journal homepage: www.intl.elsevierhealth.com/journals/ijmi

Propagation of program control: A tool for distributed disease surveillance

Johan Gustav Bellika[a,*], *Toralf Hasvold*[b], *Gunnar Hartvigsen*[a,c]

[a] *Norwegian Centre for Telemedicine, University Hospital of North-Norway, P.O. Box 35, NO-9038 Tromsø, Norway*
[b] *Department of Community Medicine, University of Tromsø, Norway*
[c] *Department of Computer Science, University of Tromsø, Norway*

ARTICLE INFO

Article history:
Received 9 December 2004
Received in revised form
8 November 2005
Accepted 16 February 2006

Keywords:
Distributed systems
Epidemiology
Population surveillance
Disease surveillance
Electronic patient records

ABSTRACT

Purpose: The purpose of the study was (1) to identify the requirements for syndromic, disease surveillance and epidemiology systems arising from events such as the SARS outbreak in March 2003, and the deliberate spread of *Bacillus anthracis*, or anthrax, in the US in 2001; and (2) to use these specifications as input to the construction of a system intended to meet these requirements. An important goal was to provide information about the diffusion of a communicable disease without being dependent on centralised storage of information about individual patients or revealing patient-identifiable information.
Methods: The method applied is rooted in the engineering paradigm involving phases of analysis, system specification, design, implementation, and testing. The requirements were established from earlier projects' conclusions and analysis of disease outbreaks. The requirements were validated by a literature study of syndromic and disease surveillance systems. The system was tested on simulated EHR databases generated from microbiology laboratory data.
Results: A requirements list that a syndromic and disease surveillance system should meet, and an open source system, "The Snow Agent system", has been developed. The Snow Agent system is a distributed system for monitoring the status of a population's health by distributing processes to, and extracting epidemiological data directly from, the electronic health records (EHR) system in a geographic area.
Conclusions: Syndromic and disease surveillance tools should be able to operate at all levels in the health systems and across national borders. Such systems should avoid transferring patient identifiable data, support two-way communications and be able to define and incorporate new and unknown diseases and syndrome definitions that should be reported by the system. The initial tests of the Snow Agent system shows that it will easily scale to national level in Norway.

1. Introduction

The severe acute respiratory syndrome (SARS) outbreak in 2003 showed that existing electronic health record (EHR) systems as well as healthcare and disease surveillance systems are not adequate tools for disease surveillance. Disease outbreaks can range from a local cluster of a communicable disease to a global threat, as in the case of SARS [1]. A

* *Corresponding author*: Tel.: +47 95 74 80 49; fax: +47 77 75 40 98.
E-mail address: johan.gustav.bellika@telemed.no (J.G. Bellika).
1386-5056/$ – see front matter © 2006 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.ijmedinf.2006.02.007

syndromic or disease surveillance system should therefore serve all levels, from the local healthcare service to the national and global level [2,3].

Syndromic or disease surveillance has two distinct phases: a detection phase, the goal of which must be to detect an outbreak as early as possible, and a monitoring phase, where the actual disease is known and the goal is to prevent the disease from spreading. In the latter case, it is important to detect new cases and to track existing cases of the disease.

To enable early detection of disease outbreaks and the monitoring of the diffusion of a disease through the identification of abnormal disease patterns, "sensors" in the form of a software system should be distributed to all the points in the healthcare system where patients present, or where they leave traces of evidence of illness [4]. Lober et al. [5] provide a good description of the information sources that can be monitored to detect an outbreak as early as possible. Where affected people present may depend on the type of disease and on the organisation of the healthcare service. In Norway, which has a system of personal general practitioners (GPs) with gatekeeper functions to secondary and tertiary levels of care, patients will typically consult their GP or the casualty clinic in the first instance. Patients are only likely to present at a hospital emergency unit if they are acutely ill (for example, if they have been injured in an accident, or referred to the casualty clinic by the GP). A telephone survey in the USA, in New York, showed that 29.1% of persons with influenza-like illness (ILI) visited a physician, 21.4% called a physician for advice, 8.8% visited the emergency department, and 3.8% called a nurse or health hotline [6]. The EHR systems used by GPs are therefore likely to be very good sources for detecting disease outbreaks in a healthcare context where most people normally consult their GP in the first instance if they get ill. However, using patient data stored in the EHR system is difficult because of the privacy issues involved in exporting data from the EHR systems [7–11].

The EHR systems used by GPs in Norway use the International Classification of Primary Care (ICPC), which is more specifically symptom-related than ICD-9 and ICD-10, making it more applicable for syndromic and disease surveillance use. Unfortunately, there is no messaging standard such as HL7 available for primary care services or hospitals in Norway, which could enable the use of an available open source solution for syndromic or disease surveillance [2,11,12].

Syndromic or disease surveillance is relatively easy to implement for all well-defined and existing diseases. If a new disease appears, as in the case of SARS, or as result of a deliberate spread, disease surveillance may become more difficult. First, we do not have codes for specifying *confirmed*, *probable* or *possible* cases of the new disease. This may easily be solved if it is sufficient merely to update the coding system. How fast such an update can be performed depends on a number of factors: How fast can the coding system be updated, how fast can the new version be disseminated, how fast can the updated code-set be employed, etc. Updating a disease classification is normally done very infrequently and the reliability of such classifications is therefore problematic when the goal is to count occurrences of a new disease. The creation of an additional coding system for contagious diseases, which enables quick updating, may provide a solution to this problem. However, this is also problematic, because duplication of coding systems adds to the quantity of information, which must be entered into the EHR systems. Basing surveillance of new diseases on existing classification categories is therefore problematic and adds to the complexity of the problem. However, without such disease codes we cannot automatically count the number of cases and get a picture of the geographical diffusion of the disease.

Another solution may be to use a list of prodromes or early symptoms (syndrome definition) as basis for definitions of possible and probable cases, but we cannot guarantee that a new disease will be covered by such predefined lists of symptoms. Conditions with similar symptom patterns may be interpreted as possible or probable cases, thereby polluting the diffusion pattern. In the case of SARS, a visit or contact with a person who had visited a geographic area where there were reported cases was a main indicator of probable or possible cases. Such conditions are not known beforehand [13]. In the case of SARS, the definition of possible and probable cases evolved over the weeks following the outbreak from initially being very vague to becoming a very specific set of symptoms. It may therefore be necessary to distribute new and updated syndrome definitions if local classification of cases based on the syndrome definition is required. This dynamic behaviour of a disease or syndrome definition makes it hard to build software that automatically identifies and distinguishes between possible, probable and confirmed cases.

There are at least two approaches to building a surveillance system: a centralised and a distributed approach. All of the systems reported by Lober et al. [5] and Kun et al. [14] are based on the centralised storage and analysis of data. These systems have been classified as using first- or second-generation architectures for data integration [9]. The Real-time Outbreak and Disease Surveillance system (RODS) from the University of Pittsburgh [12] also uses a centralised approach. The emergency departments covered by the system send HL7 messages containing ICD-9 codes in real time to a centralised inference system where outbreaks are detected. When a disease outbreak is detected by RODS, the appropriate authorities are informed about the situation. The outbreak detection process is run periodically on the data available to the system. The Bio-Surveillance, Analysis, Feedback, Evaluation and Response (B-SAFER) system [2,15] is based on a federated approach involving the open source OpenEMed infrastructure. OpenEMed [16] (formerly known as TeleMed [17]) is based on the Clinical Observation Access Service (COAS) [18] and Patient Identification Data Service (PIDS) [19] CORBAMed specification of the Object Management Group. B-SAFER and OpenEMed take a more interoperable approach that demonstrates the value of the federated approach to syndromic surveillance, which makes it possible to view and manage information at several levels ranging from local to regional, national and global levels [2].

We have constructed a system, which we have named "The Snow Agent system", after the famous Dr. John Snow. In the Snow Agent system we also take a distributed approach, where all contributors of data to the system also have the opportunity to access information about the current situation in a geographic area directly from the system. In that sense, the Snow Agent system is a true "peer-to-peer" network among general practitioners. According to Lober's classification [9], this system would be classified as using third-generation data integration. The Snow Agent system does not depend on the centralised storage of patient information, and may therefore be used by any group of health institutions or general practitioners that want to share data about the health of the population, without exposing any patient-sensitive information. It is designed for everyday use by GPs who want to share information about contagious diseases and disease outbreaks in their local community. It also avoids the problematic privacy issues, because no patient-identifiable information ever leaves the EHR system. It also supports the requirement of regular usage, which is important in being prepared for an emergency [3,20].

The Snow Agent system has a number of other applications besides syndromic or disease surveillance and epidemiology. Most important is the extraction, collection and visualisation of a patient's health record from distributed EHR systems for authorised persons. The system also makes it possible to generate statistics about the activity in the health institutions covered by the system. The messaging system used by the Snow Agent system may also be used for transferring any kind of data between health institutions, which is a valuable spin-off. However, in this paper we focus on the requirements that we argue syndromic, disease surveillance and epidemiology systems may need to meet when facing a local, national or a pandemic outbreak of a communicable disease. Some of these requirements also correspond to the requirements for extraction, collection and visualisation of a patient's health record. The paper presents the core principle utilised in building a distributed system for monitoring and interrogating the status of the population's health, and discusses the results of the initial system tests.

2. Materials and method

The work reported here has been conceived within the engineering paradigm. The processes involved in building a system can be divided into the distinctive phases of analysis, system specification, design, implementation and testing. The system described here has gone through several iterations of analysis, system specification, design, implementation, and testing.

The requirements listed in the section below are derived from the following sources: (1) a project that evaluated the needs and conditions for access to patient information from geographically distributed electronic health records [21]; (2) an analysis of the SARS outbreak in March 2003; (3) an analysis of a whooping cough outbreak in Troms County in North Norway in October/November 2003. The requirements were validated and augmented by a literature study of syndromic and disease surveillance systems and solutions directed towards early detections of bio-terrorist attacks [2–5,8–15,20,22–40].

The Snow Agent system is a redesign and a complete re-implementation of the Virtual Secretary system [41] in terms of the requirements reported here. The initial Virtual Secretary system was developed at the Department of Computer Science at the University of Tromsø. The current version is built on top of the Jabber Instant Messaging and Presence system. (see details in Section 4, "propagation of program control" below.)

To test the system components, we implemented a distributed epidemiology service which extracts epidemiological data from simulated EHR databases. The test data used for constructing the EHR databases is described in the next section. The hardware and software used for system testing is described in Section 2.2 below.

