

**Flexible Data Entry for  
Information Warning and Response Systems**

**Alice M. Mulvehill**

BBN Technologies  
10 Moulton Street  
Cambridge, MA 02138  
617-873-2228  
Fax: 617-873-2794  
amm@bbn.com

**James Reilly**

Air Force Research Lab (AFRL)  
525 Brooks Road  
Rome, NY 13441  
315-330-3333  
Fax: 315-330-2885  
James.Reilly@rl.af.mil

**Brian Krisler**

BBN Technologies  
10 Moulton Street  
Cambridge, MA 02138  
617-873-4899  
Fax: 617-873-2794  
bkrisler@bbn.com

# Flexible Data Entry for Information Warning and Response Systems

**Alice M. Mulvehill**

BBN Technologies  
10 Moulton Street  
Cambridge, MA 02138  
617-873-2228  
Fax: 617-873-2794  
amm@bbn.com

**James Reilly**

Air Force Research Lab (AFRL)  
525 Brooks Road  
Rome, NY 13441  
315-330-3333  
Fax: 315-330-2885  
James.Reilly@rl.af.mil

**Brian Krisler**

BBN Technologies  
10 Moulton Street  
Cambridge, MA 02138  
617-873-4899  
Fax: 617-873-2794  
bkrisler@bbn.com

## Abstract

The need to collect data that can provide warnings to avert crisis situations is paramount to many modern military and civil response systems. These systems allow the end user to report a variety of anomalies about the world. The U.S. Air Force's Integrated Information Management System (IIMS) has been developed to support the collection and interpretation of data that may be indicative of potential chemical or biological attacks. Data-entry forms are generally provided to end-users for description of incidents of interest. Since all possible data types cannot be realized in the design process, it would be useful to provide forms that can adapt to the data being collected.

Tracker is a tool that allows a user to define and use one or more XML-based templates (forms) to support problems such as crisis-action mission planning. Tracker provides users with a suite of tools that allow them to easily add or modify fields in a given template or instance. Since this feature is the type of capability that the IIMS developers wanted for their system, an experiment was conducted linking Tracker to IIMS. In this paper we present the results of this experiment.

## Introduction

The need to collect data that can provide warnings to avert crisis situations is paramount to many modern military and civil response systems. These systems allow the end user to report a variety of anomalies about the world. The ideal system, while it must impose a certain amount of consistency of form to the data, must also somehow offer enough flexibility to allow the user to describe events that were not anticipated and therefore not designed into the data collection forms. In this paper we describe an experiment that attempted to enhance the flexibility of the data-entry forms in a crisis-support system called the Integrated Information Management System (IIMS) [1].

*Tracker* is a tool that allows a user to define and use one or more XML-based templates (forms) to support a variety of problem-solving contexts, e.g., crisis-action mission planning. In many applications, a template is equivalent to a blank form that describes a set of fields and provides specifications about the value of a given field. With *Tracker*, templates can be developed to have an *active* quality. This means that information elements (fields) defined in a template can be linked to other information elements that appear in one-or-more related templates. When these relationships are specified, the modification of a given information element will result in *active* data propagation or change in the related elements. In the systems that have been developed with *Tracker* to date, we have noticed that this *active* feature not only reduces the data entry effort, it can also be used to manage data entry from multiple users who are working together on a given problem in a distributed-computing environment.

*Tracker* also offers another feature that is useful in problem domains where unanticipated extensions to data-entry mechanisms are the norm. *Tracker* offers the end user a method for easily adding fields to any template instance (a template that is being used to make a report) to handle unanticipated data entry or planning needs.

The experiment we describe in this paper focused on the collection and integration of data specifically associated with biological and chemical incidents as collected and analyzed in IIMS.

IIMS is a suite of information technologies that support the command and control (C2) required for effective Nuclear-Biological-Chemical (NBC) modeling, situation awareness and analysis. The technology underlying IIMS provides the means for fusing information from the many passive, active, and human data sources that are associated with the detection and tracking of chemical and biological attacks, both overt and covert.

