AMAETHON - A WEB APPLICATION FOR FARM MANAGEMENT AND
AN ASSESSMENT OF ITS UTILITY

A Thesis
Presented to
the Faculty of California Polytechnic State University
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

by
Tyler Yero
December 2012
COMMITTEE MEMBERSHIP

TITLE: Amaethon - A Web Application for Farm Management and an Assessment of its Utility

AUTHOR: Tyler Yero

DATE SUBMITTED: December 2012

COMMITTEE CHAIR: Gene Fisher, Ph.D.

COMMITTEE MEMBER: Franz Kurfess, Ph.D.

COMMITTEE MEMBER: Alex Dekhtyar, Ph.D.
Abstract

Amaethon - A Web Application for Farm Management and an Assessment of its Utility

Tyler Yero

Amaethon is a web application that is designed for enterprise farm management. It takes a job typically performed with spreadsheets, paper, or custom software and puts it on the web. Farm administration personnel may use it to schedule farm operations and manage their resources and equipment. A survey was conducted to assess Amaethon’s user interface design. Participants in the survey were two groups of students and a small group of agriculture professionals. Among other results, the survey indicated that a calendar interface inside Amaethon was preferred, and statistically no less effective, than a map interface. This is despite the fact that a map interface was viewed by some users as a potentially important and effective component of Amaethon.
Contents

List of Tables vii
List of Figures viii

1 Introduction 1
   1.1 Description of the Problem . . . . . . . . . . . . . . . . . . . . 1
   1.2 Overview of the Solution . . . . . . . . . . . . . . . . . . . . . 2
   1.3 Focus and Limitations . . . . . . . . . . . . . . . . . . . . . . . 2
   1.4 Outline of the Thesis . . . . . . . . . . . . . . . . . . . . . . . . 3

2 Background and Related Work 4
   2.1 Related Work on HCI . . . . . . . . . . . . . . . . . . . . . . . . 4
   2.2 Similar Applications . . . . . . . . . . . . . . . . . . . . . . . . 5
   2.3 Brief Background of Amaethon Requirements Acquisition . . . . . 8

3 Scenario of System Use 9
   3.1 Map Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
   3.2 Calendar Interface . . . . . . . . . . . . . . . . . . . . . . . . . 13
   3.3 Table Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
   3.4 Other Resources . . . . . . . . . . . . . . . . . . . . . . . . . . 19
      3.4.1 Fields . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
      3.4.2 Crops and Varieties . . . . . . . . . . . . . . . . . . . . . . 21
      3.4.3 Equipment . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
      3.4.4 Facilities . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

4 Overall System Design 26
   4.1 SQL Database . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2 Website</td>
<td>28</td>
</tr>
<tr>
<td>4.2.1 Map</td>
<td>29</td>
</tr>
<tr>
<td>4.2.2 Calendar</td>
<td>30</td>
</tr>
<tr>
<td>4.2.3 Other Pages</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Web Service</td>
<td>34</td>
</tr>
<tr>
<td>5 User Feedback and Analysis</td>
<td>37</td>
</tr>
<tr>
<td>5.1 Survey Design</td>
<td>37</td>
</tr>
<tr>
<td>5.2 Quantitative Results of Map and Calendar</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Qualitative Feedback</td>
<td>40</td>
</tr>
<tr>
<td>6 Conclusion and Future Work</td>
<td>42</td>
</tr>
<tr>
<td>Bibliography</td>
<td>44</td>
</tr>
<tr>
<td>A Database Statements</td>
<td>47</td>
</tr>
<tr>
<td>B Amaethon Survey</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Features comparison with similar applications .................. 7

5.1 Map and calendar ease of use ................................. 40
5.2 Map and calendar use frequency ................................. 40
5.3 Amaethon as a web or desktop application ...................... 41
List of Figures

2.1 AgSquared map interface ................................................. 6
2.2 Farmlogs map interface .................................................. 7

3.1 The login screen of Amaethon ............................................. 10
3.2 Map screen with selected operation ..................................... 12
3.3 Map screen with add dialog ............................................. 13
3.4 Calendar screen in week mode .......................................... 14
3.5 Calendar screen showing drag and drop ................................ 15
3.6 Calendar screen in month mode with edit dialog ..................... 16
3.7 Operations table screen .................................................. 17
3.8 Filtering the operations table .......................................... 18
3.9 Fields screen .................................................................. 20
3.10 Plotting a field ............................................................. 21
3.11 Crops and varieties screen .............................................. 22
3.12 Variety creation .......................................................... 23
3.13 Equipment screen ......................................................... 24
3.14 Facilities screen .......................................................... 25

4.1 Fields screen ............................................................... 27
4.2 Object model of Amaethon ................................................. 28
Chapter 1

Introduction

Amaethon, named for the Celtic god of agriculture, is a web application to assist with farm management. Its primary audience is management personnel in medium to large farming enterprises. These people are not doing the farm work themselves, but rather they manage the operations of the farm from an office. They schedule operations, run reports, and make business decisions.

