In the previous chapter, we discussed how design principles could be operationalised in practice. In this and the next chapter, we focus on the District Health Information Software (DHIS), as an example of how to inscribe these design principles in practice. While in this chapter, we discuss what DHIS is, in the next, we discuss how to go about setting it up in a new context be it a country, region, district or health facility. In this chapter, we describe and discuss three aspects of the DHIS as a project and an artifact that embodies principles of a data warehouse:

- We start by giving the background and trajectory of the process of developing the DHIS, from its start in South Africa in the 90s and to the present day.
- Then we will describe the DHIS application, and its overall structure.
- Finally, we describe its various components and the open source philosophy surrounding its development and evolution.

### 6.1 The Process of Developing the DHIS

#### 6.1.1 Context and Origin

The DHIS project is born out of the political processes of change in South Africa following the fall of apartheid, and as a synergetic collaboration between public health activists from the anti-apartheid struggle and information system developers from the Scandinavian tradition. DHIS emphasises the use of information for action and improved health services, user participation and ‘live’ (in real contexts), agile and rapid prototyping. The DHIS software development effort was organised within the HISP network, and has since its inception been embedded in a synergetic mixture of public health and participatory design perspectives.

The HISP approach to action research and information systems design was initially influenced by a number of union-based action research projects in Scandinavia in the 1970s and 1980s. The focus in the earlier participatory design projects was on empowering workers who were affected by or threatened by new technology, by exploring ways in which their influence over technological solutions could be ensured. Later projects shifted toward actively producing technological alternatives by involving workers in cooperative design at the workplace. The philosophical grounding of the design approaches followed by HISP may be seen as a continuation of these perspectives, with a goal to explore ways in which disadvantaged communities, regions and countries could appropriate ICTs for their own empowerment. User
participation and rapid prototyping in the context of use, combined with capacity building, all at multiple organisational levels of the health hierarchy, became the basis for the development approaches used to pursue this goal. The original key members of the HISP team had backgrounds as social/political activists in the anti-apartheid struggle and other social movements, and were thus explicitly political actors in a larger development process.

The context of South Africa and the process of rapid change following the fall of apartheid became decisive for the design and development of the DHIS, and its robustness through time. In South Africa at that time, the health system and data requirements were extremely fragmented and constantly changing. To give a picture of the fragmentation: During apartheid (1948–1993) and until May 1994, there were 14 departments of health at the central level: the ‘general’ Department of National Health and Population Development, 3 specific ‘white,’ ‘Asian’, and ‘coloured’ administrations, and 10 for ‘blacks’, ‘homelands’, and ‘self-governing states’. As one consequence of this fragmentation, there were numerous data collection tools, procedures and data definitions in use, and there were no national standards for data collection, and, as all the ‘homelands’ were included in the provinces; each province used different data standards.

As DHIS started out to support the decentralisation and integration of the health services, the design focus was on a flexible data structure; it had to be easy to add and change data elements, and to add and remove health facilities and districts, and to change organisational boundaries.

6.1.2 Development Process

The first phase of DHIS development can be characterised as an intensive three-year evolutionary process of participatory design, which took the system from a district pilot to a country-wide standard for HIS in South Africa. Between the years 1997–1998, HISP developed a free (open-source) database application based on Microsoft Office 97/2000, selected mainly because it was the existing defacto standard amongst potential users.

Development efforts were carried out, in line with participatory design practices, and a series of increasingly refined prototypes were tested in close collaboration with users, to enable information for local action. To some extent, the prevailing post-apartheid reform goals of decentralisation and local empowerment were consciously ‘inscribed’ into the software. Given the agenda of supporting political change in South Africa, the software design process started out with a set of objectives and scenarios the design team wanted to inscribe in the software; that is to try to enable certain ‘desired’ actions through particular functionalities and to make it less for the user to perform ‘undesired’ actions:

- Shift of control of data and information handling from central towards local levels, that is, toward more equal control between central and local levels.
- Local flexibility and user orientation – it should be easy to adapt the software to local conditions.
- Support for health sector reform towards decentralisation and the development of health districts; that is integrating the vertical flows of data from various health programs at district level.
Empowerment of local management, health workers, and communities – by providing access to their own data on their conditions.

Horizontal flow of information and knowledge, based on the principle of free access to all anonymous, aggregated health data/information.

These objectives were translated into concrete inscriptions through key principles laid down during the development of the first prototype 1997/1998:

- The application must support the hierarchy of essential datasets, that is, allowing users to add, modify, or delete local data elements, indicators, and so forth.
- The application should be designed in such a way as to support the drive toward decentralised capture, analysis, and use of data – in particular, support the push toward having the facility staff responsible for data collection also doing data capture, quality checking, initial processing, and output.
- The application should be easy to use for new areas (provinces, districts), and should allow users to tailor the geographic scope of their datasets to their needs. This resulted in the use of a front-/back-end solution in Access, where the back-end data files covered different geographical areas and the user could switch between them at will.
- The application should as much as possible rely on the flexible and powerful analytical and display tools already available within Office 97, such as Pivot Tables in Excel, even if this increased the learning curve.
- The application should be based on free (open-source) software – both gratis and with free distribution and redistribution of the source code.

6.1.3 Participatory Prototyping of the Early DHIS Application

The first DHIS application was developed in Visual Basic and Access by a team of two core developers and with a team of about 10 HISP members acting as mediators between users and the developers. The first DHIS prototype aimed at capturing and analysing routine monthly data (‘the MD module’), which was released for pilot testing in the HISP pilot districts in March 1998, and went through a series of very rapid prototype cycles during the next 4 to 6 months. New ‘builds’ were sometimes released on a weekly or even daily basis. The informal mechanisms for reporting bugs and requesting new functionality – all tightly integrated with user support – proved popular and encouraged users to provide feedback to the development team. This combined with the rapid deployment of new or corrected versions astounded many users, who previously had experienced many long drawn-out tender processes, fully pre-specified development projects that often ended in frustrating delays or fiascoes. Requests for new functionalities and/or new modules had to be filtered, prioritised and moderated by the HISP development team depending on the number of users making a request, its importance and team capacity, but all relevant requests were logged and prioritised even if they could not be implemented immediately.

The development process went through several phases, emphasising performance and progress and rapid response to user demands over any established prototyping model. Within the institutional framework in which HISP was operating, consisting of a variety of hierarchical levels and organisational and political structures, more formally organised user participation would have been impossible or inefficient. Formal user groups would easily become battlegrounds due to the ongoing large-scale
political transformations of South Africa’s administrative structures. The methodology for participation and development used was, thus more informal and to a significant degree based on improvisation, whereby any interested or innovative user, regardless of his or her place in the hierarchy, had full access to the development team – a meritocratic approach. This access was either direct or indirect via the other DHIS trainers/facilitators, and users were encouraged to use whatever channels they preferred. Access was not regulated, but the development team would normally have to guide users to a significant degree in understanding their own requests and how they could be implemented in practice. Such guided user participation was obviously time-consuming and only possible with a limited number of users.

After the first phase of very rapid prototyping, the user base increased and the software and user requests stabilised, and releases of versions became more controlled, and super users in advanced and early districts and provinces were used to test new versions before national releases. By 2001, the DHIS was implemented in all provinces and districts in South Africa.

6.1.4 Expanding the DHIS Project

While the rapid prototyping and iterative design process during the first phase of the DHIS project produced a close fit with the needs for reforming the health system in South Africa, the system accumulated both rigidities and a messy overall architecture. This proved problematic when it was subsequently introduced in countries such as Mozambique, India, Vietnam, and Cuba, after the turn of the millennium. To address this situation, the project embarked on a completely revised and internationalised version of the software in 2004, called DHIS version 1.4. This new development was started from a full remodelling of the database. The developer team was still confined to Cape Town, and employed the same technology (MS Access), but this time, as in contrast with the first phase of development, the users were primarily elsewhere. South Africa kept the existing installations (DHIS version 1.3), which were stable and well-adapted to the local situation. Through extensive travelling of project staff, supplemented by e-mail communication, the new internationalised DHIS (version 1.4) was developed as a participatory process between the developers in Cape Town and implementation teams in Botswana and Zanzibar.

As the new DHIS1.4 was introduced to new countries, the two-person development team became a bottleneck to supporting the expanding network of users with specific requirements. While the technology enabled rapid prototyping, the DHIS version 1 architecture was not suited for distributed development, and because of the small and co-located team, no source code sharing tools had been employed. Another shortcoming of the architecture used for version 1 was the dependence on the Microsoft platform, which meant that even though DHIS itself was open source, it required the full MS Windows and MS Office stack to run. These factors triggered yet another and simultaneous redevelopment of the software. Development of DHIS version 2 began in 2004 under the leadership of the University of Oslo, but aimed at distributing development activities to a number of the countries in the HISP network, in order to bring software development closer to the contexts of use.

A stack of ‘bleeding edge’ Java-based technologies was selected for the development of DHIS2, and in parallel a distributed development platform similar to those employed by many FOSS projects was set-up. However, re-implementing DHIS as a modular web
application proved quite difficult for various reasons. The radical break in technologies as well as too much reliance on the new online – electronic medium as in contrast to face-to-face communication presented a formidable obstacle to the involvement of existing technical staff in different countries and sites. The new flexible but complex architecture in fact hindered participatory design efforts in the field, as it took over a year and a half before DHIS2 could initially be deployed, first in a clinic in Kerala, India, and even then much important functionality was lacking. The system improved significantly through early use in India and Vietnam, and later also in Sierra Leone, as well as through the involvement of new software developers recruited locally in countries. While engaging with the global source code, their main task was to support local implementations, in the process trying to bridge the divide between users and developers.