2.1. *Test data*

Getting access to production data and system is problematic from both a privacy and resource point of view. We therefore chose to create fictitious EHR databases based on data from the microbiology laboratory at the University Hospital of North Norway. This laboratory serves a large geographical area and many GP clinics, and therefore has data that is well suited to our needs.

The data used in the system test was laboratory test data from the microbiology lab. The dataset contained whooping cough (pertussis) test results performed between 10 November 2002 and 1 January 2004. The lab tests were ordered by GPs from the three northernmost counties in Norway. The test dataset contained the following information:

The test data:

reks_status	CHAR(1)	always "V"
report_status	CHAR(1)	always "F"
test_received_date	CHAR(8)	date and time for receiving test
test_received_time	CHAR(4)	
result_sent_date	CHAR(8)	Date and time for sending test result
result_sent_time	CHAR(4)	
patientid	CHAR(8)	ID number of the patient in the laboratory system
test_requester_code	CHAR(8)	code to identify requester in the laboratory system
requester_municipal_code	CHAR(4)	code for the municipal location of the requester
patient_gender	CHAR(1)	gender of the patient
patient_age	CHAR(3)	age of the patient
patient_postal_zip_code	CHAR(5)	the patient's residential postal code
patient_municipal_code	CHAR(4)	Municipality where the patient lives
analysis_type	CHAR(4)	Number identifying analysis in the laboratory system
analysis_name_proffdoc	CHAR(30)	Name of analysis in the Proffdoc EHR system
analysis_name_winmed	CHAR(10)	Name of analysis in the WINMED EHR system

test_result	CHAR(6)	NEG,POS or number value
Notification	CHAR(4)	Must the disease be reported, always "N"

From the pertussis lab dataset we constructed a very limited EHR database by using a number of assumptions:

1. We assume that it takes 1 day to deliver the test sample to the hospital laboratory. The consultation date with the GP is therefore test_received_date minus 1 day, and consultation time equals test_received_time.
2. We assume that all GPs have the same behaviour regarding entering data into the electronic patient record (EHR) system.
3. They all create a consultation entry in the EHR system using the date and time values specified above.
4. They all do a test from the posterior nasopharynx using darcon or calcium alginate swabs, which they send to the hospital laboratory and order a "*B. pertussis* PCR" (*Bordertella pertussis*) laboratory test (ICPC2 process code-33 Microbiological/Immunological Test) (code 6232 and 6192 in the lab system).
5. If the lab report is negative, the GP does nothing. If the lab test is positive, the doctor creates a telephone consultation with the patient using "R71 Whooping cough" as the diagnosis, and prescribes *erythromycin*. Five days later the patient is assumed not to be contagious to others. (Pertussis is most contagious during the first week. Three weeks after the onset of symptoms, the patient is seldom contagious.)
6. The "*X* days since onset" value for the whooping cough patients will probably vary widely. We assume the distribution of *X* is Gaussian. We assume that most people will not see their GP before the second stage of the disease has started (7–14 days after onset of symptoms), when the severe cough with whooping and vomiting begins. This value may also vary with the age and gender of the patient. We assume it is likely that babies under the age of one (below 1 year in the test dataset) probably see their doctor earlier than older children do. We also assume male patients wait longer to see their GP than female patients do. Knowledge about an outbreak probably also affects how early a patient sees their GP. We assume that children <2 years see their GP 2–5 days after onset of symptoms, children ≥2, <8 years see their GP 3–7 days after onset, and that children ≥8, <18 years see their GP 5–10 days after onset. Female ≥18 years see their GP 10–15 days after onset and male >18 years see their GP 12–20 days after onset.

From negative laboratory reports, the following EHR documents were created: First consultation, lab request and lab report. From positive laboratory reports, the following EHR documents were created: First consultation, lab request, lab report, second consultation (telephone), drug prescription (erythromycin).

The dataset contained 7939 rows of lab results from 3997 different patients. The dataset contained 6463 negative lab result records and 198 lab result records with "POS" results, 794 patients with test result ≠ "NEG" (non-negative blood sample test results) and 194 patients with a "POS" test result. The dataset contained 487 test requesters, including hospital internal requesters. Of these test requesters, 459 were from outside the hospital and 28 from inside the hospital. The dataset covered 372 postcodes, 153 postcodes with test result ≠ "NEG" (non-negative blood sample test results) and 54 postcodes with test result = "POS". The dataset contained patients from 138 municipalities, where 62 municipalities had test result ≠ "NEG" (non-negative blood sample test results) and 25 municipalities with test result = "POS".

There were 4528 female patients and 3411 male patients in the data set, including 103 female patients and 95 male patients with a "POS" test result, and 779 female patients and 697 male patients with test result ≠ "NEG" (non-negative blood sample test results). In the system test, we only used confirmed cases as test data (test result = "POS").

In addition to the laboratory dataset, a map was needed, on which to plot the data. The map was created from bitmaps made available in the NorgesHelsa system [42]. The bitmaps were first converted to SVG [43] polygons. Then the map polygons were inserted into a MySQL database [44] in a convenient format for map generation, and coded with geographical information.

2.2. Hardware and software used

In our experiment, we wanted to determine the performance of the system in a close to reality configuration of the system. In our experimental setup, we used computers located in Tromsø in Norway, Valencia in Spain and Brisbane in Australia. Server 1 was a desktop computer with Intel Pentium III 533 MHz processor, 1061 BogoMIPS [45,46], 60 MB RAM, and a 100 MBit/s Internet connection. Server 2 was a desktop computer with Intel Pentium 4, 2.4 GHz processor, 4767 BogoMIPS, with 512 Mb RAM running RedHat Linux v. 8.0, with a 100 Mbit/s connection to the internet. Server 3 was an IBM ThinkPad T21 2647 with a Pentium III 800 MHz processor, 1582 BogoMIPS, with 376 MB RAM, using a wireless network adapter (11 MBit/s). Server 4 was an IBM Thinkpad 390X with 191 MB RAM with an Intel Pentium II 447 MHZ, 891 BogoMIPS, processor connected by a wireless network adapter. Server 5 was an IBM Thinkpad 390X with 319 MB RAM with an Intel Pentium II 447 MHZ, 891 BogoMIPS, processor connected by a wireless network adapter. Server 6 was an Intel Pentium II 233 MHz (466.94 BogoMIPS) with 160 MB RAM using Slackware 8.1, Linux 2.4.18. Servers 2 and 3 were located behind a firewall, while servers 4 and 5 were located behind a wireless router connected to the Internet using an ADSL modem (512 kbit/s download and 256 kbit/s upload speed). Server 6 was connected to the Internet using a 128 kbit/s line. Server 1, 3, 4, and 5 used RedHat Linux version 9.0 OS. Table 1 summarise the hardware used.

3. Requirements

The results of the analysis described in Section 2 above identified the requirements for the tools to be constructed for distributed syndromic or disease surveillance. We have divided

Table 1 – Hardware and OS used for the system tests and in experiments conducted

Server	Location	OS	Memory (MB)	Network connection	Bandwidth (Mbit/s)	
					Upload	Download
1	Tromsø, Norway	RedHat Linux 9.0	60	LAN/WAN	100	100
2	Brisbane, Australia	RedHat Linux 8.0	512	LAN/WAN	100	100
3	Brisbane, Australia	RedHat Linux 9.0	376	802.11b/LAN/WAN	11	11
4	Brisbane, Australia	RedHat Linux 9.0	191	802.11b/ADSL	0.512	0.256
5	Brisbane, Australia	RedHat Linux 9.0	319	802.11b/ADSL	0.512	0.256
6	Valencia, Spain	Slackware 8.1, Linux kernel version 2.4.18	160	ADSL	0.128	0.128

the requirements into two sections: basic requirements for distributed disease surveillance systems, and hard requirements.

3.1. Basic requirements

The requirements which distributed surveillance and epidemiology systems should fulfil, are shown below.

3.1.1. Flexible connection models

While hospitals can afford a real-time connection to a centralised system such as RODS [12], this solution is not very suitable for all GP clinics. For some GP clinics a more suitable solution may be connection to a shared interconnecting network on a regular basis for reading emails, receiving electronic lab results and electronic hospital discharge letters. This has been the most common solution for GP clinics connected to the North Norwegian regional health network. However, the GP clinics are now migrating to an alternative solution, in which they are constantly connected to the health network. To be able to cover all connection alternatives, a distributed solution for epidemiology and healthcare surveillance must support periodic connection to a shared interconnecting network, minimise communication costs, and tolerate narrow bandwidth.

3.1.2. 100% coverage

Coverage is perhaps the most important aspect of a distributed epidemiology and surveillance system. The best approach to achieving coverage is through developing a free and open-source solution that has a minimum of hardware and software costs. The RODS [12] system and B-SAFER [2] system, and several other tools used in outbreak detection, use this approach.

3.1.3. Support an internal EHR standard or format

The heterogeneity of EHR solutions for general practices is problematic because of the range of different ways in which they store medical data. Existing systems may have problems converting information stored in the internal system to a standardised format [9,47–49]. They may also lack important information items such as the onset date of a communicable disease. A distributed epidemiology and disease surveillance system would benefit from using a standardised data format internally. A standardised data format will make querying of data easier.

3.1.4. Automatic and independent

A syndromic or disease surveillance system must be automatic (independent of manual data entry for surveillance purposes) to be sustainable [22,37]. The primary task of healthcare workers is to treat patients, not to perform computer system configuration or system maintenance. However, many institutions in primary care are small, and do not normally employ professional system administrators. In some areas, there is a shortage of available staff with the skills for system administration. A system should therefore ideally require zero configuration and maintenance. This requirement may be hard to meet because most systems need at least a minimum of configuration and maintenance to operate smoothly. This can be difficult to achieve because of the need for manually specifying and ensuring a security policy (see Section 4.7 below).

3.1.5. Asynchronous

A system intended to enable the computation of global epidemiological queries must be asynchronous. It is difficult to anticipate how a globally distributed computation can be performed while a user is waiting for a system response. If we allow participating servers to connect periodically to a shared network, we must ensure that the computations performed on the servers are independent of the computation requester being available and online. Asynchronous computations are also more autonomous.

3.1.6. Dynamic reporting frequency

A surveillance system must be dynamic in terms of when and how often calculations are performed. The need for reporting frequency may span from running a query every week to several times a day. If an outbreak is detected, the frequency of calculating the diffusion of the disease may need to be increased. The responsibility for switching a GP clinic system into online mode (constantly connected to the shared interconnecting network) may need to be external to the GP clinic. This decision should only be made after a local authentication and authorisation of the request.

