ITcon
www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

AN ARCHITECTURE FOR DECISION SUPPORT IN AD HOC SENSOR NETWORKS

PUBLISHED: June 2009 at http://www.itcon.org/2009/21

EDITORS: Kazi A S, Aouad G, Baldwin A

William J. O'Brien, Assistant Professor,
Department of Civil, Architectural, and Environmental Engineering, The University of Texas at Austin
wjob@mail.utexas.edu

Christine Julien, Assistant Professor,
Department of Electrical and Computer Engineering, The University of Texas at Austin
c.julien@mail.utexas.edu

Sanem Kabadayi, Assistant Professor,
Department of Computer Engineering, Istanbul Technical University
kabadayi@itu.edu.tr

Xiaowei Luo, PhD Candidate,
Department of Civil, Architectural, and Environmental Engineering, The University of Texas at Austin
xiaowei @mail.utexas.edu

Joachim Hammer, Associate Professor,
Department of Computer and Information Science and Engineering, University of Florida
jhammer@cise.ufl.edu

SUMMARY: *The intelligent jobsite is becoming a reality as applications using sensors and mobile computing devices are being developed and deployed commercially. This creates an opportunity for workers to access data in an ad hoc manner as they move through a site. However, retasking and reuse of sensors in a dynamic setting presents significant challenges including ad hoc identification of sensors in a local environment, generalization of data, and the use of dynamic information for decision support. To achieve a generalized approach to using local data, we describe an architecture that abstracts functionality into three tiers: a layer for physical communication among devices; a layer for data processing and abstraction; and a top layer for decision support. At each tier, the level of abstraction increases, allowing for development of decision support applications at the top level that are not directly tied to specific devices. The industry vision for the intelligent jobsite is driving considerable research and development in mobile technologies. However, the bulk of this development is application specific and directly ties hardware (e.g., RFID tags, sensors) to data processing applications. We seek to decouple sensors from computing hardware and application development by providing a flexible and robust middleware. Realization of the middleware will enable more flexible reuse of data to make it available to a range of decision support applications, while at the same time speeding development of such applications. This paper describes an architecture informed by a working first generation prototype. Details of the prototype, lessons learned, and specific advancements are detailed. Future commercial implementation of the architecture will make construction-specific visions for ubiquitous computing possible by enabling flexible and robust discovery and use of data in an ad hoc manner.*

KEYWORDS: *mobile ad hoc network, ubiquitous computing, intelligent jobsite, decision support, site safety.*

REFERENCE: *O'Brien W, Julien C, Kabadayi S, Luo X, Hammer J (2009) An architecture for decision support in ad hoc sensor networks, Journal of Information Technology in Construction (ITcon), Vol. 14, Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning, pg. 309-327, http://www.itcon.org/2009/21*

1. INTRODUCTION

Advances in microelectromechanical systems (MEMS) are enabling the development of small, inexpensive, wireless sensor nodes. The construction industry is just one of many areas that are increasingly employing sensor networks to support everyday applications. These sensor network deployments enable opportunities for process reengineering, performance improvement, and improved decision support for the construction professionals on a jobsite. Specifically, the construction jobsite has witnessed the introduction of a variety of advanced computing technologies: 3-D laser scanning for collection of as-built data on the site (Trupp et al. 2004), RFID for construction resource management (Jaselskis and El-Misalami, 2003, Song et al., 2007, Song et al, 2005), wireless sensors for monitoring concrete curing (Lee et al., 2006), and a wireless mesh network connected to the Internet and used for real-time data exchange among construction resources (Nuntasunti and Bernold, 2006). These and similar technologies can provide construction site supervisors a continuous awareness of the activities and resources on the site; in combination, they enable the *intelligent jobsite* documented by FIATECH's capital projects technology roadmap (FIATECH, 2003). In this paper, we present a distributed software architecture enabling decision support in dynamic ad hoc sensor networks, which enables rapid and robust implementation of an intelligent jobsite.

The organization of the remainder of this paper is as follows. In Section 2, we present the motivation behind the development of our architecture and, in doing so, review related work in the application of wireless computing technologies to construction jobsites, decision support in construction applications, component-based software development, and data routing in sensor networks. In Section 3, we summarize the requirements in enabling expressive decision support in construction applications and introduce a three-tiered software architecture that meets these requirements. Section 4 details the architecture's implementation, based on the two decision support applications for safety management on construction sites were developed; these applications are described in Section 5. Based on the key issues we identified during the development of these applications, Section 6 provides a generalization of the open, related research challenges and issues, mainly as they relate to component-based application development, the processing of imperfect information, processing queries over dynamic data streams, and communication issues in sensor networks. Finally, Section 7 concludes with three future research directions.

2. BACKGROUND

In this section, we first discuss existing applications of wireless networks to construction jobsites. We then describe the concept of the intelligent jobsite in more detail, including elucidating requirements of applications deployed to such dynamic and unpredictable environments. Finally, we conclude this section by covering existing literature as it relates to our architecture for decision support in intelligent construction sites.

2.1 Motivation

Preliminary studies of the opportunities and benefits related to introducing wireless communication and computing technologies to construction sites have shown that these innovative technologies can facilitate information flow and therefore improve performance on the jobsite (in terms of cost, schedule, and quality) (de la Garza and Howitt, 1998, Fuller et al., 2002). Encouraged by these results, efforts have more recently focused on a variety of specific applications: integration of wireless and speech technologies for on-site data collection (Tsai et al., 2007); tablet PC based applications for safety management (Sunkara, 2005) and for integrated design and construction (Elvin, 2004); and bar codes, RFID, and GPS for resource management on the jobsite (Caldas et al., 2006, Goodrum et al., 2006, Jaselskis and El-Misalami, 2003, Song et al., 2007, Song et al., 2005, Stone et al., 2000, Tserng and Dzeng, 2005). Recently, some approaches have attempted to integrate these new technologies into a single system and model for jobsite management (Köseoğlu, 2004, Riaz et al., 2006, Sacks et al., 2005). Some related efforts for first response using project knowledge have shown the potential for dynamic decision support in ad-hoc environments (Chen et al., 2007). Collectively, this research has demonstrated the potential for systematic deployment of intelligent technologies to the job-site.

There are numerous opportunities for ad hoc (re-)use of dynamic data within an intelligent jobsite. Fig. 1 illustrates a future intelligent jobsite that integrates a range of mobile and sensor devices with wireless technologies to provide a variety of safety and resource management applications. Different sensors, including RFID, hazardous materials sensors, weather sensors, etc., are deployed around the site to monitor the environment, collecting relevant data and transmitting it to the mobile devices and/or to a centralized computer for real-time or historical data analysis and decision support. For example, in the top of fig. 1, a pallet of bricks or other materials can be tagged with RFID sensors. If and when the materials enter or leave the jobsite, RFID readers positioned at the gates can read the data from the RFID sensors, generating data that can be shared among mobile and centralized devices for a variety of materials supply management, cost management, and planning applications. A worker with a mobile RFID reader and a GPS device can also walk the jobsite, detecting available RFID signals, which may aid in locating displaced or needed materials around the site. With respect to safety applications, fig. 1 also depicts sensors attached to a crane. A sensor on the hook of the tower crane can collect data regarding the load and location of the hook. This data can be aggregated and shared to monitor the crane's safe working range. Data collected by a GPS unit embedded in a worker's mobile device can be compared to the data from the crane to ensure that the worker's movements are safe with respect to the movement and load of the crane. Another example for safety management is shown in the center of fig. 1; a worker's mobile device can interact with hazardous materials sensors to monitor air quality on the site and alert the worker of dangerous conditions.

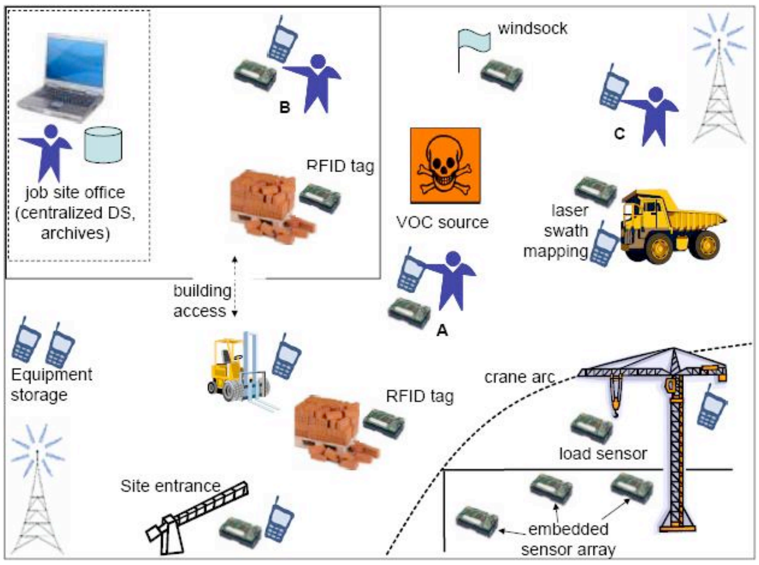


FIG. 1: An Intelligent Jobsite

This example of an intelligent jobsite demonstrates the potential of broader use of existing sensing and communication technologies in the construction industry. Currently, a range of wireless computing and sensing devices (e.g., wearable computers, RFID tags, various sensors) are being adopted to implement specific decision support applications within the broader vision of the intelligent jobsite. However, these existing applications are independently designed for specific tasks, such as tracking materials on the site using RFID. Such a stove-piped approach to application development requires one to independently design new applications for new functional requirements on the site. However, the complexity and cost of development can be significantly reduced if we leverage the commonalities across applications. To increase the reusability of highly coupled communication-based software components in the independently developed applications requires a standard to guide the design of these decision support applications. More specifically, an abstract and generic architecture for decision support applications is needed to facilitate their development.