1.1 Description of the Problem

Running a farm enterprises involves making farm plans, gathering data about resources, generating reports from that data, and making decisions based on it. Many farmers struggle with staying on track with farm plans as the season progresses. The storage of historical data can also be problematic. Computers can help farmers with their job, but any software solution needs to make farm management easier and less time-consuming. Farmers do not want to struggle with computer software because they would rather be farming.
1.2 Overview of the Solution

Amaethon replaces spreadsheets and custom software with a web application designed specifically for the farming domain but generic enough to be useful to more than a single company. Its approach is useful and natural considering the expanding functionality and availability of the web [13]. Because the management personnel of farming enterprises are usually in an office with access to broadband internet, a web application makes sense as their primary tool.

Amaethon gives its users the ability to schedule and view farming operations and manage several different types of resources on the farm. These include farming equipment, crops, crop varieties, facilities, and fields. There are plans to add reporting services to Amaethon to give valuable output for farm management.

Amaethon has several types of user interfaces to visualize and organize farm operations. Users can view operations in a familiar spreadsheet table form, on a calendar, or geographically within a map.

1.3 Focus and Limitations

Amaethon’s current focus is on operation scheduling and resource management. It makes no reference to environmental management, water usage, accounting, or other features present in more specialized applications in this field of study. Amaethon is currently limited in scope and features and is considered a prototype for web applications in the domain of farm management. It includes operation scheduling and basic resource management, but the different kinds of farm resources it supports could be improved. Currently all types of operations have the same basic set of attributes stored about them. Ideally operations would
have more attributes associated with them, with certain attributes being specific
to different types of operations. For example, a plowing operation would have a
different set of information stored about it than a planting operation.

1.4 Outline of the Thesis

In Chapter 2, this thesis first explores the background and related work in the
field of study. Then in Chapter 3 the intended use of Amaethon is presented in
detail. Chapter 4 presents the overall system design and some excerpts of its im-
plementation. Chapter 5 explains the results and feedback of user studies. These
studies include an analysis of the map and calendar interfaces in Amaethon as well
as qualitative questions about the application. Chapter 6 presents conclusions
and future work.
Chapter 2

Background and Related Work

Computer-based information systems have been applied in assisting agriculture production for decades [21]. These solutions can range from spreadsheets to general Geographic Information Systems (GIS)/Remote Sensing packages to custom software [18]. Some customized systems are independent of GIS platforms while others exchange data with them.

The scope and purpose of these applications vary greatly. However overall, they seek to improve the business and efficiency of running a farm.

2.1 Related Work on HCI

One of the most challenging aspects of developing Amaethon was making its map interface as useful as it could be. This interface is shown in Figure 3.2 and will be discussed in detail later. Several usability metrics have been proposed for map interfaces [5, 20]. In addition to traditional metrics, eye movement recordings have been encouraged. Studies have shown that recordings help to
reveal the amount of cognitive processing a display requires and where these cognitive resources are required. This can establish how a display may or may not facilitate task completion [5].

The study conducted for Amaethon only included the traditional usability metrics of task time (to measure efficiency and productivity) and test level satisfaction (for overall ease of use).

One of the main potential clients of Amaethon expressed significant interest in a map feature for Amaethon [17]. This feature seemed like it would be very useful given the geographic nature of farming. However, as the results chapter indicates, it is seemingly no more useful than a calendar interface for the specific usage study presented in this thesis.

2.2 Similar Applications

There are several similar applications to Amaethon already on the web. Some are aimed at smaller specific farms, and others are designed to be more general. Two applications that are very similar to Amaethon in scope and purpose are FarmLogs and AgSquared. Both of these applications are hosted on the web and both feature map and calendar components [8, 1].

AgSquared is an application developed in 2012 that aims to replace spreadsheets for small farmers and enthusiasts [3, 1]. In contrast to Amaethon, it does not contain any mention of farming equipment, crop varieties, or different types of farm operations. However, it has a map interface that includes outlines around the buildings and fields that the user has entered. Clicking an object on the map brings up basic information about it but does not associate it to any operations.
AgSquared does offer reports to its users to help them track the statistics of their farm over time.

FarmLogs is another application that streamlines record-keeping and task management [8]. It includes farm equipment and implements, as well as a detailed calendar interface very much like Amaethon’s. Its map interface is somewhat limited because it currently only shows markers at the center of each field instead of an outline around the field. It also contains detailed attributes for each type of operation, this is especially useful and should be included in future versions of Amaethon. FarmLogs also offers reporting services to its users.

Figures 2.1 and 2.2 show samples of the map interface in the AgSquared and FarmLogs applications. Table 2.1 compares specific features of Amaethon and the other two applications.

![Figure 2.1: The map interface of AgSquared.](image)
As the table shows, the three applications have very comparable features. If Amaethon were to be introduced into the marketplace, it would require the addition of a reports feature and thoughtful marketing that emphasizes the superiority of its user interface in important areas.
2.3 Brief Background of Amaethon Requirements Acquisition

The requirements for Amaethon were first gathered with an in-person meeting with a potential client [17]. After this, a combination of email correspondence and in-person meetings were used to create mockup diagrams of the interface and to make design decisions about the data model.

An iterative approach was used to create Amaethon and, after the first few meetings, a live prototype of Amaethon itself was used as a platform to discuss new and existing features. The original mockups do not therefore provide any useful insight into the end-user requirements that is not in the existing operational interface. Hence the end-user scenarios in chapter 3 provide both a description of the final interface, as well as use cases of functional requirements.