3.2. The hard requirements

To be able to meet our objective of not revealing patient identification, and of not being dependent on centralised storage of information about individual patients, the system needs to be distributed. Classification of cases into disease or syndrome groups must therefore be done locally, within each EHR system. Many of the syndromic surveillance systems in the USA utilise the 'chief complaints' free text field, and ICD-9 or ICD-10 codes in the HL7 messages to perform classification of events into syndrome groups. In these systems, the classification of cases against disease and syndrome groups is done centrally, which makes it easy to define and put into use new disease definitions.

In Norway and several other European countries, the GPs use the ICPC classification. ICPC is more directly related to symptom recording than either ICD-9 and ICD-10, which makes it more applicable for syndromic and disease surveillance. However, recent travel to affected areas or contact with infected persons is not part of the classification. Nor are explicit values of biometrical measurements such as body temperature $>38.4\,^{\circ}$C. These features were among the criteria for identifying a possible SARS case. The most difficult requirement for existing EHR systems to meet is therefore the capacity to incorporate new syndrome definitions that need to be reported. It is important that these syndrome definitions are distributed and automatically incorporated into the EHR system, because of the limitations which paper-based forms have in an outbreak situation, as identified by Foldy [13].

For some EHR systems, a new disease or syndrome definition may require that new software is added to the existing EHR system. An example that illustrates this issue is the use of biometrical measurements such as body temperature, pulse, or blood pressure as part of the disease classification. This problem does not apply to all EHR systems, as we will see below. Constructing, distributing and updating EHR system software are all time-consuming activities that preferably should not be necessary at the point when an outbreak of a new and dangerous disease is detected.

In the case of SARS, we saw that the definition of the disease evolved and became more specific over time. As new knowledge about a disease becomes available, it needs to be disseminated to all systems participating in surveillance. This may require that the EHR system is revised to include the new definition of the disease. Experience from Milwaukee, Wisconsin [13] shows that manual routines for updating screening forms are problematic.

The "counting" part of a distributed disease surveillance system also needs to be highly dynamic and to allow for inclusion of new disease definitions "on the fly". To be able to monitor the diffusion of a new disease, we argue that we need systems able to include new definitions of diseases in real time. Communication in such systems therefore must be two-way, enabling the flow of information about diseases or syndrome definitions in one direction, and information about occurrences and diffusion of the condition in the other direction.

4. Propagation of program control

The requirements above seem to present a perfect case for mobile code, also known as mobile software agent solutions [50,51]. Many implementations of mobile software agent solutions already exist. Mobile software agent technology makes it possible to build highly dynamic distributed systems that support asynchronous and autonomous computations. New functionality in EHR systems may be defined and disseminated as source code to the target systems. However, are mobile-code-based solutions imaginable in healthcare? The risks of mobile code are that it has the capacity to bring down the EHR system in a GP's office due to programming errors, deliberately or unintentionally dump patient data on the Internet, take all computing resources away from the GP, etc. Even if the source code is certified, it should be asked whether it is appropriate to trust software providers which assert that they "will not be held responsible", if patient data finds its way onto the Internet. Yes, indeed, it may be difficult to accept mobile-code-based solutions in healthcare information management.

Is the mobile-code-based solution the only solution able to meet the requirements? Mobile code is not suitable, because the physician responsible for the patient data should be able to know exactly what the software that accesses the patient data do. If not, it is unlikely that a physician will authorise the use of such a system. Authorisation is closely connected to the tasks performed. If the physician does not know beforehand what the software does, it will be inappropriate to provide authorisation.

Instead of moving code, we can get some of the flexibility inherent in mobile-code-based approaches by moving program control and data. Propagation of program control allows program control to be moved between hosts, in the same manner as mobile code, by transferring a task specification and data between the hosts. The principle is located somewhere between process migration and remote procedure call [52]. Instead of letting a process or software agent wander autonomously at its own will between hosts, we ensure that the software agent is under the control of several authorities at all times. These authorities may terminate the software agent at any time.

By applying this principle, we gain control over the code executed, but lose some of the flexibility of mobile-code-based solutions. As described, the Snow Agent system would not appear to provide a simple solution to the hard requirements outlined above. The system may however be used as a dissemination tool for technology that is able to meet the hard requirements above. We will come back to this issue in Section 4.8, below.

The current version of the Snow Agent system is a redesign and a complete reimplementation of the Virtual Secretary system developed at the Department of Computer Science at the University of Tromsø [41] in 1993–1998. The first version of the system was initially a mobile-code-based solution for mobile software agents. However, due to the security issues with mobile code [53–55], we have moved away from this approach. The current version of the Snow Agent system does not support mobile code, but allows program control to propagate between hosts using two daemon processes as the mobility enabling system services. These two services are the Agent daemon and the Mission Controller (MC). Their tasks and responsibilities are explained in more detail below. The Agent daemon and the Mission Controller are built as extensions to the Jabber extensible open source server version 1.4.2 [56], as shown in Fig. 1. Jabber implements the Extensible Messaging and Presence Protocol (XMPP) (IETF RFCs) [57,58], which allows users to show their presence, chat, and exchange instant mes-

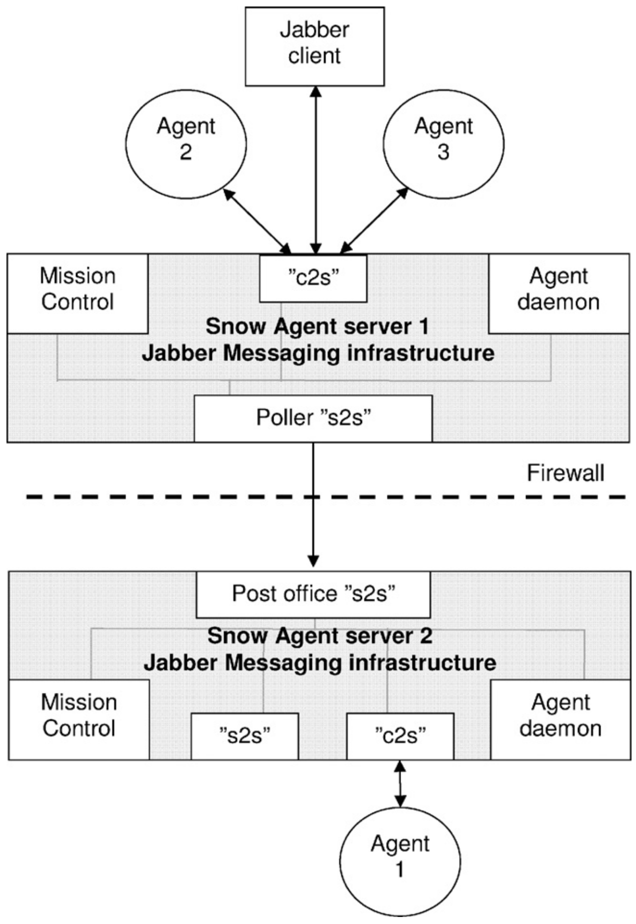

Fig. 1 – Snow Agent system server components implemented as extensions to the jabber XMPP server.

sages (XMPP messages). The XMPP messages are expressed using XML. The Agent daemon, the Mission Controller, Snow agents and users of the Snow Agent system also communicate using XMPP messages. The Jabber server communicates with Jabber clients using the "c2s" (client-to-server) component. Jabber talks to other Jabber servers using the "s2s" (server-to-server) component.

4.1. *Agents, missions and mission agents*

We define an agent as a software entity that performs a series of operations on one or more host computers on behalf of a user. An agent may be mobile and able to migrate or propagate between host computers. An agent has a mission that it seeks to accomplish. This mission is specified by the user (or "mission requester") and is expressed as a mission specification using XML. We use the term "mission agent" for the incarnation of an agent that runs on a single computer. The mission agent is instantiated by an Agent daemon that creates the mission agent processes. An agent (or agent mission) may therefore consist of several mission agents that run in parallel on multiple hosts or serially as a relay. A mission agent may also employ sub-missions by sending a mission specification to a Mission Controller.

4.2. *The Agent daemon*

The Agent daemon has more or less the same responsibility as traditional server technology and middleware such as CORBA, J2EE etc., and can easily be replaced by such technology. The Agent daemon manages local resources such as processor, disk, network, and agent applications. Its main responsibility is to instantiate, evict and restore mission agents. It negotiates agent missions with remote MCs and performs authentication of instantiated mission agents. In Fig. 1, above, the topmost Agent daemon has instantiated two mission agents. These mission agents may have been requested by Agent 1 or by a user using a Jabber client. The Agent daemon also provides information about running mission agents to system administrators, local and remote agents and users if allowed by the local system administrator and each Agent.

4.3. *The mission controller*

The Mission Controller functions as an intermediary between the user client and the Agent daemon which instantiate the processes based on the specification provided by the user. By performing mission control on behalf of the user, the user becomes independent of the agent execution and the agent becomes more autonomous and independent of the user. The MC knows the set of applications (or mission types) available for each user based on user identity.

The MC performs mission control by receiving mission specifications from end users, negotiating mission agent instantiation with remote Agent daemons, keeps track of the whereabouts of the agent and notifies the mission requester when a mission is finished, when mission results are available, or if an error occurred. The Mission Controller performs the migration service for agents executing on a host by receiving the mission specification from the local agent. The Mission Controller also provides mission control to agents that want to use sub missions on remote hosts.

4.4. *The Snow Agent*

A mission agent is a process that is instantiated by the Agent daemon based on a specification received from a MC. The mission specification is expressed in XML and contains the name and version of the application to run, user identity, certificates and signatures proving the identity and authorisation of the user, public keys for encryption of mission results, time and mobility constraints, list of hosts to visit and data used during the mission. The code used for constructing the mission agent must exist on the target host before the mission agent can be instantiated. The system owner needs to consent to supporting the agent type before the agent type can be instantiated.

A Snow agent has the capability to communicate with any entity using XMPP messages. It can ask either the Mission Controller or the Agent daemon, or both, for a list of local or remote agents. It can participate in a conference with users or other agents or even obtain presence information about local or remote agents and users.

The agents may send mission results directly to the mission requester using ordinary XMPP messages with encrypted or unencrypted contents, or to any other valid recipient.

