗	✗	✗	✗
Audit-Logging	R15	✓	✗	✓	✗	✗	✗	✗	o	✗	✗	✗	✗	✗

Since related information and experimental results are usually stored in additional documents, a management system, where files can be linked to an arbitrary number of samples (R5), would be most useful. Another requirement is that raw data previously entered into the system can be exported to various file formats. This requirement is usually met through an integrated reporting mechanism (R6).

Flexibility in deployment. Academic laboratories are often part of an existing IT infrastructure, but support is in many cases limited, e.g. to a single database management system (DBMS), such as MySQL. LIMS deployment should thus be as flexible as possible not be bound to a specific operating system or DBMS. While the first requirement is fulfilled by all LIMS considered in this paper, multiple database support remains an issue (R7). Furthermore, if a suitable server is not available, deployment locally (R8) or to a cloud service (R9) is advantageous.

User acceptance and excess value. Triplet et al. have identified approachability as a major hurdle in the acceptance of a LIMS[9]. Modern web-technologies like Ajax allow for a more responsive and intuitive user interface, which in turn improves the user experience and reduces the learning period. Another crucial requirement for a successful adaptation of a LIMS is good documentation (R10). User acceptance can also be improved by offering an excess value over traditional spreadsheet tools, for instance by incorporating the use of barcodes (R11), label printing (R12), and mobile devices, such as smartphones (R13). A further advantage would be the incorporation of data analysis tools directly within the LIMS (R14).

Security. LIMS typically address security concerns by restricting access through secure user logins and different user roles. Security would also be enhanced by audit logging features (R15), where a version number is added to each database entry. Any change will then result in a copy of the entry with a new version number, so that accidentally overwritten entries can be restored.

OpenLabFramework. We present OpenLabFramework (OLF), a laboratory information management system (LIMS) primarily targeted at advanced sample and storage management in mid-sized laboratories with less than 50 users. It facilitates a seamless integration of virtual and real world storage handling by making use of mobile devices, which are carried by lab personal anyways, in combination with cheap and fully integrated barcode labeling

technology. In the following we shed a light on how OLF fulfills the LIMS requirements that we have identified before (R1–R15). A brief comparison with existing open-source LIMS is given in Table 2.

Modularity and extendibility. OLF is published as open-source (R1) and, due to its modular structure, it can be adapted to different types of laboratory data and sample types. New functionality can also be added and integrated (R2). Various features are covered by the following modules.

- GeneTracker:
  GeneTracker is intended to fulfill requirements specific to the hierarchical organization of genes, gene variants, vector constructs, and genetically engineered cell lines, thus helping to keep track of extensive sample libraries in the field of targeted genomics. The organization of these samples is further supported through OLF's built-in user and project management features.
- Sample Storage:
  The Storage module adds options for tracking and organizing samples in a customizable storage infrastructure (R3–4). This infrastructure is hierarchical, starting from buildings and rooms and ending in individual freezers and storage boxes. Interactive grids help the user to assess the content of a storage box at a glance. Together with GeneTracker, samples can be added or removed from storage in an intuitive manner, while providing an overview of remaining copies and related samples.
- File Uploads:
  The FileAttachments module allows users to up- and download arbitrary files, allowing for a better organization of their results and documents. Files are stored with a combination of timestamp and original file name to avoid conflicts arising from identical file names. Files are uploaded to a configurable folder on the server and not to the database itself. They can be linked to an arbitrary number of samples, so that other users can quickly obtain an overview of files relevant to a sample (R5).
- Barcode and Label Support:
  The functionality of the Storage module is complemented by the Barcode module, with which a user can create and print barcode labels (R11–12). These can later be used to locate a sample in OLF by scanning the barcode using a USB-connected scanner or a mobile device (R13). The Barcode module currently requires a connected DYMO® label printer but can be extended in the future to support other devices.

Reporting. Apache POI is utilized to export lists of samples to various file formats, including Excel (XLSX), Open Document Spreadsheets (ODS), PDF, and comma separated values (CSV). This feature is currently available for lists of genes, vector constructs, and cell lines. The storage hierarchy and individual boxes can also be exported to Excel spreadsheets (R6).