The requirements scenarios of Chapter 3 also present the end-user view of the underlying Amaethon data model. These requirements are represented directly in the end-user interface. During the requirements acquisition process, it was not considered necessary to use any other form of user-level data modeling, such as a data dictionary or other tabular summaries of the data.
Chapter 3

Scenario of System Use

Amaethon is accessed by its users via a web browser. After entering their user credentials on the login page (Figure 3.1), they are logged into the website. The same website has the potential to host many different farming companies. A user is associated with a certain company, so upon logging in they will only view and manage data from their farm. They may schedule or view operations, manage their resources (crops, crop varieties, fields, and equipment), or manage the users of the system at their company if they are an administrator.
Figure 3.1: The login screen of Amaethon.

An operation is a task or activity to be performed on the farm. Currently, an operation includes:

- An operation type
- Starting time
- Ending time
- Field to be performed on (optional)
- Crop (optional)
• Equipment used (optional)
• Facility used (optional)
• Notes (optional text)
• A ’Completed’ status

The current operation types are planting, harvesting, tilling, plowing, fertilizing, and spraying. The fields, crops, and equipment that can be included in operations are created by the user on their respective screens as explained later in this chapter.

Scheduling operations is the most common task performed in Amaethon. There are three different interfaces available to the user to schedule and view operations: a map, calendar, and table. The same functionality is available in each interface but in different ways.

3.1 Map Interface

On the ’Map’ page, one can overlook a geographic representation of the farm. Using a Google Maps component, the fields of a user’s farm are plotted with green polygons. These plots can be irregular shapes and are created by the user as one of the attributes of a field. When a field is clicked, an information box will appear above that field. This also happens if an operation that features that field is clicked in the list of operations on the left of the screen, as Figure 3.2 shows.
Figure 3.2: The map screen showing a selected operation and the corresponding field information box.

An operation can also be created or edited with a dialog that appears when a user clicks the '+ New Operation' button or double clicks on an operation in the list on the left, as shown in Figure 3.3. The contents of this dialog are the same attributes of an operation that were listed at the beginning of this chapter.
Figure 3.3: The map screen showing the dialog to add a new operation.

3.2 Calendar Interface

The user can also view upcoming or past operations on the 'Calendar' page. Many applications feature calendars for time-dependent data, so users are likely to be familiar with this type of interface. The calendar has three modes: month, week, or day. The calendar defaults to a week view, as shown in Figure 3.4.
Figure 3.4: The calendar screen in week viewing mode.

The operations are plotted onto the calendar with different colors. Each color represents a different type of operation. On each operation there is a label that describes the field and equipment used so a user can distinguish the operations at a glance.

On the left side of the screen there is a panel that contains a legend that matches the colors with their respective operation type. There is also a small set of filters that a user can use to limit the operations that are shown on the calendar.

As Figure 3.5 shows, the operations can be selected and dragged to change
their start and end time(s). They may also be resized to change their duration.

Figure 3.5: The calendar screen in month viewing mode demonstrating an operation being dragged to a different date.
3.3 Table Interface

A user may also manage operations using a table interface on the 'Operations' page. The columns of the table represent all the attributes of an operation and the rows are each record. This puts the information in a familiar spreadsheet-like format.
Figure 3.7: The operations table screen listing the operations and their attributes. This example shows the operations being sorted by their start times in an ascending order (soonest to latest).

Figure 3.7 represents a sample viewing of the table interface screen. There is a filter icon on each column header that a user may press to add their own constraints to each attribute to limit the operations shown, demonstrated by Figure 3.8. The columns may also be sorted by clicking on the column header’s title.
Figure 3.8: The operations table screen showing the ability to filter which operations are shown. The green color around the 'Completed' column's filter indicates that a filter is currently being applied to that attribute. The Start Time filter is opened and is being set to the first of the month.

The right-most column contains two buttons for every row, an edit button and a delete button. Pressing on the edit button will bring up an edit dialog for that particular record. Pressing on the delete button will display a confirmation dialog to delete that individual record.

The 'Completed' column contains a checkbox to signify whether that particular operation has been marked as complete by farm administration. A user may toggle the checkbox and change the completed status of that particular operation.
without bringing up the edit dialog. This is to facilitate the task of marking an
operation as completed.

3.4 Other Resources

The crops, fields, equipment, and users of the farm are managed on their own
pages with a table interface. However, filtering has not been implemented for any
of these tables.

3.4.1 Fields

Fields are an important entity to model correctly. They contain several at-
tributes, including their geographic plots as mentioned earlier. Their plot can be
edited with a second dialog after opening the initial edit dialog for a particular
field. See Figure 3.10.

Fields also have attributes for their currently planted crop, varieties of that
crop, and the planting date. Currently, these attributes are set directly by the
user from this screen. A useful enhancement would be to make them a computed
attribute that is based on the operations scheduled for that field. So instead of
setting the crop of the field manually, the system would know which crop was on
that field based on a previous planting operation.

Also included are the fairly static attributes of gross acreage, plantable acreage,
and root zone depth. These do not change unless there is crop rotation or re-
structuring of the fields.
Figure 3.9: The fields screen listing the fields and their attributes.
Figure 3.10: The fields screen showing the plot dialog for editing the shape of a field. The user uses the tools within the map area to draw the desired shape.