The first real implementation of DHIS2 was in Kerala, India, January 2006, and from there, after a hectic period of development, it was implemented from 2008 onwards in more than 20 Indian states, as described elsewhere in the book. India is by far the largest DHIS2 open source implementation for the health sector in the world.

6.1.5 Open Source and Shared Development

Open Source Software development in the HISP context is intimately linked to the continuous development of the DHIS software for managing health data for decision-making in various countries in Africa and Asia over the last 15 years. Since 1997, over 15 years, a tremendous amount of man-hours have gone into the development of the DHIS application, its business logic and use cases, on-site in multiple countries in Africa and in Asia, since its inception in South Africa. For any developing country, to build their own such HIS data warehouse application from scratch, with equivalent levels of embedded knowledge and functionality of the DHIS would be a huge undertaking far beyond individual resources. This is one of the philosophical underpinnings of the HISP and DHIS project, namely that because the creation of software to support HIS in countries is so complex and so huge a task, while the requirements in many countries are quite similar, it makes a lot more sense to collaborate as a big virtual team, than to work in isolation and reinvent the wheel. Distributed development of open source software in collaborative networks of development, implementation and use, including regional and country health authorities, open source communities, and research institutions, make-up both the arena and methodology for the DHIS project.

This ideal aim of distributed and collaborative development of new functionalities and modules into a common shared toolbox, where innovations in individual countries are shared and spread in the network, is, however, a complicated project to achieve. Logically, it can be likened with the shared writing of a big document where a number of writers around the world, all with their own local experiences that they would like to capture and explore further in various subsections of the document. Despite all writers using the same draft_1 as their point of departure, bringing all contributions together in a shared draft_2, is a complicated process. It would require both appropriate tools for editing and a certain level of organisation and specification of responsibilities. While Google Doc would provide a tool for such writing, an editor combined with agreed rules for what to write and how update the document, would make-up the organisation. In open source and distributed software development, such as the DHIS project, similar challenges of coordination and management of the ‘master’ copy, is extremely critical. This is illustrated by the following example:
In 2003, the Ethiopian HISP team developed a module for International Classification of Diseases (ICDs) registration in the DHIS. In order to address the age and sex breakups of ICDs, they developed the option of multi-dimensional data elements, that is, the core data element is the singular ICD code and disease and age groups and male/female represent dimensions of it. This ICD module was elegantly implemented and well received. The problem, however, was that at that time the development of the DHIS was not organised as a distributed project, and the multi-dimensional functionality was not included in the master version. The Ethiopian DHIS version became a ‘fork’, not compatible with the ‘master’ source code. The drawback from this development was as a principle, twofold:

- The Ethiopian HISP team was stuck with the particular DHIS version in which they had hard coded the multi-dimensionality – the DHIS version 1.3.67, and they could not benefit from future improved versions of the DHIS.
- The innovation was not shared and distributed, it did not become part of the shared toolbox, and the global DHIS network was prevented from taking the Ethiopian innovation into use more broadly.

Forking is thus disadvantageous, both for the individual ‘forker’ and for the network as a whole, as sharing necessarily requires a common code base, which is not possible. It was this situation that triggered the DHIS2 development. The DHIS2 started out with a data model quite similar to the redeveloped DHIS1.4, which did not include multi-dimensionality. However, when a similar situation arose in the DHIS2 project; an Ethiopian developer implemented multi-dimensionality as a response to user requirements in trying to manage large and complex datasets in Tajikistan, considerable efforts were invested in making multi-dimensionality part of the DHIS2 meta data model. However, it takes more than a distributed development platform and an enabling architecture to solve the problems of shared and distributed development, as illustrated by the following example:

When DHIS2 was introduced in 2006, the HISP India team had already several years of experience with the DHIS version 1, and mastering the totally different and unknown technologies used in version 2 represented a huge challenge. As the DHIS2 at that time still did not have reporting and graphing tools, the India HISP team needed to develop such reports, but they had the problem of ‘finding’ their data. The database model in version 2 was quite similar to version 1, but the technology differed; version 2 included a database abstraction layer, inscribing the use of a Java Application Programming Interface (API) instead of directly accessing the database. As the Indian developers were unfamiliar with this technology, they bypassed the API and worked directly against the database in developing a range of ‘hard coded’ reports. An additional problem at that time was that DHIS2 had poor capacity to process the large amounts of data being managed in India, and the API was also here seen as a hindrance in providing yet another incentive for working directly on the database; why bother with a slow aggregation of a data mart for general use when you could make specific hard coded reports that were generated much quicker?

The dashboard module developed by the India team illustrates the problem facing the open source sharing philosophy. This module was developed in close collaboration with health managers and enabled ‘on-the-spot’ analysis and presentation of indicators, and became very popular among users. However, as it was a hard-coded workaround, it could not become part of the global repository and toolbox. To solve this problem,
the Norwegian and Indian developers spent a considerable amount of time together in reprogramming the module, but also in developing a shared understanding of the DHIS2 architecture and to develop concrete procedures for how to manage development and decision-making in the core cross-country development team. Today the country teams are working more closely together on releases and the merging of code bases. See Box 6.1 for more details on the organisation of DHIS2 development.

Box 6.1 Organisation of the DHIS2 development

The DHIS2 software development process is an open and transparent process similar to other typical open source software projects with source code, bug reports, blueprints, and discussion lists all accessible to anyone on the Internet.

**DHIS2 development** is a distributed effort with developers spread around the world. The coordinating office is the University of Oslo, where a team of 4–5 core developers, including the lead developer sits, with India and Vietnam being the two other major hubs of development. The developers in India are mostly developing local modules to support their large user base, and Vietnam hosts a small team of 3–4 core developers working under close supervision of the lead developer in Oslo, a team that has made considerable contributions to the global code base. These three teams of developers are all paid salaries and code more or less full time on DHIS2. Beyond these three teams, there are other local technical teams in Asia and Africa that mostly deal with customisations and some local module development. There have also been a considerable contribution by Masters students at the University of Oslo, and from 2005–2009, these were the main contributors to the source code. There is still a considerable student contribution, but now the main bulk of the work is done by full time programmers. The DHIS2 project is hosted on the Launchpad software collaboration platform, a website supporting development and maintenance of more than 22800 projects, and in 2011, DHIS2 became one of the 25 featured projects listed on the launchpad.net front page, due to its active use of Launchpad’s services.

The source code is maintained in so-called branches, and each release (version), as well as the current in-development code (called trunk) has its own branch under the DHIS2 project on Launchpad (www.launchpad.net/dhis2). The developers can download the source code from these branches through the use of a version control tool called Bazaar. Bazaar is installed locally on each developer’s computer and helps to synchronise local code against the central repository. Any anonymous user can download the DHIS2 source code from Launchpad, but only a team of registered and accepted (by DHIS coordinators) users (currently 42 people) can commit any changes back to the repository. A commit will automatically add a new revision to the central code and send out a message to all developers informing about the changes made to the central repository. All developers are then encouraged to update their local code against the latest revision of the code on Launchpad. Experimental work can be done in the developers’ private branches, which are also public and linked to the DHIS2 project on Launchpad. This code can also be viewed by the other developers, and allow for code revision and feedback from peers as well as the lead developer. Mature work from these experimental branches can at any point in time be merged into the main branch, and new developments on the main branch are similarly merged into the experimental branches to make sure they remain compatible with the main code.
All in all, the source code hosting arrangement at Launchpad provides mechanisms for distributing the source code and keeping everyone updated on its progress, and it supports distributed development on the same application, so that developers around the world can collaborate efficiently.

**Bug reporting** is done directly on the Launchpad website by filling out an online form with details on the problem encountered with the DHIS2 software. This is typically used by the developers themselves and the more advanced DHIS users and trainers. A bug report is automatically sent to all registered users on the mailing list, and further discussions and work on the bug is also distributed to the list. The lead developer usually assigns bugs to the different developers and the bug report list on Launchpad can show who is working on the problem and helps to stay updated on its status (for example, confirmed, assigned, fix committed).

**Blueprints** or new requirement specifications are also handled through online forms on Launchpad. Advanced users, coordinators, and developers can add requests and suggestions for new functionalities, which are often also discussed on the mailing list, and then finally assigned to a future release by the lead developer. A release of the DHIS2 software then consists of a list of bugs that have been fixed and a list of blueprints that have been addressed. Anyone can view the plan for future releases and see the planned release date as well as which bugs and blueprints will be targeted in the coming release. The release cycles in DHIS2 have in 2011 stabilised at around 2 months, which means that a new version or release is ready for download by the users every 2 months. Since the beginning in 2005 until May 2011, there have been 18 ‘full’ releases of the DHIS2, so the pace of releases has increased during 2010/2011.

Releases of DHIS2 are announced on the developer and user mailing lists and made available for download on the dhis2.org website. This dhis2.org website presents the key features of the software, contains contact information to the coordinators, provides downloads of the different releases and user manuals, contains information on how to join the mailing lists, and explains how to get started as a new developer.

The **modular architecture** of DHIS2 is designed to support different user needs across a range of contexts (more than 20 countries by January, 2012). Local modules or additions can be packaged together with the global modules (the core) and the end product appears as one coherent application to the users. DHIS2 has a portal structure which provides a common look and feel and a uniform and pluggable menu system so that new modules appear as additional menu items and blend into the application by following the same standards for the user interface programming. Which modules to be packaged together is decided during the build process, which takes place after downloading the source code and before starting up the application. By adding or removing a few lines of text in an xml file, a technical person with some training (no need to be a programmer) can select exactly which modules to include in the local application. A build tool called Maven builds the application based on the downloaded source code and the build specification file (the xml file) and produces a so-called .war file (Java web archive) which is ready to be deployed on a java web server. While inside the DHIS2 user interface (in the running web application), the local system administrator can specify further which modules that are visible to the different user roles (groups of users), roles that have been defined by the same administrator.
After the first implementation in Kerala in 2006, the DHIS2 was rolled out to many states in India, and from 2008, as it has matured, it has also been spread to many countries in Africa and Asia. An important point in this development was the decision by the HMN to use DHIS2 in their country pilot project to implement an integrated HIS in Sierra Leone. This also meant that it became part of a WHO recommended stack of FOSS tools for public health. The Sierra Leone project was a high profile HMN ‘first wave’ country pilot and was carried out during 2008–2009 as a very rapid prototyping project with constant demand and pressure from users at multiple levels; district, national and international (HMN/WHO) levels. Such international pressure and demand was important in order to both constantly improve the software and also to get exposure to the global community with regards to requirements and existing products.