Flexibility. Grails applications are not bound to a specific database management system and will even work with non-SQL solutions, such as MongoDB (R7). OLF is compiled either as WAR file, which is suitable for deployment on a large number of Java-based web containers, or as locally executable JAR file, which comes packed with its own web container and file-based SQL solution (R8). It should be noted that OLF has only been tested thoroughly on Tomcat versions 6 and 7.

Cloud deployment. Grails also offers a plug-in for cloud deployment using the VMware Cloud-Foundry service (http://www.cloudfoundry.com/) (R9). Apart from CloudFoundry credentials and memory settings, no further configuration is needed. Upon deployment CloudFoundry automatically configures a suitable database to work with the application.

Mobile support. OLF utilizes the Spring Mobile Grails plug-in to distinguish mobile clients from desktop clients. If a mobile device is detected, a different view is shown that is tailored for the small-sized screen and touch-screen interaction (R13).

User approachability and excess value. OLF offers a modern web-interface that is clearly organized and intuitive (Figure 1), and allows for responsive user interaction. The Compass-powered search engine allows users to locate required information quickly and conveniently. Users can also develop effective laboratory work-flows using the sample tracking feature together with barcode labels and mobile devices (Figure 2). OLF validates all user entered data for validity and will, where applicable, provide a list of viable options in form of select boxes. In this way, OLF effectively avoids ambiguity and ensures consistency of sample data. Finally, OLF comes with online documentation that introduces the system to users, administrators, and software developers (R10).

Example for practical implementation and user acceptance - olf at the nanocan center. Within the past three years since the introduction of the first version of OLF in 2010 at the Lundbeckfonden Center of Excellence NanoCAN, around 780 genes, 1,200 vector constructs, and 300 cell lines have been added to the system, along with 1,500 associated sample locations. Data are stored on a Microsoft® SQL Server 2008 installation with a database size of approximately 7 MB. The more than 20 scientists engaged in functional genomics projects were introduced to the system through a one hour feature presentation, which enabled them to use OLF productively. The system was reported to be intuitive, albeit only one user had previous experience in using a LIMS system. Missing functionality considered useful for increased productivity and user convenience, such as handling of barcoded labels, were added to the system subsequently.

Recommended system configuration. OLF relies on a database backend for storing sample related data. However, only primitive data types, such as numbers or text fields, are persisted to the database itself, while all documents and files are stored in a folder and merely linked in the database. In this way, we expect a database size of less than 10–20 MB for most use cases. For smaller laboratories with less than 20 members, the file-based SQL database that is part of OLF's standalone version is appropriate. Since no significant processing of data is required, OLF is expected to be responsive even on systems with a single CPU core. For larger laboratories, however, we recommend using a system with multiple CPU cores and a dedicated database management system, such as MySQL for efficiently dealing with concurrent access in a responsive manner. Due to its dynamic nature, OLF has a large memory footprint and we strongly recommend providing a minimum of 768 MB of RAM on Linux and 1024 MB on Windows.

Discussion

Numerous commercial and open-source laboratory information management systems (LIMS) exist today. However, since commercial licenses are expensive and lack the possibility to be adapted to specific needs without additional costs, academic laboratories usually focus on finding an open-source solution to their sample management issues. Moreover, none of the existing solutions seems optimal for all given tasks (Table 2). Naturally, most LIMS are dedicated to a specific field of research and are thus not generally suited for other fields. Some solutions, on the other hand, focus on certain general aspects of laboratory work, such as sample tracking, protocols, or work-flows.

OpenLabFramework (OLF) was developed to address the need for an open-source LIMS solution for covering vector constructs and cell line library sample tracking. Acknowledging that many LIMS remain limited to their research domain, we created OLF in a strictly modular and extensible fashion, with dedicated modules for sample track-