3.4.2 Crops and Varieties

Crops and crop varieties share a screen. This was done because each entity does not have many attributes and because these two entities are related to one another. A crop variety is meaningless without a crop associated with it. When a new crop variety is created, the user can associate that crop variety with a particular crop, as shown in Figure 3.12
Figure 3.11: The crops screen listing the crops and their varieties.
Figure 3.12: An example of creating a variety called 'Vidalia’. This is an onion variety so Onions will be selected as the crop.

3.4.3 Equipment

Many farm operations require the use of specialized farming equipment. Amaethon provides functionality to allow users to create their own equipment and types of equipment. These functions are shown in the same screen because of their close association. See Figure 3.13. This is another area that will likely need enhancement in a future version of Amaethon. Currently, there is no means to create farming implements that can be attached to a piece of equipment. For instance, a sprayer is an implement that can be attached to a tractor to spray chemicals
onto crops.

Figure 3.13: The equipment screen listing the equipment and the types of equipment.

3.4.4 Facilities

Facilities are a resource meant to include turnouts, pumps, booster stations, reservoirs, or other stationary pieces of equipment. These usually store or transport water in some way. This is an underdeveloped part of Amaethon. There is no 'Irrigation' type of operation, but there are plans for one in the future. Irrigation management is a complex subset of farm management that has many detailed
requirements [4]. Currently, facilities are included in Amaethon for completeness but their role is limited.

Figure 3.14: The facilities screen listing the facilities and the types of facilities.
Chapter 4

Overall System Design

The application consists of three main parts: a website, web service, and SQL database. All three work together to provide responsive data output and a clean experience for the user. The whole application is hosted in Windows Azure, Microsoft’s cloud platform [23]. The website runs in the user’s browser, but the web service and database are in Windows Azure. This means that all application data is stored in Windows Azure servers.

Figure 4.1 provides a diagram of the overall architecture of Amaethon.
Figure 4.1: The overall architecture of Amaethon. Arrows represent the flow of data.

4.1 SQL Database

All of the persistent data of Amaethon is stored in Windows Azure’s SQL Database. SQL Database is a database platform offered as a service. It is built using SQL Server technology [23]. Thus, Amaethon stores its data in a relational environment.

The schema is fairly simple and will likely change significantly if Amaethon is to become a commercially viable application. For the purposes of academic work, the schema is as shown by Figure 4.2. The SQL statements used to create the database are in Appendix A.
4.2 Website

The only interface to Amaethon is its website. The website is designed for full-screen desktop computers, and may work satisfactorily with a smaller tablet device. The interface is not designed with specific features to make it convenient for use on a smaller portable device like a smartphone. The simple horizontal navigation at the top of the screen attempts to maximize the content area in the
The login screen uses ASP.NET’s Forms Authentication [2]. This same authentication also governs access to the web service discussed later. Once logged in, the browser is redirected to the map screen.

Amaethon makes heavy use of AJAX, which is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. [10] JavaScript is employed on the client-side to asynchronously request data after the initial page is finished loading. Once any necessary additional data has been retrieved, the JavaScript controls manipulate the Document Object Model (DOM) of the webpage to construct its final form [10].

For example, on the fields page, see Figure 3.9, the initial page contains little markup and is loaded quickly. However, the grid control in the middle has no data and appears empty. The grid immediately requests data from Amaethon’s web service to fill itself. Once the data is retrieved new rows will dynamically be added to the grid to display the information. This technique allows the entire page to be responsive and fluid even as the grid is requesting its data.

Most of the UI controls used in Amaethon are from a JavaScript framework called Kendo UI. This framework is sold by Telerik [16]. Amaethon also makes use of jQuery, a popular JavaScript library [14].

4.2.1 Map

The map screen contains a Google Maps component on the right. This map is fixed to the right-hand side of the page and scales with the browser window. It uses Google Map’s API to overlay the colored polygons on the fields of the farm [11]. The information box that appears when clicking on a field is also from
this API. The map defaults to the 'Satellite' mode, which means that it uses
real images of the area. However, these images will probably be out of date and
could cause confusion. A road map or geographic map could also be made the
default viewing mode [11]. During development, the default mode of the map
was considered and potential clients indicated that they preferred the satellite
mode [17].

On the bottom left of the screen there is a small table that contains a list of
recent farm operations. This is a Kendo UI Grid control and this control is used
extensively in Amaethon. The panel on the left of the screen also has several
Kendo UI controls to add filter parameters to the web service call that is used by
the map and recent operations table. An example of this mechanism is shown in
the Calendar section.

4.2.2 Calendar

The calendar screen contains a JavaScript calendar control on the right. This
calendar is fixed to the right-hand side of the page and scales with the browser
window. It is from FullCalendar, an open-source calendar plugin for websites
[9]. The calendar’s API provides callbacks for different events that occur on the
calendar, like dragging and dropping, and requests for new operations to plot
when the user changes the viewing period. The calendar only fetches data for
the date range that it needs.

The panel on the left of the screen uses more Kendo controls to add filters
parameters to the web service call used by the calendar. The following code is
the definition of the drop-down list that the user sees in the filter panel:

```javascript
('#operationTypesFilter').kendoDropDownList({
```
When the value of this control is changed, a filter will be added and immediately applied to the calendar. This means that an asynchronous call to the web service will immediately be made. The call will only return data that fits the constraints of all filters that have been applied. This same mechanism is used on the map screen.