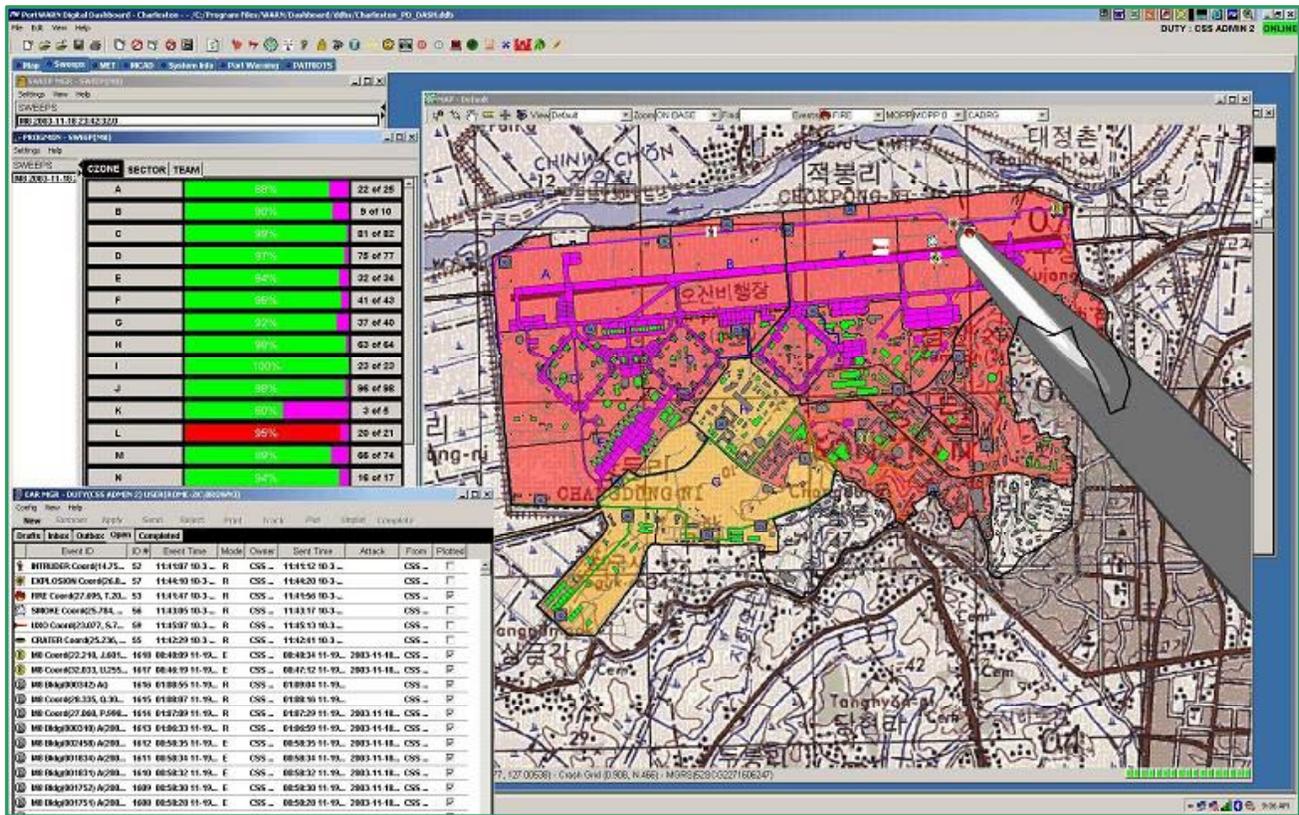


Figure 1: IIMS Digital Dashboard

In order to provide C2 decision makers with timely situational awareness, IIMS leverages the concept of a Digital Dashboard. In many decision support environments, Digital Dashboards are often used to gather information about a problem and provide a set of visualization methods that describe the state of a problem domain at a given time [2]. Digital Dashboards can be hand crafted, as is the IIMS Digital Dashboard, or they can be commercially obtained, then tailored to suit a particular problem. For example, a commercially available Digital Dashboard tool, Executive Dashboard by ServiceWare [3] provides Analysis, Alerts and Reporting tools in a single easy to use application. This tool relies on the Microsoft Office XP Digital Dashboard [4] developer environment, which incorporates Web-based elements (such as news, stock quotes, and so on) called WebParts, allowing these parts to be integrated into a single Office like application. Office dashboards can be customized in a multitude of ways and named accordingly, for example as a general corporate or enterprise dashboard, or more specifically, as a commander's or CEO dashboard.

The IIMS Digital Dashboard (see Figure 1) was developed to support decision makers specifically concerned with chemical and biological incidents. This Dashboard features a java-based graphic-information system and a suite of situational awareness tools that can be customized by the operator to facilitate the flow of hazard and damage information into a single integrated operational picture.