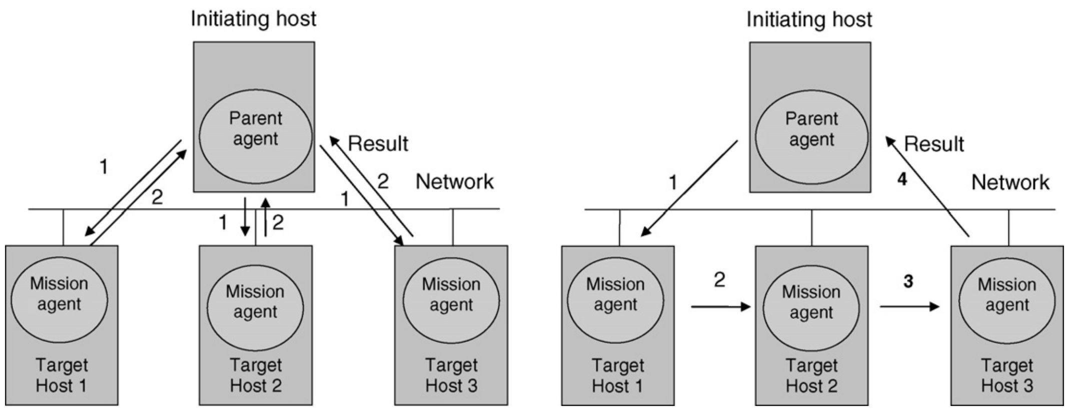

Fig. 2 – Distribution schemes: (a) single jump multiple mission agents approach (left). (b) Multiple jumps single mission agent approach.

4.5. *Agent phases and distribution schemes*

The Mission Controller is able to control two kinds of agent distribution schemes: Spread mode missions and jump mode missions, as shown in Fig. 2. In Fig. 2 an agent, labelled the "parent agent", requests a sub-mission to three hosts by sending a mission specification to the local MC (not shown).

In Fig. 2a, the mission specification specifies a spread mode mission, in Fig. 2b it specifies a jump mode mission. Fig. 3 shows the phases of the mission agent's life cycle on the hosts in a spread mode mission. In spread mode missions, the Mission Controller forwards the mission specification to the remote Agent daemons, indicated by the arrows labelled "1" in Fig. 2a. The Mission Controller negotiates with several remote Agent daemons simultaneously. This is illustrated by the negotiation phase in Fig. 3. The Agent daemons may respond with a waiting time for execution. This is illustrated by the waiting phase shown in Fig. 3. After the potential waiting phase, the mission agents are instantiated by the Agent daemon and the mission specification is transferred (through the connection to the c2s component). This stage is shown as the activation phase in Fig. 3. When the complete mission specification (including the task specification and data) is transferred, the mission agents become operational and start performing their tasks. This is illustrated by the working phase in Fig. 3. Mission agents are potentially instantiated on each host covered by the mission and run in parallel until the mission has ended. When a mission agent has finished its task, the mission result can be transferred to the mission requester as illustrated by the arrows labelled "2" in Fig. 2a.

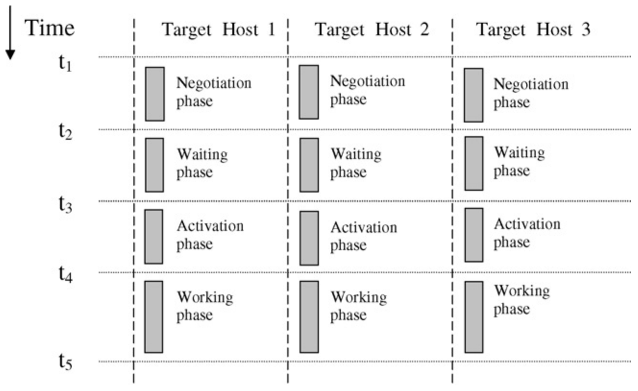

Fig. 3 – Mission Agent phases in spread mode missions.

In jump mode (Fig. 2b), the Mission Controller sends the mission specification to the Agent daemon on the first target host. This is illustrated by the arrow labelled "1" in Fig. 2b. A negotiation phase may follow while the Agent daemon and the Mission Controller try to reach an agreement on the mission specification. This is illustrated by the negotiation phase (t1–t2) in Fig. 4. After the negotiation phase, a waiting, activation and working phase may follow, as in the spread mode case. When execution on that site has finished, the mission agent asks the local Mission Controller for migration service to a new host. This is illustrated by the migration phase (t5–t6) shown in Fig. 4. The local Mission Controller negotiates the migration with the Agent daemon on a remote site by transferring and negotiating the contents of the mission specification. The main Mission Controller (on the initiating host in Fig. 2b) regains control when the agent has migrated to the new host. The first migration is illustrated by the arrow labelled "2" in Fig. 2b. This scheme continues until all hosts in the agent's itinerary have been visited as illustrated by the arrows labelled "3" and "4" in Fig. 2b.

4.6. *Snow Agent system clients*

The users of the Snow Agent system may use any Jabber client that supports sending raw XML messages or specialised plug-ins that produces the mission specifications. Jabber clients such as Exodus [59] and Jeti [60] provide a plug-in interface. Specialised Graphical User Interfaces (GUI) to the Snow Agent system applications may be implemented as Exodus or Jeti plug-ins and provide GUI for specifying missions and interacting with the Snow Agent system.

The Snow Agent system clients communicate with Mission Controllers, create missions, obtain mission results, terminate, restore and remove agent missions, and show incoming or stored mission results. Snow Agent system missions are not dependent on the Jeti or Exodus Jabber clients. Any program or Jabber client (or even telnet) capable of communicating with

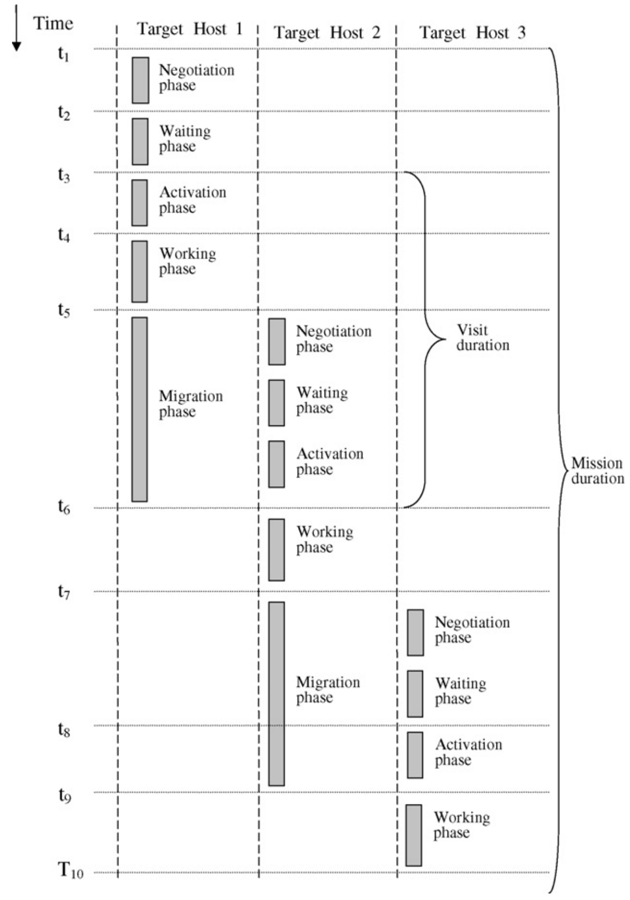

Fig. 4 – Mission Agent phases in jump mode missions.

a Jabber server may be used to deploy Snow Agent missions and receive mission results.

4.7. *Meeting disease surveillance requirements*

In this section we describe how the Snow Agent System meets the requirements stated in Section 3 above.

4.7.1. *Flexible connection methods*

The Snow agent system may use several connection or organisation models.

4.7.1.1. The peer-to-peer model. The simplest model is the peer-to-peer model where the Snow Agent servers must be constantly connected and available for mission requests. Each Snow Agent server is responsible for one GP clinic or patient clinic. Specialised Snow Agent servers, supporting agent types that coordinate large agent missions, may also be used. Communication between the servers is done using the standard "s2s" Jabber component.

4.7.1.2. The hierarchical model. In this model, one Snow Agent server serves as post office for a number of Snow Agent servers that poll the post office for messages and processing requests. A post office may serve all GPs in a county or a single municipality. If the EHR installation needs to be protected by a firewall that blocks incoming connections from the network into the EHR system, the hierarchical model is needed. The Snow Agent system uses two additional Jabber extensions to enable periodical connections to the centralised Snow Agent server (Jabber + Snow Agent system extensions). In this organisation, the Snow Agent servers have two distinct roles, the "poller" and the "post office" roles. The "post office" server provides offline storage and relay of XMPP messages on behalf of the "poller" which is the EHR installation. The Snow Agent server behind the firewall, the poller, must take the initiative to connect to the post office (outgoing connection). When the connection is established, and authentication is performed, the poller will receive all messages and processing requests stored by the post office server while the poller was offline (see Fig. 1 above.) The poller may also choose to stay constantly connected to the post office. These two components make it possible to treat a hierarchically organised network as a peer-to-peer network.

The hierarchical model enables GP clinics protected by firewalls to periodically receive computing requests (in the form of a mission specification). It also satisfies the Norwegian Data Inspectorate's requirements for how small health institutions and GP clinics can connect to a common health network [61]. These requirements state that it should be impossible to open connections from the network into the EHR system protected by the firewall.

The connection frequency of the Poller component is configurable. If an outbreak is detected, an external authority may request the component to stay permanently connected. The poller component is also capable of connecting to several post office servers, which enables participation in several organisation models or "networks" simultaneously.

4.7.2. *Automatic, independent and dynamic reporting mechanism*

The Snow Agent system is designed to be an automatic epidemiology and surveillance service. As long as healthcare workers continue to enter data into the EHR system, the system will be able to collect statistical data, given that the necessary consent for collecting the data is obtained. Contents and collection frequency are however dependent on the configuration of the participating hosts. The system can only collect information about diseases explicitly supported by the GPs. The reporting frequency depends on the connection model supported by the GP clinics. If all GP clinics are constantly connected (using the peer-to-peer or hierarchical model), then the system supports a high reporting frequency. If the GP clinics connect periodically, then the GP clinic with the lowest connection frequency decides the maximum reporting frequency if 100% coverage and up-to-date data is necessary. The Snow Agent system epidemiology service does however cache epidemiological data (mission results) locally and on the "post office" servers, which makes it possible to have a higher reporting frequency if the data "freshness" requirement can be relaxed.

4.7.3. *Asynchronous*

The computations performed by the mission agents within the Snow Agent system operate independently of its requester. The requester may however terminate or query the state of the mission agent at any time. The important issue is that the computation is independent of an online mission requester. A Snow Agent mission may run for days or weeks without the requester being available and online.

4.7.4. *Must provide adequate security mechanisms*

A Snow Agent system server capable of connecting to and extracting data from an EHR system needs to operate under a security policy. The security policy states which agent applications may be allowed to run by which user. Snow Agent applications necessarily generate network traffic and the associated communication costs. Such issues may also influence the policy for hosting Snow Agent applications.