### 4.2.3 Other Pages

All other pages of the website use Kendo UI Grid controls to display tables of information and the corresponding dialogs to edit their records. Every Grid control uses a Kendo DataSource object to interface with the web service. The following is the definition of a DataSource object used by the grid that displays
the farm’s fields:

```javascript
var fieldsDS = new kendo.data.DataSource({
    transport: {
        read: {
            type: 'POST',
            url: '/API/Fields',
            contentType: 'application/json',
            dataType: 'json'
        },
        update: {
            type: 'PUT',
            url: '/API/Fields/Field',
            contentType: 'application/json',
            dataType: 'json'
        },
        destroy: {
            type: 'DELETE',
            url: '/API/Fields/Field',
            contentType: 'application/json',
            dataType: 'json'
        },
        create: {
            type: 'POST',
            url: '/API/Fields/Field',
            contentType: 'application/json',
            dataType: 'json'
        },
        parameterMap: function (options, type) {
            if (type === 'update' || type === 'create' || type === 'destroy') {
                options.CropVarieties = selectedVarieties;
            }
        }
    }
});
```
```javascript
options.PlotString = fieldPlotString;
}
return JSON.stringify(options);
}
},
schema: {
total: 'Total',
data: 'Data',
model: {
id: 'FieldId',
fields: {
  Name: { type: 'string', validation: { required: true } },
  GrossAcres: { type: 'string', validation: { min: 0 } },
  PlantedAcres: { type: 'string', validation: { min: 0 } }
},
Crop: {
  defaultValue: { CropId: '' },
  CropId: { type: 'string' },
  Name: { type: 'string' }
},
CropPlantDate: { defaultValue: null, type: 'date' },
SoilType: {
  defaultValue: { SoilTypeId: '' },
  SoilTypeId: { type: 'string' },
  Name: { type: 'string' }
},
RootZoneDepth: { defaultValue: null, type: 'number', validation: { min: 0 } },
PlotString: { type: 'string' },
CropVarieties: [
```
The Transport object within the DataSource specifies the settings for loading and saving data. The schema defines the expected JSON schema to be received and uploaded to the web service [15]. There is also a property to specify the default page size for the table. This means that the web service will only return 25 records at the most for one request. This same design is used throughout the Equipment, Crops, and Users pages.

4.3 Web Service

A RESTful ASP.NET web service was written using C# to provide a useful API to the website. All of the AJAX on the website uses this API to manage data. This includes creating, reading, updating, and deleting records (CRUD). The web service focuses on having meaningful URIs for every resource [19]. The HTTP verb being used in the request dictates which operation is being performed.
Thus, the same URL is used for all CRUD operations for a given resource and the verb tells the service which operation to perform. The following example is for reading all of the farm’s scheduled operations:

GET 'API/Operations'

The verb being used is the HTTP GET, and the URL is ’API/Operations’. To read a single operation, the following is used:

GET 'API/Operations/Operation/id'

The unique ID of the operation must be in the request to return the correct operation. Using this system, the POST verb is used for creation, PUT for updating, DELETE for deletion, and GET for reading. This same structure is used for every entity stored in Amaethon.

To interface with the SQL database, the web service uses Microsoft’s Entity Framework (EF) [6]. This is an object-relational mapping tool that can generate data access objects in C# from an existing SQL database. The following is an example of its use in the Create method for the Crop entity.

```csharp
public CropDTO Create(CropDTO item) {
    item.CropId = Guid.NewGuid();
    item.UserId = Membership.GetUser().ProviderUserKey as Guid?;

    using (var db = new AmaethonEntities()) {
        db.Crops.Add(ReverseMap(db, item));
        db.SaveChanges();
    }

    return item;
}
```
The 'AmaethonEntities' class is generated by EF. It enables the program to query a database and group together changes that will then be written back to the store as a unit. In this case, the only change is the addition of one new Crop entity.

The Create method does not return an object created by EF, but rather maps it to a data transfer object (DTO). The DTOs are simple and contain no business logic. These objects are turned into JSON when sent over the network. Here is an example of a crop object that the website would ultimately receive from the Create function:

```json
{
   'CropId': 'abbbe2b3-80dd-4a2f-867c-54be34fd5d26',
   'Name': 'Onions'
}
```

The map, calendar, and every dialog and table in Amaethon uses this API for their data needs. The way this service is designed keeps the requests consistent across all pages of the application and for every type of resource.
Chapter 5

User Feedback and Analysis

An electronic survey of Amaethon was taken by 32 people. The populations were 14 computer science undergraduate students, 15 agri-business undergraduate students, and 3 people currently working in the farming industry. The survey consisted of two main parts, and its goal was to provide a summative evaluation of the usability and effectiveness of Amaethon. The complete survey is presented in Appendix B.

5.1 Survey Design

The survey included a brief introduction to Amaethon and its features. Instructions were given on how to complete the tasks that the users were asked to perform. These included exposition about the details of the map and calendar interfaces interlaced with screenshots.

The first part of the survey focuses on the utility of the map and calendar interfaces. Maps are becoming more ubiquitous in software applications [12],...
and one of the main points of evidence-based HCI research is to question if an interface type is actually useful [7]. This study does not attempt to answer if maps are useful in general or even in farming applications. Rather, it asks the specific question: is the map interface useful in Amaethon? As noted earlier in Chapter 2, a map interface is a seemingly natural fit for Amaethon. Given this, it could be reasonable to expect that it would be a useful form of interface. This expectation was the motivation of the experiment to compare the utility of the calendar and map interfaces. This study included the traditional usability metrics of task time (to measure efficiency and productivity) and test level satisfaction (for overall ease of use).