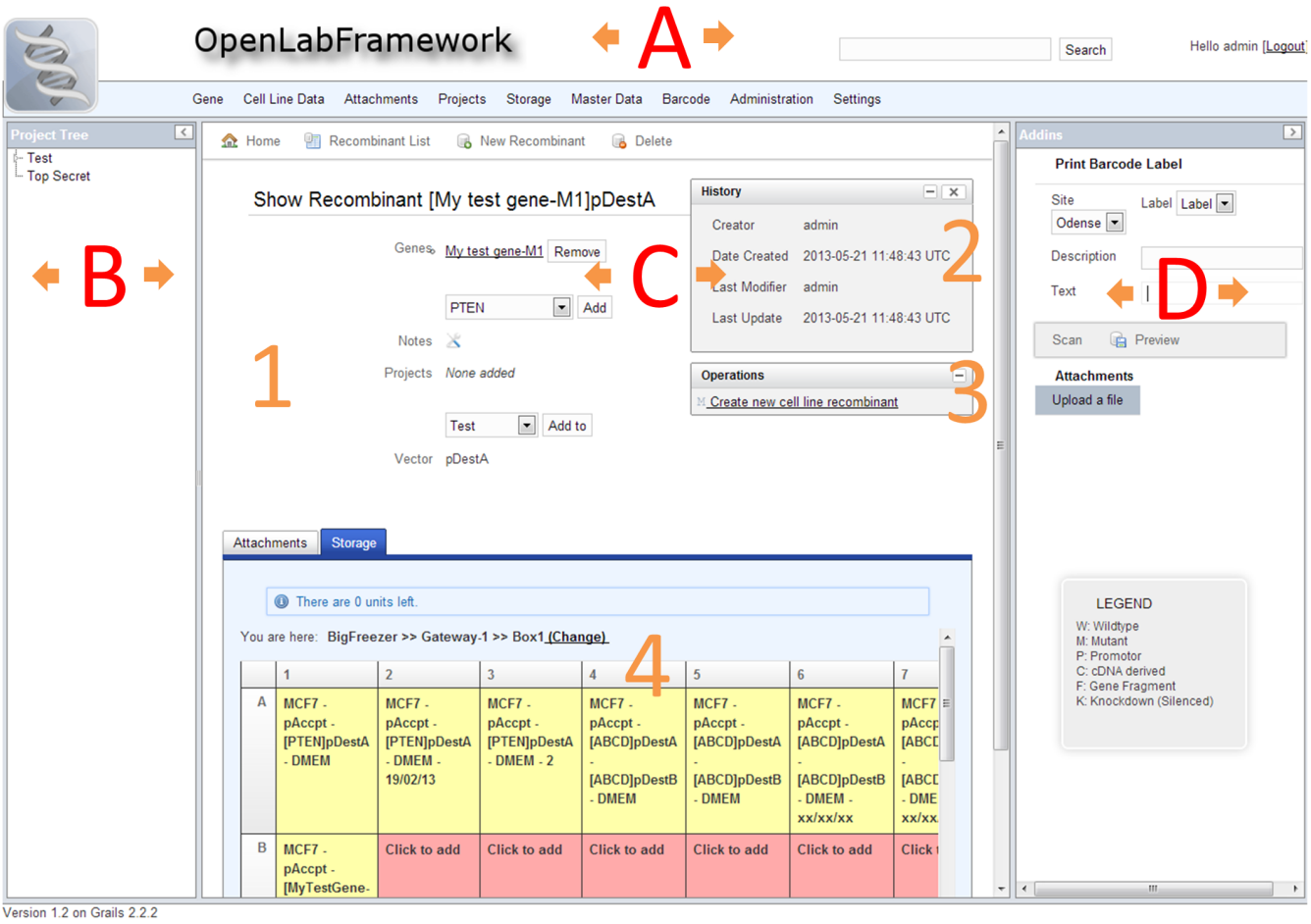

Figure 1 | The web-interface of OLF is divided into four main parts. (A) The header contains a menu and a search field for navigation. (B) A hierarchical project tree found in the left column can also be used for navigation. (C) The main panel is used to render the actual page. It can be further divided into a central panel (1), where properties can be altered, an object history box (2), an operations box with additional links (3), and at the bottom a set of tabs (4) where related information is available and can be interacted with. (D) Additional interaction possibilities are provided through add-ins that can be customized by each user in the right column. (Screenshot by ML. OLF logo by JT).