The Digital Dashboard also acts as a container application and framework for the integration of one or more loosely coupled and dynamically configurable software components. These components, called cells, can be run as stand-alone applications, as an applet in a Web page, or contained within the Digital Dashboard.

In theory, the *Tracker* application is capable of supporting IIMS as either a stand-alone application, as a cell, or in both modes. While *Tracker* can support the distributive collaborative usage of templates during planning and execution, its strength and most attractive feature to the IIMS developers was the ease that the *Tracker* system affords users in the creation, usage and extension of templates/forms.

## Background

### IIMS Overview

The genesis of the IIMS Digital Dashboard began with the Restoration of Operations (RestOps) Advanced Concept Technology Demonstration (ACTD) [5]. This ACTD was designed to help fixed military sites, such as air bases and seaports, protect themselves against and recover from chemical or biological attacks. The RestOps Information Management (ROIM) tool replaced manual reporting processes with computer-based data entry and display capabili-

ties in order to enhance situational awareness of Command and Battlestaff, and unit-control-center personnel.

IIMS continues to evolve these capabilities by providing unparalleled versatility and interoperability. IIMS has been showcased in the 2004 Joint Warfighter Interoperability Demonstrations (JWID). During this JWID effort IIMS was used to display data originating from players throughout the United States and from worldwide coalition partners. As IIMS matures and continues to integrate its capabilities, it is envisioned to play a role as the C2 backbone for various warning and response systems.

The developers of IIMS are interested in supporting incident response and battle management with tools that enable operators to *interact* with the system to both describe the incident and to respond appropriately to the incident. This is currently supported in IIMS by the following:

- *Digital Dashboard Command Post Software*, which is a data-fusion system providing a suite of applications designed to consolidate, display, and manage day-to-day data, Chemical-Biological contingency data, and related hazard data from sensors, reconnaissance reports, and hazard modeling.
- *A detection network*, which is established by using electronic signal control devices that provide a communication link and a computer interface to integrate dissimilar, remotely located devices (e.g., detectors, sirens, warning lights, GPS receivers, and meteorological sensors) into a common network.
- *Warning devices*, consisting of both audio systems and light systems that are used to disseminate alarm and environmental condition information.

*Tracker* is intended to extend the capabilities of the existing Digital Dashboard by allowing end users to easily extend description elements of the existing reporting tools that constitute a component of the Digital Dashboard, called the Electronic Activity Report Manager. Conveniently, the Digital Dashboard is also the component of choice for the integration of new tools. The Discussion section of this paper describes how *Tracker* was integrated as both a cell and as a stand-alone application to support IIMS.

## Tracker Overview

The *Tracker* prototype is one of the many active-forms tools that were developed as part of a research program sponsored by DARPA called Active Templates [6]. The *Tracker* software was developed to support template construction, and then to allow planners at different levels of the C2 structure to use the templates (make instances of them) to support crisis action mission planning. *Tracker* also provides an infrastructure to support the collaborative, distributed development and execution of plans. For example, *Tracker* has been successfully used to support a group of military planners in the generation of mission-planning folders that described the state, objectives, resource availability and other aspects of missions. During these experi-

ments, *Tracker* templates were used and modified to support mission requirements as the mission evolved [6]. Note, however, that for IIMS *Tracker's* collaborative infrastructure is not needed. IIMS needs only the flexible capabilities that *Tracker* provides for template (and instance) construction and modification.

A pallet of information element widgets is available in *Tracker* for use in the construction of a template. With this pallet of design widgets, the *Tracker* application has been successfully used to prototype templates to provide a variety of planning and execution contexts [7, 8]. Experiments with *Tracker* have demonstrated that the *Tracker* interface can enable a user who is not skilled in the development of a template/forms-based graphical user interface to rapidly design such an interface. If the interface is to be part of *Tracker*, then there is no further development, but if the interface is to be used in another application, *Tracker* provides a set of tools to allow the export of the template design to the other application. Of course, methods in the receiving application have to exist for that application to make use of the *Tracker* exported template(s).

At the time of *Tracker* inception, few tools were available that could provide the types of template creation, usage, and modification capabilities that are currently offered by *Tracker*. Other tools are starting to appear, though most are based on XForms [9], a World Wide Web Consortium (W3C) recommendation designed for forms on the web. XForms-based templating tools are more of a transformation of a graphical form into XML, and do not provide any of the widget libraries or other features that are now a part of *Tracker*. However, XForms does separate data, logic, and presentation, which frees the use and manipulation of data from platform constraints, and allows data to be accessed by any type of data browser—telephone, web site, etc. The XForms working group is moving toward making this a standard within the W3C.