**Summary of the DHIS Development and Current Trends**

Uncountable man/woman days have gone into the DHIS project since its inception in 1997, and as part of the early HISP even since 1994. Rapid prototyping and close user participation have been the dominant approach throughout. We use the term ‘prototyping’ here to denote a methodology where the software application is being continuously further developed based on feedback from users during testing and real use. A system as the DHIS is never ‘finished’, it will always be subject to large challenges of user’s requirements and new functionalities to be addressed. The design principles in this approach are characterised by learning through use and innovations combined with a development approach emphasising improvisations, experiments and rapid cycles of testing, learning and implementation. The design of the

---

*The local modules* typically make use of the core interfaces and underlying core code, but provide new functionality in the user interface, for example, reports that are tailor-made to local needs. The source code of the local modules are maintained by the local developer (in the country or region), but mostly hosted together with the global code on Launchpad. The central code repository has a folder for local modules and inside a subfolder for each country. All these local modules, currently 22 modules for the three countries India, Vietnam, and Bangladesh, are thereby also shared on the Internet to anyone interested, and can be downloaded and reused in other locations in need of similar functionality. Keeping all the code in one place also makes it easier for the lead developer to make sure that local modules are compatible with the latest revision of the global (core) code, and allows for code updates across all modules, for example, when updating the version of a third party tool or framework, or when changing the release version of DHIS2 in each module. This reduces the risk of local forking considerably. Forks are local modules that are no longer compatible with the global core and develop into complete local applications that are not able to benefit from the new developments on the global code. There have been examples of such forks in DHIS2 development in the past, but since 2010 all of the relevant local modules are hosted centrally and synched with global development. In the past, we have seen that quite a few local modules and functionality that have been recognised and used by many other locations (other than the origin) and gradually transitioned into global modules. This usually requires a collaborative process between a core (global) and local developer in making sure the standards for global modules are met and that it is fully compatible with the other global code. Some great innovations that today are core features of DHIS2 have been developed in this way, emerging from local use.
application is kept ‘open’ and decisions that will close and lock future opportunities are deliberately not taken. It might be a misnomer to call this a 25 years process for prototyping, but the point is that the application is under continuous and further development.

Prototyping in an ‘offline’ environment, that is, the dominant paradigm in the DHIS project until before 2010, is quite cumbersome in that new versions will have to be distributed to and implemented in all sites. The most ‘rapid’ part of prototyping has therefore, typically, been confined geographically to smaller groups of super users and only installed everywhere for ‘production’ when the new version was reasonably stable. Typically, therefore, the most active part of the user centered prototyping of the new DHIS1.4 was carried out in new countries like Zanzibar, where the overall country systems were being redesigned, and not in South Africa, where the stable version 1.3 was in countrywide ‘production’.

Online country implementations of DHIS2 have changed the conditions for prototyping; now, new functionality can be provided countrywide at a ‘key stroke.’ While online computing made it much easier to scale geographically, as discussed in Chapter 4, it also drastically changed the conditions for prototyping. Online state wide installations have been used in India since 2009, but for Africa this was still regarded as impossible, when in October 2010, we tested the Internet for DHIS2 browsing in a district in Kenya. We wanted to check how many installations in the national system that could run an online application. Everything worked fine, until a power-cut occurred, and despite the hospital’s generator quickly bringing the power back, the fixed Internet line went down. We were about to conclude that ‘Africa was not yet ready,’ when we started to test the modem to the mobile Internet, and found that it worked perfectly well. This sparked a 100% change in strategy; Internet over modem was quickly tested in all parts of the country and, as it worked well, the decision was taken to go for an online installation of the DHIS2. Following this, a period of very rapid prototyping followed. In fact, the whole DHIS2 lab was literally moved from Oslo to Kenya as the system was implemented in the first pilot province. A lot of development addressing ‘limited Internet’ and the new concept of ‘semi-online’ deployment followed: optimising data entry; automatic generation and download of data mart for offline generation of Excel pivot tables, customised reports, dashboard, and other required features. All these functionalities were implemented directly on the online server and made available for all users on the spot, which again helped to generate feedback, adjustments and requirements for new functionalities. Furthermore; through being involved with users in the field, and facing the problem of ‘how to quickly make users aware of new functionalities?’, the idea of implementing a ‘DHIS2 Facebook Light’ was born; messaging across DHIS users in Kenya, by district or province or nationwide; informing about new functionalities, feedback, and support.

From a digital gap and African development perspective, the new opportunities for Internet based online use and development, even given the limitations of the mobile network, are revolutionary in character; making it possible for the first time in Africa what has been taken for granted for quite some time in the North. Once these opportunities are explored through use and innovation, it might be that the vast user base and rapid expansion of the mobile network combined with creative use and untapped needs that will become the driving force for the development of new and innovative Internet based services. This mirrors the process when mobile telephone banking was innovated and first spread to a large use base in Kenya.
District Health Information Software – What it is

The DHIS development strategies based on agile approaches such as prototyping (which have been ongoing since 1997!) and on-site user participation, are becoming the dominant trends in current web development environments. The new opportunities for rapid cycles of development and use of software applications in the Internet and web era are of course general, across domains. In addition to the opportunities given by the web, the need for ‘rapid changes’ has also increased, as the web-based development environment is also in constant flux and changing continuously. These conditions have triggered the current trends in web development which emphasise rapid releases based on deadlines on ‘time’ rather than on planned functionality (for example, Google every 6 weeks), and thereby enabling innovations and pivoting changes when needed.

This paradigm shift towards rapid calendar based, as opposed to functionality based, releases is well illustrated by the changes in the Firefox’ release policy; they hung on to their version 3.6 for ages because they were not able to release version 4 with all the promised features. Then, in 2011, with their version with the long awaited release of version 4, and soon after version 5, they changed to a rapid 6 weeks release policy.

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Stage</th>
<th>Use and Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997–2000</td>
<td>From pilot to national system</td>
<td>DHIS v1 was developed in South Africa by one software team and a network of users. Essential health domain knowledge and practical applicability were translated to the DHIS metadata model and overall design, and the surrounding network of users and implementers.</td>
</tr>
<tr>
<td>2000–2005</td>
<td>Expansion and networking across countries</td>
<td>Multiple countries, but mostly pilot projects. HISP focus is on networking and research. DHIS ‘offline’ was not representing an attractive technology from countries, it was seen as something of the past.</td>
</tr>
<tr>
<td>2004–2010</td>
<td>Technological change: Transition to web-based</td>
<td>DHIS2 based on web technologies started and gained momentum. Technology was seen as part of the future and attractive. Full scale country projects initiated. DHIS v1 and v2; maintained two different teams.</td>
</tr>
<tr>
<td>2010-</td>
<td>Paradigm shift: Online cloud architecture</td>
<td>Cloud-based infrastructures and online implementations in India and Kenya demonstrates the potentials, also for developing countries. DHIS2 is riding on the ‘surf of development’ and gains further momentum. The two DHIS software teams are coming together.</td>
</tr>
</tbody>
</table>

6.2 More on DHIS2

In this section, we further elaborate on the question ‘what is DHIS2’ by:

- First, we continue where the last section ended; DHIS in the age of online computing; we describe DHIS2 in the wider perspectives of data warehousing and Business Intelligence.
- Second, describing and discussing DHIS2 as a platform, and comparing it with the perspective of an architecture.
Third, we discuss the alternatives for server hosting, in the ‘cloud’ or in the ‘basement’.

6.2.1 Data Warehousing and Business Intelligence

In the age of online computing and cloud based infrastructures, access to information is assumed to be web-based and available from everywhere through the Internet. Online access to information for decision-making, stored and managed in data warehouses, is becoming increasingly important, and applications providing such analysis and information are termed BI. DHIS2 is providing users online access to dashboards and maps presenting indicators targeting decision-makers, and is, therefore, also appropriately understood as a BI application. In this section, we describe and discuss DHIS2 as a combined data warehouse and BI application, or rather a platform, as it can be used to build modules and new applications and linkages to other systems. Also, as described earlier, while data warehousing is basically about integrating data flows and getting data ‘in’, BI is about analysing, presenting and disseminating the processed data, getting the data ‘out’.

A key point we bring forward is that BI and data warehousing in the context of the health sector in developing countries, are qualitatively different from how these application types are understood, designed and used in the commercial sector in, typically, industrialised countries. The reason for this is that data providers and data users are much closer and intimately connected in our health context, they may even be the same, than what is the case in data warehousing and BI as practiced in commercial sectors and as described in the literature. In Figure 6.1, we give a schematic illustration of the relationship between data warehouse as input and BI as output. We see that data sources are paper-based or in other ways close to the social context of data generation. In fact, all health facilities in the country may include direct data sources. While there may also be computer-based sources of data to the data warehouse in the context of health sector of developing countries, these are not dominant. They may, however, become increasingly computerised at the source, in the facilities, such as local medical records, but they will co-exist with paper-based systems for the foreseeable future.