Our overarching goal is to create a generic software architecture for decision support in wireless mobile sensor networks. We state three driving requirements for this architecture. The architecture should:

- be abstract and not domain-specific, this allows the architecture to be extended to domains similar to but other than the construction industry,
- consider the effects of wireless mobile sensor network characteristics on the decision support application's development and address these issues directly in the abstract architecture, and
- support data components and functional components; this reduces the complexity and cost associated with the development of decision support applications.

2.2 Related Work

In addition to our review of the application of wireless sensor network technologies to construction sites, we divide our review of the literature into three components. First, we look at the state-of-the-art in decision support applications. Then, we give an overview of the notion of component-based software development. Finally, we detail efforts related to data routing and aggregation in wireless sensor networks.

2.2.1 Decision Support

Four powerful tools exist for decision support applications: data warehouses, online analytical processing (OLAP), data mining, and web-based decision support. Conventional relational databases achieved success in processing data stored in the database to directly support decision making for the database's users. However, in today's dynamic computing worlds, decision makers require more precise and complex decision support that can be based on a large amount of historical data. Analysis-oriented data warehousing was introduced to meet these requirements (Inmon, 2005). OLAP has emerged as a technology to enable decision makers to access and analyze the accumulated historical data in a fast, consistent, and interactive manner (Codd et al., 1993). Data mining provides an even more powerful set of tools that enable complex decision making (Tan et al, 2006). These approaches include: statistical analysis, Bayesian belief networks, and machine learning. As the Internet has emerged as a driving computing technology, web-based decision support systems have also become a popular research area (Shim et al., 2002). Many of the decision support tools and techniques focus on complex decision making problems based on a large amount of information and require a nontrivial running time.

2.2.2 Component-based Software Development

In software engineering, a component is an independent piece of a program that undertakes a specific function in the software system. Component-based software engineering develops software by first decomposing the functionality into these pieces and then assembling them to form a complete software system. In this way, the software system becomes more loosely coupled, and the components can be replaced and reused by other systems with only minor adaptation. The use of component-based software engineering reduces the time and cost required for system development and simplifies the system. Component-based software engineering has been widely used in commercial software development and in online software evolution (Wang et al, 2006). Current research mainly focuses on (but is not restricted completely to) three areas: component design and specification, component storage and retrieval, and component assembly. To facilitate component retrieval and assembly in later development phases, each component must have a detailed specification, which is often XML-based (Wang et al., 2006). Searching for and retrieving available components that meet a specific requirement is another major challenge in component-based software engineering. There are three main genres of retrieval approaches: free-text keyword searching, faceted index searching, and semantic-based retrieval, each of which has certain advantages and disadvantages (Sugumaran and Storey, 2003). Among these approaches, semantic-based component retrieval has attracted the most attention since it provides greater query flexibility and user satisfaction in comparison to the other approaches. Various component retrieval system prototypes have been proposed and implemented (Wang et al., 2006, Yao and Etzkorn, 2004). WREN is an environment for component-based software development and uses an Argo/UML-based component assembly design editor to show the relationship between selected components (Lüer and Rosenblum, 2001). Aside from WREN, there are numerous other environments for component-based software development and assembly (Wang and Shin, 2002, Mezini and Ostermann, 2002, Mørch et al., 2004).

2.2.3 Data Routing and Aggregation in Sensor Networks

In wireless sensor networks, one of the most significant challenges is finding an efficient and stable data transmission route that adequately supports applications and maximizes the lifetime of the network. Several data routing approaches exist including flooding, gossiping, directed diffusion (Intanagonwiwat et al., 2000), geographic routing, and energy-aware routing (Heinzelman et al., 2000). These latter approaches are more efficient because they consider the energy efficiency of nodes in the network and maximize the network lifetime. Since communication is more expensive than computation for a wireless sensor node, another approach to maximizing network lifetime is data aggregation in the network. Several techniques have recently been developed to perform such in-network aggregation and conserve energy (Lee et al., 2004, Madden et al., 2002).

2.3 Discussion

This prior work guides the development of our software architecture for enabling lightweight decision support on intelligent construction sites. Specifically, by examining the existing work, we find that the available approaches do not satisfy the requirements we listed above in the following ways:

- Although the idea of component-based software development can easily engender the development of a decision support component that can integrate with a variety of other software components, many challenges remain related to the cost and complexity associated with developing such a module.

- Existing decision support approaches have shown great potential in decision making for large and complex problems where significant computing cycles are available for processing data. However, these techniques are based on large quantities of data and require nontrivial running times. Many of the decision support problems on construction sites require near real-time decision making, so new approaches are required.
- Current data routing and aggregation algorithms for wireless sensor networks have considered energy and computing capacity constraints. However, in our targeted applications, system users do not know in advance what data or even what sensors are available at a given moment. Changes are required to enable opportunistic resource discovery in a wireless sensor network on a construction site.

In the next section, we detail a software architecture that solves these research challenges by dividing the concerns into three relatively independent layers of functionality.

3. THREE-TIERED ARCHITECTURE DESIGN

In this section, we introduce a three-tiered architectural approach for lightweight, near real-time decision support in mobile ad hoc sensor networks to meet the unsolved requirements given in the previous section. Our architecture can be deployed in configurations of varying sizes, enabling applications to minimize the functions deployed on the resource-constrained sensors and to increase the functions deployed on more powerful devices such as handheld PDAs and tablet PCs. We first give an overview of the entire architecture, then describe each of its component tiers in more detail.

3.1 Overview of the Architecture

Our software architecture must support the implementation of functions in three layers, shown in fig. 2. The top layer, the Decision Support Layer (DSL) provides decision support for application users. A middle layer, the Data Processing Layer (DPL) enables expressive data processing in the form of aggregation, fusion, and other processing. The bottom layer, the Sensor Communication Layer (SCL), enables generic communication among mobile computing devices. Based on the application's specific requirements, an end user selects and executes small decision support applications, or *chunks* in our architecture. The DPL takes query instructions from the chunks and decomposes their data requirements into one or more concrete query plans. The DPL employs the SCL to send the queries' data requests to available sensors to collect the data. When the collected data is received, it is passed into the DPL for processing and onto to the application chunks through the DSL. A domain ontology within the DSL allows the DPL and SCL to be domain independent. In addition, it allows the architecture to be easily extended to new domains. We next discuss the details of each layer in this architecture.

3.2 The Decision Support Layer

The functionality within the DSL is decomposed into small applications, or chunks, each of which performs a specific decision support task, such as monitoring the concentration of a hazardous compound and alerting users when it exceeds a threshold. Such fine-grained decompositions of functionality keeps chunks loosely coupled with each other and with particular applications, easing the development of the DSL and allowing reuse of chunks on a variety of small computing devices in different environments. Since these chunks are domain-specific, they need to be developed by domain experts, which we refer to as "chunk developers." To ease the reuse and assembly of these chunks, a formal standard for the description of chunks is applied during chunk development. The internal description of a chunk may include (but is not restricted to) the function, required input, required output, behaviors, system requirements, and preferred performance. Chunks and their internal descriptions are stored in a domain-specific chunk repository, accessible to any chunk developer for reuse. These ready-to-use chunks can be assembled to implement more powerful composite functions. The chunk assembly is completed by site experts, which we refer to as "chunk assemblers." To facilitate chunk assembly, the repository embodies an efficient retrieval mechanism employing the expressive chunk descriptions. Chunk assemblers can query the repository for the relevant available chunks, given the application's unique characteristics and circumstances. Depending on the availability of predefined chunks, a chunk assembler may need to create new chunks for tasks unmatched by existing chunks. The end-users (e.g., the workers on a construction site) simply execute this developed composite application.