Many of the tools that use XForms [10] provide support primarily for the browsing and viewing of existing forms, with less emphasis on the dynamic creation of forms. For example, the Oracle Application Server 10g Wireless Client [11] uses XForms only to display information on a variety of mobile devices for both online and offline form processing.

Another template/form engine is Microsoft's InfoPath. [12] InfoPath is a Microsoft Office based tool that can be used to generate dynamic, content-rich forms that can be shared and reused effectively. This tool provides additional capabilities beyond XForms, but without adhering to the W3C recommendation. InfoPath provides a common, well known, editing environment (the Microsoft Office environment). It also allows the quick creation, modification and completion of dynamic forms, making it a comparable tool to *Tracker*.

However, unlike *Tracker* (which is lightweight, inexpensive, and platform independent), InfoPath requires the purchase of the Microsoft Office software, which must be installed on a modern computer running the most current

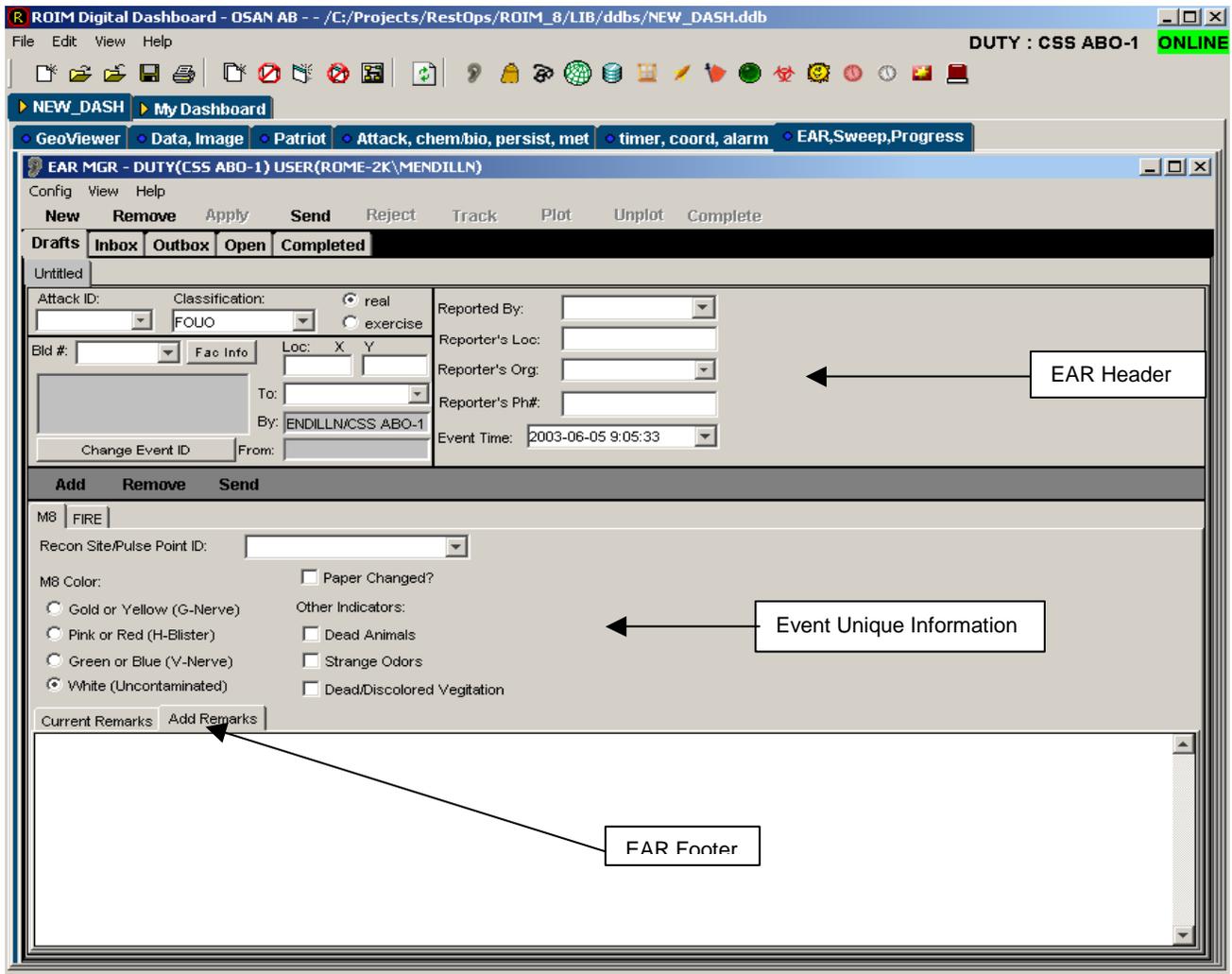


Figure 2: Example Electronic Activity Report (EAR)

Microsoft operating system and Microsoft Internet Browser [13]. Another difference from *Tracker* is that InfoPath forms cannot be viewed externally to the InfoPath client application. This prevents the use of InfoPath to generate or modify an existing form that would then be used inside of another application such as the IIMS Digital Dashboard.

### Premise and Goals

One of the existing tools available through the IIMS Digital Dashboard for data reporting is the Electronic Activity Report Manager. This application is a workflow-collaboration tool that provides a user-friendly interface for submitting, receiving, forwarding, and tracking Electronic Activity Reports (EARs). The fields of the reports have been specifically designed to address general EAR information as well as the specific information associated with a given category of incident response (see Figure 2).

IIMS Electronic Activity Reports are pre-defined and adhere to a database schema that cannot be readily changed