A central term in modern data warehousing and in BI is ETL (Extract, Transform, Load); Extract data from (electronic) sources, Transform data to required format, and Load data to the destination data warehouse. This is the process that is catered for by SDMX-HD in the case of DHIS2. In reality, however, most data in DHIS2 is captured from paper-based sources or directly from the source, which necessitates a very different design of the DHIS2, than of the traditional Data warehouse and BI as described in contemporary literature.

The data flows from the source to the data warehouse, the ETL, the central part of the whole information management cycle, stretching from the data source to the data warehouse to the BI provision of processed data to the users. In our contexts, data providers and data users are intimately linked; data to manage health services, districts or hospitals are generated, collected, processed and used, all, primarily, within the same realm. Systems design, therefore, needs to include all these steps.

We see much more direct interaction between data sources, which are in fact human beings and users of the data themselves, for example, health workers carrying out
District Health Information Software – What it is

Implementing Good Design

an immunisation campaign, and the end-users of the information; various levels of management as well as the health workers providing the data themselves. The situation is thus very different from the traditional ‘Western’ data warehouse scenario, which operates in a fully computerised environment. Such applications are all secondary data processing units, in the sense that they get all their data loaded electronically from other computerised systems, which are ‘primary’ production or transaction systems (for example, banking or retail transactions, or hospital admission records) or other ‘secondary’ data processing systems, such as systems providing updated currency rates. Data interoperability is carried out through ETL: Extract data from a data source, Transform data to the required format, and Load it into the data warehouse.

There are two current trends in BI directly related to Internet and cloud based infrastructures; real-time BI and pervasive BI, and another more general trend linked to BI used for business performance management, which we translate into health services performance management. In the following, we describe DHIS2 in relation to these trends.

Real-time BI

To influence decisions while the data is still fresh is the reasoning behind real-time BI being made possible through online Internet-based infrastructure. In managing on-demand stock control, air traffic control and similar high intensity tasks in highly computerised environments, latencies in the range of seconds, minutes and hours, may be both possible and important. In the context of health services management, the required latency is typically given by the frequency of the routine reporting, whereas disease reporting may need to trigger immediate action. In DHIS2, due to large amounts of data, data processing, that is aggregating data, calculating indicators and loading it into the data marts, we carry that out at night. This gives one day latency; data is processed, analysed and available ‘countrywide’ the day after it is

Figure 6.1 DHIS2 as data warehousing and business intelligence application

![Diagram of DHIS2 as data warehousing and business intelligence application](image-url)
captured. In DHIS2, however, the wider information management cycle, including data collection and capture, is as important in terms of latency. We may distinguish the following types of latencies:

- **Data registration and collection latency**: This is the time it takes to collect the data. Whether data registration is paper-based or computerised, and whether data is captured directly at the source of data collection, is important in eliminating reporting latency.

- **Data reporting and capturing latency**: In the traditional paper-based system, the sending of paper reports from the source at the health facility to the block or district office where it is captured in DHIS2, is the critical process. Then there may be further delays until data is captured in the DHIS2. The strategy to reduce this latency is to move data capture to the facilities, at, or closer to the source of the data, using mobile telephones or laptops with the Internet modem. Problems related to available manpower to actually capture data is also a widespread problem, but not of a technical character.

- **Data transmission latency**: In the offline DHIS2 sites, data needs to be transferred to the national data warehouse by way of e-mail attachment or memory stick. This latency is eliminated with online installation of DHIS2.

- **Data processing and analysis latency**: Data is processed and data marts and indicators updated once data is in the data warehouse, that is, every night, meaning one day latency in nationally online central server installation. In an offline installation one may, of course, process data when appropriate.

- **Data feedback and dissemination latency**: Providing feedback to providers and users of information is traditionally a very delayed process. In Botswana, it took 2–3 years before the annual statistics were disseminated, and there were not much other feedback. In DHIS2, the feedback is available in dashboards, reports, GIS once it is processed. Immediate access to information available through online DHIS2 installations, but of course there must be institutional practices in place that encourage feedback.

At the technical level, for real-time DHIS2, latency is thus primarily related to whether online or offline installations are used, and to what extent data capture is carried out in facilities or close to the source, using mobile telephones, PDAs, laptops or desktops. At the organisational level, however, latency depends on how the wider information management cycle is managed (or not).

**Health Services Performance BI**

The use of indicators to monitor achievements of health programmes and services provision according to targets is the signature element of the HISP public health approach and key part of the DHIS design. In the BI world, the use of dashboards and other means to present continuously updated ‘real-time’ indicators on business performance graphically are key to its management. DHIS2 enables the presentation of dynamically updated information, using dashboards, GIS and various graphical reporting tools.

**Pervasive BI**

The spread of the Internet and cloud-based infrastructures have given way to the term “pervasive BI”, meaning that since web-based systems are available from everywhere
with the Internet, the user base is drastically increased and all potential users in an organisation or sector may be reached. Following the online deployment of DHIS2 in Kenya, for example, we have seen a drastic increase in availability of online and updated information and its access to online users.

6.2.2 DHIS: As a Platform

Whether one regards DHIS as an application, a platform, or even an architecture depends on the context and perspective. When the AWARE project; ‘Action for West Africa Region – Reproductive Health’, addressing limited access to health care and low quality of services in countries in West Africa, uses DHIS for their internal reporting across countries; or when WHO Afro is setting up DHIS to monitor malaria vector control in African countries, or when in Tanzania DHIS is set-up to monitor the Payment for Performance, ‘P4P’, programme, in each place or instance, we may talk about DHIS as an application designed to address particular user needs. When in Kenya, the DHIS is designed and set-up to cover most of the data flows in the health sector and support the range of user needs expressed by a multitude of stakeholder and users, DHIS may also be regarded as an application, although a very wide and diversified one with a heterogeneous user-base, and very different from the former three examples. While in each instance, DHIS may be regarded as a unique application, multitude of applications serving very different user needs, there is also the generic ‘tool’ from which all these different applications are generated, which we may label ‘platform’.

Platform as a term is maybe most often used for generic software applications and frameworks, such as Firefox and Windows, which are context free also in the sense of not even when designed in particular ways are targeting particular organisations or end-users. The defining elements that make them differ from applications are their support to a heterogeneous and growing user-base, constant generification, and that they form a framework for a suite of IT capabilities.

Box 6.2 gives working definitions for different terms related to ‘platform’.

<table>
<thead>
<tr>
<th>Box 6.2</th>
<th>Working definition of terms related to ‘platform’.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiwana et al 2010, provide a coherent set of concepts, spanning the range from Platform to Architecture;</td>
<td></td>
</tr>
<tr>
<td>• <strong>Platform</strong>: The extensible codebase of a software system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Module</strong>: An add-on software subsystem that connects to the platform to add functionality to it.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Ecosystem</strong>: The collection of the platform and the modules specific to that platform.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Interfaces</strong>: Specifications and design rules that describe how the platform and modules interact and exchange information.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Architecture</strong>: A conceptual blueprint that describes how the ecosystem is partitioned into a relatively stable platform and a complementary set of modules that are encouraged to vary, and the design rules binding on both.</td>
<td></td>
</tr>
</tbody>
</table>
In concrete terms, DHIS can be perceived as a platform on several levels. First, the application database is designed ground-up with flexibility in mind. Data structures such as data elements, organisation units, forms and user roles can be defined completely freely through the application user interface. This makes it possible for the system to be adapted to a multitude of locale contexts and use-cases. We have seen that DHIS supports most major requirements for routine data capture and analysis emerging in country implementations. It also makes it possible for DHIS to serve as management system for domains such as logistics, labs and finance.

Second, due to the modular design of DHIS it can be extended with additional software modules. These software modules can live side by side with the core modules of DHIS and can be integrated into the DHIS portal and menu system. This is a powerful feature as it makes it possible to extend the system with extra functionality when needed, typically for country specific requirements as earlier pointed out.

The downside of the software module extensibility is that it puts several constraints on the development process. The developers creating the extra functionality are limited to the DHIS technology in terms of programming language and software frameworks, in addition to the constraints put on the design of modules by the DHIS portal solution. Also, these modules must be included in the DHIS software when it is built and deployed on the web server, not dynamically during run-time.

In order to overcome these limitations and achieve a looser coupling between the DHIS service layer and additional software artifacts, the DHIS development team decided to create a Web API. This Web API complies with the rules of the REST architectural style. This implies that:

- The Web API provides a navigable and machine-readable interface to the complete DHIS data model. For instance, one can access the full list of data elements, then navigate using the provided hyperlink to a particular data element of interest, then navigate using the provided hyperlink to the list of forms which this data element is part of.

- Data is accessed through a uniform interface (URLs) using a well-known protocol. There are no fancy transport formats or protocols involved - just the well-tested, well-understood HTTP protocol which is the main building block of the Web today. This implies that third-party developers can develop software using the DHIS data model and data without knowing the DHIS specific technology or complying with the DHIS design constraints.

- All data including meta-data, reports, maps and charts, known as resources in REST terminology, can be retrieved in most of the popular representation formats of the Web of today, such as HTML, XML, JSON, PDF and PNG. These formats are widely supported in applications and programming languages and gives third-party developers a wide range of implementation options.

There are several scenarios where additional software artefacts may connect to the DHIS Web API.

First, Web portals may be built on top of the Web API. A Web portal in this regard is a website which functions as a point of access to information from a potential large number of data sources which typically share a common theme. The role of the Web portal is to make such data sources easily accessible in a structured fashion under a common look-and-feel and provide a comprehensive data view for end users.
Aggregate data repository: A Web portal targeted at the health domain may use the DHIS as the main source for aggregate data. The portal can connect to the Web API and communicate with relevant resources such as maps, charts, reports, tables and static documents. These data views can dynamically visualise aggregate data based on queries on the organisation unit, indicator or period dimension. The portal can add value to the information accessibility in several ways. It can be structured in a user-friendly way and make data accessible to inexperienced users. It can provide various approaches to the data, including:

- **Thematic**: Grouping indicators by topic, such as immunization, reproductive health, HIV / AIDS, notifiable diseases and environmental health.
- **Timelines**: Trends over time.
- **Geographical**: Grouping data by states / provinces / regions and districts. This will enable easy comparison of performance and workload.
- **Presentation**: Using maps, charts, tables, etc.