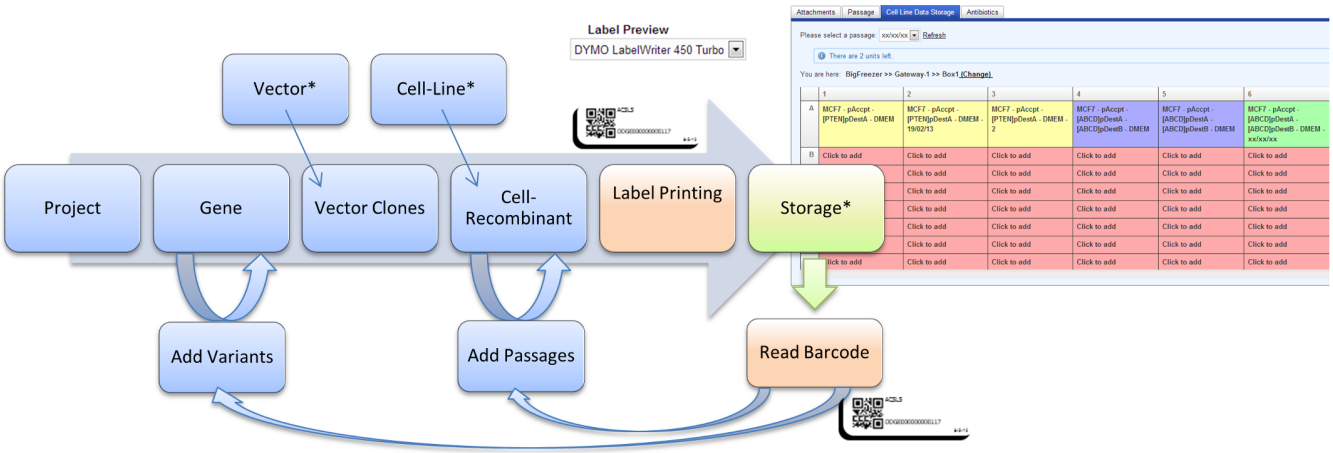

Figure 2 | Initially, the administrators set up master data, such as vectors, cell-lines, medium compositions, as well as the storage infrastructure (*). Users then create projects and link genes to them. Vector clones are created from the genes, which in turn can be used to create cell-line recombinants. Samples are labeled using a DYMO® label printer and added to physical, as well as virtual storage. At a later point, the barcode can be used for efficient retrieval and updating of sample information. Moreover, new gene variants and passages can be added with respective new labels and storage locations conveniently. Files and documents can be added to samples and genes, in order to make experimental results and additional information such as related publications, available to other users. (MT, TK, QT, JB and JM).

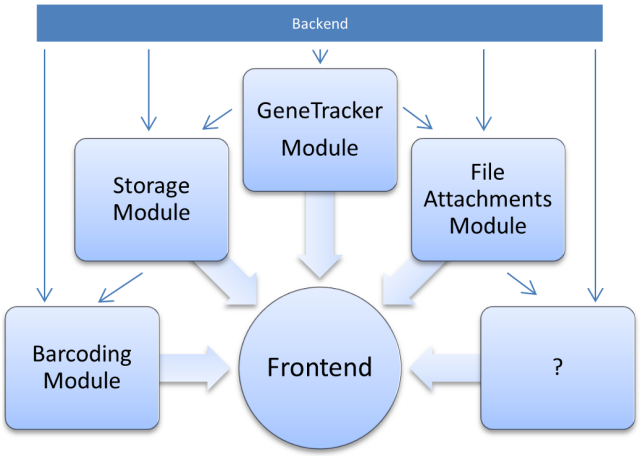

Figure 3 | OpenLabFramework is built in a strictly modular fashion. A back-end module provides the basic functionality, including project and user management, as well as base classes for other modules. Additional modules extend the base classes and integrate with existing ones. Finally, the front-end module creates views for all defined content and allows for interaction through a responsive web-interface. (MT, TK, QT, JB and JM).

ing, barcoded label printing, and file management. We expect that OLF can be adapted to other research fields and biomaterials with minimal developmental effort by implementing a content module similar to GeneTracker, which is then complemented by plugging in additional features as needed.

We built OLF using Grails and its extensive plug-in eco-system. This allows OLF to satisfy basic software development requirements, such as flexibility, reliability and simplicity. The use of a solid framework allows developers to focus on user-specific requirements, as for instance the support for mobile devices, and to keep the application up-to-date, since it will improve together with the underlying framework. The use of Grails, which can be considered the most dynamic and flexible web-application framework available in Java, together with its plug-ins, poses a significant simplification when adding new web-application features. This advantage separates OLF from comparable LIMS, which also embrace the concept of modularity or the use of web-application frameworks.