by end users. With *Tracker*, IIMS developers could allow end users to modify EAR templates to better to support their immediate “in-the-field” data reporting needs. *Tracker* would also allow end users who are disconnected from the central platform a mechanism for accessing and using IIMS templates. As templates were used and possibly modified, the users would then upload their completed template reporting data into the IIMS database. Any added data elements would be filtered and provided directly to an IIMS administrator for processing. The two direct advantages to this approach are that *Tracker* would allow the end users to enter the reporting data required by the EAR templates as well as any extra data that was of relevance to the incident. In this way, *Tracker* would provide a method to enable the IIMS system to “learn” about new data-element requirements, and to use this learned knowledge to support data-form modification.

Our experiment was driven by the following main goals:

- To convert *Tracker* to run as a cell within the IIMS Digital Dashboard software (DDS). Since IIMS was designed to provide a single environment for data collection, hav-

ing *Tracker* run as a cell within the DDS would minimize potential data integration problems for the end user.

- To use *Tracker* to enable the DDS to convert new and existing EARs into XML-based templates/instances. Providing the DDS with the ability to convert forms and templates into an XML data format would expand the ability of IIMS applications to communicate with XML-based applications, such as service based informational and analysis systems.
- To store *Tracker*-developed templates and Instances into the IIMS Oracle Database. The IIMS developers would need to build a method to later support the analysis of any added data-collection fields to determine whether they should become part of the permanent IIMS database schema.

## Procedures and Results

We attempted to integrate the IIMS Digital Dashboard software (DDS) with *Tracker*. Our intent was to add dynamic form generation capabilities to existing EAR forms to allow end users the ability to adapt the forms to match their data collection needs and not restrict the users to old, sometimes outdated data collection forms. Our experiment was sensitive to the fact the DDS is developed on a legacy Oracle database with a well-defined schema that is used by many vendors, and therefore the schema does not lend itself to frequent changes or updates. We needed to develop a method for using current and future schema-based data while also storing our new non-schema based template data in the database (where it could later be analyzed and used) thus allowing for a transparent integration between *Tracker* and DDS.

Because the IIMS Digital Dashboard was designed to be a large container of many different *cells*, the ideal imple-