Relating to the discussion on BI in the previous section; the portal is providing end-users access to the data in the DHIS as a BI application.

Mash-ups: The Web portal is not limited to consuming data from a single Web API - it can be connected to any number of APIs and be used to “mash up” data from auxiliary systems within the health domain. If available, the portal might pull in specialized data from logistics systems for tracking and managing ARV medicines, from finance systems for managing payments to health facilities and from lab systems for tracking lab tests.
for communicable diseases. Data from all of these sources might be presented in a coherent and meaningful way to provide better insight in the situation of the health domain.

**Document repository:** The Web portal can act as a document repository in itself (also referred to as content management system). Relevant documents such as published reports, survey data, annual operational plans and FAQs might be uploaded and managed in terms of ownership, version control and classification. This makes the portal a central point for document sharing and collaboration. The emergence of high-quality, open source repository/CMS solutions such as Alfresco and Drupal makes this approach more feasible and compelling.

**Knowledge management:** KM refers to practices for identifying, materializing and distributing insights and experience. In our context it relates to all aspects of information system implementation and use, such as:

- Database design – data warehousing and Business Intelligence.
- Information system usage and how-to.
- End-user training guidelines.
- Document and documentation management – content management systems.
- Data use, analysis and interpretation.

Knowledge and learning within these areas can be materialised in the form of manuals, papers, books, slide sets, videos, system embedded help text, online learning sites, forums, FAQs and more. All of these artefacts might be published and made accessible from the Web portal. In this way, the portal may be designed as a “Knowledge management system.”

**Forum:** The portal can provide a forum for hosting discussions between professional users. The subject can range from help for performing basic operations in the HIS to discussions over data analysis and interpretation topics. Such a forum can act as interactive source for information and evolve naturally into a valuable archive.

Second, third-party software clients running on devices such as mobile phones, smart phones and tablets may connect to the DHIS Web API and read and write to relevant resources. For instance, third-party developers may create a client running on the Android operating system on mobile devices targeted at community health workers who needs to keep track of the people to visit, register vital data for each encounter and receive reminders of due dates for patient care while travelling freely in the community. Such a client application might interact with the patient and activity plan resources exposed by the DHIS Web API. The developer will not be dependent on deep insight in the DHIS internal implementation, rather just basic skills within HTTP/Web programming and a bit of knowledge of the DHIS data model. Understanding the DHIS data model is made easier by the navigable nature of the Web API.

Third, information systems developers aiming at creating new ways of visualising and presenting aggregate data can utilise the DHIS Web API as the service layer of their system. The effort needed for developing new information systems and maintaining them over time is often largely under-estimated. Instead of starting from scratch, a new application can be built on top of the Web API. Developer attention can then be directed towards making new, innovative and creative data representations and visualisations, in the form of e.g., dashboards, GIS and charting.
DHIS, architecture and portal design

The architecture perspective we have discussed so far has been illustrated by interoperability between the data warehouse for aggregate data and transaction systems such as Human Resource and Medical Records systems. To this perspective, the Web API and portal approach add at least two additional dimensions: first, it provides user interface and access point to a range of end-users and the general public, and second, it provides an approach to group together components that are naturally belonging together in an enterprise design perspective, but which are "technically" incompatible; such as e.g., data processing and document management. Together these two dimensions of the portal approach make up a powerful part of the architecture; while the easy end-user access contributes to the aim of increased dissemination and use of information, the new design possibilities help provide richer and more targeted information.

6.2.3 Servers: In the Basement or in the Cloud

Using ‘cloud’ as a metaphor for the Internet, ‘cloud computing’ is a loose term used for everything from the more narrow meaning of virtual servers available over the Internet, to software based services more generally and even to denote ‘everything’ outside the organisation’s ‘firewall’. In our context, ‘cloud computing’ is illustrating the fact that the server may be anywhere as long as you have Internet connection to it. When the user in a remote rural health center in Kenya is accessing DHIS through the wireless Internet over the mobile network, cloud computing is indeed an apt metaphor. Furthermore, an important reason to the successful implementation of DHIS in Kenya was that the focus could be on the users’ access to the Internet and not on the proper functionality of the server and its connectivity. In the case of the Kenya implementation, the server was in the ‘cloud’, located in London. When implementing online web-based systems, the server is extremely critical as it needs to be optimally accessible next to 100% of the time. The new paradigm of online web-based systems is breaking with the tradition of each organisation and Ministry having their own computer centre and servers. Implementing DHIS ‘on the web’, means that it does not rely on any particular physical location. Generally, therefore, we would recommend modern cloud-based infrastructure, where all the technical parts of managing and maintaining the servers are done externally, in the ‘cloud’.

It maybe argued that modern cloud-based computing is a new infrastructure, which is actually providing countries with poor infrastructure the same opportunities as the rich countries. That is, of course, only the case if the country has sufficient Internet, good connectivity to the international grid. We have seen that in Kenya, as an example, the Internet is good. The advantage of renting external server capacity, in the cloud infrastructure, is that it makes everything such as setting up and maintaining the system done easier and faster, while at the same time the general services such as security, availability and scalability are guaranteed. The cloud infrastructure is providing a unique elasticity, while there are virtually no need for any up-front commitments on payment or rentals, there are ‘infinite’ resources on demand, for example, if the trial is successful.

But can one trust the ‘cloud’, will the data be safe, also in the future? These are of course important political questions. The fact that the US Government has decided to move significant parts of their data processing to the cloud has been used as an argument for other government also to move towards cloud hosting. However, the
“cloud” used by the US Government is firmly located in a high security “basement” on their own territory, which, again, is emphasising the differences in infrastructure between countries. Therefore, the use of the “cloud” by rich countries is underlining the fact that the cloud is only a metaphor and that in reality the clouds will always be physically located somewhere. In Kenya, the DHIS2 was initially implemented on a rented, external server located in London. This was necessary because the server and network in the Ministry of Health were not working. There is an ongoing project to refurbish the computer room and install a donated server. The plan is, therefore, to move the DHIS2 to the local server once it is running properly. The start of a project to implement the DHIS2, or any such online web-based system, is very critical. It is, therefore, important that the servers are working properly. The advantage of using external servers in the startup of a project is that it only takes a few hours to set up the system and it is easy to manage. This is very important in the first cycles of prototyping a new system, as we do not want to be distracted by the technicalities of the server, when the software and capacity building is the focus.

Obviously, there are political arguments for countries to remain in physical control of the data and the servers. Rich countries in Europe and the US are all making sure their data are confined within their own territory and under their ‘physical’ and maybe also legal control. All countries are concerned with security, but it should generally not be seen as the Ministry of Health’s primary task to manage servers. If the government has hosting facilities serving the public sector, or if there are local companies with hosting services, that is fine.

6.3 The DHIS Metadata Model

The first version of the DHIS was designed and developed in the HISP pilot districts in Cape Town aiming at supporting a public health approach to build the new health services in South Africa. The key questions to address through the DHIS were related to health status of the population and achievements of the health services. The focus in the public health approaches at that time was on epidemiology and health management. The drawing in Figure 6.2 was used in training sessions in pilot districts and at the Universities in Cape Town, the base of the HISP at the time, and asked two questions:


These questions are represented in the picture:

Answering these questions went into the design of DHIS. The ‘why?’ would rather be linked to analysis, but data on the more exact questions of what? where? when? where captured in the DHIS data model, as illustrated in Figure 6.3. The question about ‘where’ is linked to the health facility, where the data is registered. The health facility is located both geographically, with address and x,y coordinates, and administratively, in the hierarchy of the health sector, called org.unit hierarchy.

Figure 6.4 illustrates how the ‘reality’ of Figure 6.3 is represented, or mapped, into a data structure. If we take the information management cycle one step further, we see
the same data representation analysed and presented graphically in Figure 6.5, where the ‘what’ is malaria incidence, the ‘where’ is Kenya, and the ‘when’ is by each month of the last quarter 2010 and the first quarter 2011.

Data elements (or data variables, or counts) in the DHIS are handled as singular atomic units, which can be broken up in dimensions, such as age groups and sex. The handling of data elements as atomic units, and not as columns in a table, is what enables the flexibility in adding and removing data elements, characterised in the DHIS. Keeping data elements as atomic units is also important in making the metadata structure transparent and easy to understand and update by information managers;
Datasets, indicators and evaluation rules are easy to define and add. The different foundation concepts of the DHIS data model are described in Box 6.3.

**Box 6.3 Components of the DHIS**

*DHIS Components*

*Data elements, the What dimension in DHIS:* The data element is an atomic unit of data and the key building block in DHIS. It is the raw data in a statistical system, called ‘counts’ in some contexts, being collected through tally sheets or other tools or generated through ‘transactions’ in computerised systems, e.g. patient admissions in a hospital. ‘Confirmed malaria cases’ and ‘Measles (antigens) doses given’ are typical data elements in DHIS that may be collected using paper forms, or in fact generated from paper register books or computerised patient registries. We may distinguish between frequent routine data from health programmes and services, e.g. monthly, and less frequent, semi-permanent data, such as population (census) data, data on staff, and equipment.

Data elements may be disaggregated in dimensions called categories, such as age and sex, which are individual atomic components of the data element. The data element is thus not an ‘absolute’ atomic unit, but for management, analysis and aggregation of data, it is best understood as an atomic unit.