4.8. *Meeting the hard requirements*

In terms of the hard requirements outlined above, it may be necessary to distribute new disease or syndrome definitions to be able to automatically count occurrences of *possible*, *probable* and *confirmed* cases of a new disease. Such definitions may be built using archetypes as used in the OpenEHR approach [62]. Archetypes are explicit definitions of the structure and content of clinical data and of the constraints on the data. The archetypes may be built by an epidemiologist using an archetype editor. The archetype (disease or syndrome definition) may be read by a form generator component to build a graphical user interface for the input of data according to the structure, contents and constraints in the archetype. The screen representation may also be built manually, as in the PropeR system [63], which is based on OpenEMed [16]. The data produced by the GUI may be validated by a validator component that uses the archetype definition as input, to ensure that the data stored by the input system is valid.

To count cases of a new disease such as SARS, specialised archetypes may be used to define concepts like “possible SARS”, “probable SARS” and “confirmed SARS” cases. The counting of cases may be done by evaluating stored data against these archetypes. The specialisation mechanisms in the OpenEHR Archetype Definition Language (ADL) [64] allow for revision of a disease definition and may ensure that the definition of a new disease evolves without losing the cases defined by earlier definitions of the disease.

The Snow Agent system and the OpenEHR or the PropeR approaches fit perfectly together because the Snow Agent system is able to distribute a disease or syndrome definition on a global scale. The definition, represented by explicit archetypes, can be used to: (1) generate GUI for data input, (2) validate user-supplied data, and (3) provide definitions or screening forms of concepts such as “possible disease X”, “probable disease X” and “confirmed disease X”. These definitions may be used by Snow Agents monitoring the diffusion of disease X in a geographical area.

4.9. *Status of the system*

A first version of the system has been developed at the Norwegian Centre for Telemedicine and at the Distributed System Technology Centre (DSTC) at the University of Queensland, in Brisbane, Australia. The major parts of the system will be open source and available for download and use from the project web page at http://www.telemed.no/opensource/snow/ or http://mit.cs.uit.no/snow/.

5. System test

To learn more about the strengths, weaknesses and performance of a system based on propagation of program control, we performed a series of tests and experiments to see how fast a distributed epidemiology and surveillance system can provide useful information to the system’s users. This question is interesting because we expect that a distributed system based on propagation of program control would need to have considerably slower response time than a centralised system. The reason for this is that such a system is more complex and may involve low-performance servers using low-bandwidth network connections. We also wanted to test the scalability of the Mission Controller component. This is necessary because MC is the component that handles most of the communication in the system, mainly because of its role as an intermediary between users, agents and Agent daemons.

5.1. *Test data*

The task for the epidemiology system was to plot the pertussis occurrences on an interactive map implemented in SVG [43]. The pertussis occurrences were extracted from a set of simulated EHR databases. The EHR databases were generated from microbiology laboratory data as described in Section 2.1 above. The map polygons were created from bitmaps made available in the NorgesHelsa system [42] and converted to SVG polygons. Then the map polygons were inserted into a MySQL database [44] in a convenient format for map generation and coded with geographical information. The interactive map was based on the Vienna example provided by carto.net [65].

5.2. *Design of the experiments*

In our experiments, we wanted to determine the performance of the system in a close to reality configuration of the system. In Norway, this means that we needed to use the hierarchical connection model with one post office server and several GP installations polling for processing requests. We used two mission agents, the “Main-epidemio” mission agent, which ran on the post office server, collecting and merging epidemiological data from the participating hosts, and the “EHR-epidemio” agent, which computed on our GP clinics servers. The EHR-epidemio mission agent queried the local EHR database for pertussis cases in the given time period, and reported the result to the main-epidemio agent.

In our experimental setup, a computer located in Tromsø, Norway, acted as post office server for the GP clinic Snow Agent servers. The poller components in the five GP clinic servers were configured with a permanent connection to the post office. This configuration provides the fastest system response possible. (See Section 2 above for description of the hardware and software used on the Snow Agent servers.)

5.3. *Experimental result*

In our experiments we first tested the computation time for each participating host. In this case the hosts acted as both post offices and GP clinic servers. The databases and mis-

Table 2 – Processing times for main-epidemio and EHR-epidemio missions using a single host

Server	BogoMIPS	Average processing time			S.D.		
		Main	EHR	Diff	Main	EHR	Diff
1	1061	13.66	11.53	2.13	0.58	0.58	0.001
2	4767	6.54	4.42	2.12	0.45	0.37	0.186
3	1582	9.77	7.45	2.32	0.37	0.36	0.040
4	891	14.36	12.15	2.21	0.69	0.61	0.319
5	891	14.19	12.02	2.40	0.89	0.76	0.246
6	466	23.59	21.22	2.37	0.48	0.48	0.008

Table 3 – Processing times for the Main-epidemio and EHR-epidemio missions using multiple hosts

Servers	Number of hosts	Average			S.D.		
		Main	EHR	Diff	Main	EHR	Diff
1 + 6	2	25.12	23.04	2.09	0.35	0.35	0.040
1 + 6 + 4	3	24.80	22.73	2.08	0.59	0.59	0.001
1 + 6 + 4 + 5	4	25.21	23.33	2.09	1.59	1.59	0.011
1 + 6 + 4 + 5 + 3	5	25.69	23.57	2.12	0.79	0.79	0.002
1 + 6 + 4 + 5 + 3 + 2	6	25.01	22.85	2.16	0.55	0.55	0.002

sion specification were identical for all hosts. We measured the amount of time used from receiving the mission or sub-mission request until the mission result notification message was sent by the Mission Controller. Our results are shown in Table 2. The table shows the average processing time from 10 execution runs on each target.

In Fig. 5 we have plotted the processing time for an identical mission specification on each of our target hosts (shown as points on the line), which shows that the processing time is dependent on processor speed. The time difference between the EHR agent and the Main agent (line labelled Diff) consists of processing time used to prepare the sub-mission specification and process sub-mission result (from EHR-epidemio agents) and producing the final SVG file (113 kb in this case).

In our next experiment, we wanted to see how the cost of administrating several target hosts in the sub-mission affected the total mission duration. To test this, we deployed the main-epidemio agent on our post office server. We deployed the EHR-epidemio mission to an increasing number of target hosts, starting with the slowest host first. Again, we averaged the processing time over ten consecutive runs. We used the same database and mission specification as in experiment 1. This should make the computation needs for the sub

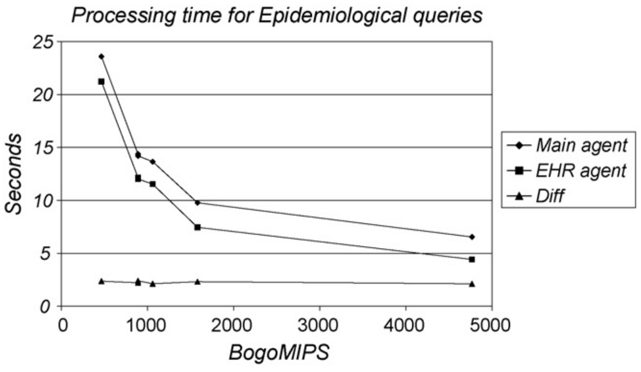

Fig. 5 – Processing time as a function of processor speed.

missions similar to experiment 1. However, the administration cost of the sub mission has increased because the mission specification was transferred over the Internet to an increasing number of participating hosts. This is likely to cause an increased delay.

In Table 3, we see that the increase of total processing time, compared to that in Table 2, stems from an increase in the EHR mission processing time measured by the Mission Controller on host 1. We believe the variations in processing time result from varying network conditions, as the participating hosts were spread around the globe. There is an increase in processing needs required by the Mission Controller to administrate the sub mission, but these processing costs are probably less than the variation in network conditions we experienced. Some evidence for this may be found in the standard deviation columns for the processing time.

The good news from the second experiment is that the total processing time seems to be minimally affected by increasing the geographical area covered by the epidemiological query. This result is according to our expectations. However, we have only tested the system on a minimal number of hosts.

The processing time reported in Table 3 shows a wait of about 25 s for getting updated information directly from the EHR systems. A 25-s wait may be a bit too long for busy GPs. With the expectation of slow response times in mind, we designed the epidemiology service with a cache. Use of the cache reduces the processing time to an average of 1.53 s (over 10 execution runs with S.D. 0.57) when interrogating the server in Tromsø, Norway from Brisbane, Australia. By relaxing the requirement for freshness of data, and scheduling periodic epidemiology missions to slow hosts, it seems to be possible to achieve an acceptable response time from the epidemiology service even if servers with 1997 technology (server 6) are used. In Fig. 6 we see that the processing times involving six hosts are shorter than those involving five hosts. This effect is explained by the varying bandwidth conditions on the Internet during the experiments, and represents a worst-case scenario as intended in the design of the experiment.

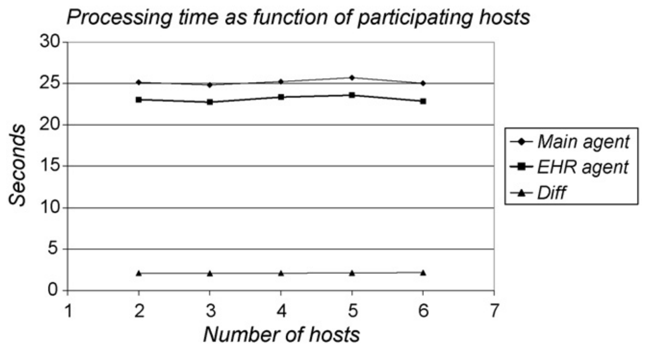

Fig. 6 – Processing times as function of the number of participating hosts.

5.4. *Scalability tests*

In Fig. 7 we see the UML sequence diagram for an agent mission involving one target EHR host. The diagram shows that the Mission Controller is the component, which handles most messages. The scalability of the system is therefore partly dependent on the ability of this component to scale when the number of target hosts in a spread mode mission grows. In the second experiment the "Service provider host" and the "Post office" was the same host, which means that a single Mission Controller handled both agent missions shown in the sequence diagram. To establish a rough estimate of the scalability of the Snow Agent server and the Mission Controller we computed the processing requirements and then performed two tests: (1) message throughput test for the Snow Agent server and, (2) Mission Controller and the Snow Agent server's ability to send outgoing messages.

In Norway, there are approximately 1800 GP clinics [66]. The distribution of GP clinics by county is shown in Table 4. A hypothetical deployment of the Snow Agent system could be configured with one Snow Agent post office server in each county, to which all GP clinics in that county connect. In this configuration, the Snow Agent post office servers need to be able to support 180–200 EHR system connections simultaneously.