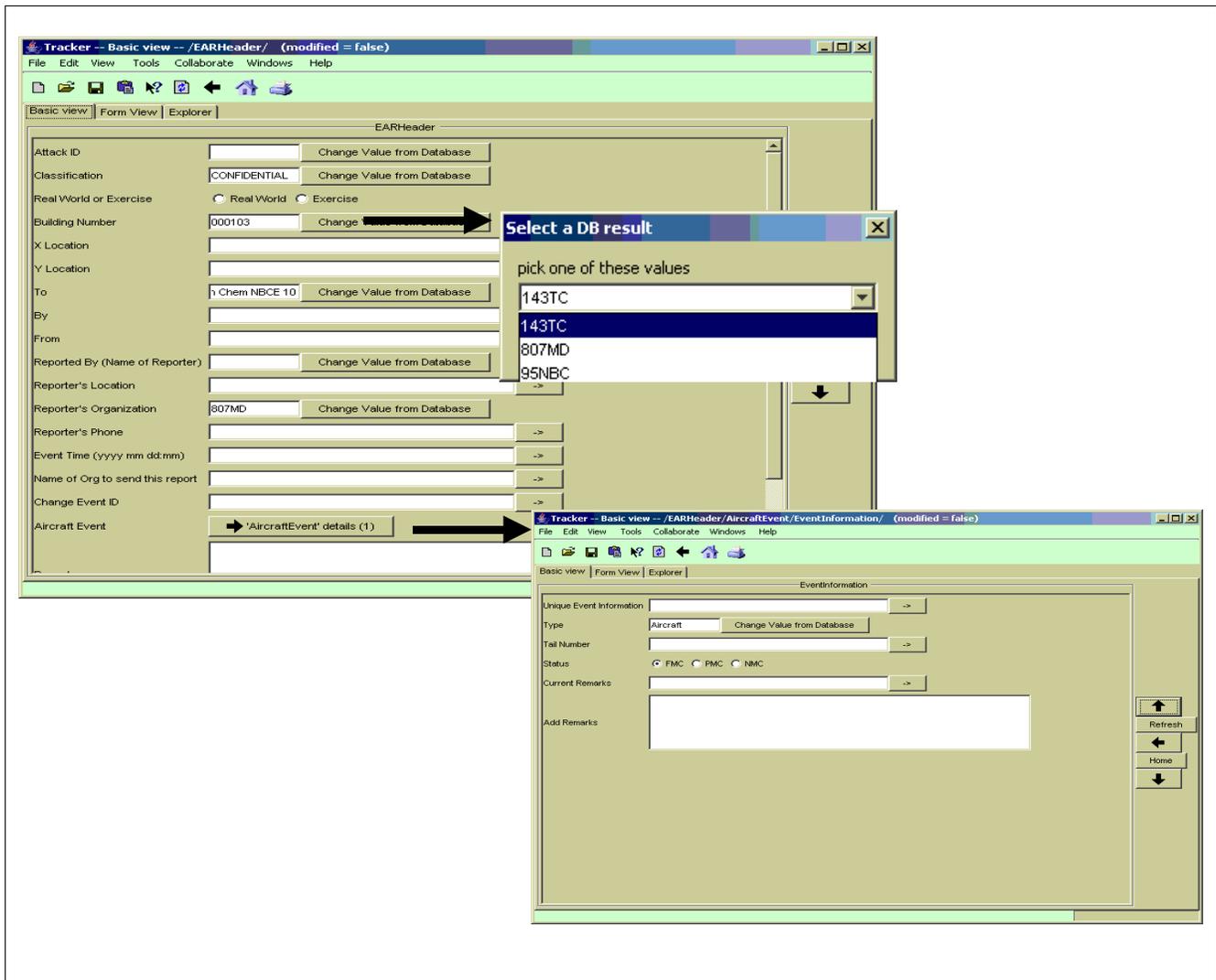


Figure 3: Tracker Version of an EAR Report with Database Calls and Sub-Templates

mentation would be to have *Tracker* run as a cell within the DDS. However *Tracker* was designed to run only as a stand-alone application. When we attempted to make the required modifications to allow *Tracker* to run as a cell we discovered that this effort would require quite a large extension to *Tracker*. For example, all of the window listeners attached to the menus and the forms widgets in *Tracker* would require modification in order to operate correctly as an IIMS cell. This would involve the separation of the display components into a Model View Controller (MVC) pattern that would centrally locate the listeners. Although the MVC pattern is a common design pattern in GUI software development today, *Tracker* was developed under another research paradigm and was not intended for extreme portability. Because of the overhead required to run *Tracker* as a Cell, we decided to instead integrate it as a Standalone application.

Another goal of the integration experiment was to use *Tracker* to display EAR reports. This goal could be satisfied with *Tracker* as a Cell or as a Standalone application. Our initial approach was to use *Tracker* to read the IIMS database records and automatically generate a set of templates from those records. Unfortunately EARs are comprised of multiple database records, and the automatic method did not apply. Instead, we opted to create a handful of templates to represent a set of EAR templates (see Figure 3). The generation of these EAR templates provided us with recommendations to IIMS developers on future template development and re-use. For example, although each EAR addresses a given incident category, each EAR contains a header that contains information types that are invariant across EARs. Therefore, the example EAR that is displayed in Figure 3 is comprised of several sub-templates (in that example the EAR header and the aircraft specific template). Once the EAR header template is defined with *Tracker*, it can then be re-used to develop other EAR templates. Additionally, we developed a custom database field (also displayed in Figure 3) that draws value options directly from the IIMS database.