Datasets are used to bundle and organise data elements for the purpose of data entry and for export and import between instances of DHIS. Datasets are linked to services that are provided by a health facility. Health facilities, or
organisational units, are assigned the particular datasets they use to collect data, that is, the services provided. Datasets are not linked directly to the data values, only through their data elements and frequencies, and as such a dataset can be modified, deleted or added at any point in time without affecting the raw data already captured in the system, but such changes will of course affect how new data will be collected.

Datasets are used for the construction of Data entry forms in DHIS2. Once assigned to an organisational unit, the dataset is made available for data entry for that unit. There are three options of increasing flexibility for visual design of the data entry screen and correspondingly more work to set it up:

The default data entry form is quite simply a list of the data elements in the dataset and a column for inputting data. If data dimensions (categories) are defined, these will be automatically generated. Section forms allow for more flexibility in setting multiple tables and subheadings. Custom forms allow you to construct the data entry forms in a HTML editor.

The organisational hierarchy and units, the where dimension in DHIS: The organisational hierarchy is defining the health facilities, geographical areas and administrative units being addressed by the particular DHIS instance and their relationships in the health system through a system of child-parent relationships. The hierarchy is defined with one root unit (for example, Ministry of Health) and any number of levels and nodes below. The nodes in this tree-structure are called organisational units, and they may be health facilities, that are providing services, or administrative units. The design of the hierarchy will determine the geographical units of analysis available to the users as data is collected and aggregated in this structure.

The organisational units, orgunits in DHIS language, are categorised and defined using group sets, for example, ‘Type’ and ‘Ownership’ (default group sets), and groups for each group set, such as ‘Dispensary’, Health Centre,’District hospital’, for type, e.g. ‘Central government’, ‘Local Government’, ‘Faith-based organisation’ for ownership.

Indicators represent the most powerful data analysis feature of the DHIS2. While data elements represent the raw data (counts), the indicators represent formulas providing coverage rates, incidence rates, ratios and other formula-based units of analysis. An indicator is made up of a factor (for example, 1, 100, 100, 100000), a numerator and a denominator, the two latter are both expressions based on one or more data elements. For example, the indicator ‘Measles coverage <1 year’ is defined by a formula with a factor 100, a numerator (‘Measles doses given to children under 1 year’) and a denominator (‘Target population under 1 year’). The indicator ‘DPT1 to DPT3 drop-out rate’ is a formula of 100% x (‘DPT1 doses given’- ‘DPT3 doses given’) / (‘DPT1 doses given’).

Due to the management of data in atomic units, which is also used for denominator data such as population data, makes it easy for users to define indicators by selecting data elements and formulas for numerators and denominators. Indicators can be added, modified and deleted at any point in time without interfering with the data values in the database.
Validation rules are data quality checks used to improve the quality of the data being collected. Rules are composed by left and right side expressions built by data elements and mathematical operators and numbers, with a logical operator, such as ‘less than’, ‘greater than’, ‘equal’, ‘not equal’, between the two sides. Validation rules may be absolute, meaning cannot be true, like ‘number of live births’ + ‘number of still births’ cannot be less than ‘number of deliveries’. Or they may be logical, or expert rules, such as ‘number of live births’ should not be more than ‘number of deliveries’ + 20%. Such rules need to be used with caution, and typically not made mandatory for ‘all users’.

The rules can be run in data entry, after filling each form, or as a batch process on multiple forms at the same time, for example, for all facilities for the previous reporting month. The results of the tests will list all violations and the detailed values for each side of the expression, where the violation occurred, thus making it easy to correct the values using the data entry.

Information dissemination; reports, tables, charts, pivot-tables, dashboard, GIS: DHIS2 provides the range of reporting facilities used by Business Intelligence applications. Standard reports in DHIS2 provide aggregated data by organisational units or levels, of data elements and indicators, and over time. External report designers such as Business Intelligence and Reporting Tools (BIRT) and iReport may be used for customised reports. Pivot tables are dynamic and online, or datamart may be downloaded to generate Excel pivot tables. A chart dialogue guides the user through the creation of various types of charts with data on indicators, organisational units and periods. These charts can easily be added to one of the four chart sections on the dashboard and be made available right after log in (Figure 6.6).

Figure 6.6 DHIS2 Dashboard. Users add their own charts, which are dynamically updated to the chart sections on DHIS2 main page.
6.4 DHIS2: Modules and Components

In this section, we describe three important modules and fourthly the components used in the DHIS2. These include:

- The patient, or tracking module.
- The GIS module, which is an important part of the DHIS2 analysis and presentation layer.
- The module for mobile reporting.
- Finally we will discuss components, by providing a description of the Open Source Frameworks and tools using which DHIS2 has been built.

6.4.1 From Aggregate to Discrete Data: The Generic Case, Event, Patient, Tracking Module in DHIS

DHIS2 is a tool for collection, validation, analysis, and presentation of aggregate statistical data. As the use and technology are both maturing, however, and as discussed in Chapter 4, vertical scaling in terms of requests for more comprehensive solutions and more granular data are gaining momentum. The DHIS2 case based 'discrete' data module is developed as a response to demands for in particular cases being able to handle the roll up and roll down of the data captured, from singular discrete cases to statistics, and vice versa, to be able to move quickly from singular cases, or events, to statistical analysis. The registration of vital events, such as registration of deaths according to the ICD10, is a typical use case. In this case, the registration is one-off, a way to organise data as both singular cases and statistics, and to enable aggregation and disaggregation one step further within the same framework. Maternal death is another typical use-case; by registering each case of maternal death much richer data can be registered, and following the ease of aggregation, much richer data analysis may be carried out.

Line-listing is another use case; each case or patient encounter is typically registered on a line in a registry. In the traditional paper-based system, data are then aggregated by month from these register books and compiled into monthly reports, typically then entered into the DHIS2. In many countries, for example, in India and Ghana, for particular diseases, or deaths by diseases from hospitals, paper-based line-listing are being reported. In such cases, names are not the issue, as each line is to be regarded as a 'discrete event' for statistical purposes.

One step up in terms of increased complexity from the singular case registration is the tracking of cases, for example, infants for immunisation, pregnant women for Antenatal Care visits and delivery, and ultimately, the tracking of the pregnant woman through lifecycle of the different ANC visits to delivery and then to postnatal visits, and the immunisation scheme of the baby. Other important use cases are linked to tracking of patients with chronic diseases necessitating continuous follow-up such as diabetes or tuberculosis. A key principle in the design of the DHIS case based discrete entity tracking module is that the singular cases, or events, and the aggregates are using the same data elements. This allows for the registering of each case, event, count or occurrence, or the individual patient or client that are then aggregated to become the data value stored in the DHIS2 ‘aggregate’ module. This is illustrated through the Figure 6.7.
6.4.2 DHIS GIS – GIS Module in DHIS2

Background
Spatial representations of health data, enabled through GIS, represents a powerful way to present data and indicators, including for identifying spatial trends of diseases, identifying access patterns to health facilities, understanding determinant of diseases, and various others. HISP was early to recognise the importance of spatial visualisations and GIS, and the importance of linking spatial and non-spatial data in one database. To arrive at the current powerful Open Source GIS module in DHIS2 has taken time involving significant amounts of research and development. During 2009–2011, the focus has been on developing the WHO OpenHealthMapper, the web version of their earlier desktop based Health Mapper. This has been bundled as a module inside the DHIS2 – which we call DHIS GIS. The development was enabled through an agreement between the WHO and University of Oslo.

The Technical DHIS GIS Solution
The cartographic engine in the DHIS GIS is an open source Javascript library called OpenLayers. It provides an API for building rich web-based geographic applications similar to Google Maps and Bing Maps. The library includes components from the Rico JavaScript library and the Prototype JavaScript Framework. Health data is presented graphically through Scalable Vector Graphics (SVGs) on the OpenLayers map helped by the MapFish JavaScript client. MapFish is an open source web mapping development
framework that extends the Pylons development framework and provides support for spatial data.

The GIS user interface is mainly built by the use of ExtJS and GeoExt JavaScript frameworks. ExtJS is a JavaScript library for building interactive web applications using techniques such as Ajax, DHTML and DOM scripting. Originally built as an add-on library extension of YUI by Jack Slocum, Ext includes interoperability with jQuery and Prototype. GeoExt brings together the geospatial know how of OpenLayers with the user interface of Ext JS to help you build powerful desktop style GIS applications on the web with JavaScript.

Setting up of DHIS GIS is basically a matter of populating the coordinates field of the organisation units in the database as the maps are generated from the coordinates information that are linked to the organisation units. No additional files are needed. As soon as the organisation units have coordinates, the maps will be available in the GIS module. Although it is possible to add/edit coordinates directly in the Edit Organisation Unit window (Maintenance → Organisation Units). It is recommended doing this in a batch job using the general import process in the import/export module of the DHIS2.

The import process will need a GML file with at least two properties of Name and coordinates. To generate the GML file, we have to start by installing the open source tool called ogr2ogr. This should be available for most Linux distros (‘sudo apt-get install gdal-bin’). For Windows, we have to get FWTools (http://fwtools.maptools.org). The most common format for GIS data is the ESRI shapefile, which consists of three identically named files with extensions .shp, .shx and .dbf (you can use ogr2ogr to convert between any formats, see the example below). After opening the .dbf in a spreadsheet application (for example, MS Excel), we have to make sure there is a field (column) called Name, which has organisation unit names, and that these organisation units are already existing in DHIS2. We have to make sure that all spellings are identical since the matching is done on this name. On Windows, we must open the FWTools Shell and navigate to the folder with the shapefile. The following command has to be issued (replace ‘output’ and ‘input’ with the actual names): ogr2ogr -F GML output.gml input.shp.