The second part of the survey was simply a questionnaire that asked the users qualitative questions about Amaethon and its features. The survey questions were refined with the assistance of a statistical consultant in the Cal Poly Statistics Department [22].

5.2 Quantitative Results of Map and Calendar

All survey users were asked to schedule 10 operations using the calendar interface and 10 operations using the map interface. The time needed to complete that task for each interface was recorded by the user. Thus, concrete results were obtained for just how long it actually takes people to use one interface versus the other. In hindsight, the times should have probably been gathered programmatically to be more precise and because some users forgot to note their times.

The users were divided into two groups, computer science students in group 1, and agri-business students and industry personnel in group 2. Each group of
users was further stratified to have half of them do the task using the calendar first and the other half using the map first. This is to mitigate the variance that comes from subjects benefiting from their experience with the first interface they used.

Over the two groups, the average task completion times were 10 minutes 45 seconds for the map interface and 10 minutes and 1 second for the calendar interface.

A General Linear Model - Analysis of Variance was done on this data. The two factor model adjusts for the difference of which group the person was in. The primary quantitative analysis was performed on the following null hypothesis: *the mean amount of time to complete the task is the same for the map and calendar interfaces*. With a p-value of .335, we fail to reject this hypothesis. That is, there is no statistical difference in the time it takes to complete the experimental scheduling tasks using the map versus calendar interfaces.

Also, two qualitative questions were asked about the map and calendar interfaces:

- Which screen was easier to use to schedule and modify operations?
- Which screen do you think you would use more often?

The results are shown in Figures 5.1 and 5.2. The result set is too small to perform a meaningful chi-squared test, however they offer some insight into the preferences of users.
<table>
<thead>
<tr>
<th></th>
<th>Group 1 (Comp. Sci.)</th>
<th>Group 2 (Ag)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>14.3% (2)</td>
<td>33.3% (6)</td>
</tr>
<tr>
<td>Calendar</td>
<td>71.4% (10)</td>
<td>38.9% (7)</td>
</tr>
<tr>
<td>Both the same</td>
<td>14.3% (2)</td>
<td>27.8% (5)</td>
</tr>
</tbody>
</table>

Table 5.1: Which screen was easier to use to schedule and modify operations?

<table>
<thead>
<tr>
<th></th>
<th>Group 1 (Comp. Sci.)</th>
<th>Group 2 (Ag)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>14.3% (2)</td>
<td>22.2% (4)</td>
</tr>
<tr>
<td>Calendar</td>
<td>78.6% (11)</td>
<td>55.6% (10)</td>
</tr>
<tr>
<td>Both the same</td>
<td>7.1% (1)</td>
<td>22.2% (4)</td>
</tr>
</tbody>
</table>

Table 5.2: Which screen do you think you would use more often?

### 5.3 Qualitative Feedback

Because the survey had a qualitative section with open-ended questions, many suggestions from users were documented. The most common suggestions, with the number of participants who suggested them in parentheses, were:

- The ability to delete operations from the calendar and map screen. (15)
- Sorting and filtering abilities in every table in the application. (8)
- Allowing the page size for all tables to be changed by the user at will. (4)
- Adding and modifying fields (and their plots) from the map screen. (5)
- Making the ‘Filters’ region of the map and calendar page collapsible. (3)
- Allowing fields to have sub-regions defined within them upon which operations can be scheduled. (2)
- User-customizable attributes for all entities. (Fields, crops, etc.) (2)
• Add reports for past operations and crop yields. (All historical data) (2)

Fortunately, most of these changes are relatively small. Some will require changes to the data model, as discussed earlier.

<table>
<thead>
<tr>
<th></th>
<th>Group 1 (Comp. Sci.)</th>
<th>Group 2 (Ag)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>71.4% (10)</td>
<td>38.9% (7)</td>
</tr>
<tr>
<td>Desktop</td>
<td>7.1% (1)</td>
<td>38.9% (7)</td>
</tr>
<tr>
<td>No preference</td>
<td>21.4% (3)</td>
<td>22.2% (4)</td>
</tr>
</tbody>
</table>

Table 5.3: Would you prefer to use Amaethon in its current form as a web application, or instead as a desktop application?
Chapter 6

Conclusion and Future Work

Web applications like Amaethon could represent the future for farm management. The ability to use multiple devices and be in different locations while managing a farm makes web applications like this a good choice for farm managers. The contributions of this thesis include:

- Providing the prototype for a potentially very useful application
- A focused study on the utility of a map interface in such an application
- A qualitative study of the utility of such an application

While limited as a prototype, with a few extra features Amaethon could definitely be used by some enterprises with a Software as a Service (SaaS) business model. Amaethon can fulfill the goal to simplify operation scheduling, resource management, and reporting.