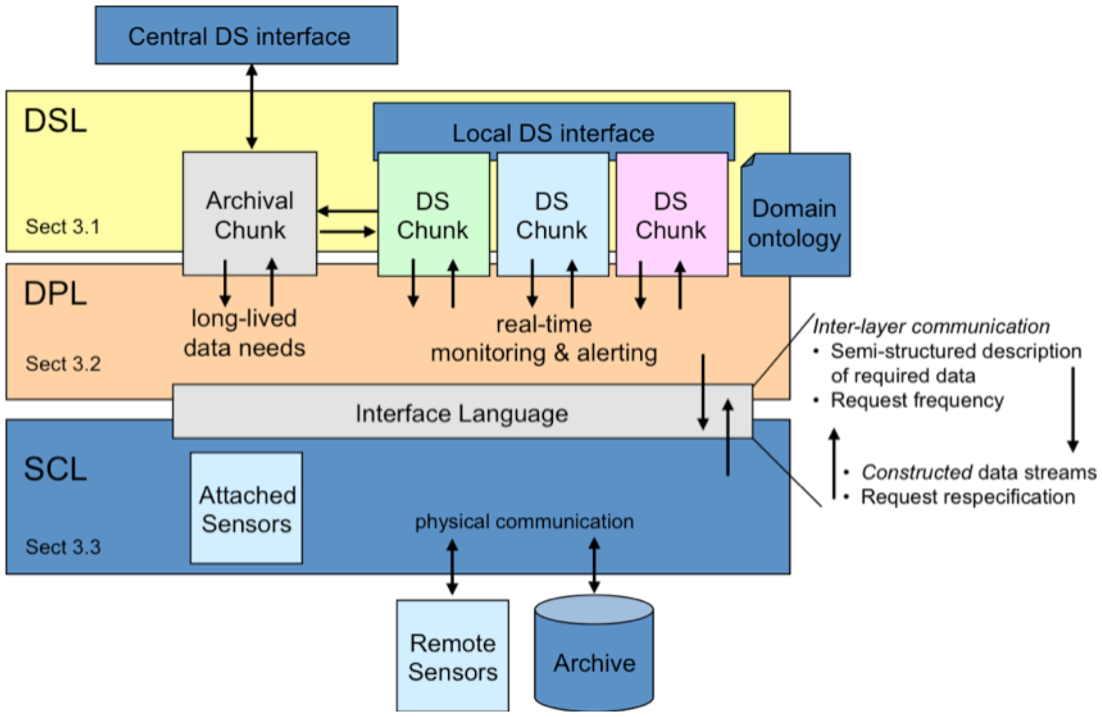


FIG. 2: Three-layer architecture for decision support in mobile ad hoc sensor networks

Decision support chunks in the DSL can be categorized into two types: (i) chunks that provide local and near real-time decision support for a particular user (*Decision Support Chunks*), and (ii) chunks that provide archival decision support for longer-lived information gathering with a more global perspective (*Archival Chunks*).

Decision support chunks address tasks that require direct access to local information and real-time decision support. When these chunks direct information to be collected from the local environment, the delivery of that data to the chunk causes it to respond to predetermined events (e.g., a user entering a dangerous area around a crane). The output of the decision analysis may appear on a user interface in the form of a value, a figure, or an alert. In addition, decision analysis output may serve as input to anther decision support chunk in some aggregate composition. As one example, consider an application interested in monitoring and responding to the presence of a hazardous concentration of a volatile organic compound (VOC) on the construction site. In implementing this application in our architecture, a single decision support chunk may continuously monitor VOC emissions and alert the user in the case of danger. A second chunk may take this measurement output as input, and, when a hazardous condition occurs, may trigger monitoring of wind direction to provide information about the possible movement of a dangerous gas plume.

Archival chunks, on the other hand, collect data available in the network and keep track of its changes over time, but they do not provide instant decision analysis to the local user. The data collected will be uploaded to a central archive whenever a reliable connection to one is available. Archival chunks may periodically monitor the available local information, decide what local information to archive for later use based on pre-described needs, and conduct analysis based on the data in the local archive. An example of an archival chunk may be finding the environmental causes of workers entering the dangerous area under a crane. By monitoring and analyzing the conditions that lead to such events, one may be able to learn to post better warnings for future sites.

3.3 The Data Processing Layer

When a user loads a chunk into the architecture, the DSL evokes the query text associated with this specific chunk and passes it to a query execution engine within the DPL. This description-based query text describes the desired data and the constraints on the data. The query execution engine translates the query text into a logic-based query plan that not only describes the query's data requirements (e.g., data type required, frequency of readings, etc.) but also the operations to be performed on the data (e.g., aggregation or filtering). The DPL then forms requests for data that it sends to the SCL for resource discovery and data collection.

A domain ontology defined for a particular application domain customizes the DPL's interaction within the particular realm of interest. The DPL uses the domain ontology to glean a semantic understanding of a chunk's requests for data to be able to create expressive query plans to send to the SCL and to understand the data returned by the SCL. The DSL can then interpret the received data before it passes it on to the chunk in the DSL. We have intentionally designed the query engine and other components of the DPL and SCL to be domain independent. By allowing query plans to be decoupled from domain knowledge, the DPL and SCL do not need to be aware of the particular application domain or the particular application chunk's intentions. In addition, developers of domain-specific decision support applications do not need to know the complex details and mechanisms within the DPL and SCL implementations. In this way, domain programmers can customize applications to specific site needs without making a large investment in understanding the details of specific sensors and related devices and technologies.

3.4 The Sensor Communication Layer

The SCL executes the query plan components it receives from the DPL. To then return the required data, the SCL interacts with the information-rich environment to discover the requested resources. The scripted query plan is unaware of the particular physical sensors available in the network and instead provides only a description of the desired data (e.g., VOC concentration in some regional area), desired query types (e.g., a one-time query or a long-lived monitoring query), the frequency of required readings, and the sizes of result windows. When the SCL receives a query from the DPL to be processed, the SCL discovers sensors that can provide the requested data type and communicates with these data sources to generate the requested data value or stream. We use Cross-layer Discovery and Routing (CDR) (Julien and Venkataraman, 2006) to discover and communicate with the available data sources. The SCL initializes communication between the data sources and the device generating the query to provide the required data to the DPL. The routing process is dynamic and real-time; as a worker with a handheld device moves on a jobsite, the connection is updated and linked to different physical sensors according to the worker's position.

Some query requests from the DPL require a combination of data from a set of different physical sensors. The SCL coordinates the set of sensors required and aggregates the information from the sensors to meet the DPL's data requirements. The SCL can also perform a limited degree of data integration to reduce the message overhead in the network, allowing more efficient use of the potentially limited network bandwidth. The SCL can also undertake energy and resource optimization tasks.

4. IMPLEMENTATION

In the previous section, we introduced our three-tiered architecture and the allocation of its underlying functional allocations. To demonstrate how to apply this architecture to make development and deployment of dynamic decision support applications simpler, especially for the domain developer, we implement an abstract framework for our architecture. This framework is written in Java to make interfacing with developers more straightforward. The Java framework interacts with low-level sensor implementations written for the TinyOS operating system. In this description, we define the commonly used classes and supported operations in the three layers introduced previously. An application developer can build a decision support application based on the classes provided in the framework and extend these provided classes to customize the framework's operation for a particular application or domain. Furthermore, since a basic query processing engine and sensor communication subsystem are embedded in this framework, the domain developer can focus on domain application development and does not have to worry about how queries are processed and data is retrieved from the network. This section first introduces the classes provided in the framework. It then shows the typical information flow among these classes.

4.1 Abstract Framework

Our abstract framework contains the necessary abstract classes to implement the three layers of our architecture. We start with a description of a set of common utility classes.

4.1.1 Common Utility

The common utility package includes several classes that can be employed by multiple layers. One class in this category is the *Query* class, used to hold and share the content of a data query. The query constructor contains (at a minimum) the following parameters: (i) the domain object, or the data the query is looking for; (ii) the query continuity, or whether the query is a one-time query or a continuous query; and (iii) the frequency of requested data for a continuous query. Operations within this class allow others to initialize and retrieve the values for these three variables. The *ResultListener* class is another utility class that implements a callback when results are received for a particular query. The *DomainObject* class provides an abstraction of data items that can be requested from and provided by sensor devices in the network. This class helps the sensor communication layer determine which available sensor to use to provide the requested information.

4.1.2 Decision Support Layer

Within the DSL implementation, the *ChunkLoader* class enables retrieval of information about the available decision support chunks in the chunk repository. In our current implementation, the *ChunkLoader* shows all the available chunks with a brief description of each. Future versions will include keyword searching and category-based retrieval. The most important class in this layer's implementation is the abstract *DSChunk* class. Implementation of applications' chunks must extend this abstract class and fill in its required functionality. When a *DSChunk* is loaded, end-users can initialize queries and dispatch query plans to the DPL. Specifically, the *DSChunk* relies on a built-in implementation of a *submitQuery* method to register and execute a query and the *stopQuery* method to stop the execution of a (continuous) query.