The column in the .dbf file with the orgunit name would then have been converted to an XML element inside the GML file. Next, the GML file has to be opened in a text editor (for example, Notepad++ for Windows or Geany for Linux) and do a search/replace to make sure this element is called exactly ogr:Name (case sensitive), for example, <ogr:Name>Badjia</ogr:Name>. Import the GML file into DHIS2 through the import process in the import/export module (under services menu). There is no need to zip the file. Change import type to preview before doing the import to see which changes that will be made and resolve any issues with orgunit name matching that may exist. After importing, the coordinates will be added to the organisation units’ metadata and also be available from the organisation unit edit window.

Basic Usage and Concepts of DHIS GIS

Thematic mapping: The left panel of DHIS GIS lets you use the defined organisation units and health data for thematic mapping. All you need to do is select the desired indicator/dataelement-period-organisation units combination in the left side menu. Then by clicking either the boundary or level, the organisation unit selection
window is opened. Here the user can select both the parent organisation unit and the level of its children. If the database has coordinates for these organisation units, they will appear on the map. Calculation method alludes to the size of the legend classes. Set to equal intervals, they will be ‘highest map value – lowest map value/number of classes’. Set to equal group count the legend creator will try to distribute the organisation units evenly. Choose fixed bounds and you may define your own class break values, type for example, ’20, 40, 60’ using a comma to separate each of them. The map view combo box lists all map views (favorites) saved by the user. The settings that are stored in the map view will be automatically applied to the thematic map panel.

Favorite map views: This window will save the current thematic map view in order to restore it whenever you want via the map view combo box in the thematic map panel. By adding your views to DHIS2 Dashboard, you may access them there by inserting map views into one of the link areas.

Legend sets: A legend set may be connected to many indicators, but an indicator may only have one legend set. Thus, multiple indicators can be selected while creating a legend set. When an indicator that has a legend set is selected in the thematic map panel, the number of classes, low colour and high colour is automatically set. Example usage (vaccination coverage):

- Create the legends that are going to constitute the legend set.
- The first one could be “Low bad” (display name), 0 (start value), 30 (end value), red (colour).
- Then create “Medium”/30/70/yellow and finally “High good”/70/100/green.
- Now, open the legend set panel, type for example, “High is good” as display name and select the desired legends below.
- Multi-select the three legends by pressing and holding the Ctrl/Shift button when selecting.
- Then click the register button to store the legend set.
- Assign indicators/data elements to the legend set in one of the two last panels.
- Select the legend set in the combo box and multi-select items in the list that is given below.
- Click the assign button to update the legend set.

Export map images: Click the image icon on the map toolbar and the print window will open.

- Title: Image title, will appear as a headline in the image.
- Layers: Choose whether polygons, points or both will be printed.
- Width/Height: The pixel resolution of the image. Choose among the predefined small (800 × 600), medium (1190 × 880), large (1920 × 1200). If you want to exclude the legend from the image, uncheck the legend checkbox. Finally click the export button to print the image on png format.

Layer options: Right or left click any layer in the layer tree in the upper right corner to open the layer options context menu. There are currently three layer options available for vector layers: Locate feature, Show/hide labels and Opacity
District Health Information Software – What it is

Implementing Good Design

Figure 6.8 User interface in the DHIS GIS module
Locate feature (organisation unit): Click one of the units in the list and its colour in the map will change to the colour you have specified in the Highlight colour combo box. If the list of organisation units is long you can easily create a text filter in the text field below.

Show/hide labels: By clicking this option, a label with the organisation unit name and its value (based on the current indicator/period selection) appears on the map.

Opacity: Adjust the opacity/transparency of the layer in the map.

6.4.3 DHIS Mobile: Mobile Module in DHIS2

Mobile telephones and Internet over the mobile network are drastically changing the way health information is interchanged, managed, accessed and used in the DHIS2. What we see is that the different types of devices and technologies are converging, not into one ‘thing’, but into a seamless range of computing technologies. DHIS2 primarily being a web application can be hosted in the cloud and can be accessed by many mobile devices ranging from cheap feature phones to full blown desktops/laptops that use wireless modems and connect with mobile phone networks to communicate with DHIS2.

In Table 6.2, the range of mobile technologies and their implementation in DHIS2 is outlined:
### Table 6.2 Integration models of mobiles with DHIS

<table>
<thead>
<tr>
<th>Application</th>
<th>Primary Use</th>
<th>Complexity</th>
<th>Status in DHIS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formatted SMS</td>
<td>Data is typed as a formatted text, with codes representing different meaning of terms and values</td>
<td>Difficult to use, simple to implement</td>
<td>To send data from any phone with need to install application on phones. This is also used to send feedback or notification from DHIS2 to the user.</td>
</tr>
<tr>
<td>Form-based SMS</td>
<td>A form-based application is used to capture data and then data is transmitted as SMS</td>
<td>Simple to use, somewhat complex to implement</td>
<td>This has been implemented in DHIS2 since Mar’09 and has been used by more than 6000 users to report monthly, weekly or daily data.</td>
</tr>
<tr>
<td>Form-based GPRS</td>
<td>A form-based application is used to capture data and then data is transmitted to a web application using GPRS or other modern data services</td>
<td>Simple to use, simple to implement once web application is deployed</td>
<td>This has been developed to download forms from the Internet onto the phones. These are used to report aggregate data, but also for individual patient-data and also provide activity plans for health workers.</td>
</tr>
<tr>
<td>Web Access</td>
<td>The web browsers in mobile phones are used to connect to a web application and data is directly fed into the web application. Browsers use data services in mobile phones to communicate with web servers hosting the web application.</td>
<td>Somewhat complex to use, easy to implement once web application is deployed</td>
<td>A light version of DHIS2 allows users with smart phones to use the phone browsers to view the DHIS2 web application. This allows users to do data entry, view graphs and perform some analysis. The full DHIS2 application will be available to be compatible with mobile browsers.</td>
</tr>
<tr>
<td>Interactive Voice Response (IVR) Systems</td>
<td>Mobile phones are used to make calls to a phone number and then keypad is used to respond to questions heard on the phone call.</td>
<td>Easy to use, complex to implement and requires hardware/software for setting up the system</td>
<td>Many projects use this to collect data by allowing users to type-in keypad responses to the questions asked on the phone.</td>
</tr>
</tbody>
</table>

While there are other hybrid ways to capture and send data, the above table illustrates the broad governing principles and approaches for data capture and reporting through
mobile phones. Mobile phones can be used with existing DHIS2 deployment including datasets and forms. Mobile phones cannot be looked at as standalone devices in an infrastructural context free mHealth perspective, but as part of the overall health information system and integrated architecture. In the following table a SWOT analysis on Strengths, Weaknesses, Opportunities and Threats, regarding mobile technologies and the integrated architecture is given in Table 6.3:

<table>
<thead>
<tr>
<th>SWOT</th>
<th>Mobile technology and Internet over mobile network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strengths</td>
<td>Cheap devices; easy to use; low power usage; deep marked penetration; widespread network coverage, allows easy use by mobile health workers.</td>
</tr>
<tr>
<td>Weaknesses</td>
<td>Limited visualisation; small screen size; low processing power; low memory availability; small keypads.</td>
</tr>
<tr>
<td>Opportunities</td>
<td>Scalable, less investments to maintain, already established installed base; Smartphones and portable computers are getting cheaper and network/GPRS getting better, opening up for seamless convergence across the range of technologies.</td>
</tr>
<tr>
<td>Threats</td>
<td>New vertical initiatives provide end-to-end infrastructure for fragmented sections of the Health Information System; cloud infrastructure leads to outsourcing of software based services from poor to rich nations; outsourcing of both the ‘value adding chain’ and the ‘learning, innovation and entrepreneurship chain’ to rich companies in the North.</td>
</tr>
</tbody>
</table>

The integrated disease surveillance and response system is an example of a module inside DHIS2, which is adapted to mobile technology for reporting and receiving data (Figure 6.10). The module is processing response on aggregate data about communicable diseases, preferably through mobile phones because of timeliness of data reporting.

The system is based on generic rules for triggers that are created by data managers or disease surveillance officers. Based on these rules, triggers of actions are performed and these could range from notifying a group of users over email, SMS or on DHIS2 user interface to perform ‘suggested actions to be taken’ as a response to the trigger. The rules for the triggers are created by using min/max ranges of data value for a data element, making comparisons with values of other data elements, indicators, geographical groups/distances, and known disease prevalence rates.

6.4.4 Layers and Open Source Frameworks in DHIS2

After having discussed three key modules in DHIS2, we elaborate in this section on the open source components that are used in DHIS2. From its inception in late 2004, open source principles have been important for the development of DHIS2. This has several dimensions; not only is the software distributed under a specific open source license, but there are implications for the libraries and tools utilised and for the organisation of the development.
The particular legal regime chosen for DHIS2 is the well-known BSD license,¹ which stipulates very few restrictions on what people can do with the programming source code. As with any FOSS project, the license meets the Open Source Definition,² which means people are free to modify and redistribute the source code. Additionally, they may even incorporate the source into proprietary software without releasing the source of the derivative work. This last point is the main contrast with the relatively restrictive General Public License (GPL), which is the most widely used open source license.

A number of leading open source Java frameworks are employed in the DHIS2 development which help to structure and generalise the code. The core frameworks, the various components, and the developer toolkit are themselves the product of large

---

¹ See http://www.linfo.org/bsdlicense.html for a thorough discussion of the BSD license and how it compares to the GPL

² http://www.opensource.org/docs/osd
collaborative communities, which often also includes large commercial companies. These communities represent both rich and varied demands as well as a number of resources, which together help ensure continued refinement and rapid development. In several cases, this has resulted in the projects taking a leading role within their fields, often emerging as de facto standards. Since the frameworks themselves are freely available without license and cost constraints, they form a common platform which attracts large numbers of people to help with further refinement while using it for their own needs. Their wide recognition and the open licenses also serve as guarantees against vendor lock-in: any number of developers around the world can be hired to improve code built on these frameworks, making them a suitable basis for stimulating the local software industry in the South. However, as with commercial software, support in the form of documentation, training, and certifications is key.