The introduction of OLF allows controlling sample logistics effectively, which is a particular challenge upon movement, turn-over of lab staff, and improper labeling of samples. OLF may further increase productivity by including modern technologies so far disregarded by most other open-source LIMS, such as printing and reading barcode labels. The high degree of automation and standardization that can be achieved by this may substantially reduce user-caused errors in sample assignment. A web-layer for mobile devices provides an additional advantage. In this way, samples can now be removed from physical and virtual storage at the same time, thus limiting the risk of forgetting this step after the work with the sample is completed. As illustrated in Figure 2, the implementation of OLF in a laboratory environment can lead to a significantly more productive work-flow.

Finally, unlike most LIMS, OLF is not bound to a specific database or web-container. OLF can be coupled with a large number of database management systems, including non-SQL solutions like MongoDB. If a suitable server is not available, OLF can be installed locally or even be deployed to the cloud with little effort, as demonstrated in our demo application. This flexibility will reduce technical hurdles in the introduction of OLF to a new laboratory.

OLF offers efficient and user-friendly management of sample information and location in the field of high-throughput biology and functional genomics. Being extensible, it can be adapted to satisfy additional requirements with little developmental effort. One

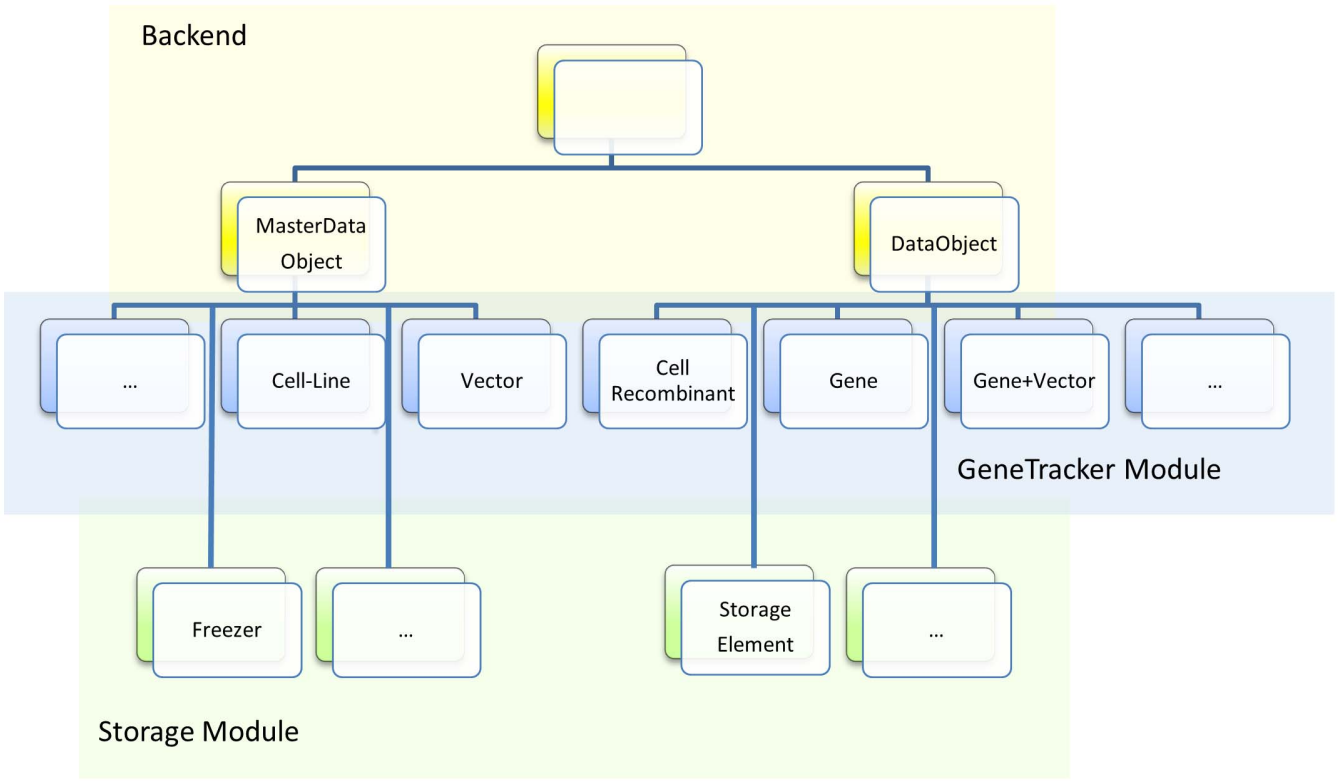

Figure 4 | The back-end provides two classes MasterDataObject (MDO) and DataObject (DO) that are extended by the different modules. MDOs can only be created by administrators, whereas DOs can be created by any user. Existing DOs and MDOs can be combined freely by module developers in order to build complex hierarchies. (MT, TK, QT, JB and JM).