4.1.3 Data Processing Layer

The data processing layer handles the queries it receives from the decision support chunks. These queries arrive through the *DPLAccess* class, which is used for outside access to the DPL. The *QueryPlanGenerator* class parses the queries and creates a concrete query plan. For efficiency, we have implemented the query plan class as a binary tree. The query plan execution engine uses the queries and their associated result listeners to execute the query. This execution engine supports two basic operations: query plan execution and query plan stopping. A *DataCache* class is used to store the result returned by the result listener from the query plan execution engine before the result is passed to the DSL.

4.1.4 Sensor Communication Layer

In this layer, the *SCLAccess* class allows the DPL to send the queries from the query plan to be dispatched to the physical sensors. The SCL uses query information from the DPL to register a new query in the SCL. As a connection interface between the DPL and the SCL, the *SCLAccess* class provides *queryRegister* and *queryDeregister* operations. A Java class dedicated to interfacing between applications and the low-level sensor code allows the connection between the Java implementations and the low-level sensor hardware.

4.2 Data Flow among the Layers

We next use an abstract application example that involves only a single decision support chunk to demonstrate the data flow through the classes described above that implement the three layers of our architecture. This data flow is depicted pictorially in fig. 3. When the *ChunkLoader* submits a chunk for execution, the *DSChunk* is loaded and takes its query parameters from the user or application. As the *DSChunk* then submits the query, the constructed query passes into the DPL through the *DPLAccess* class and into the *QueryPlanGenerator*, which generates the concrete query plan. The query plan execution engine accepts the query plan and parses it into small queries that are then sent to the SCL for processing through the *SCLAccess* class. This class extracts the domain object information contained in the query to determine if the requested information can be directly retrieved from the physical sensors. If direct retrieval from the sensors is possible the SCL sends the query to the physical sensors (shown at the bottom of fig. 3). Ultimately, the result is returned through the *SCLAccess* class to the DPL where they are stored in the *DataCache* before being transmitted to the *DSChunk* for application processing. If the *DSChunk* issues an instruction to stop a query, that information is passed through the layers along the same path.

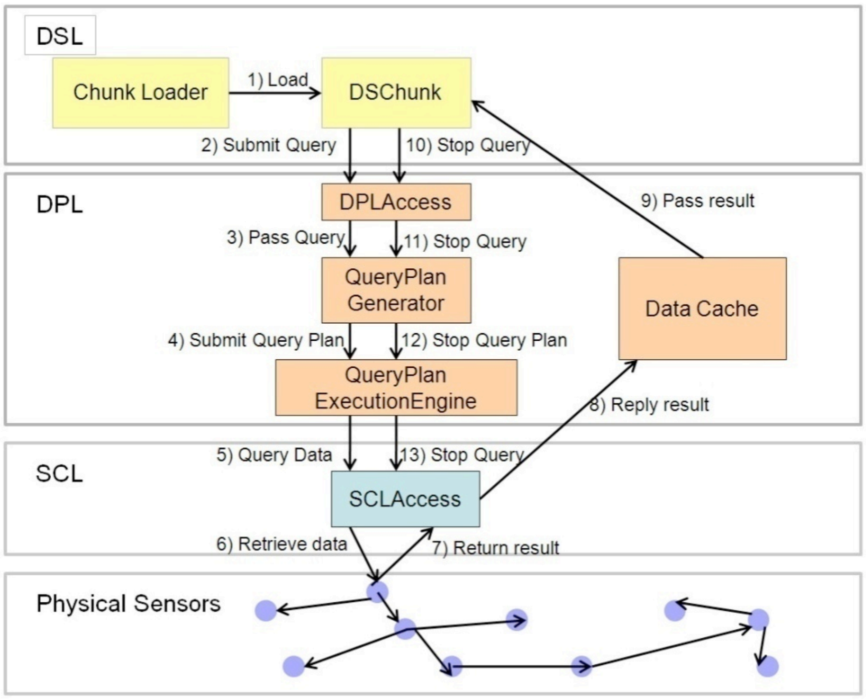


FIG. 3: Abstract depiction of data-flow through the three-tiered architecture

5. EXAMPLE DECISION SUPPORT APPLICATIONS

In this section, we will describe the development of two example decision support application on a construction jobsite. These examples serve to concretize the architectural concepts implemented in the abstract framework described in the previous section. The first of these applications is a hazardous gas detection application. The second notifies a worker if he is in a dangerous area surrounding a crane on the site.

5.1 Example 1: Hazardous Gas Detection

In our first example, the decision support application interacts with a local network to monitor and respond to dangerous concentrations of a hazardous gas.

5.1.1 Application Background

A variety of construction materials used in building interiors (e.g., paint, glue, carpet backing) have the potential to emit dangerous volatile organic compounds, or VOCs, such as benzene and alcohols, which contaminate the air. Such contaminations have been identified as a possible cause of various acute health conditions (such as headaches, fatigue, and respiratory tract infections) and discomfort to workers who inhale them (Andersson et al. 1997). To avoid the adverse effects of VOCs on construction workers, it is necessary to monitor and control the levels of various VOCs on a jobsite. Furthermore, it is important to perform this monitoring in real time so that proper corrective actions can be taken. Whenever a detected VOC's concentration exceeds a predefined threshold, the affected workers and a safety supervisor can be alerted. Additional information such as wind direction and speed can be collected and used in conjunction with the VOC concentration for further evacuation planning. To provide real-time VOC control and decision support for workers in the site, we use our framework to develop a decision support application that runs on a user's mobile handheld device.

5.1.2 Chunk Development

There are two important functional requirements for this hazardous gas detection decision support application: i) detecting if the nearby gas concentration has reached a dangerous threshold and ii) if so, showing the user the gas concentration distribution trends over the recent past to help him find a possible evacuation path. These two functions are implemented by two separate decision support chunks: the *DS_gasMonitor* chunk addresses the first, while the *DS_evacuationPath* chunk addresses the second. *DS_gasMonitor* takes in provided decision support parameters (query continuity type, query frequency, and gas concentration threshold) and instantiates a new query object to retrieve local gas concentrations. The query containing these three parameters and a result listener is dispatched through the DSL to the DPL for query resolution. When passing through the query to be processed, the chunk associates a result listener that will ultimately receive the query results. When the query is resolved by a data source in the SCL, the chunk's result listener receives the query result through the DPL's data cache. If the result received in the chunk indicates that the concentration has exceeded the specified threshold, the chunk alerts the user through a simple interface notification. The *DS_evacuationPath* chunk takes similar parameters to initialize a process that retrieves wind information from a local wind sensor. When the wind information is available, the chunk fuses the wind data with the VOC concentration data to generate predicted gas dissipation trends. This information is combined with the user's location to compute a possible evacuation path, which is displayed on the user's handheld device's screen. There is no user input for this decision support chunk because it is automatically triggered when the gas concentration exceeds the specified threshold.

5.1.3 Chunk Assembly

Many decision support applications require functions provided by various separate decision support chunks, necessitating intelligent chunk assembly and integration. Our gas detection application is a simple application that involves only two chunks. In our approach, we extract process diagrams to facilitate the chunk assembly process. Fig. 4 shows this diagram for our gas detection application. The majority of the processes are already included in the independent processes for the two decision support chunks, combined with a trigger (or a conditional controller) lying between the two chunks. Although the relationship between these two chunks is very clear in this case, the ability to assemble chunks in a straightforward manner requires careful consideration within the chunk's interface design.