Another *Tracker* feature that was used to develop EARs is the *Tracker Domain* functionality. Although *Tracker* was developed to allow users to define one or more templates dynamically in support of many problem solving contexts, our experience to date has indicated that certain form widgets form widgets (e.g., text, graphics, date/time, radio buttons, web calls) are often tailored to support a given problem domain, hence the *Tracker Domain* concept. When a Domain is loaded into *Tracker*, Custom widgets become available to support that problem context. For example, a form can be created in *Tracker* with a custom widget that performs a database lookup and displays a selection list based on the database values returned. *Tracker* will store the SQL as a parameter within the widget. This allows for complete freedom in form modification without requisite knowledge of the *Tracker* sources. However, for this widget to work correctly within another application, the widget knowledge of how to perform database lookups and building of a selection list must be transferable to the

new application. The *Tracker Domain* controls some of this background work for the user. This detailed knowledge is specific to the *Tracker* custom *Domain* and is not stored within the generated XML templates.

To allow *Tracker* access to the legacy database we created an Application Program Interface (API). The API was also developed to provide access to a new set of IIMS database tables that were developed to support the storage of templates developed with *Tracker*. Because these new tables communicate only with the DDS, they can exist outside of the larger existing IIMS database and not require changes to the legacy schema.

To store the *Tracker* templates into the database we experimented with separating the base template from any user-entered modifications. We did this in order to (1) build a table of templates that could be reused within IIMS, possibly as a cell, independent of *Tracker*, and (2) to implement a special table within the database that would be used to store any added or modified fields. This table could then be used by the DDS/IIMS administrator to track the frequency of any fields that were added or modified by users with the *Tracker* EAR forms. The theory was that the frequent addition of a given field indicated a requirement for a modification to the existing data entry form. It would be the job of the IIMS administrator to determine if this requirement should be implemented as a change to the IIMS database schema and associated existing legacy forms.

Due to limited time constraints, we decided to store the XML templates as a single entity in the database. Due to database schema requirements, we were required to store the entire template as a BFILE. This storage prevented template analysis via simple database queries, however we didn't intend to perform analysis on the original templates so we did not view this as an immediate issue. Instead, we assumed that future development would result in a template storage method that would facilitate parsing and analysis of templates using database calls.

When we attempted to store the template instances (templates with data and possible new or modified fields), we ran into several problems. First we discovered that separating templates and instances was not as easy as we originally expected. When *Tracker* stores an instance, the instance contains the complete template with the instance data plus any new and modified fields. We wanted to extract this information so that we could store the instance data in it's own table. We found that this would required the modification of *Tracker*. We needed to have a tag that indicated what template field the data applied. Without this tag, we were not able to re-assemble the template outside of *Tracker* with completed instance data.

While we did develop a method to tag both templates and template instances, our integration effort did not mature to the point where we could write this data directly to the IIMS database. More work would be required to meet this goal.

We also discovered that while separating template structure from the template data was the correct approach from

a database point of view, this design did not provide the required flexibility required to meet our experiment demands for data form flexibility. For example, we found that having the template structure in one table and the template data in another table did not easily accommodate new or modified fields. Future work would be required to develop a better, more flexible database design that allowed for the addition and modification of template fields, data and metadata, while also allowing for the easy reconstruction of template forms in applications outside of *Tracker*.

Finally, because existing *Tracker* widgets are tightly integrated into the templates and *Tracker* itself, we were not able to share them with IIMS (as a cell) as easily as we expected. Future work would require the development of an API that could allow easier exporting of the *Tracker* widgets to the IIMS library.

### Conclusion and Future Work

During our research we discovered that the concept of adding a dynamic templating capability to the IIMS Digital Dashboard would indeed provide a flexible interface for incident response operators to quickly adapt their forms and templates to changing data gathering requirements at a moments notice. Although we were not able to get the API between the two tools working sufficiently to test our hypothesis that IIMS would benefit from dynamic templating, we were able to demonstrate the concept through our work with the development of *Tracker* EAR templates. In addition, our work identified the need for additional modifications to both *Tracker* and IIMS.