requirement currently neglected is the extensive integration of additional web tools, which would make OLF more attractive to end-users. Although result data files can be uploaded and linked to samples, we envision that direct interfacing with laboratory equipment would make result data management significantly more convenient. The implementation of RESTful web services could expose OLF data sets and functionality to other tools and information systems. Along with this, the reporting capabilities could be improved, allowing for customized reports, where information from several instances is pooled. This would also help in establishing data analysis directly within OLF or through integration of additional tools (R15). Another important aspect worth considering is that OLF might in some instances require a more fine-grained access to the data. This functionality could be added through additional user roles or access control lists. Finally, the creation of a dedicated app for mobile platforms, such as Android or iOS, would improve the mobile user experience.

Conclusions

In order to retain an overview over large sample libraries typically found in nowadays laboratories, an efficient system for management and tracking of samples is required. OpenLabFramework (OLF) has been developed with focus on vector construct and cell line libraries. Thanks to its modularity, however, it can be adapted to new scenarios. OLF can be deployed using different database management systems either locally, to a server, or to the cloud. The incorporation of modern technologies, such as mobile devices and printing of barcode labels may increase productivity even further. These properties together may be considered characteristic for a next-generation LIMS and should provide the potential for widespread adaptation.

OLF embraces open-source in the hope of attracting not only laboratories in need of a LIMS, but also a community of software developers willing to adapt OLF to new scenarios. We intend to contribute in the future by developing further modules, e.g. for seamless evaluation of experimental data by integration of third party tools. Consequent utilization of community-proven open-source libraries make OLF's backend already highly reliable. With the support of its own community, OLF as a whole is expected to reach the same high quality standard in the future.

Methods

Grails web-application framework. We considered the Java™ based framework Grails to be the most promising candidate for a building a web-application. Grails is a VMWare®/SpringSource® product and builds on the company's experience in industry-standard frameworks such as Hibernate or Spring, which also form the core of Grails. In Grails, plug-ins deliver high-quality solutions for non-trivial web application tasks, e.g. database search (Compass and Apache Lucene), spreadsheet im- and export (Apache POI) and a user/security management (SpringSecurity). Grails embraces the paradigms convention over configuration and separation of concerns to keep the code concise and clean. Furthermore, Grails hides the complexity of data persistence with an object relational modeling technique, which encapsulates all database interactions and models them through Java domain classes. These characteristics allow faster development and integration of new features.

OpenLabFramework. As illustrated in Figure 3, OLF has a modular structure, in which a back-end plug-in provides the necessary base classes, as well as user, project and security management. Content plug-ins can then add arbitrary classes and view templates, and integrate with other plug-ins. All plug-ins are finally merged in the front-end application, which utilizes the scaffolding mechanism of Grails to dynamically create Ajax-driven views for user interaction.

The back-end of OLF introduces two base classes called MasterDataObject and DataObject (Figure 4). MasterDataObjects can be extended by classes representing master data that are maintained by system administrators, e.g. wildtype cell lines or vector systems in the GeneTracker module, or freezers and storage locations in the Storage module. More dynamic content is expressed through DataObject classes, which can be created and modified by regular users, e.g. genes or genetically engineered cell lines in the GeneTracker module, or StorageElements, which contain location data for other DataObject instances, in the StorageModule. This hierarchy makes OLF highly generic, but also allows fine-grained interactions between more specialized classes.

OLF strictly follows established patterns found in software architecture. Code is separated into different functional layers according to the model-view-controller pattern. In addition, business logic related code is bundled in service classes and tag libraries to avoid code duplication. Furthermore, OLF introduces the concept of content modules, which can be attached to existing pages. Content modules can either contribute new links to OLF's menu, render additional add-ins or tabs, or provide additional links called operations (Figure 1). By providing a clear structure and suitable interfaces, other developers can thus easily contribute to OLF.