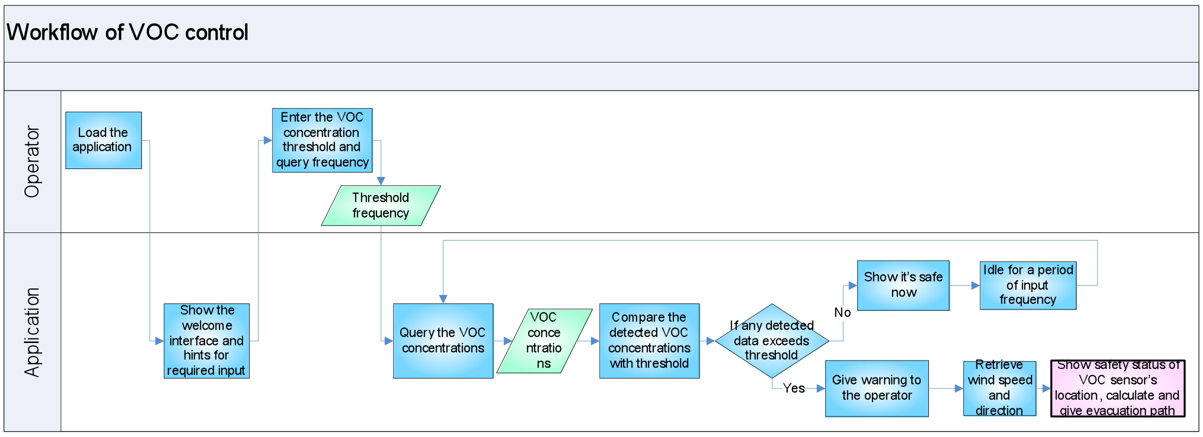
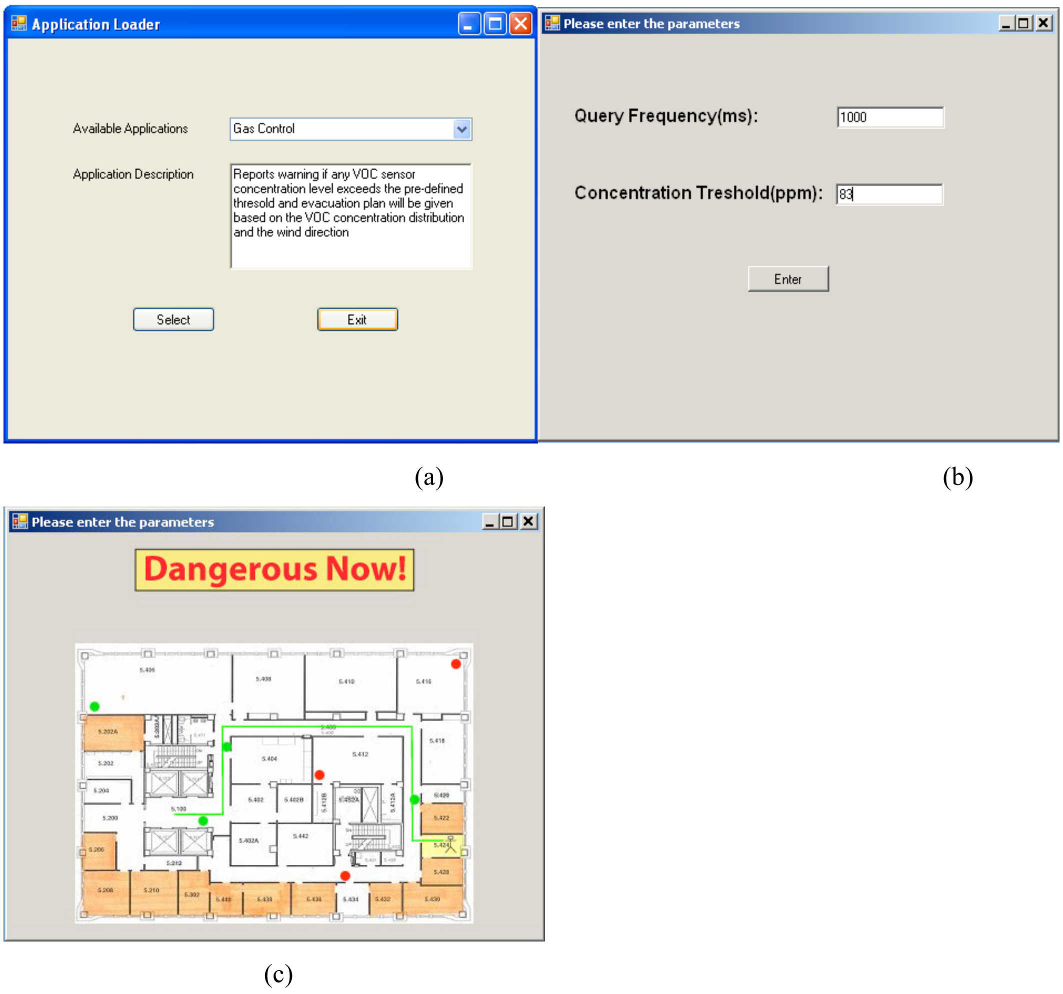

Workflow of VOC control
Operator
Application
Load the application
Enter the VOC concentration threshold and query frequency
Threshold frequency
Show the welcome interface and hints for required input
Query the VOC concentrations
VOC conce ntratio ns
Compare the detected VOC concentrations with threshold
If any detected data exceeds threshold
No
Yes
Show it's safe now
Idle for a period of input frequency
Give warning to the operator
Retrieve wind speed and direction
Show safety status of VOC sensor's location, calculate and give evacuation path

FIG. 4: Process diagram for gas detection chunk assembly

5.1.4 Demonstration

Fig. 5 shows the stages in the user interface of the final application. The chunk loader simply lists the available decision support chunks in the chunk repository on this device. Once the user selects a specific chunk (e.g., Gas Control in fig. 5(a)), the chunk's interface (if any) is loaded on the device. In this example, the interface initially consists of a prompt to the user to receive the required parameters (i.e., the query frequency, and concentration threshold, as shown in fig. 5(b)). Once submitted, the query containing these parameters passes through the DSL to the DPL where a query plan is generated. The query plan execution engine executes the query by sending concrete requests for data to the SCL, which retrieves the desired data from physical sensors. The qualified sensors compared their detected concentration levels to the threshold and send back a message indicating if the concentration is above the threshold or not. The DPL first stores the message in the data cache and then sends it back to the decision chunk in the DLS. When the returned message indicates that the gas concentration is over the threshold, a warning alert appears on the user's screen, and the evacuation path calculation chunk is automatically triggered to retrieve the wind information from the windsock. Finally, the chunk shows the result (a possible evacuation path) on the screen (as shown in fig. 5(c)).



Reports warning if any VOC sensor concentration level exceeds the pre-defined thresold and evacuation plan will be given based on the VOC concentration distribution and the wind direction

Query Frequency(ms): 1000

Concentration Treshold(ppm): 83

FIG. 5: User interface for the VOC detection application example

Fig. 6 shows an abstract depiction of the devices and distributed data flow in this example. In the figure, the numbers indicate steps in the application. The application first loads the chunk that monitors the VOC concentration. As this figure shows, this chunk sends an instruction through the DPL to the SCL to sense VOC concentration. Using the SCL for discovery and routing, the worker's handheld device can locate and contact the VOC sensor which performs long-lived sensing. If the sensor detects that the acceptable concentration threshold has been exceeded (step 4), the sensor notifies the handheld device, again through the SCL. Notice that the resource-constrained VOC sensor does not need to support any part of the architecture other than the SCL; it is not responsible for any query planning or aggregation tasks. When the worker's handheld receives the notification that a dangerous condition exists, it notifies the chunk, which, in addition to alerting the user, activates the second chunk (step 7). Again, this generates a query plan in the DPL, which generates queries sent via the SCL. In this case, however, a weather sensor may be less resource constrained. In particular, we depict the case where a leader of an array of wind sensors can aggregate wind information locally (step 13) before sending it back to the worker's handheld device.

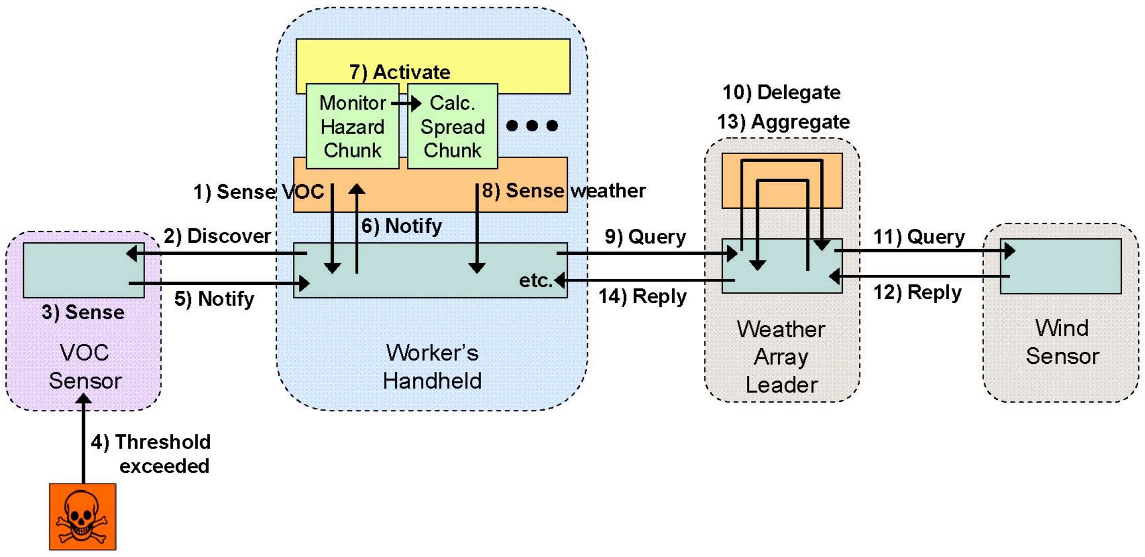


FIG. 6: An example deployment of the gas detection chunk application

This particular example demonstrates the functional scalability of our approach in the manner in which it allocates tasks to devices with varying resource capabilities. A particular device may support one to three layers of our framework. Devices with high computing capacity and abundant energy (e.g., handheld devices or tablet PCs) can be equipped with all three layers. These devices are also likely to support user interface interaction. Resource-constrained devices (e.g., wireless sensors) may be equipped with only one or two layers to support limited functionality while conserving resources.