Table 4 – GP clinics per county in 2003 in Norway. See [66] page 27

County	GP clinics
Akershus	170
Aust-Agder	55
Buskerud	86
Finnmark	26
Hedmark	79
Hordaland	181
Møre og Romsdal	77
Nordland	101
Nord-Trøndelag	56
Oppland	81
Oslo	179
Rogaland	148
Sogn og Fjordane	37
Sør-Trøndelag	81
Telemark	75
Troms	56
Vest-Agder	63
Vestfold	87
Østfold	133
Total	1771

The mission specification used in the above experiment is about 1 K of data for the EHR epidemiology mission. For each additional host participating in the mission, the mission specification grows by about 100 b. The output requirement for the Mission Controller therefore has an N squared growth, or O=N^2 in big-oh notation. The output requirement is shown in Fig. 8.

The input requirement, shown in Fig. 9, is linear. To achieve this we had to modify the data format returned by the EHR agent running on the EHR hosts, about 2 kb for 15 postcodes for a period of 34 days. The system internal messages shown in Fig. 7, the "OK", "Alive" and "Finished" status messages require 179, 179 and 180 b, respectively.

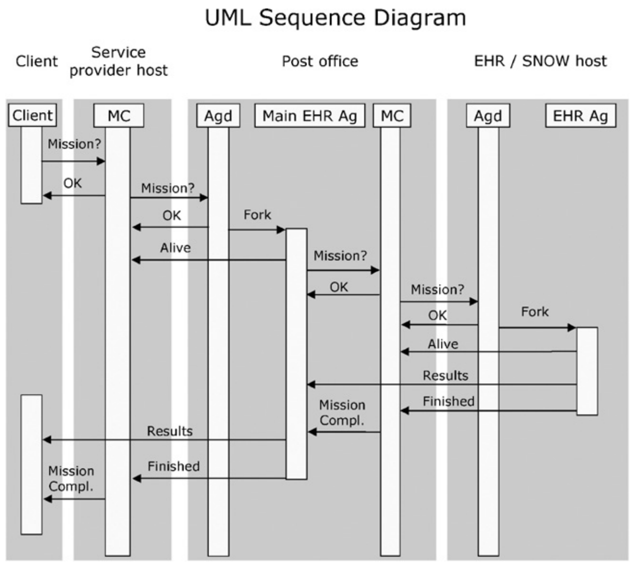

Fig. 7 – Simplified UML sequence diagram showing a spread mode mission to a single EHR host.

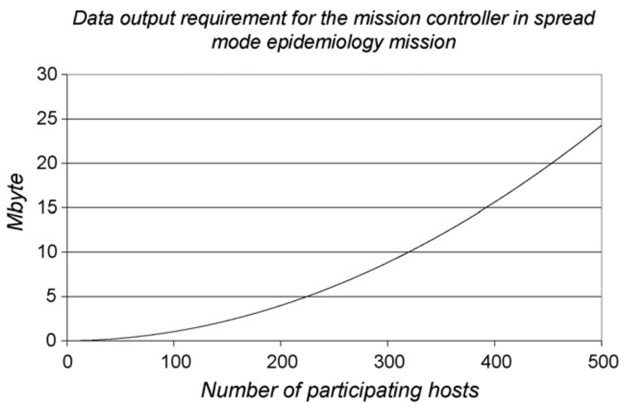

Fig. 8 – Data output requirement for spread mode epidemiology mission. Mission Controller output requirements as number of host grows.

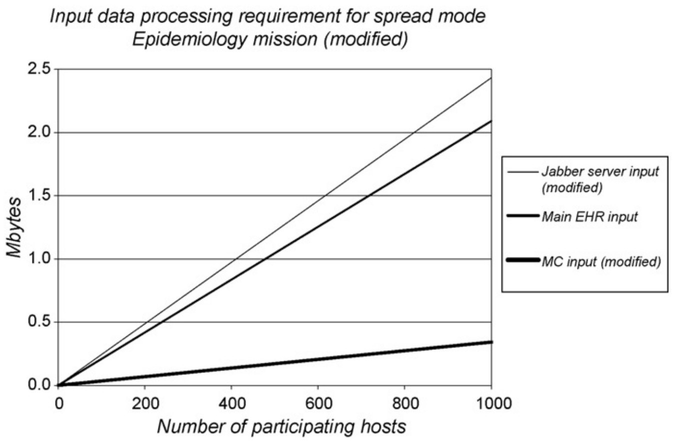

Fig. 9 – Data input requirement for spread mode epidemiology mission as number of participating EHR system hosts grows.

First, we tested the message throughput of the Snow Agent server. In this test we used server 2 (a Pentium 4, 2.4 GHz processor) and 3 (a Pentium III 800 MHz processor) connected through a 100 Mbit/s network. We measured the time taken to transmit messages (4 kb) from a client program on server 3 over the network to server 2 and back to a second client on server 3. Because server 3 had a slower processor than server 2, we only observed a 30% load of the Snow Agent server running on server 2. The throughput in the test stabilised at 0.8 Mb/s or 197 messages per second. This throughput may actually reflect the maximum performance of server 3, which both sent and received the messages.

In the second test we measured how fast the Mission Controller and the Jabber server were able to output messages. The Mission Controller on server 2 sent 1000 messages to a client on server 3. We measured the time from when the Mission Controller sent the message, until the client received the message (using synchronized clocks). Because of limitations in the test software, the maximum message size was 4 kb. The test showed that the Mission Controller can send approximately 400 messages per second. However, a 4 kb mission specification will cover 31 hosts. To cover 200 hosts, the corresponding mission specification will be 20 K, and the total output would need to be 4 MB. In the test, the first 200 messages were sent by server 2 and received on server 3 after 0.6 s. Four megabytes was transmitted and received after 1.6 s (output rate 2.5 Mb/s). How fast a real system would be able to distribute the mission specifications depends on many parameters, which makes it difficult to predict a realistic estimate for a deployed system. However, our results show that it is realistic to support 200 EHR host per post office server.

The theoretical bandwidth of a server connected to a 100 Mbit/s network is an output of 12.5 Mb/s, which corresponds to a mission with approximately 350 participating hosts. With utilisation of about 70% of the network adaptor, the server can support 295 hosts (8.6 Mb Mission Controller output) in 1 s.

For the scalability of the input, it would be beneficial to distribute the activation of the sub mission agents over time, because this will distribute the load on the network and the receiving server over time. The input is also dependent on how many cases of infected persons are identified in a single query. In a situation with many infected persons, the size of the mission result will increase towards the maximum of all postcodes covered by the GP clinic. The realistic capacity requirement of the system can therefore be estimated by counting the number of postcodes covered by the GP clinic and calculating the maximum throughput required for the system.

6. Discussion

The goal of syndromic and disease surveillance systems is to detect disease outbreaks as early as possible. However, syndromic surveillance systems has not yet proved its capacity to discover outbreaks earlier than traditional surveillance methods [24,29,38]. Also, the reliance on the ability of “astute clinicians” to detect disease outbreaks in time has been questioned [32]. In the future, syndromic surveillance may be based on data provided by human sensors, as described by Årsand et al. [67]. The use of human sensors may enable earlier detection of disease outbreaks than what is possible today.

A study of healthcare seeking behaviour in New York showed that in that healthcare context, only 8.8% of patients with ILI symptoms attend an emergency department. About 50% of the patients consulted a physician (GP) [6]. Because most patients in Norway see a GP as their first point of contact with the healthcare system, we should provide the general practitioners with the necessary tools to discover local disease outbreaks. A syndromic surveillance system should therefore include data from the EHR system used by doctors to increase the coverage of the system. In Norway this implies using EHR data from GPs for surveillance purpose. This is feasible, because Norway has close to 100% coverage of EHR systems among GPs. Most GP surgeries are now becoming connected to the national health network.

The current disease surveillance system in Norway (MSIS) [40] is based on reporting both from laboratories and from GPs. The GPs report the cases on pre-printed paper-based forms provided by the laboratory that performed the disease-confirming lab test. In addition to the mandatory reporting undertaken by all GPs, approximately 200 GP clinics report weekly, during the winter, about the occurrence of a selection of diseases (including ILI). Achieving good coverage and high accuracy has been a problem in Norway as in other countries [39,68–71] and is also limited by the question of cost. Previous attempts to report the number of cases of selected diseases on pre-printed postcards proved too expensive to be sustainable. The Norwegian Institute of Public Health produces a weekly report (the MSIS report) containing the number of occurrences at a national and county level. The reporting frequency has been too slow and the geographic level too coarse-grained to be of any practical use in triggering general practitioners’ awareness of disease outbreaks. However, the Norwegian Institute of Public Health is changing its reporting system to a web-based interface that supports a higher level of geographical granularity.

The inclusion of data from a large number of EHR systems used by GPs raises the need to rethink what technology might be appropriate for this purpose. Current middleware approaches seem to be focused on the idea of a centralised, highly available and powerful server (or pool of servers), which utilises the request reply call semantics for distributing the computations between clients and servers or tiers. In contrast, our need is to have distributed autonomous computations on many EHR system servers, with potentially periodic connectivity and narrow bandwidth that cannot support the request reply semantics for security reasons.

From our point of view, a peer-to-peer system among GPs within the local communities seems to be the appropriate model on which to base a tool for discovering local disease outbreaks. Such a system could also avoid the privacy issues associated with the use of EHR data. However, building a distributed and decentralised system for syndromic or disease surveillance raise new challenges not present in centralised systems such as RODS [12]. Distributed and decentralised surveillance systems need to address scalability, system maintenance and the need for two-way communication, as outlined in the requirements section above.

We have described the requirements for such a system, and found that an ideal surveillance system is one that: (1) enables epidemiologists to define the disease to be monitored, including codes, symptom lists and constraints on biometrical measurements, (2) enables distribution of a disease definition to all systems participating in disease surveillance, (3) enables clinicians to start reporting cases in their normal working tool, the EHR system, immediately after receiving the disease definition, (4) immediately enables collection of statistical data after dissemination of a disease definition, and, (5) supports revision of the disease definition as more knowledge becomes available about the disease. This procedure needs to be achievable in a minimum of time. The human factor should be the main time consumer. The surveillance system needs to support flexible connection methods, enabling automatic, independent and dynamic reporting mechanisms. It must support asynchronous (and autonomous) computations and must provide adequate security mechanisms.

Syndromic and disease surveillance systems traditionally report events. An event is typically a positive laboratory test that confirms a diagnosis. In our system, it seems more natural to report the number of cases affected by the disease or syndrome classification used. This is natural because the EHR system is constantly updated with new electronic lab reports. It also provides the opportunity to review the development of cases retrospectively, because historical data is also available in the EHR system.