Geographic map interfaces have potential in this domain, but must be designed in a way that focuses on usability and productivity for its users. While
maps may seem to be a natural fit for a tool like Amaethon, the results of this study did not show that a map interface is more useful than a calendar.
Bibliography


Appendix A

Database Statements

CREATE TABLE [dbo].[SoilType]( [SoilTypeId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL )

CREATE TABLE [dbo].[OperationType]( [OperationTypeId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL )

CREATE TABLE [dbo].[FacilityType]( [FacilityTypeId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL )

CREATE TABLE [dbo].[EquipmentType]( [EquipmentTypeId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL )

CREATE TABLE [dbo].[Equipment]( [EquipmentId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL, [EquipmentTypeId] [uniqueidentifier] NULL, [UserId] [uniqueidentifier] NULL )

CREATE TABLE [dbo].[Facility]( [FacilityId] [uniqueidentifier] NOT NULL, [Name] [nvarchar](120) NOT NULL, [FacilityTypeId] [uniqueidentifier] NULL, [UserId] [uniqueidentifier] NULL )
CREATE TABLE [dbo].[CropVariety](
    [CropVarietyId] [uniqueidentifier] NOT NULL,
    [Name] [nvarchar](120) NOT NULL,
    [CropId] [uniqueidentifier] NULL,
    [UserId] [uniqueidentifier] NULL)

CREATE TABLE [dbo].[Crop](
    [CropId] [uniqueidentifier] NOT NULL,
    [Name] [nvarchar](max) NOT NULL,
    [UserId] [uniqueidentifier] NULL)

CREATE TABLE [dbo].[Field](
    [FieldId] [uniqueidentifier] NOT NULL,
    [Name] [nvarchar](120) NOT NULL,
    [Plot] [geography] NULL,
    [PlantedAcres] [decimal](18, 4) NULL,
    [GrossAcres] [decimal](18, 4) NULL,
    [CropId] [uniqueidentifier] NULL,
    [SoilTypeId] [uniqueidentifier] NULL,
    [RootZoneDepth] [int] NULL,
    [CropPlantDate] [datetime2](7) NULL,
    [PlotString] [nvarchar](max) NULL,
    [UserId] [uniqueidentifier] NULL)

CREATE TABLE [dbo].[FieldCropVariety](
    [FieldId] [uniqueidentifier] NOT NULL,
    [CropVarietyId] [uniqueidentifier] NOT NULL)

CREATE TABLE [dbo].[Operation](
    [OperationId] [uniqueidentifier] NOT NULL,
    [OperationTypeId] [uniqueidentifier] NOT NULL,
    [StartTime] [datetime2](7) NULL,
    [EndTime] [datetime2](7) NULL,
    [Completed] [bit] NOT NULL,
    [Notes] [nvarchar](max) NULL,
    [FacilityId] [uniqueidentifier] NULL,
    [FieldId] [uniqueidentifier] NULL,
    [UserId] [uniqueidentifier] NULL)
Appendix B

Amaethon Survey
Amaethon is a web application designed to assist farmers and farm management personnel with farm record keeping. After you read a bit about how Amaethon works, you will be participating in an experiment to study Amaethon's user interface. The experiment involves scheduling farm operations like plowing fields and harvesting crops.

In a separate browser window, please browse to: http://amaethon.cloudapp.net

Login with the username and password that you were given.

As you can see in the Amaethon window, you start out in Amaethon on the Map page. There are links at the top of the page to browse to different pages of Amaethon. The two links you will be following in the experiment are Map and Calendar.

The Map page consists of a map on the right side of the screen. This map contains shapes of the fields. If you click on one of the shapes, more information is displayed about that field.

On the left side, there is a grid containing recent operations and some filters to limit which operations are shown.
If you click a row in the operations list in the bottom left, then the field associated with that operation will display an information box. If you double click an operation in the list a window will appear to allow you to edit that operation. You may add new operations by using the link in a field's information box or by clicking the '+ New operation' button in the top left corner of the screen.
This dialog window allows you to add or update a single operation. To edit the type, field, or equipment - click on the drop down lists and select the appropriate item.

The start and end times can be changed by clicking on the button to the right or typing a new date into the box. Note: this is the dialog window you will be using to add operations during the experiment.

Please go to the calendar page. This page has a calendar on the right and filters on the left. Operations are plotted on the calendar. These operations can be moved to a different date by clicking and dragging them. An edit window will appear for an operation if you click on it in the calendar.
This is the week view of the calendar. You can add a new operation by using the button in the top left corner of the screen or simply click and drag with your cursor on an empty part of the calendar.
You may switch to a monthly view using the buttons at the top right of the screen. You can also click and drag an operation to different times on the calendar.

You will see specific instructions for the experiment when you press the 'Next' button below. Before you start the experiment, please take a moment to familiarize yourself with the map and calendar screens. You can click around on the screens, and add a few operations if you wish. Don't worry about deleting them.
I am going to ask you to perform two sets of scheduling tasks, one set using the map page and one set using the calendar page. Each set is explained on the following pages.

Upon completion of these tasks, I will ask you to reflect on and assess your ease of use in using the map and calendar scheduling tools in Amaethon.

Let's get started.
1. Please enter your starting time here now (including seconds).