DHIS2 uses a three-tiered architecture, including:

- **A data layer**: This is where the data is stored and retrieved and kept as neutral and independent as possible with regard to its usage by applications and the business layer. This separation of the data from the logic is important for scalability.

- **A business logic (service) layer**: This is where the knowledge of the health sector ‘business’ is handled, such as the metadata structure, calculation of indicators, evaluation and aggregation rules, and the actual processing of data. It also serves as a bridge between the data layer and presentation views. The data can also be exposed as web services which can also be used by other applications.

- **A presentation layer**: This is a layer which handles the user interfaces. The presentation layer translates information from the layer below and formats it into user views, and transmits the user’s commands to the lower layers. In fact, the layered architecture allows for alternative presentations of data from the service layer – for example, specialised to small screens like those on mobile phones. In the DHIS GIS module, for example, the dialog boxes, colour pickers and navigation controls (zoom, pan, and so on) that are used to select and design maps are part of the user interface (the presentation layer), whereas the actual polygon and point coordinates and indicator values are provided by the layers below. Furthermore, for online installations, map images are pulled in from external services like OpenStreetMap and Google maps as background to the map views.

This layered architecture is depicted in Figure 6.11.

Such clearly defined layers allow different objects to follow the same rules, enabling modularisation, which is the key to collaborative development among teams distributed across countries and continents.

- The data layer consists of a standard SQL database and Hibernate\(^3\) is used to link Java objects to the database and thus also allows for abstraction of the data access layer and independence from the particular database chosen. DHIS2 comes with an in-memory Java database called H2, but is usually deployed using an external database engine such as PostgreSQL or MySQL.

---

\(^3\) [http://www.hibernate.org/](http://www.hibernate.org/)
For the service layer, the Spring framework is used to tie together the server side components application logic. It provides a consistent programming and configuration model, and comprises a range of services: A container for the various DHIS2 modules, transaction management, authentication and authorization via the Spring Security sub-project and messaging between components.

Struts 2 is a well-established and popular framework for developing sophisticated web applications. The presentation layer also includes CKEditor for designing input forms, Jasper Reports for analytical output and charts, and GeoExt and MapFish for the GIS module, as well as the powerful JQuery library for interactive JavaScript based functionality (for example, updating only parts of a web page using AJAX instead of reloading the whole page).

In addition to the core frameworks, the system also relies on a range of open source Java libraries for functionality like XML processing and unit testing. All of this is made possible because of the Free and Open-Source Software culture of sharing and avoiding reinventing the wheel.

Furthermore, the whole development process is transparent, as all changes to the source code can be tracked on the online hosting platform, Launchpad. The same openness applies to bugs and their resolution, as well as feature requests (called

---

4 http://www.springsource.com/
5 http://struts.apache.org/2x/index.html
6 http://launchpad.net/dish2
‘blueprints’), which are assembled into a roadmap for future releases. In addition, the developers use a suite of open source tools which include Maven, Bazaar, Eclipse or NetBeans, and a number of Firefox browser add-ins to work with and build the code.

### 6.5 Priorities for Future Development of the DHIS2

Looking at the current landscape, by 2012, and looking into the future, we identify and outline the following priority areas for the further development of the DHIS2+:

1. **Integration from ‘below’ and integration from ‘above’**: Developing DHIS2 further as a platform for integration of data sources – from below and as a platform for information user web services:
   - **From ‘below’** – Integration of data sources; continue to strengthen the data warehouse and interoperability capabilities of the DHIS. The SMX-HD standard for exchange of metadata and statistical data illustrates the open standard based approach which we see as a priority to expand into real use, whereas to establish practical and useful ‘production’ use-cases will be the main approach to achieve this. We will work with in particular our partners in the iHRIS (human resources), OpenMRS (medical records) and OpenLMIS (logistics) to establish such use cases.
   - **From ‘above’** – Develop effective web-API (interface) to the DHIS2. In order to foster innovation in and around the DHIS user and implementer communities, it is necessary to strengthen the ease with which new modules may be developed and interfaced to DHIS2. This aim may be seen as targeting two levels of use and development:
     - Make it easy to develop add-on extensions, ‘apps’ or modules, using data and functionalities in DHIS2, mainly for external developers. For example, a web site for health information in a country accessing data and/or analytical tools in the DHIS2.
     - Make the interfaces more durable and general so that more ‘internal’ developers and DHIS implementers find it easier to make extensions and plug-ins that are easily brought to new versions of DHIS2 and not ‘locked’ to versions that quickly become past versions.

To develop effective portal solutions is another priority which may also be grouped under the integration from above label. Portals will be key mechanisms in disseminating data and providing targeted information to a range of users, from super users to the general public.

2. **‘Semi-online’; optimising Internet connectivity and functionality in developing countries; ‘being online even while offline’**: Internet connectivity in the developing world, and in disadvantaged regions, will be relatively poor for the foreseeable future. As argued earlier, online web computing and one central server represents a tremendous improvement to the traditional offline installations of DHIS. In order to achieve this in more constrained networking contexts, the DHIS is optimising the availability of the Internet both at the levels of data use and data entry (Figure 6.12):
   - Providing offline access to data and analytical tools. Data for selected areas and periods are downloaded and updating a datamart, by only adding new data. The datamart is then used to load Excel pivot tables and eventually other
analytical tools. Archive facilities for downloaded maps and reports are also provided. This functionality provided by a small offline application was ready for testing first quarter 2011 and will be made more user-friendly and usable in the future.

- Providing offline data capture. The HTML 5 standard is including offline database and storage facilities. We expect to have this functionality running during 2011.

3. **Business intelligence and decision support systems**: A key longer term goal is to strengthen DHIS2 as a platform for supporting decision-making processes and monitoring organisational performance. We need to get the DHIS2 project fully part of the current advances in Business Analytics and BI technologies, now increasingly labelled, not only BI, but BAI technologies. In addition to the general use linked to decision-making and monitoring and evaluation in health, increasing focus on ‘payment for performance’ in health, make it important to translate the BAI approaches developed in business and commercial domains to the context of global health. GIS, dashboards, monitoring ‘live data’ against thresholds calculated from, e.g. surveys as an effective way to integrate survey data with routine data, and so on, represent some of the areas where DHIS2 is already working and which plan to be further strengthened.

4. **Mobile web interface**: Better and stable web and GPRS-based mobile and touch screen solution and data capture using low-end devices: We see the
mobile Internet and the converging of computers, iPads, netbooks and mobile telephones into online computing devices as the key way to achieve the web-based DHIS in developing countries. Interchanging between touch screens and/or key boards over web and/or GPRS will be the new standard for online computing in developing countries. Developing robust solutions in this area is thus an important priority area for DHIS.

DHIS also needs solutions for contexts with poor infrastructure written for low-end java-enabled feature phones. These can work offline from the DHIS2 instance, but sync with the server through web APIs when GPRS is available and also send data reports over SMS. Where mobile infrastructure is mature and lower-end android smart phones are starting to become available, an important focus is on extending and tuning the DHIS 2 Web interface for use in the browser on these devices.

5. **DHIS academy and documentation:** Given the rapid scaling of the DHIS2, training and documentation are now the key obstacles. The DHIS Academy will be developed to become a self-contained ‘package’ of material and live and generically updated documentation. User-friendly documentation is a related area which will be focused. Various training materials being developed within the different HISP nodes will be converged using online learning platforms like Moodle.

6. **New modules:** Where to develop new modules as part of DHIS, and where to work with partners depends on context, situation and priorities. The following are current priorities:

- Further development of the (vital) event-based module and lightweight patient/client tracker and aggregation. To develop use cases and real use of the aggregation functionality will be prioritised.
- Improve basic logistics and inventory module. Functionality in the data entry form for logistics ordering form; calculate order based on stock and consumption.

### Summary

In this chapter, we have provided a kaleidoscope of the DHIS artifact including its past development history, current modules and components, and some of the identified future priorities. Key summary points include:

1. The first version of DHIS was based on MS Office platform and distributed free. It has been developed in South Africa and continues to be supported from there.
2. Early DHIS development was based on a strong participatory design philosophy, and principles of user empowerment and local control were key design aims that were inscribed in the design, development and implementation of the software. The interaction between DHIS development and use was mediated with a strong public health focus.
3. From 2004, a process of remodeling of DHIS1.3 as DHIS1.4 was initiated, where internationalisation was a key design aim.
4. Also from 2005, the DHIS2 development was initiated by Oslo based on a stack of bleeding edge Java technologies, and using principles of open source and shared distributed development.
5. Guiding aspects of DHIS2 architecture include:
   a. A three-tiered architecture including data layer, business logic layer, and presentation layer.
   b. Based on data warehousing and BI functionalities.
   c. Built as a platform to support the development of supporting systems and modules.

6. A recent development is of the Web API based on REST architecture style in order to achieve a loose coupling between the DHIS service layer and additional software products.

7. Key DHIS2 modules include:
   a. Patient and tracking.
   b. GIS.
   c. Mobile reporting.

8. Priorities for future development:
   a. Integration from “below” as well as “above”.
   b. Being “online” even while offline.
   c. Supporting BAI (Business Analytical Intelligence) functionalities.
   d. Building a stable web and GPRS based mobile integration.
   e. Strengthening of DHIS Academy for regional capacity building efforts.

Reference