The Real-time Outbreak and Disease Surveillance system (RODS) [12] uses a centralised approach to perform syndromic surveillance. An interesting feature of RODS is the use of a Health System Resident Component (HSRC) that runs within the health institutions' firewall. This component has much more data available because it can operate on patient-identifiable clinical data. HSRC is able to link and use laboratory and radiology data in its inference and is able to achieve much higher specificity of patient categorization through access to more information. These benefits also apply to our system.

The SARS outbreak showed that a disease outbreak can easily cross borders. This represents a problem when an outbreak occurs on the border between two disease surveillance systems. The use of a common disease surveillance system can help to overcome such problems. The decentralised approach used in the Snow Agent system has the potential to solve the problem of cross-border disease surveillance because it removes the need to export patient related information to a centralised storage for processing.

From our survey of available literature, we have not discovered any systems that are able to update the coding system or symptom list as a feature of the disease surveillance system. However, the centralised inference and detection system could easily be updated to use new diagnosis codes or syndrome definitions. In distributed and decentralised surveillance systems, this issue can be solved by enabling two-way communication, passing disease or syndrome definitions in one direction and data about the distribution of cases in the other direction.

While RODS is a monitoring system that feeds data from many sources into a centralised inference system, the Snow Agent system is more a "peer-to-peer" approach aimed at helping clinicians to discover ongoing outbreaks in the local community and aiding the diagnosis process on a daily basis. We believe such use of the system will have a positive effect on data quality in primary care. The Snow Agent system approach has the advantage that it is capable of accessing high quality data that precedes disease outbreak detection in a laboratory by many hours. The time advantage may sometimes be as much as a day, depending on geography and sample delivery routines. If the system counts the occurrences of prodromes or even just the rate of lab test requests, it can raise the awareness of local GPs about an outbreak long before the laboratory sees enough cases to issue an alarm. The Snow Agent system also has the advantage that it can count cases that are diagnosed using epidemiological techniques, for instance in situations where a whole family is affected. Such cases may never be confirmed by sending a sample to a laboratory. These two features are unique advantages using the peer-to-peer approach between GPs.

Our performance test shows that the system will scale to national level in Norway. The test shows that a normal desktop computer running the Linux operating system can distribute a computing request to all GP clinics within any Norwegian county within a reasonable time. The results from the epidemiology processes running on all EHR system installations can be merged, and may be depicted on an interactive map by an end user anywhere, using a normal web browser with a SVG viewer component installed. If caching of computing results is performed, the response time may be reduced dramatically. By scheduling epidemiological queries at regular intervals, it is possible to achieve an acceptable level of responsiveness by the system. The asynchronous nature of the system, together with the autonomy of the mobile agents, makes it possible to compensate for the use of old servers with narrow bandwidth and periodic connectivity.

We believe that the system can scale further, by adding more levels to the hierarchical organisation implemented by the system. The Mission Controller component is the critical component regarding the scalability of the system. The Jabber server supports several server configurations that enable the server to handle many server connections. This feature makes it possible to scale up the processing capacity of a Snow Agent server dramatically.

The Snow Agent system's epidemiology service is not yet sufficiently developed to qualify as an ideal surveillance system. However, it has some benefits over existing approaches. Most important, it avoids the privacy problems associated with using EHR data. The two-way communication supported by the Snow Agent system's epidemiology service makes it possible to update the computer systems that feed the disease surveillance system with new kinds of data. The system is designed for everyday use by general practitioners who need epidemiological information in their daily work, which is beneficial for surveillance systems [3]. The system may also be used as a feeder system to a centralised outbreak detection system such as RODS.

7. Conclusions

In this paper we have presented the requirements that we propose distributed surveillance and epidemiology systems must meet when facing a local, national or a pandemic outbreak of a communicable disease. The surveillance systems should make use of the EHR systems used by GP clinics, because most patients consult their GP as the first point of contact with the health service when they get ill. From our point of view, such systems should be distributed and decentralised to provide doctors with an appropriate tool to discover local disease outbreaks, and to avoid the associated privacy issues. Such a system must support two-way communication, to enable the participating EHR systems to incorporate new disease or syndrome definitions.

The Snow Agent system makes it possible to collect epidemiological data directly from the EHR system in a geographical area based on a task specification. The core principle utilised to achieve this is propagation of program control, which lets computing processes or programs migrate between hosts in the same manner as a mobile-code-based system. In our approach, we do not move code, because of the security and authorisation problems inherent in mobile-code-based systems. Instead, we propagate program control by moving a task specification and data between the hosts.

We have tested our approach in realistic settings to learn about the efficiency and scalability of a system based on the propagation of program control principle. In our test, we placed servers around the globe. The results of our test show little or no performance penalty associated with covering a larger geographical area in an epidemiology query. The response time of the slowest server limits the response time of the epidemiological service. By applying a cache, we reduced the response time of the system from about 25–1.5 s, to produce an interactive map containing the epidemiological data. Our scalability tests indicate that the system can easily scale to national level in Norway.

Acknowledgements

This work has been funded by the Norwegian Research Council, Grant no. 142162/320 and by the Norwegian Centre for Telemedicine. Many individuals have contributed to its realisation, and their help is very much appreciated. In particular, we would like to thank Gudleif Arnosen at the Microbiology Laboratory at the University Hospital of North Norway, for providing lab data for our system; Ivar Ness for donating time, bandwidth and processing cycles on his server in Spain to the project; Andreas Neumann for letting us use the JavaScript source code from the Vienna SVG map example; and DSTC, in particular Dr. Hoylen Sue and the Titanium project members, for all their help and technological insight which they have so willingly shared. We are also very grateful for the help from Hilde Pettersen and Eva Henriksen with proofreading of this manuscript.

What was known before the study:

The utility of syndromic surveillance systems, compared to traditional surveillance systems, has so far not been demonstrated [24,29,38].

Syndromic surveillance systems utilise many different information sources and data from emergency departments is an important one. A telephone survey from New York showed that about 50% of patients with influenza like illness consult a physician, about 9% visit an emergency department and about 4% called a nurse or a health hotline [6]. However, very few of these systems utilise data from general practitioners, GPSURV from Auckland, New Zealand is one exception [72].

Syndromic and disease surveillance systems should complement, rather than replace the “astute clinician” [26,30].

Protection of patient privacy is an important issue to address for disease surveillance systems [7–11].

What this study contributes:

The propagation of program control principle provides a platform for mobile autonomous agent solutions that can be applied in the healthcare environment. This platform have been used to build an epidemiology service that may be expanded to include a distributed syndromic and/or disease surveillance system.

Distributed syndromic and/or disease surveillance systems can be implemented without being dependent of transferring patient identifiable data which eliminates the privacy issue.

A surveillance system built using the propagation of program control principle can easily scale to National level in Norway and is able to provide up to date data from the participating source systems within 25 s, even if server technology from 1997 is used.

Distributed syndromic and/or disease surveillance systems should provide two-way communication to enable distribution and update of syndrome and disease definitions for use by the EHR systems contributing data.

REFERENCES

[1] H. Liang, Y. Xue, Investigating public health emergency response information system initiatives in China, Int. J. Med. Inform. 73 (9–10) (2004) 675–685.

[2] D.W. Forslund, E.L. Joyce, T. Burr, R. Picard, D. Wokoun, E. Umland, et al., Setting standards for improved syndromic surveillance, IEEE Eng. Med. Biol. Mag. 23 (1) (2004) 65–70.

[3] S.L. Foldy, Linking better surveillance to better outcomes, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 12–17.

[4] F.C. Tsui, M.M. Wagner, V. Dato, C.C. Chang, Value of ICD-9 coded chief complaints for detection of epidemics, Proc. AMIA Symp. (2001) 711–715.

[5] W.B. Lober, B.T. Karras, M.M. Wagner, J.M. Overhage, A.J. Davidson, H. Fraser, et al., Roundtable on bioterrorism detection: information system-based surveillance, J. Am. Med. Inform. Assoc. 9 (2) (2002) 105–115.

[6] K.B. Metzger, A. Hajat, M. Crawford, F. Mostashari, How many illnesses does one emergency department visit represent? Using a population-based telephone survey to estimate the syndromic multiplier, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 106–111.

[7] J.W. Buehler, Review of the 2003 National Syndromic Surveillance Conference—lessons learned and questions to be answered, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 18–22.

[8] D. Drociuk, J. Gibson, J. Hodge Jr., Health information privacy and syndromic surveillance systems, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 221–225.

[9] W.B. Lober, L. Trigg, B. Karras, Information system architectures for syndromic surveillance, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 203–208.

[10] J.W. Loonsk, BioSense—a national initiative for early detection and quantification of public health emergencies, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 53–55.

[11] J.U. Espino, M. Wagner, C. Szczepaniak, F.C. Tsui, H. Su, R. Olszewski, et al., Removing a barrier to computer-based outbreak and disease surveillance—the RODS Open Source Project, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 32–39.

[12] F.C. Tsui, J.U. Espino, V.M. Dato, P.H. Gesteland, J. Hutman, M.M. Wagner, Technical description of RODS: a real-time public health surveillance system, J. Am. Med. Inform. Assoc. 10 (5) (2003) 399–408.

[13] S.L. Foldy, E. Barthell, J. Silva, P. Biedrzycki, D. Howe, M. Erme, et al., SARS Surveillance Project—Internet-enabled multiregion surveillance for rapidly emerging disease, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 215–220.

[14] L.G. Kun, D.A. Bray, Information infrastructure tools for bioterrorism preparedness. Building dual- or multiple-use infrastructures is the task at hand for state and local health departments, IEEE Eng. Med. Biol. Mag. 21 (5) (2002) 69–85.

[15] D. Forslund, E. Umland, J.C. Brillman, E. Joyce, P. Froman, T. Burr, et al., Results from the fielding of the Bio-surveillance Analysis, Feedback, Evaluation and Response (B-SAFER) system in Albuquerque, New Mexico, AMIA Annu. Symp. Proc. (2003) 842.

[16] The OpenEMed Project. URL: http://openemed.org/ Accessed 11.04.2005.

[17] D.G. Kilman, D.W. Forslund, An International Collaboratory based on virtual patient records, Commun. ACM 40 (8) (1997) 111–117.

[18] Object Management Group. Clinical Observations Access Service, 2001, URL: http://www.omg.org/technology/documents/formal/clinical_observation_access_service.htm, Accessed 4.10.2005.

[19] Object Management Group. Person Identification Service, 2001, URL: http://www.omg.org/technology/documents/formal/person_identification_service.htm, Accessed 4.10.2005.