5.1.5 Discussion

This application development example shows how a domain developer typically designs and implements a decision support application based on the architecture and abstract framework described in the previous section. In this example, the domain developer can concentrate on only the application level concerns, namely component design and implementation and user interface design. Therefore, domain experts with fundamental programming knowledge can handle the development task. Every component in the application corresponds to a specific function and can potentially be reused by other developers. However, as indicated in the discussion on chunk assembly, we still require a universal standard to facilitate the chunk assembly process. In addition, one may envision a more intelligent decision support application that can account for noisy or missing data. For example, if the data collected is noisy and reports an incorrectly elevated concentration, unnecessary evacuation measures may cause undue concern on the site. Future work will explore how to handle these spurious data issues, including understanding and managing incomplete or noisy data.

5.2 Example 2: Crane Safety

As the previous example demonstrated, safety applications are critical on construction sites and can be straightforward to implement with our framework. In this section, we demonstrate the development of a second such application: an application that monitors the unsafe area surrounding a crane and notifies workers when they or equipment move within this area.

5.2.1 Application Background

Cranes are widely used on construction jobsites. In the United States, there are nearly 125,000 cranes in use in the construction industry, and more than 250,000 crane operators, construction laborers, and non-construction workers are exposed to the risk of injury and death as a result of crane accidents every year. Crane-related injuries and fatalities account for most of the machinery-related accidents in the construction industry (Pratt, 1997). A number of crane operation and work safety guidelines have been introduced to improve the safety of crane operations on construction jobsites (CMAA, 2006, NSAI, 2004, Reich et al., 1994). These guidelines explicitly specify the required spacing for crane operations. However, workers might still unknowingly enter the dangerous area around a crane due to the constricted working space or because they misjudged their position in reference to the crane. An efficient way to warn workers that they are in danger with respect to a crane can reduce the risk of injury and death. Fig. 7 depicts the dangerous area of an operating crane (Khaled and Ahmed, 2005). In the figure, *Area I* is the actual working area of the crane, where the load or parts of the load may fall from the boom. *Area II* does not contain a danger with respect to falling objects (e.g., it represents an area *behind* the crane's arm), but it does present danger with respect to crane collapse. In this section, we demonstrate a decision support application for crane safety. If a worker's location indicates that he has entered *Area I*, his device will warn him to leave the area; if he enters *Area II*, his device will warn him to be increasingly careful of his surrounding environment.

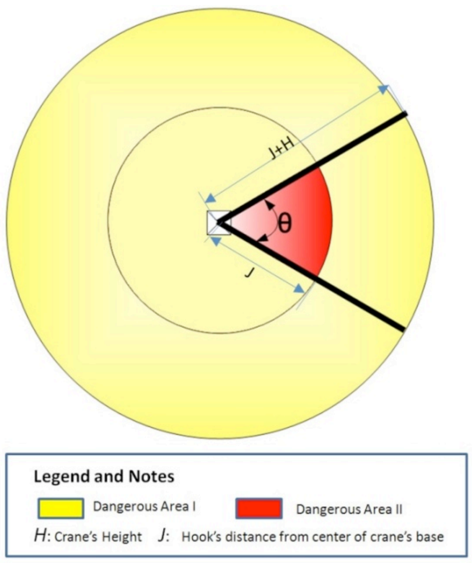


FIG. 7: Dangerous areas around an operating crane

5.2.2 Chunk Development

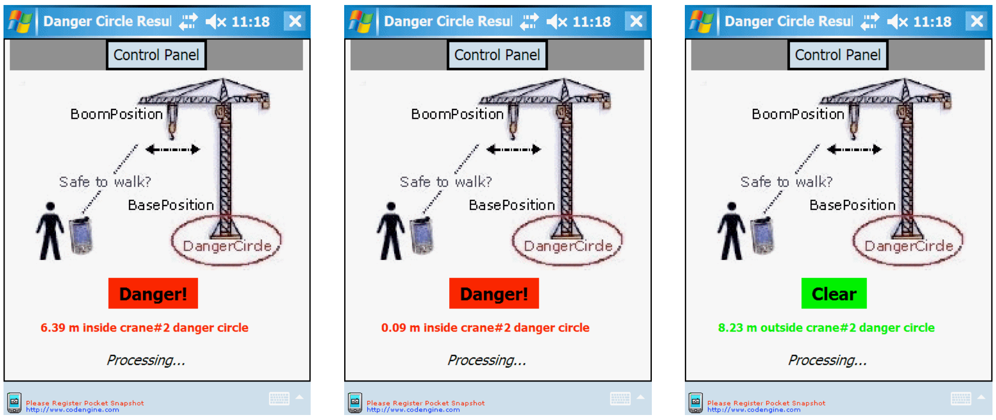
Given the functional requirements of this application, one simple chunk can serve to provide the required decision support. The implementation of this chunk is similar to that of the gas detection chunk presented in the previous section. The major difference lies in the type of the data queries. In this decision chunk, the chunk queries for: i) the boom's operating angle, ii) the hook's current location on the boom, iii) the crane's base's location, iv) the crane's height, and v) the user's location. Once collected, this information is aggregated within the DPL on the user's device. The hook's location and the crane base's location are used to compute the value of *J* in fig. 7. This value, in combination with the other sensed values are used to judge which area depicted in fig. 7 contains the user's location.

5.2.3 Application Optimization

In the naïve implementation above, the decision chunk queries the sensors attached to the crane to retrieve the hook's current location and the crane base's current location. This data is sent back to the user's device through the SCL and used to calculate the value of *J*. Moving the computation of *J* into the sensor network reduces the number of messages sent in the network and the sensors' energy consumption. We use a *virtual sensor* to optimize the energy the sensor nodes spend in answering this application query (Kabadayi et al., 2007). The virtual sensor in this application takes the hook's location and the crane base's location and computes *J* directly. This value is stored on a software sensor *in the network* that provides the same interfaces to the SCL as any hardware sensor. This enables multiple users' devices to reuse this calculation, saving precious energy and bandwidth resources.

5.2.4 Demonstration

The application interface for this decision support application is shown in fig. 8. In this application, the chunk passes queries to the DPL, which are turned into query plans and then into specific queries that are dispatched to the SCL and on into the physical sensor network. If the discovery phase in the SCL locates a virtual sensor that is not yet in use, it activates that sensor and starts collecting data from it. If the virtual sensor is already active, the application can simply connect to it to receive data. Fig. 8 shows the progression of a worker who has entered the dangerous area around a crane and is moving away from it until he is clear of danger. In addition to the handheld device demonstration, we have also created a small-scale crane equipped with sensors to actively monitor the movement of the crane (Kabadayi et al., 2007).


Danger Circle Resul 11:18
Control Panel
BoomPosition
Safe to walk?
BasePosition
DangerCircle
Please Register Pocket Snapshot http://www.codengine.com

Danger!

6.39 m inside crane#2 danger circle

Processing...

Danger!

0.09 m inside crane#2 danger circle

Processing...

Clear

8.23 m outside crane#2 danger circle

Processing...

FIG. 8: Crane safety application demonstration

5.2.5 Discussion

In the crane safety application, we consider the network constraints by introducing simple in-network aggregation techniques to achieve better overall performance. In this way, a simple computation task is pushed into the sensor network to make use of the sensor nodes' (limited) computation capabilities. In this application, we consider a single tower crane on a site to make the application simple. On real construction sites, there may be several cranes at work. In these cases, the application needs to be extended with data about which location data belongs to which crane. However, additional possibilities exist then in monitoring the relationships between the cranes to prevent accidents involving multiple cranes.

6. RESEARCH CHALLENGES

The implementation of two decision support applications described in the previous section demonstrate that our three-tiered architecture can be used to develop a variety of application using the support of various sensing, computing, and communications devices. These applications also highlighted the architecture's capability to support data abstraction, to adapt to dynamic environments, and to provide opportunistic data collection and coordination. In implementing these applications for our architecture, we identified several additional research challenges whose resolution could even further advance the use of sensor networks on real-world construction sites. In the decision support application development, our architecture would be enhanced by a careful design of mechanisms to support efficient chunk assembly, including tool support for chunk development. In addition, as domain programmers access the chunk repository to construct their applications, a precise formal understanding of the performance characteristics, behavioral properties, and other tradeoffs of different chunks could provide helpful guidance in the development task. In the data processing and sensor communication layers, understanding how to effectively pre-fetch data for future queries is also an interesting area of future research. In addition, explicitly handling the unpredictability of these environments due to the movement of both the query issuer and the underlying sensor network is an important research challenge. We discuss these newly revealed challenges in the remainder of this section.