Using the Map page, please schedule the following 10 operations:

Plowing on November 20th from 12pm to 1:30pm on field F-100 with Tractor 1
Plowing on November 20th from 1:30pm to 3pm on field F-101 with Tractor 2
Fertilizing on November 21st from 8am to 12pm on field F-102 with Tractor 1
Fertilizing on November 22nd from 8am to 12pm on field F-103 with Tractor 1
Spraying on November 22nd from 8am to 9am on field F-104 with Sprayer 1
Spraying on November 22nd from 9am to 10am on field F-105 with Sprayer 1
Harvesting on November 23rd from 6am to 12pm on field F-106
Harvesting on November 24th from 6am to 12pm on field F-107
Harvesting on November 25th from 6am to 12pm on field F-108
Harvesting on November 26th from 6am to 12pm on field F-109

Now modify these 4 operations:

Fertilizing on November 22nd from 8am to 12pm on field F-103 with Tractor 1 should change to
Fertilizing on November 22nd from 8am to 12pm on field F-103 with Tractor 3

Spraying on November 22nd from 9am to 10am on field F-105 with Sprayer 1 should change to
Spraying on November 23rd from 9am to 10am on field F-105 with Sprayer 1

Harvesting on November 25th from 6am to 12pm on field F-108 should change to
Harvesting on November 25th from 6am to 11am on field F-108

Harvesting on November 26th from 6am to 12pm on field F-109 should change to
Harvesting on November 26th from 6am to 11am on field F-109

2. Please enter your finishing time here now (including seconds).
3. Please enter your starting time here now (including seconds).

Using the Calendar page, please schedule the following 10 operations:

Plowing on December 3rd from 12pm to 1:30pm on field F-100 with Tractor 1
Plowing on December 3rd from 1:30pm to 3pm on field F-101 with Tractor 2

Fertilizing on December 4th from 8am to 12pm on field F-102 with Tractor 1
Fertilizing on December 5th from 8am to 12pm on field F-103 with Tractor 1

Spraying on December 5th from 8am to 9am on field F-104 with Sprayer 1
Spraying on December 5th from 9am to 10am on field F-105 with Sprayer 1

Harvesting on December 6th from 6am to 12pm on field F-106
Harvesting on December 6th from 6am to 12pm on field F-107

Harvesting on December 7th from 6am to 12pm on field F-108
Harvesting on December 7th from 6am to 12pm on field F-109

Now modify these 4 operations:

Fertilizing on December 4th from 8am to 12pm on field F-102 with Tractor 1 should change to
Fertilizing on December 4th from 8am to 12pm on field F-102 with Tractor 3

Spraying on December 5th from 9am to 10am on field F-105 with Sprayer 1 should change to
Spraying on December 6th from 9am to 10am on field F-105 with Sprayer 1

Harvesting on December 6th from 6am to 12pm on field F-106 should change to
Harvesting on December 6th from 6am to 11am on field F-106

Harvesting on December 7th from 6am to 12pm on field F-108 should change to
Harvesting on December 7th from 6am to 11am on field F-108

4. Please enter your finishing time here now (including seconds).
5. How easy to use is the map screen?
- Very easy
- Moderately easy
- Slightly easy
- Not at all easy

6. How helpful is the map screen when scheduling and modifying operations?
- Extremely helpful
- Very helpful
- Moderately helpful
- Slightly helpful
- Not at all helpful

7. Please rank the following map features in order of importance

- Ability to add operations from the map screen
- Having field outlines and field information boxes on the map
- Opening a field's information box when clicking an operation in the list
- Filtering the operations list on the map screen

8. Please add any comments you have about the map screen.
9. How easy to use is the calendar screen?
- Very easy
- Moderately easy
- Slightly easy
- Not at all easy

10. How helpful is the calendar screen when scheduling and modifying operations?
- Very helpful
- Moderately helpful
- Slightly helpful
- Not at all helpful

11. Please rank the following calendar features in order of importance.
- Ability to add operations from the calendar screen
- Ability to edit an operation by resizing it
- Dragging and dropping operations on the calendar
- Filtering of operations on the calendar screen.

12. Please add any comments you have about the calendar screen.

13. Which screen was easier to use to schedule and modify operations?
- Calendar
- Map
- Both about the same
14. Which screen do you think you would use more often?
   - Calendar
   - Map
   - Both about the same

15. How easy is it to navigate to different screens within Amaethon?
   - Very easy
   - Easy
   - Neutral
   - Difficult
   - Very difficult

16. Would you prefer to use Amaethon in its current form as a web application, or instead as a desktop application?
   - Prefer in current form as a web application
   - Would prefer as a desktop application
   - Have no preference

17. While you were using it, how often did Amaethon freeze, crash, or produce errors?
   - 0 times
   - 1-2 times
   - 4-6 times
   - 7 or more times
18. Please describe your reaction to the following statement:

I'm OK with a user interface that may take (a little) longer to use, if that interface is more pleasurable to use than a quicker one.

- Strongly agree
- Agree
- No opinion
- Disagree
- Strongly disagree

19. Were there any parts of Amaethon that you found particularly confusing or difficult to use?

20. Rate your overall satisfaction with Amaethon

- Extremely satisfied
- Moderately satisfied
- Slightly satisfied
- Neither satisfied nor dissatisfied
- Slightly dissatisfied
- Moderately dissatisfied
- Extremely dissatisfied

21. Do you have any suggestions for how Amaethon can be improved?

The following questions only apply to people who are currently working in the farming industry:
22. If you are currently using another software system to manage your organization's operations, please briefly explain the system and compare it to Amaethon.

23. Would you choose Amaethon to manage a farm enterprise?

- [ ] Yes
- [ ] No