1. Sanchez-Villeda, H. *et al.* Development of an integrated laboratory information management system for the maize mapping project. *Bioinformatics* **19**, 2022–2030 (2003).
2. Jayashree, B. *et al.* Laboratory information management software for genotyping workflows: applications in high throughput crop genotyping. *BMC Bioinformatics* **7**, 383 (2006).
3. Voegele, C. *et al.* A sample storage management system for biobanks. *Bioinformatics* **26**, 2798–800 (2010).
4. Morris, C. *et al.* The protein information management system (PiMS): a generic tool for any structural biology research laboratory. *Acta Cryst.* **D67**, 249–60 (2011).
5. Ponko, S. C. & Bienvenue, D. ProteinTracker: an application for managing protein production and purification. *BMC Res Notes* **5**, 224 (2012).
6. Droit, A. *et al.* PARPs database: a LIMS systems for protein-protein interaction data mining or laboratory information management system. *BMC Bioinformatics* **8**, 483 (2007).
7. Morisawa, H., Hirota, M. & Toda, T. Development of an open source laboratory information management system for 2-D gel electrophoresis-based proteomics workflow. *BMC Bioinformatics* **7**, 430 (2006).
8. Haebel, P. W., Arcus, V. L., Baker, E. N. & Metcalf, P. LISA: an intranet-based flexible database for protein crystallography project management. *Acta Cryst.* **D57**, 1341–1343 (2001).
9. Triplet, T. & Butler, G. The EnzymeTracker: an open-source laboratory information management system for sample tracking. *BMC Bioinformatics* **13**, 15 (2012).
10. Olhovsky, M. *et al.* OpenFreezer: a reagent information management software system. *Nat. Methods* **8**, 612–613 (2011).
11. Stocker, G. *et al.* iLAP: a workflow-driven software for experimental protocol development, data acquisition and analysis. *BMC Bioinformatics* **10**, 390 (2009).
12. Melo, A. *et al.* SIGLa: an adaptable LIMS for multiple laboratories. *BMC Genomics* **11**, 8 (2010).
13. Burgarella, S., Cattaneo, D., Pinciroli, F. & Masseroli, M. MicroGen: a MIAME compliant web system for microarray experiment information and workflow management. *BMC Bioinformatics* **6**, 6 (2005).
14. Krestyaninova, M. *et al.* A system for information management in biomedical studies–SIMBioMS. *Bioinformatics* **25**, 2768–2769 (2009).
15. Truong, C. V. C., Groeneveld, L. F., Morgenstern, B. & Groeneveld, E. MolabIS–an integrated information system for storing and managing molecular genetics data. *BMC Bioinformatics* **12**, 425 (2011).
16. Mollenhauer, J. *et al.* David versus Goliath. *Nanomedicine: NBM* **6**, 504–509 (2010).

Acknowledgments

This work was supported by the Lundbeckfonden grant for the NanoCAN Center of Excellence in Nanomedicine, the Region Syddanmarks ph.d.-pulje and Forskningspulje, the Fonden Til Lægevidenskabens Fremme, and co-financed by the INTERREG 4 A-program Syddanmark-Schleswig-K.E.R.N. with funds from The European Regional Development Fund.

Author contributions

M.L. and J.Th. developed the software requirement specification. S.S. and J.Tr. specified requirements specific to the biomedical domain and contributed extensively to bug-tracking and regular testing of OLF. ML implemented the application. M.T., T.K., Q.T., J.B. and J.M. jointly supervised the project, designed graphics and work-flow diagrams. All authors contributed equally to drafting the manuscript. All authors read and approved the final manuscript.

Additional information

Availability and Requirements: Project name: OpenLabFramework, URL: https://github.com/NanoCAN/OpenLabFramework, Wiki: https://github.com/NanoCAN/OpenLabFramework/wiki, Demo: http://www.nanocan.dk/openlabframework/demo, (user: admin, password: demo0815), Operating system(s): Platform independent, Programming language: Java, Groovy, Java-script, Other requirements: Java 1.6 or higher, Tomcat 6.0 or higher, License: GNU GPL v3.

Supplementary information accompanies this paper at http://www.nature.com/scientificreports

Competing financial interests: The authors declare no competing financial interests.

How to cite this article: List, M. *et al.* Efficient Sample Tracking With OpenLabFramework. *Sci. Rep.* **4**, 4278; DOI:10.1038/srep04278 (2014).