By providing *Tracker*/IIMS with specific tables in the legacy database we discovered that we can collect the required information that would allow the creation of analysis tools that could discover frequently added data fields, thus providing an element of learning that could help to shape future data collection forms.

Our experiment with the integration of *Tracker* with IIMS indicated the benefits of both modifying *Tracker* templates to match a common format such as XForms, and providing for the separation of the widget library from the core of *Tracker*. First, by following a defined data standard such as XForms, *Tracker* templates could be integrated more easily into other software projects without affecting its adaptive, unstructured abilities. In some instances, integration might only require the existing templates and widget libraries to have full functionality.

Second, the separation of widgets from the core of *Tracker* would also improve the portability of *Tracker* towards non-desktop based implementations such as PDA or mobile telephone. Since mobile devices do not usually contain the full suite of graphical components, a specific library could be created to mimic the missing components on a device such as a PDA.

To allow the proper sharing of widgets, a widget library with a common API needs to be defined. This API would allow the development of multiple platform specific librar-

ies without the need to modify the core of *Tracker*. For example, if we used a design approach such as the *dependency of interjection approach* [14], we could have incorporated widget functionality in lightweight components. With this approach, swapping platform widgets libraries could be as simple as changing a parameter within a configuration file.

Our research indicates that the provision of unstructured, flexible data entry systems like *Tracker* can offer the end user the ability to modify and update templates that have schema-specific structure. Further efforts would require better APIs that support the integration of the new data elements with existing database schemas. Perhaps some of the research with evolving ontologies could help to support this type of capability.

### References

- [1] Air Force Research Laboratory News Release, "Information Directorate Participates in JWID 2004," July 23, 2004, *AFRL-R 04-86*, [http://www.rl.af.mil/div/IFO/IFO/IFOIPA/press\\_history/pr-04/pr-04-86.html](http://www.rl.af.mil/div/IFO/IFO/IFOIPA/press_history/pr-04/pr-04-86.html).
- [2] Dursteler, J. C., "Digital Dashboard," *InfoVis.net*, April 12, 2004, <http://www.infovis.net/printMag.php?num=143&lang=2>.
- [3] ServiceWare Executive Dashboard, <http://www.infoimage.com/solutions/executivedashboard.asp>.
- [4] Microsoft Digital Dashboard, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/modcore/html/decondigitaldashboardoverview.asp>.
- [5] Garamone, J., "Advanced Technologies Program is on the Battlefield," *American Forces Press Service*, 2003; <http://www.dau.mil/pubs/pm/pmpdf03/March/afps-0503.pdf>.
- [6] Mulvehill, A., "Authoring Templates With *Tracker*," *IEEE Transactions on Intelligent Systems*, 2005.
- [7] Mulvehill, A., Callaghan, M., Hyde, C., "Using Templates to Support Crisis Action Mission Planning," *2002 Command and Control Research Technology Symposium*, June 2002.
- [8] Mulvehill, A., Benyo, B., Rager, D., DePalma, E., "ACT – The Automated Clearance Tool: Improving the Diplomatic Clearance Process for AMC," *2004 Command and Control Research Technology Symposium*, June 2004.
- [9] *XForms - The Next Generation of Web Forms*: <http://www.w3.org/MarkUp/Forms/>.
- [10] *XForms Implementations*; <http://www.w3.org/MarkUp/Forms/#implementations>.
- [11] Oracle Corporation, *Oracle 10<sup>g</sup> Wireless Application Server Data Sheet*. Retrieved December 2004 from [http://www.oracle.com/technology/tech/wireless/mobilebrowser/OracleAS\\_Wireless\\_Client\\_DS.pdf](http://www.oracle.com/technology/tech/wireless/mobilebrowser/OracleAS_Wireless_Client_DS.pdf).

- [12] *Microsoft InfoPath Website*. Retrieved December 2004 from <http://office.microsoft.com/en-us/FX010857921033.aspx>.
- [13] Dubinko, Micha, (2003), *XForms and Microsoft InfoPath*; Retrieved December 2004 from <http://www.xml.com/pub/a/2003/10/29/infopath.html>.
- [14] Fowler, Martin. (2004), *Inversion of Control Containers and the Dependency Interjection Pattern*. Retrieved December 2004 from <http://martinfowler.com/articles/injection.html>.