6.1 Decision Support Component Development

Of the challenges elucidated above, one of the most important is providing development support for decision support chunks. This is especially important since we intend for domain programmers, who are not experts in programming languages or techniques, to develop decision support chunks. Several techniques have been widely used for component-based software engineering development support (Szyperski, 1997). These techniques provide specific guidelines on how to design the interfaces, internal structure, and specifications of particular components. These issues include determining a component's ideal granularity, specification standards for component self-description, and efficient component retrieval mechanisms. In our architecture, when developers begin chunk design, they must perform a similar set of tasks. Therefore, we require the creation or adaptation of a business-process-based chunk design tool to facilitate chunk definition and assembly. Our architecture requires an approach different from previous ones due to the focus on decision support applications, the drive to support domain programmers, and the unpredictability of our target environments.

Each chunk in our architecture encapsulates a specific decision support task. However, due to the distribution and mobility inherent in our target environment, the decision support problems within a particular chunk are not well-defined, and the data available to support those tasks may change over time as a result of changing sensor availability. To handle similar problems and thereby provide the best decision support possible, a range of techniques have been proposed (Greco et al., 1999, Grzymala-Busse and Hu, 2000, Scheffer, 2002, Zhu et al., 2004). These techniques were formulated for well-defined data mining processes, based on large data sets. They do not map well to our target environment because they suffer in meeting the near real-time requirements we have on decision support applications. Therefore, new decision support approaches are required to support chunk development and operation, especially with respect to providing support in the face of missing or unreliable data.

Another concern related to chunk development has to do with how developed chunks are stored and shared for reuse. Different developers write different decision support chunks that are inserted into the chunk repository. These different developers may use different denotations for the same concept, making efficient chunk retrieval difficult. In this work, we introduced a *domain ontology* for describing elements of domain in a common way. How to use the information in this ontology for efficient chunk retrieval is an open question. Furthermore, an efficient chunk retrieval engine that can support retrieval both locally and over the Internet presents more challenges to design and implementation.

6.2 Decision Chunk Assembly

Many applications will take the form of our first example application, in which multiple decision support chunks are combined to provide the application's expressive decision support behavior. While we kept our example small for simplicity, the interactions between these chunks may be complex and multidirectional, making chunk assembly a nontrivial task. The users who serve as chunk assemblers for a specific site or project have significant expertise in their site or sites like theirs, but not in programming. Therefore, this chunk assembly task must be made very simple without limiting the expressiveness and flexibility of the architecture. We expect that, to fully support the power of our architecture, we require the development of a chunk assembly environment. This tool must be language independent and must be accessible to nonprogrammers.

6.3 Performance Measurement

Traditional software performance metrics do not map well to component-based software engineering; however, approaches exist for evaluating component-based software development (Gill and Grover, 2003). To fully measure the performance of an application developed for our architecture, we need to develop a group of practical metrics that can communicate the effectiveness and power of the developed applications.

In addition, understanding the performance of different chunk implementations and different query plan implementations is important to provide to developers at all levels. Challenges related to the DPL stem from ensuring effective operation while maintaining the domain and application independence of the layer's implementation. As the DPL is not necessarily aware of the physical sensors and is not designed with knowledge of specific chunks, generalized approaches to abstract query processing present a significant challenge. Effective integration of data queries in the DPL also requires further consideration to achieve better performance (e.g., data query latency, network resource consumption, etc.).

A related challenge stems from the ad hoc nature of sensor networks, which may require fusion of multiple data streams with different integration algorithms depending on the data available in the SCL. A key feature of the SCL is its opportunistic operation, both in terms of identification of supporting sensors and of communication with these selected sensors. Beyond identifying an arbitrary set of supporting sensors (that may change as the user moves through the site), efficient interaction is a challenge. In an environment with multiple sensors and handheld devices, it is desirable to limit bandwidth and power requirements on resource-constrained sensors (Kabadayi and Julien, 2007). Routing and data aggregation between devices that support the SCL are central research challenges.

7. DISCUSSION: INDUSTRIAL IMPLICATIONS

The major industrial contribution of the three-tier architecture described by in this paper will be to speed deployment of various applications that support the intelligent job-site. Development to-date has largely been tied to specific hardware with few tools that allow generic development of applications. This has hindered deployment in several ways: First, it has required developers to have extensive knowledge of both hardware and software at multiple levels, making commercial development more expensive and hence less likely until a market can be proven. Second, beyond cost, it ultimately limits innovation as few individuals or small teams have the necessary expertise to pilot prototype applications. This is particularly troublesome for construction as the complexity of project execution requires considerable domain expertise to tailor useful applications. Third, the expense and limited expertise problem is also seen in academia where research prototypes tend to be restricted and hence difficult to test robustly. Similarly, the expense limits technology transfer as more robust applications are costly to develop. By providing an architecture that separates details of the hardware from top-level domain programmers, the necessary depth and range of expertise to develop complete applications is significantly reduced. This facilitates innovation both in academia and in the commercial sector. A broader range of applications deployed means more experimentation and validation. In turn, this should help identify the most successful pilots and speed commercial applications. At the same time, the market for commercial applications should be thickened both in supply and demand. Less cost should translate to lower price, increasing market demand. At the same time, less development cost should increase the potential supply of applications as the barriers for entry are reduced. Hardware suppliers and their partners may enter the market, and small developers with domain expertise will find a broader range of tools they can develop applications with and for.

More efficient development of applications will make commercial realization of the intelligent job-site both more rapid and more extensive. The current literature on IJS deployment has seen utilization on job-sites by larger companies. Of course, large construction companies make up a small proportion of the market; most construction firms are small outfits with limited ability to invest in new technologies. It remains likely that the larger firms will continue to be the initial adopters of IJS technologies. However, lower cost will make these technologies more attractive to both large and small companies and hence should help spread the benefits of IJS across the industry. Perhaps the most important benefits of such technologies will be in safety. While IJS safety application remain very much a research topic, the potential benefit in accident avoidance is great. Safety monitoring that operates continuously in the background represents a radical departure from traditional safely planning and 'work-safely' programs. While not a replacement for such programs, the provision of background safety monitoring may help to avoid accidents in much the same way electronic safety programs have helped automobile safety by helping drivers avoid putting their vehicles in unsafe conditions that may, for example, cause a roll over. Productivity and materials management are the most likely immediate beneficiaries of more rapid deployment of IJS technologies. Existing commercial applications focus on RFID supported materials tracking. Likely logical extensions would be to integrate materials tracking on-site with off-site logistics, supporting practices such as vendor managed inventories and lean construction planning. Tools for supporting dynamic field sequence planning also are a logical development, linking 3D/4D technologies with up-to-date and highly accurate monitoring of field conditions. More accurate monitoring would also enable automatic progressing for the site, aiding both project management and project controls functions with faster, more extensive, and more accurate data. These projected applications are likely to be just a few of those that will be developed with time and experimentation. A final thought for industrial applications is that broadly applied IJS technologies will allow more complete documentation of job sites, presenting new opportunities for data mining and hence discovery of new paradigms and techniques for productive and safe job-sites.

8. CONCLUSIONS

In this paper, we have presented a novel architecture for decision support applications in mobile ad hoc sensor networks. The architecture's functionality is carefully divided into three layers: a layer for expressive yet approachable decision support application development, a layer for expressive data processing, and a layer for efficient sensor communication. We described our prototype implementation of the architecture and used two example applications for construction sites to demonstrate its applicability. The three-tiered architecture increases the level of abstraction and can therefore speed application development by reducing the need for domain developers to have detailed knowledge of device specifics. This also increases the reusability of software and protocols developed at all levels and makes the architecture applicable to other industries outside of construction. Future research will focus on development support for crafting different areas of the architecture, including expressive support for domain programmers.

Our research will have practical impacts for the development and deployment of innovative field technologies. The industry vision for the intelligent jobsite is driving considerable research and development; however, the bulk of this effort is application specific and directly ties hardware (e.g., RFID tags, sensors) to higher level data processing applications. Our work seeks to decouple sensors from computing hardware and application development by providing a flexible and robust middleware. Realization of the middleware will enable more flexible reuse of data on sensors to make it available to a range of decision support applications, while at the same time speeding development of applications as the developers need not be tied to specific technologies. Commercial implementation of the research will make possible construction specific visions for ubiquitous computing by enabling flexible and robust discovery and use of data in an ad hoc manner. Rather than deploying a specific application, a worker's mobile device can continuously monitor the local environment for data and provide local decision support on demand. We expect that worker safety and productivity will be enhanced accordingly.

9. ACKNOWLEDGEMENTS

C. Julien and S. Kabadayi would like to thank the Center for Excellence in Distributed Global Environments for providing research facilities and the collaborative environment that supports this work. This research was funded, in part, by the National Science Foundation, Grant #OCI-0636299. The conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

10. REFERENCES

Andersson, K., Bakke, J. V., Bjorseth, O., Bornehag, C. G., Clausen, G., Hongslo, J. K., Kjellman, M., Kjaersgaard, S., Levy, F., lhave, L., Skerfving, S., and Sundell, J. (1997). "TVOC and Health in Non-industrial Indoor Environments: Report from a Nordic Scientific Consensus Meeting at Langholmen in Stockholm, 1996." *Indoor Air*, 7, 78-91.