[20] M. Turoff, On site: Past and future emergency response information systems, Commun. ACM 45 (4) (2002) 29–32.

[21] J.G. Bellika, Prerequisites for developing and deploying a system for ubiquitous access to patient information. In: TEHRE 2001 m-Health Conference, 1st Annual Conference on Mobile and Wireless Healthcare Applications, 11–14 November 2001; London, UK, 2001.

[22] K.J. Henning, What is syndromic surveillance? MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 5–11.

[23] W.K. Yih, B. Caldwell, R. Harmon, K. Kleinman, R. Lazarus, A. Nelson, et al., National Bioterrorism Syndromic Surveillance Demonstration Program, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 43–49.

[24] A. Reingold, If syndromic surveillance is the answer, what is the question? Biosecur. Bioterror. 1 (2) (2003) 77–81.

[25] D.L. Buckeridge, J. Graham, M.J. O'Connor, M.K. Choy, S.W. Tu, M.A. Musen, Knowledge-based bioterrorism surveillance, Proc. AMIA Symp. (2002) 76–80.

[26] C.V. Broome, R.W. Pinner, D.M. Sosin, T.A. Treadwell, On the threshold, Am. J. Prev. Med. 23 (3) (2002) 229–230.

[27] M.L. Popovich, J.M. Henderson, J. Stinn, Information technology in the age of emergency public health response. The framework for an integrated disease surveillance system for rapid detection, tracking, and managing of public health threats, IEEE Eng. Med. Biol. Mag. 21 (5) (2002) 48–55.

[28] A.F. Kaufmann, M.I. Meltzer, G.P. Schmid, The economic impact of a bioterrorist attack: are prevention and postattack intervention programs justifiable? Emerg. Infect. Dis. 3 (2) (1997) 83–94.

[29] J.W. Buehler, R.L. Berkelman, D.M. Hartley, C.J. Peters, Syndromic surveillance and bioterrorism-related epidemics, Emerg. Infect. Dis. 9 (10) (2003) 1197–1204.

[30] M.D. Lewis, J.A. Pavlin, J.L. Mansfield, S. O'Brien, L.G. Boomsma, Y. Elbert, et al., Disease outbreak detection system using syndromic data in the greater Washington DC area, Am. J. Prev. Med. 23 (3) (2002) 180–186.

[31] R. Lazarus, K.P. Kleinman, I. Dashevsky, A. DeMaria, R. Platt, Using automated medical records for rapid identification of illness syndromes (syndromic surveillance): the example of lower respiratory infection, BMC Public Health 1 (1) (2001) 9.

[32] Z.F. Dembek, D.G. Cochrane, J.A. Pavlin, Syndromic surveillance, Emerg. Infect. Dis. 10 (7) (2004) 1333–1334.

[33] R. Heffernan, F. Mostashari, D. Das, M. Besculides, C. Rodriguez, J. Greenko, et al., New York City syndromic surveillance systems, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 23–27.

[34] M. Paladini, Daily Emergency Department Surveillance System—Bergen County, New Jersey, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 47–49.

[35] Z.F. Dembek, K. Carley, A. Siniscalchi, J. Hadler, Hospital admissions syndromic surveillance—Connecticut, September 2000–November 2003, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 50–52.

[36] C.M. Yuan, S. Love, M. Wilson, Syndromic surveillance at hospital emergency departments—Southeastern Virginia, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 56–58.

[37] J.M. Teich, M.M. Wagner, C.F. Mackenzie, K.O. Schafer, The informatics response in disaster, terrorism, and war, J. Am. Med. Inform. Assoc. 9 (2) (2002) 97–104.

[38] S. Balter, D. Weiss, H. Hanson, V. Reddy, D. Das, R. Heffernan, Three years of emergency department gastrointestinal syndromic surveillance in New York City: what have we found? MMWR Morb. Mortal Wkly. Rep. 54 (Suppl) (2005) 175–180.

[39] T.J. Terry Jr., A system for electronic disease reporting and management. Determining the extent/spread of problems and minimizing consequences through rapid reporting and dissemination of critical information, IEEE Eng. Med. Biol. Mag. 21 (5) (2002) 86–99.

[40] The Norwegian Institute of Public Health. Surveillance of Communicable Diseases In Norway 1999, 2000, URL: http://www.fhi.no/dav/B2C6E86EA18D4B169E8163997A07A2D2.pdf, Accessed 2004-09-27.

[41] J.G. Bellika, G. Hartvigsen, R.A. Widding, The Virtual Library Secretary—A user model based software agent, Personal Technol. 2 (3) (1998) 162–187.

[42] The Norwegian institute of Public Health. Norhealth. Program, 2004, URL: http://www.norgeshelsa.no/norgeshelsa/index.jsp?language=en, Accessed 2004-09-27.

[43] J. Ferraiolo, F. Jun, D. Jackson, Scalable Vector Graphics (SVG) 1.1 Specification W3C Recommendation. January 2003. URL: http://www.w3.org/TR/SVG/, Accessed 2004-09-27.

[44] MySQL AB. MySQL. URL: http://www.mysql.com/, Accessed 2005.04.14.

[45] L. Torvalds, BogoMips, Program.

[46] Wv. Dorst, BogoMips mini-Howto. February 2000. URL: http://www.apeldoorn.hccnet.nl/howto/mini/BogoMips.html, Accessed 2004-09-27.

[47] L. Bird, A. Goodchild, S. Heard, Importing Clinical Data into Electronic Health Records: Lessons Learnt from the First Australian GEHR Trials. In: HIC2002 Health Informatics Conference, 4–6 August 2002, Melbourne, 2002.

[48] I. Kvalstad, SEDA: Sentrale data fra allmennlegetjenesten - Teknisk dokumentasjon. Statistics Norway; 2003. URL: http://www.ssb.no/emner/03/02/20/notat_200316/notat_200316.pdf, Accessed: 2004-09-23 (in Norwegian).

[49] A. Sundvoll, I. Kvalstad, SEDA, Sentrale data fra allmennlegetjenesten. Sluttrapport fra pilotprosjektet. Statistics Norway. Report No.: Rapport 2002/13, 2002, Accessed 2004-09-23 (in Norwegian).

[50] D. Johansen, Rv. Renesse, FB. Schneider, An Introduction to the TACOMA Distributed System Version 1.0, University of Tromsø, Report No.: Technical Report 95–23, 1995.

[51] J.E. White, Mobile Agents, in: J. Bradshaw (Ed.), Software Agents, The MIT Press, 1996.

[52] A.D. Birrell, B.J. Nelson, Implementing remote procedure calls, in: Proceedings of the ACM Symposium on Operating System Principles, Bretton Woods, NH: Association for Computing Machinery, 1983, p. 3.

[53] N.M. Karnik, A.R. Tripathi, Security in the Ajanta mobile agent system, Minneapolis, MN 55455, U.S.A, University of Minnesota May 1999, Report No., Technical Report TR-5-99, 1999.

[54] J. Baumann, F. Hohl, K. Rothermel, M. Strasser, Mole Concepts of a Mobile Agent System, University of Stuttgart August 1997, Report No, Technical Report 1997/15, 1997.

[55] D. Chess, C. Harrison, A. Kershenbaum, Mobile Agents: Are They a Good Idea? IBM Research Report. T.J.Watson Research Center, Yorktown Heights, NY 10598, 1995 Accessed 2004/09/26.

[56] Jabber Software Foundation. Jabber Technology Overview, 2005, URL: http://www.jabber.org/about/techover.shtml, Accessed 2005-04-11.

[57] P. Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 3921, 2004, URL: http://www.ietf.org/rfc/rfc3921.txt, Accessed 2004-10-11.

[58] P. Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920, 2004, URL: http://www.ietf.org/rfc/rfc3920.txt, Accessed 2004-10-11.

[59] P. Millard, eXodus - escape from proprietary IM systen, 2002, URL: http://exodus.jabberstudio.org/, Accessed 2004/09/26.

[60] de Boer E. Jeti, To Jabber in Java, 2004, URL: http://jeti.jabberstudio.org/, Accessed 2004/09/26.

[61] The Data Inspectorate. Veiledning i informasjonssikkerhet for kommuner og fylker, 2005, URL: http://www.datatilsynet.no/upload/Dokumenter/infosik/veiledere/tv202_2005_1.pdf, Accessed: 20/04/2005. (in Norwegian).

[62] T. Beale, A. Goodchild, S. Heard, EHR Design Principles, 2002, Revision: 2.2.URL: http://titanium.dstc.edu.au/papers/ehr_design_principles.pdf, Accepted 2004/09/27.

[63] H. Van der Linden, J. Grimson, H. Tange, J. Talmon, A. Hasman, Archetypes: The PropeR Way, Medinfo 2004 (2004) 1110–1114.

[64] T. Beale, S. Heard, Archetype Definition Language, 2004, Revision: 1.2 DraftF.URL: http://www.openehr.org/drafts/ADL-1_2_draftF.pdf, Accessed 2004/11/01.

[65] A.M. Winter, A. Neumann, Carto:net, 2004, URL: http://www.carto.net, Accessed 2004-09-27.

[66] Noklus. Årsrapport 2003 NOKLUS, 2003, URL: http://www.uib.no/isf/noklus/rapport/2003/fil1.pdf, Accessed 2004-10-11 (in Norwegian).

[67] E. Arsand, O.A. Walseth, N. Andersson, R. Fernando, O. Granberg, J.G. Bellika, et al., Using blood glucose data as an indicator for epidemic disease outbreaks, Stud. Health Technol. Inform. 116 (2005) 217–222.

[68] A. Jansson, M. Arneborn, K. Skarlund, K. Ekdahl, Timeliness of case reporting in the Swedish statutory surveillance of communicable diseases 1998–2002, Scand. J. Infect. Dis. 36 (11–12) (2004) 865–872.

[69] C.J. Allen, M.J. Ferson, Notification of infectious diseases by general practitioners: a quantitative and qualitative study, Med. J. Aust. 172 (7) (2000) 325–328.

[70] J. Dinis, Mandatory notification of communicable diseases: what physicians think, Acta Med. Port. 13 (12) (2000) 33–38.

[71] S.S. Abdool Karim, A. Dilraj, Reasons for under-reporting of notifiable conditions, S. Afr. Med. J. 86 (7) (1996) 834–836.

[72] N.F. Jones, R. Marshall, Evaluation of an electronic general-practitioner-based syndromic surveillance system—Auckland, New Zealand, 2000–2001, MMWR Morb. Mortal Wkly. Rep. 53 (Suppl) (2004) 173–178.