Caldas, C. H., Torrent, D. G., and Haas, C. T. (2006). Using Global Positioning System to Improve Materials-Locating Processes on Industrial Projects. *Journal of Construction Engineering and Management*, 741-749.

Chen, A. Y., Tsai, M.-H., Lantz, T., Kaushik, N., Lakhera, S., Plans, A. P., Mathur, S., Pena-Mora, F., "SERMAN: A collaborative framework for Supporting Emergency Response with Mobile Ad-Hoc Networks," ASCE International Workshop on Computing in Civil Engineering, Pittsburgh, PA, USA, July, 25-28, 2007.

CMAA. (2006). *Operational Guide for Lifting Devices*, Crane Manufacturers Association of America.

Codd, E. F., Codd, S. B., and Salley, C. T. (1993). Providing OLAP (On-line Analytical Processing) to User Analysts: An IT Mandate. Arbor Software Corporation.

De la Garza, J. M., and Howitt, I. (1998). Wireless communication and computing at the construction jobsite. *Automation in Construction*, Vol. 7, 327-347.

FIATECH. (2003). Strategic overview: Capital projects technology roadmap initiative (version 1), FIATECH.

Fuller, S., Ding, Z., and Sattineni, A. A Case Study: Using the Wearable Computer In the Construction Industry. *Proceedings of the International Symposium on Automation and Robotics in Construction*, Gaithersburg, Maryland, 551-556.

Gill, N. S., and Grover, P. S. (2003). Component-based measurement: a few useful guidelines. *SIGSOFT Software Engineering Notes*, Vol. 28, No. 6.

Goodrum, P. M., McLaren, M. A., and Durfee, A. (2006). The application of active radio frequency identification technology for tool tracking on construction jobsites. *Automation in Construction*, Vol. 15, No. 3, 292-302.

Greco, S., Matarazzo, B., and Slowinski, R. (1999). Handling Missing Values in Rough Set Analysis of Multi-Attribute and Multi-Criteria Decision Problems " *Proceedings of the 7th International Workshop on New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, 146-157.

Grzymala-Busse, J. W., and Hu, M. (2000). A Comparison of Several Approaches to Missing Attribute Values in Data Mining. *Lecture Notes in Computer Science*, Vol. 2005, 378-385.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-Efficient Communication Protocol forWireless Microsensor Networks. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii.

Inmon, W. H. (2005). *Building the Data Warehouse*, John Wiley & Sons, Inc.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *Proceedings of the International Conference on Mobile Computing and Networking*, 56--67.

Jaselskis, E. J., and El-Misalami, T. (2003). RFID's role in a fully integrated, automated project process. *Proceedings of the Construction Research Congress In Construction - Wind of Change: Integration and Innovation* (R. M. Keith and S. C. Paul, editors), Honolulu, Hawaii, U.S.A., 91.

Julien, C. and Venkataraman, M. Cross-Layer Discovery and Routing in Reconfigurable Wireless Networks. *Proceedings of the 3rd International Conference on Mobile Ad-Hoc and Sensor Systems*.

Kabadayi, S. and Julien, C. (2007). Scenes: Abstracting Interaction in Immersive Sensor Networks. *Pervasive and Mobile Computing*, Vol. 3, No. 6, 635-658.

Kabadayi, S., Julien, C., O'Brien, W. J., and Stovall, D. (2007). Virtual Sensors: A Demonstration. *26th International Conference on Computing Communications: Demonstrations Track*.

Khaled, E.-R., and Ahmed, K. (2005). Trade-off between Safety and Cost in Planning Construction Site Layouts. *Journal of Construction Engineering and Management*, Vol. 131, No. 11, 1186-1195.

Köseoğlu, O. O. (2004). Construction Project Control through Wireless Networking. Middle East Technical University, Ankara, Turkey.

Lee, S.-J., Lee, C.-J., Cho, Y.-Z., and Kim, S.-U. (2004). A New Data Aggregation Algorithm for Clustering Distributed Nodes in Sensor Networks. *Lecture Notes in Computer Science*, Vol. 3262, 508-520.

Lee, U.-K., Kang, K.-I., and Kim, G.-H. (2006). Mass Concrete Curing Management Based on Ubiquitous Computing, *Computer-Aided Civil and Infrastructure Engineering*, Vol. 21, No. 2, 148-155.

Lüer, C., and Rosenblum, D. S. (2001) WREN---an environment for component-based development. *Proceedings of the 8th European Software Engineering Conference*

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. *Proceedings of the 5th Annual Symposium on Operating Systems Design and Implementation.*

Mezini, M., and Ostermann, K. (2002). Integrating independent components with on-demand remodularization. *SIGPLAN Notes,* Vol. 37, No. 11, 52-67.

Mørch, A. I., Stevens, G., Won, M., Klann, M., Dittrich, Y., and Wulf, V. (2004). Component-based technologies for end-user development. *Communications of the ACM*, Vol. 47, No. , 59-62.

NSAI. (2004). *Code of Practice- Safe use of Cranes in the Construction Industry (2004 Draft)*. National Standards Authority of Ireland, 56.

Nuntasunti, S., and Bernold, L. E. (2006). Experimental Assessment of Wireless Construction Technologies. *Journal of Construction Engineering and Management*, Vol. 132, No. 9, 1009-1018.

Pratt, S. (1997). Machinery-related fatalities in the construction industry. *American Journal of Industrial Medicine*, Vol. 32 No. 1, 42-50.

Reich, R. B., Dear, J., and Culver, C. G. (1994). *Mobile Crane Inspection Guidelines for OSHA Compliance Officers*. U.S Department of Labor, Occupational Safety and Health Administration.

Riaz, Z., Edwards, D. J., and Thorpe, A. (2006). SightSafety: A hybrid information and communication technology system for reducing vehicle/pedestrian collisions. *Automation in Construction*, Vol. 15, No. 6, 719-728.

Sacks, R., Navon, R., Brodetskaia, I., and Shapira, A. (2005). Feasibility of Automated Monitoring of Lifting Equipment in Support of Project Control. *Journal of Construction Engineering and Management*, Vol. 131, No. 5, 604-614.

Scheffer, J. (2002). Dealing with Missing Data. *Research Letters in the Information and Mathematical Sciences*, Vol. 3, 153-160.

Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., and Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems,* Vol. 33, 111-126.

Song, J., Haas, C. T., and Caldas, C. H. (2007). A proximity-based method for locating RFID tagged objects. *Advanced Engineering Informatics,* Vol. 2, No. 4, 367-376.

Song, J., Haas, C. T., Caldas, C. H., and Liapi, K. (2005) Locating materials on construction site using proximity techniques. *Proceedings of the Construction Research Congress: Broadening Perspectives*, San Diego, California, U.S.A., 108.

Stone, W. C., Pfeffer, L., and Furlani, K. (2000) Automated Part Tracking on the Construction Jobsite. *Proceedings of the International Conference on Robotics for Challenging Situations and Environments*, Albuquerque, New Mexico, 9.

Sugumaran, V., and Storey, V. C. (2003). A semantic-based approach to component retrieval. *SIGMIS Database*, Vol. 34, No. 3, 8-24.

Sunkara, P. (2005). A Tablet PC Application for Construction Site Safety Inspection and Fatality Prevention. Louisiana State University, Baton Rouge, LA.

Szyperski, C. (1997). *Component software : beyond object-oriented programming*, ACM Press, New York.

Tan, P.-N., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining.* Addison-Wesley.

Trupp, T., Marulanda, C., Hashash, Y., Liu, L. and Ghaboussi, J. (1994). Novel Technologies for Tracking Construction Progress of Deep Excavations. *Proceedings of Geo-Trans* (M. K. Yegian and E. Kavazanjian, editors), 2254-2262.

Tserng, H. P., and Dzeng, R.-J. (2005). Mobile Construction Supply Chain Management Using PDA and Bar Codes."*Computer-Aided Civil and Infrastructure Engineering*, Vol. 20, 242-264.

Wang, Q., Shen, J., Wang, X., and Mei, H. (2006). A component-based approach to online software evolution. *Journal of Software Maintenance and Evolution*, Vol. 18, No. 3, 181-205

Wang, S., and Shin, K. G. (2000) An architecture for embedded software integration using reusable components. *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems* San Jose, California, United States 110-118

Yao, H., and Etzkorn, L. (2004) Towards a semantic-based approach for software reusable component classification and retrieval. *Proceedings of the 42nd annual Southeast regional conference,* Huntsville, Alabama 110-115.

Zhu, W., Zhang, W., and Fu, Y. (2004). An Incomplete Data Analysis Approach Using Rough Set Theory. *Proceedings of the lntemational Conference on Intelligent Mechatronics and Automation*, Chengdu, China.